Vyšší odborná škola, Střední škola,
Centrum odborné přípravy

ABSOLVENTSKÁ PRÁCE

Návrh autonomní navigace mobilního robotu Robotino a vytvoření výukové videodokumentace

Sezimovo Ústí, 2026 **Autor: Tomáš Čuchna**

VYŠŠÍ ODBORNÁ ŠKOLA, STŘEDNÍ ŠKOLA, CENTRUM ODBORNÉ PŘÍPRAVY

SEZIMOVO ÚSTÍ, BUDĚJOVICKÁ 421

ZADÁNÍ ABSOLVENTSKÉ PRÁCE

Student:	**Tomáš Čuchna**
Obor studia:	26-41-N/01 Elektrotechnika – mechatronické systémy
Název práce:	**Návrh autonomní navigace mobilního robotu Robotino a vytvoření výukové videodokumentace**
Anglický název práce:	**Design of autonomous navigation for the mobile robot Robotino and creation of educational video documentation**

Zásady pro vypracování:

1. Popište hardware a software mobilního robotu Robotino.
2. Popište programy RobotinoView, RobotinoSIM a Robotino Factory.
3. Vytvořte program pro ruční řízení robotu Robotino.
4. Autonomně ovládejte robotu Robotino v daném prostoru.
5. Aktualizujete operační systém robotu Robotino.
6. Vytvořte mapu školní učebny v RobotinoFactory.
7. Vytvořte programy pro řízení robotu Robotino v učebně na základě její mapy.
8. Vytvořte scénář pro natočení videa o robotu Robotino, video natočte a sestříhejte.
9. Absolventskou práci vypracujte problémově ve struktuře odpovídající vědecké práci.

Doporučená literatura:

[1] CORELL, NIKOLAUS. *Introduetion to Autonomous Rohois. Mechanisms, Sensors, Acturators, and Algorithms*. Boca Raton, FL, USA: CRC Press, 2020. ISBN 978-1138502383.

[2] THRUN, SEBASTIAN, *BURGARD. Wolfram a FOX. Dicter Probahilistic Robotics*. Cambridge. MA USA: The MIT Press, 2005. ISBN 978-0262201629.

Vedoucí práce:	Mgr. Bc. Miroslav V. Hospodářský, VOŠ, SŠ, COP Sezimovo Ústí
Odborný konzultant práce:	Ing. Daniel Semerád, Festo, s. r. o.
Oponent práce:	Ing. Jiří Roubal, VOŠ, SŠ, COP Sezimovo Ústí

Datum zadání absolventské práce: **1. 9. 2025**

Datum odevzdání absolventské práce: **15. 5. 2026**

Mgr. Bc. Miroslav V. Hospodářský
(vedoucí práce)

VYŠŠÍ ODBORNÁ ŠKOLA, STŘEDNÍ ŠKOLA, CENTRUM ODBORNÉ PŘÍPRAVY, Budějovická 421, Sezimovo Ústí

doc. PhDr. Mgr. Lenka Hrušková, Ph.D.
(ředitel školy)

V Sezimově Ústí dne 1. 9. 2025

Prohlášení

Prohlašuji, že jsem svou absolventskou práci vypracoval samostatně a uvedl jsem veškeré použité informační zdroje, veškerý použitý software a dodržel jsem Metodiku užití „umělé inteligence“, která je součástí metodiky tvorby absolventské práce.

Prohlašuji, že jsem v průběhu příprav a psaní této práce nepoužil nástroje „umělé inteligence“.

Stvrzuji, že jsem si vědom, že za obsah této práce plně zodpovídám.

V Sezimově Ústí dne 13.5.2026

podpis

Poděkování

Velmi rád bych na tomto místě poděkoval vedoucímu své absolventské práce, panu Mgr. Bc. Miroslavu V. Hospodářskému, za odborné vedení, cenné rady, ochotu a čas, který mi při konzultacích a řešení praktických problémů věnoval.

Mé velké poděkování patří rovněž mým rodičům za jejich nesmírnou trpělivost, trvalou morální i materiální podporu a vytváření optimálního zázemí nejen při tvorbě této práce, ale po celou dobu mého studia.

Anotace

Tato absolventská práce se zaměřuje na praktické zprovoznění mobilního výukového robota Robotino od společnosti Festo Didactic a návrh jeho autonomní navigace. Pro účely řízení pohybu byla v prostředí Robotino Factory vytvořena mapa školní učebny. Samotná autonomní jízda a plnění specifických úkolů, jako je sledování čáry, detekce a přiblížení k barevnému objektu, sekvenční vyhledávání tří různých barev, autonomní vyhýbání se překážkám a orbitální kroužení kolem objektu, byly realizovány pomocí metod strojového vidění z palubní kamery nebo s využitím všesměrového podvozku. Zásadním praktickým výstupem projektu je názorná výuková videodokumentace, která demonstruje funkčnost celého řešení a usnadní jeho budoucí integraci do výuky.

Klíčová slova: Mobilní robot Robotino; autonomní navigace; Robotino View; Robotino Factory; tvorba mapy; aktualizace operačního systému; řídicí programy; výuková videodokumentace; mechatronika.

Annotation

This graduation thesis focuses on the practical commissioning of the Robotino mobile educational robot by Festo Didactic and the design of its autonomous navigation. For motion control purposes, a map of the classroom was created within the Robotino Factory environment. The autonomous driving itself and the execution of specific tasks such as line following, detection and approach to a colored object, sequential search for three different colors, autonomous obstacle avoidance, and orbital circling around an object were implemented using computer vision methods from the onboard camera or by utilizing the omnidirectional chassis. A key practical output of the project is an illustrative educational video documentation that demonstrates the functionality of the entire solution and will facilitate its future integration into the curriculum.

Keywords: Mobile robot Robotino; autonomous navigation; Robotino View; Robotino Factory; map creation; operating system update; control programs; educational video documentation; mechatronics.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Mobilní robotika se dnes dynamicky rozvíjí s obrovským přesahem do Průmyslu 4.0 i školství. Klíčovou vlastností moderních robotů je autonomní navigace, jejíž pochopení vyžaduje práci s reálným hardwarem. Mobilní robot Robotino od společnosti Festo (Festo Didactic, 2024*c*) představuje ideální nástroj, protože bezprostředně propojuje teorii s testováním řídicích systémů. Orientace v neznámém prostoru a tvorba map je naprosto nezbytná pro zrychlení logistických procesů, což z této problematiky dělá velmi žádanou inženýrskou dovednost.

Obrázek 1.1: RobotinoV3

Problematika autonomní navigace je detailně popsána v odborné literatuře. Základní principy fungování senzorů, mechanismů a řídicích procesů pro autonomní systémy komplexně shrnuje například nedávná publikace o autonomních robotech (Correll, N. et al., 2022). Detailní vysvětlení metod lokalizace a tvorby map (SLAM) přináší kniha *Probabilistic Robotics* (Thrun, S. et al., 2005), která může v této problematice určitým způsobem pomoci. Ačkoliv je teorie zpracovaná dobře, ve výuce chybí jasný postup. Dostupné práce často řeší jen počítačové simulace (přes RobotinoSIM) nebo dílčí technické problémy a nepropojují je s vývojovým prostředím, jako je Robotino Factory (Festo Didactic, 2024*c*). V současných materiálech proto schází praktický postup, který by navíc doplňovala názorná videodokumentace.

Cílem této absolventské práce je návrh autonomní navigace mobilního robotu Robotino v reálném prostoru školní učebny. K dosažení tohoto cíle je nutné zajistit jeho spolehlivé ruční řízení, vytvořit mapu školní učebny pomocí softwaru Robotino Factory a následně navrhnout příslušné řídicí programy s využitím kamery, infračervených senzorů vzdálenosti a interní odometrie Robotinu. Nedílnou součástí a významným výstupem je rovněž vytvoření scénáře a následný sestřih výukové videodokumentace, která názorně demonstruje navržené řešení v praxi.

Struktura této práce, která je vytvořena v systému LaTeX pomocí distribuce MiKTeX (Schenk, C., 2024) a editoru TeXstudio (van der Zander, B. a Sundermeyer, J., 2024), je logicky rozdělena do navazujících částí. Kapitola 2 je věnována detailnímu popisu hardwarového a softwarového vybavení mobilního robotu Robotino. V kapitole 3 jsou charakterizovány klíčové vývojové nástroje, konkrétně programy RobotinoView, RobotinoSIM a Robotino Factory. Realizace a verifikace autonomních úloh je obsažena v kapitole 4, která ukazuje celý postup oživování systému. Popisuje začátky s ručním řízením, metodiku tvorby mapy školní učebny, návrh programů kamerové navigace a zpracování finální výukové videodokumentace. Závěrečná kapitola 5 pak shrnuje dosažené výsledky a hodnotí celkový přínos práce.

Součástí práce je také podrobná videodokumentace, která obsahuje klíčové fáze oživování robotu a realizaci autonomních úloh. Toto shrnující video je pro názornost dostupné online na adrese: `https://www.youtube.com/watch?v=zD8QVryrceg`.

Kapitola 2

Hardware a software mobilního robotu Robotino

Mobilní robot Robotino, vyráběný společností Festo, představuje komplexní edukační a výzkumnou a edukační platformu. V rámci této absolventské práce je kladen důraz na specifikace třetí generace tohoto zařízení, se kterým probíhala praktická realizace zadaných úkolů. V následujících podkapitolách bude detailně popsána jeho hardwarová architektura a dostupné softwarové nástroje.

2.1 Hardwarová specifikace

Konstrukční řešení robotu Robotino se vyznačuje vysokou stabilitou a mechanickou odolností. Základem podvozku jsou tři všesměrová kola, takzvaná omni-wheels, díky nimž se robot může plynule pohybovat do libovolného směru bez nutnosti předchozího natáčení celé konstrukce. Pohon těchto kol zajišťují výkonné stejnosměrné motory vybavené přesnými enkodéry, což umožňuje detailní řízení pohybu a přesné odměřování ujeté vzdálenosti.

Obrázek 2.1: LiDAR

Za účelem vnímání okolního prostředí je systém vybaven bohatou senzorickou výbavou. Základní orientaci a detekci překážek zajišťuje devět infračervených senzorů, které jsou rovnoměrně rozmístěny po obvodu platformy. K dispo-

zici je dále integrovaný gyroskop a akcelerometr, jež poskytují důležitá data pro stabilizaci a zvyšují celkovou přesnost navigace. Pro pokročilejší úlohy, jako je zpracování obrazu, slouží kamera s vysokým rozlišením.

Významnou výhodou je možnost instalace volitelného laserového skeneru (LiDAR), který je nezbytný pro detailní mapování prostoru a navazující autonomní navigaci.

Systém je rovněž vybaven rozšířenými I/O porty, které umožňují připojení dalších analogových i digitálních senzorů a akčních členů.

Obrázek 2.2: Procesor Intel Atom (převzato z (Wikipedia contributors, 2026))

Jádrem celého systému je řídicí počítačová jednotka osazená moderním procesorem (ve verzi architektury Intel Atom nebo ARM, dle konkrétní konfigurace). Operační paměť o velikosti 4 GB RAM zaručuje dostatečný výpočetní výkon i pro náročnější úlohy, jako je právě zpracování obrazu nebo komplexní výpočty mapovacích algoritmů. Ukládání dat je řešeno pomocí rychlého SSD disku. Napájení zajišťuje efektivní lithium-iontová baterie. Díky systému pro automatické řízení spotřeby energie dosahuje robot provozní výdrže až 4 hodiny na jedno nabití.

2.2 Softwarová specifikace

Na úrovni operačního systému využívá Robotino distribuci Linuxu. Vzhledem k datu realizace praktické části práce byla jednou z priorit provedena aktualizace operačního systému na modernější verzi Ubuntu (Canonical Ltd., 2020), aby byla zajištěna kompatibilita s aktuálními nástroji. Architektura je navíc plně kompatibilní s platformou ROS (Robot Operating System), což umožňuje efektivní využití všech dostupných senzorických a periferních modulů robota.

Pro vývoj programů a přímé řízení slouží především grafické vývojové prostředí Robotino View 3 (Festo Didactic, 2024*c*). Tento nástroj disponuje intuitivním uživatelským rozhraním a obsahuje řadu předpřipravených funkčních bloků pro snadnou práci se senzory či kamerou. Kromě vizuálního programování je systém plně otevřen pro psaní kódů

v tradičních jazycích, jakými jsou C++, Python nebo prostředí MATLAB.

Důležitou součástí softwarového ekosystému je možnost pokročilé simulace. Kromě specializovaných nativních programů lze využít silné podpory balíčků v ROS pro modelování lokalizace, mapování (SLAM) a navigace (Thrun, S. et al., 2005). Samotná komunikace s robotem probíhá primárně prostřednictvím Wi-Fi, kdy robot pomocí USB adaptéru vytváří přístupový bod (hotspot) pro připojení uživatele.

Kapitola 3

Vývojová a simulační prostředí

Pro úspěšnou realizaci autonomní navigace a řízení mobilního robotu Robotino je nezbytné využití adekvátních softwarových nástrojů. Tato kapitola detailně popisuje trojici klíčových programů, kterými jsou RobotinoView, RobotinoSIM a Robotino Factory.

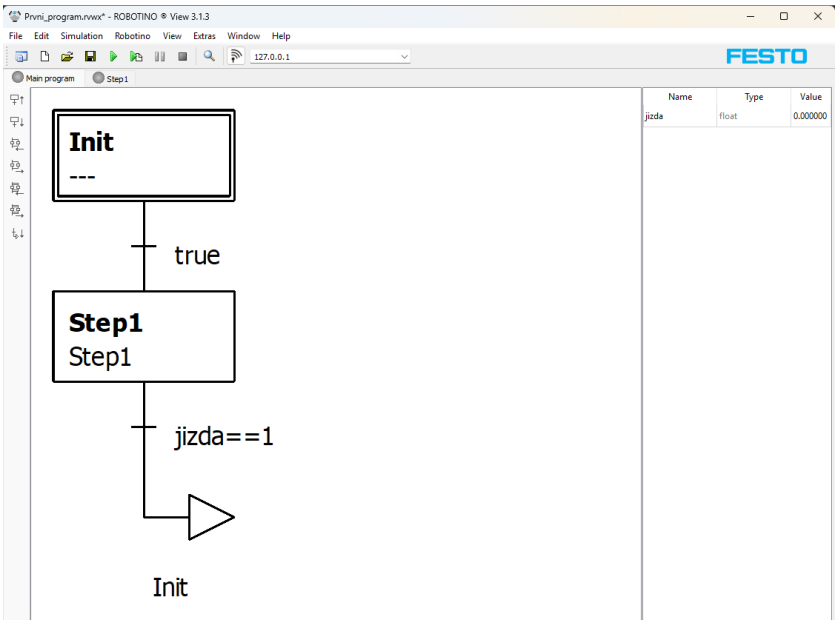
3.1 Prostředí RobotinoView

Vývojové prostředí RobotinoView představuje primární grafické prostředí pro tvorbu řídicích programů. Jeho uživatelské rozhraní, jehož celkový vzhled ilustruje obrázek 3.1, je pro maximální přehlednost rozděleno do několika logických celků. V horní části okna se nachází hlavní menu (obsahující záložky File, Edit, Simulation, Robotino, View a Help) a pod ním je umístěna nástrojová lišta pro rychlý přístup k často využívaným funkcím. Z této lišty lze program spouštět, zastavovat, ukládat a především zde probíhá zadání IP adresy pro navázání spojení s fyzickým robotem.

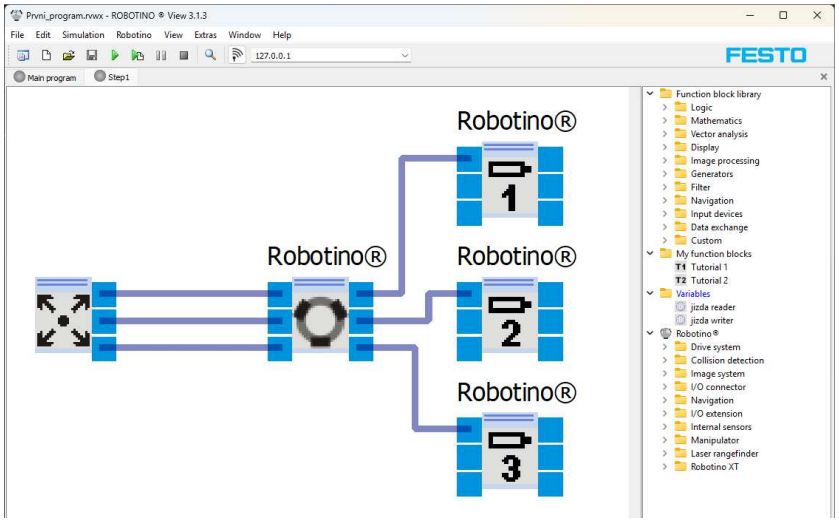
Základním pracovním prostorem je centrální plocha, na kterou se umisťují jednotlivé programové bloky (například *Init* nebo *Step1*) a propojují se linkami definujícími tok dat a řízení. Předdefinované bloky se vybírají z knihovny v levém postranním panelu. Konfigurační panel na pravé straně následně slouží k definici proměnných, jež mohou řídit například ukončení podprogramů.

Knihovna bloků je pro snadnou orientaci strukturována do několika tematických složek. Pohyb a ovládání motorů se řeší pomocí bloků ve složce *Drive System*, zatímco *Collision Detection* obsahuje logiku pro vyhýbání se překážkám na základě dat z infračervených či ultrazvukových senzorů.

Pro kamerové úlohy je vyhrazena složka *Image System*. Zpracování dat z interních snímačů (gyroskop, enkodéry) probíhá pomocí prvků v *Internal Sensors*, přesná lokalizace pak využívá moduly *Laser Rangefinder* a *Navigation*. K dispozici jsou rovněž složky *I/O Connector* a *I/O Extension* pro obsluhu vstupně-výstupních signálů, případně specifické bloky pro externí manipulátory (*Manipulator*).



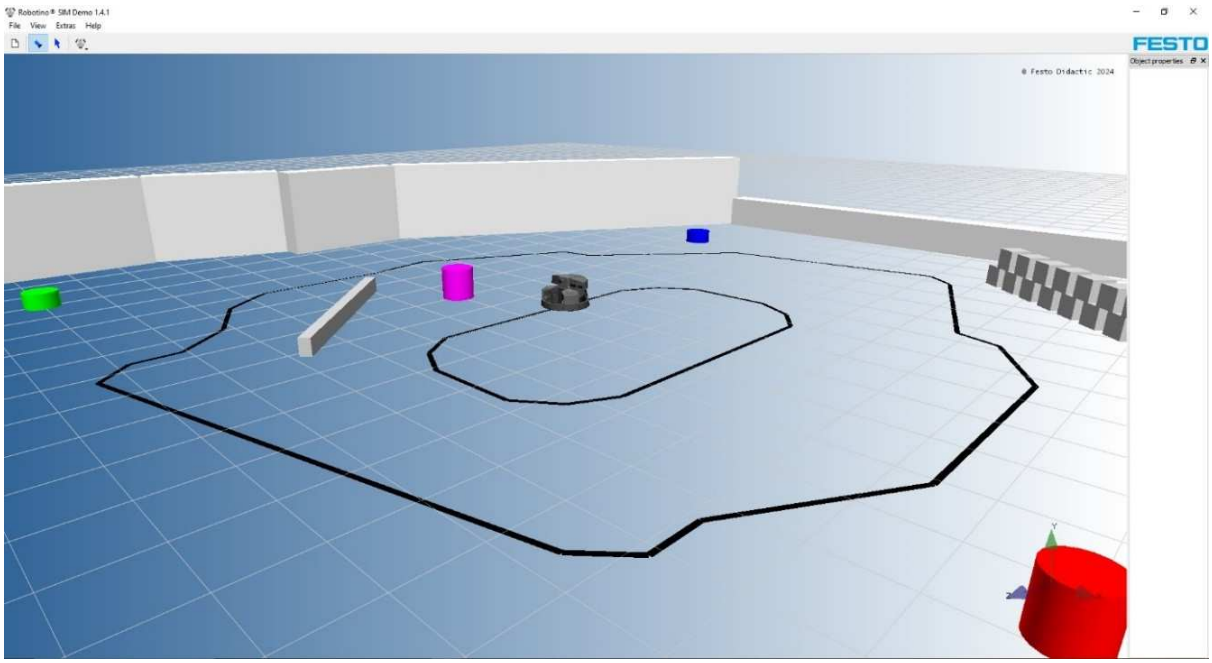
Obrázek 3.1: Hlavní uživatelské rozhraní programu RobotinoView



Obrázek 3.2: Pracovní přostředí v bloku Step1

3.2 Prostředí RobotinoSIM

RobotinoSIM je simulační nástroj navržený k bezpečnému testování vytvořených programů před jejich nahráním do reálného zařízení.



Obrázek 3.3: Simulace pracovního prostoru v programu RobotinoSIM

Jak ukazuje obrázek 3.3, hlavní část okna tvoří 3D simulační prostor. V něm se nachází virtuální model robotu a definované okolní prostředí s překážkami. Zásadní výhodou je možnost přímého propojení programu RobotinoSIM s vývojovým prostředím RobotinoView pomocí virtuální IP adresy.

Díky tomuto spojení lze vizuálně sledovat vykonávání řídicího kódu. Běh simulace se ovládá přímo z RobotinoView, přičemž do kódu je možné přidat i bloky pro manuální řízení virtuálního robotu pomocí klávesnice či joysticku. Pro resetování virtuálního vývojového prostředí do původního stavu slouží funkce vytvoření nové scény.

3.3 Prostředí Robotino Factory

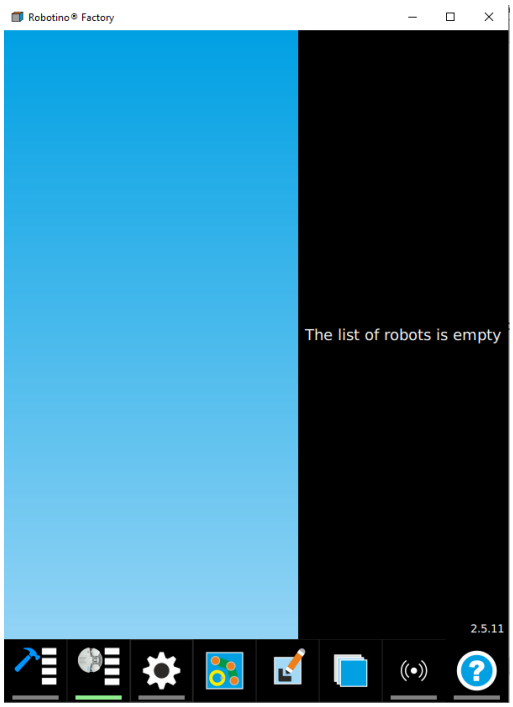
Aplikace Robotino Factory představuje komplexní nástroj pro správu, mapování a navigaci mobilních robotů (viz obrázek 3.4).

Z hlediska cílů této práce spočívá nejdůležitější funkce programu v možnosti tvorby

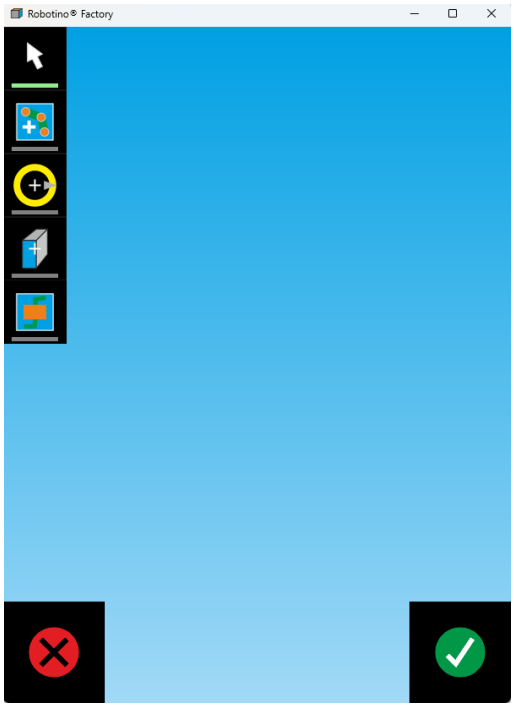
mapy reálného prostředí a následné definici navigačních úloh. Při mapovacím procesu je robot nejprve ručně ovládán (teleoperován) daným prostorem. Během tohoto pohybu systém pomocí integrovaného laserového skeneru a odometrie průběžně skenuje okolí a vykresluje jeho 2D půdorys. Tímto způsobem byla v rámci praktické realizace vytvořena mapa školní učebny.

Vygenerovanou mapu lze v prostředí programu dále detailně upravovat. K dispozici jsou nástroje pro vkládání virtuálních zdí, definování zakázaných zón (kam robot nesmí vjet) nebo vyznačení specifických orientačních a cílových bodů.

Kromě samotného mapování slouží rozhraní k přímému řízení a monitoringu. Poskytuje přehledný ovládací panel, ve kterém je možné v reálném čase sledovat aktuální stav robotu, jeho přesnou polohu na vytvořené mapě a stav nabití baterie. Uživatel zde může zadávat nové požadavky na autonomní přesun do vybraných lokací, přičemž software na pozadí automaticky vypočítá nejkratší bezpečně průjezdnou trasu a zajistí vyhýbání se překážkám.



Obrázek 3.4: Rozhraní pro mapování



Obrázek 3.5: Multifunkční lišta

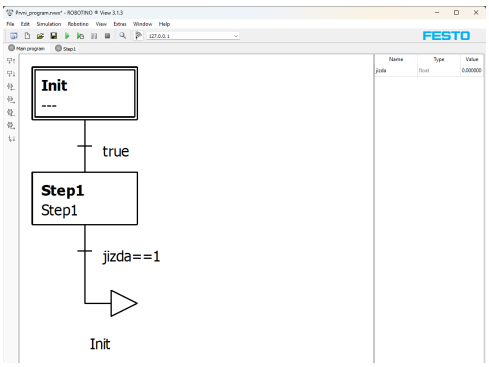
Kapitola 4

Realizace a verifikace autonomních úloh

Tato kapitola popisuje postupný proces zprovoznění a programování zadaných úkolů na mobilním robotu Robotino. Počínaje tvorbou programu pro manuální ovládání, přes nezbytnou aktualizaci operačního systému, až po komplexní úlohy spojené s mapováním v prostředí školní učebny.

4.1 Program pro ruční řízení robotu

Nezbytným krokem pro budoucí mapování prostoru je zajištění spolehlivého ručního řízení robotu. Pro tento účel byl v prostředí RobotinoView vytvořen program, který umožňuje ovládat směr a rychlost pohybu v reálném čase.

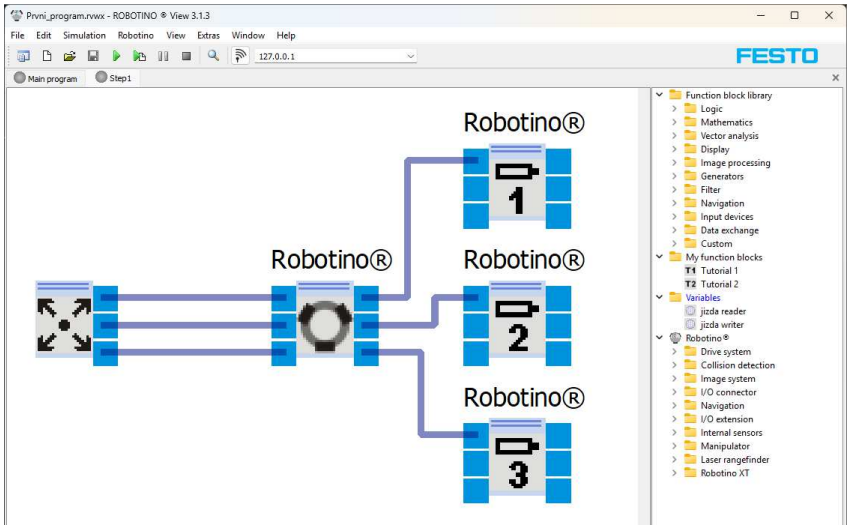


Obrázek 4.1: Struktura hlavního programu pro ruční řízení (Main program)

Na obrázku 4.1 je vidět hlavní řídicí struktura tohoto programu. Už u této základní úlohy je využito takzvané sekvenční řízení. Zajišťuje to, že se program nespustí živelně celkově, ale prochází jasně definovanými stavy. Celý proces začíná v inicializačním kroku *Init*, který připraví robota k činnosti. Následuje přechodová podmínka *true*, která funguje jako trvalý souhlas k okamžitému přechodu do hlavního pracovního kroku označeného jako *Step1*. Právě v tomto bloku je „schována“ celá řídicí logika propojující uživatelské vstupy s motory robotu.

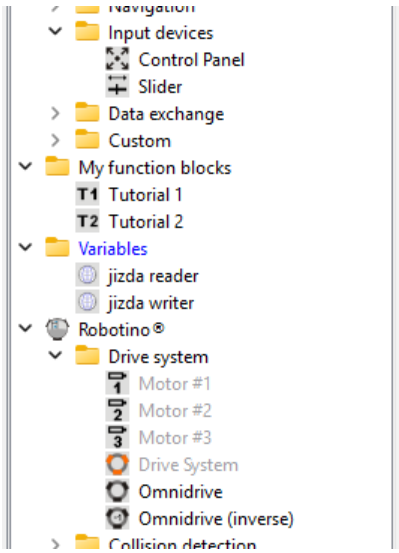
Pod krokem *Step1* se nachází výstupní přechodová podmínka *jizda==1*. Jak je vidět na pravé straně obrazovky v seznamu proměnných, v systému byla vytvořena uživatelská proměnná *jizda* (datového typu *float*) s výchozí hodnotou 0. V tomto konkrétním programu pro ruční řízení sice tato proměnná nemění svou hodnotu a slouží spíše jako demonstrace možností prostředí RobotinoView, nicméně ukazuje velmi důležitý princip práce se systémem.

Prostředí RobotinoView totiž umožňuje uživateli definovat si v hlavním programu libovolné množství vlastních proměnných. S těmito proměnnými pak lze napříč celým projektem libovolně pracovat – můžeme do nich ukládat data ze senzorů, provádět s nimi matematické operace nebo je využívat právě jako logické přepínače. V našem případě platí, že dokud se hodnota proměnné nezmění, program neustále běží v hlavním pracovním kroku. Pokud by v případné složitější verzi programu došlo (například na základě stisku tlačítka) ke změně proměnné na hodnotu 1, program by z pracovního kroku vyskočil a pomocí symbolu skoku (trojúhelník dole) by se vrátil zpět na začátek do bloku *Init*.



Obrázek 4.2: Logika ručního řízení uvnitř pracovního kroku Step1

Samotný obsah kroku *Step1* je zobrazen na obrázku 4.2. K generování povelů zde slouží blok *Control Panel* ze složky *Input devices*. Tento prvek generuje vstupní hodnoty na základě interakce uživatele a umožňuje tak intuitivní ovládání robotu, které se velmi podobá klasickému dálkovému ovladači nebo gamepadu.



Obrázek 4.3: Použité funkční bloky z knihovny RobotinoView

Výstupní data z ovládacího panelu jsou následně přivedena do výpočetního bloku *Drive System* (případně *Omnidrive*), jenž se nachází v knihovně hardwaru (viz obr. 4.3). Tento blok matematicky vypočítává požadované hodnoty otáček pro jednotlivé akční členy (*Motor #1*, *Motor #2* a *Motor #3*) na základě požadované rychlosti v osách x a y (v_x, v_y) a úhlové rychlosti rotace ω. Na stejném principu, avšak v opačném směru, pracuje doplňkový blok *Omnidrive (inverzní)*, který dokáže zpětně vypočítat reálné rychlosti v_x, v_y a ω na základě skutečných otáček, které motory v danou chvíli vykonávají.

Tyto podrobné technické definice a popisy vstupních i výstupních parametrů pro každý z těchto funkčních bloků si můžete prostudovat přímo v prostředí RobotinoView. Lze je vyvolat kliknutím pravým tlačítkem myši na konkrétní blok a zvolením položky *Help* (Nápověda).

4.2 Aktualizace operačního systému robotu

Pro zajištění plné kompatibility s nejnovějšími vývojovými nástroji, stabilitou systému a dostupností moderních funkcí pro autonomní navigaci bylo nutné před zahájením sa-

motného programování provést kompletní aktualizaci operačního systému a firmwaru řídicí jednotky robotu Robotino.

Celý proces se skládal z několika na sebe navazujících kroků. Nejprve byla provedena záloha veškerých existujících uživatelských dat a programů z interního úložiště robotu. Následně byl z oficiálního repozitáře výrobce (Festo Didactic, 2024*b*) stažen instalační obraz (tzv. image) s označením verze 4.20.10, postavený na distribuci Ubuntu 20.04 (Canonical Ltd., 2020). Vzhledem k velikosti souboru byl tento obraz distribuován ve více částech (part1 a part2).

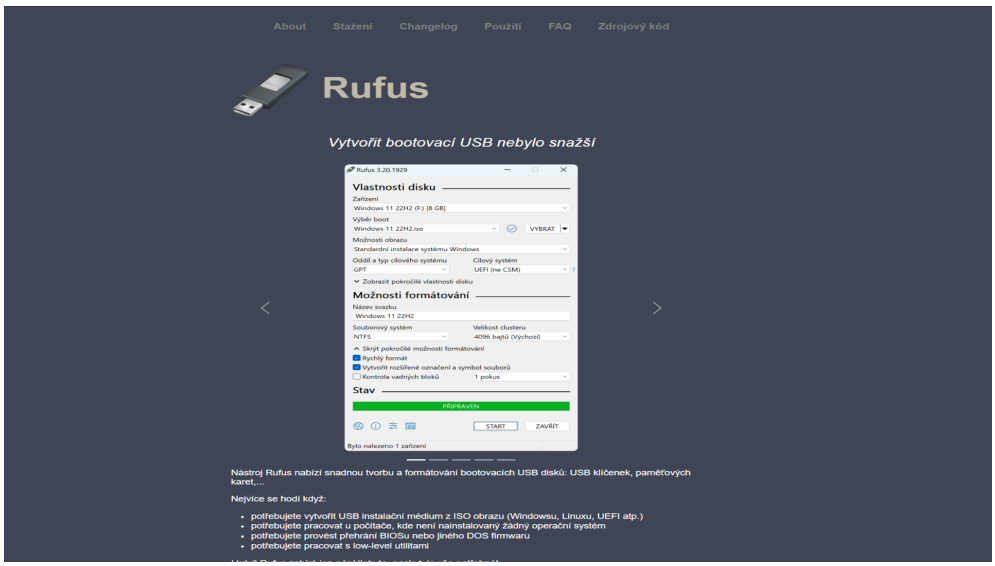
Robotino4 images

Page

Name	OS	Image	MD5
4.20.10	Ubuntu 20.04	part1 part2	md5

Obrázek 4.4: Stažení operačního systému Ubuntu 20.04 (verze 4.20.10) převzato z (Festo Didactic, 2024*b*)

Po úspěšném stažení a sloučení souborů bylo nezbytné vytvořit bootovací (spouštěcí) USB flash disk. K tomuto účelu byl využit volně dostupný softwarový nástroj Rufus, který zformátoval přenosné médium a vytvořil potřebnou strukturu.



Obrázek 4.5: Softwarový nástroj Rufus, převzato z (Akeo, 2024)

Samotný postup instalace se následně přísně držel oficiálních pokynů pro aktualizaci systému (Festo Didactic, 2024*a*):

1. **Příprava instalačního média:** Zkopírování obrazu operačního systému a firmwaru do adresáře *boot* na připraveném USB disku.

2. **Hardwarové propojení:** Zapojení USB disku do volného portu robotu. Pro vizuální kontrolu a zadávání příkazů byl připojen externí monitor k VGA výstupu a standardní klávesnice do dalšího dostupného USB portu.

3. **Spuštění bootovacího menu:** Robot byl zapnut za současného nepřetržitého stisku klávesy **F7** (tato klávesa je určena pro edici Premium, u edice Basic se využívá klávesa **F11**).

4. **Výběr úložiště:** V zobrazeném systémovém dialogu „Vyberte spouštěcí zařízení“ byla pomocí směrových šipek vybrána třetí položka, která odpovídala typu připojeného USB disku. Volba byla potvrzena stisknutím klávesy **Enter**.

5. **Instalace obrazu:** Po načtení základního instalačního prostředí bylo zadáno identifikační číslo nového obrazu a volba byla potvrzena klávesou **Enter**. Následně byl celý proces instalace zahájen stiskem klávesy **Y** (Yes) a **Enter**.

6. **Dokončení procesu:** Samotný proces flashování probíhal zcela automaticky. Po úspěšném dokončení instalace se Robotino samočinně vypnulo.

Po novém spuštění aktualizovaného systému bylo nezbytné provést finální síťovou a hardwarovou konfiguraci. Vzhledem k přeinstalování jádra operačního systému dochází k odstranění původních síťových profilů a k obnovení továrního nastavení komunikace. Z tohoto důvodu Robotino po dokončení instalace automaticky aktivovalo režim dočasného přístupového bodu (Access Point) s výchozím názvem sítě *RobotinoAP* a heslem *robotino123*.

Pro znovuuvedení robotu do plnohodnotného provozu bylo následně nutné připojit řídicí počítač přes Wi-Fi k tomuto vytvořenému přístupovému bodu a přistoupit k administraci systému zadáním výchozí IP adresy do webového prohlížeče.

Jako volitelný, avšak mnohem uživatelsky přívětivější krok k detailní konfiguraci, bylo využito webové rozhraní robotu. Na konfigurační stránce v sekci *wlan0* byly upraveny parametry bezdrátové sítě podle potřeby. Nakonec musely být pomocí integrovaného průvodce (Guide Setup) znovu aktivovány a nakalibrovány veškeré palubní senzory a periferie. Tím byl systém plně připraven pro nasazení v autonomních úlohách.

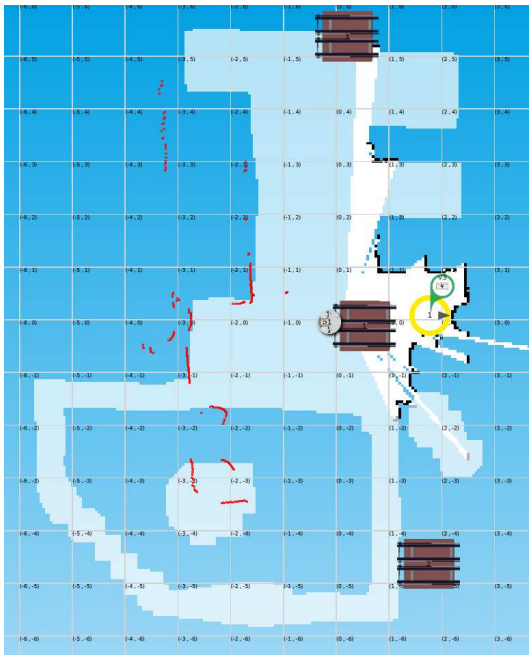
4.3 Tvorba mapy školní učebny

Nezbytným předpokladem pro realizaci autonomní navigace je vytvoření 2D mapy okolního prostředí. Pro tento účel byl využit software Robotino Factory ve spolupráci s integrovaným laserovým skenerem robotu.

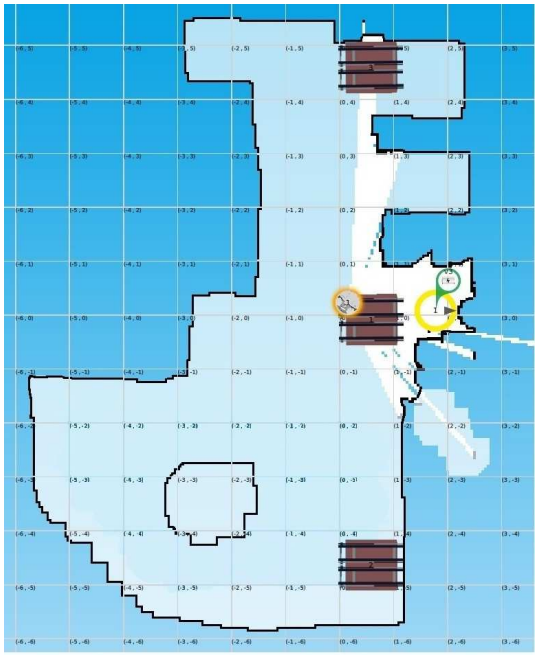
Před zahájením mapování bylo nutné nakonfigurovat obousměrnou komunikaci mezi robotem a řídicím počítačem. Prostředí webového rozhraní (přes IP adresu robotu) byly aktivovány komunikační režimy *Master* a *Slave*. Bez těchto režimů nebylo možné robota k programu Robotino Factory přes Wi-Fi hotspot připojit.

Samotné mapování probíhalo fyzickým posouváním robotu po učebně. Klíčovou roli zde sehrál 2D LiDAR skener *Hokuyo URG-04LX-UG01*, který na základě odrazů laserového paprsku od překážek generoval surový model prostoru. Jelikož LiDAR v reálném prostředí vykazuje drobné nepřesnosti, byla mapa již během skenování hrubě očištěna pomocí editačních nástrojů přímo v prostředí Robotino Factory. Surový výstup tohoto procesu zachycuje obrázek 4.6.

Vzhledem k fyzikálním limitům laserového měření (problémy s detekcí skleněných ploch či nízkých podnoží stolů) byla mapa následně exportována pro finální postprocesing do grafického editoru Adobe Photoshop. Zde byly do mapy dokresleny souvislé černé kontury, které definují pevné hranice učebny. Výsledná podoba mapy je znázorněna zde na obrázku 4.7.



Obrázek 4.6: Surová mapa učebny



Obrázek 4.7: Finální mapa učebny

4.4 Autonomní řízení a senzorická navigace robotu

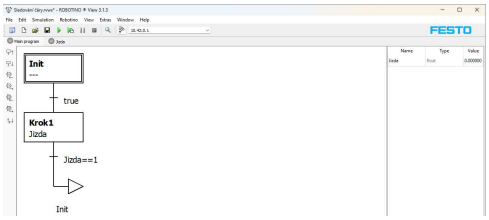
Původním záměrem v této fázi projektu bylo využití globálních navigačních funkcí programu Robotino Factory. Ačkoliv byla mapa prostředí úspěšně vytvořena a komunikace s robotem spolehlivě navázána, při zadání konkrétních cílových bodů (waypointů) do mapy nedokázal systém vygenerovat a vykonat požadovanou trajektorii. Tento jev, pravděpodobně způsobený interní softwarovou chybou nebo skrytým síťovým omezením platformy, vedl k operativnímu přehodnocení strategie autonomního řízení.

Jako alternativní a v mnoha ohledech flexibilnější řešení bylo zvoleno využití metod lokální navigace. Pomocí programu RobotinoView bylo navrženo, odladěno a prostřednictvím Wi-Fi hotspotu přímo do paměti robotu nahráno pět nezávislých řídicích programů. Tyto úlohy k autonomní orientaci v prostoru využívaly kombinaci dat z palubní kamery, infračervených senzorů vzdálenosti a interní odometrie.

Jednotlivé programy byly navrženy tak, aby postupně prověřovaly různé aspekty senzorického vnímání – od pokročilého strojového vidění při sledování objektů až po využití všesměrové kinematiky podvozku pro komplexní manévrování v blízkosti překážek. Před samotným nasazením na reálný hardware byly veškeré algoritmy důkladně otestovány v simulačním prostředí RobotinoSIM.

4.4.1 Sledování čáry

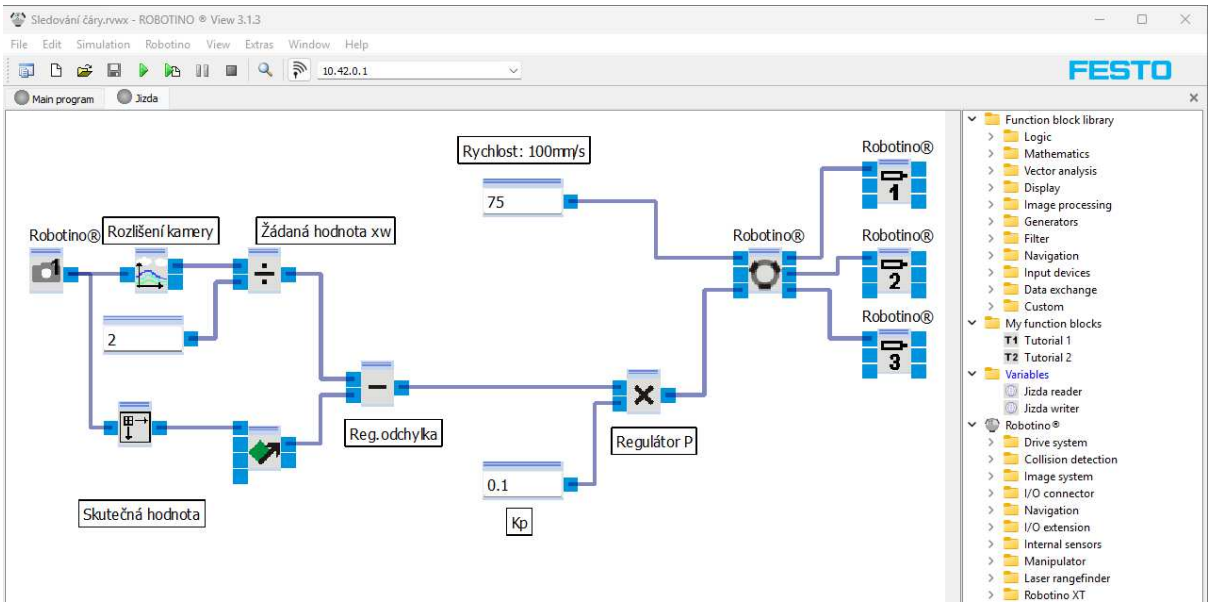
První autonomní program byl zaměřen na klasickou, avšak pro průmyslovou logistiku velmi důležitou úlohu – sledování čáry (tzv. Line Tracking). Cílem tohoto programu bylo v reálném čase zpracovávat obrazová data z palubní kamery, detekovat kontrastní čáru na podlaze a dynamicky upravovat rychlost a směr otáčení motorů tak, aby robot plynule kopíroval požadovanou trajektorii. Pro správnou funkci detekce bylo nutné mechanicky nastavit palubní kameru robotu pod úhlem přibližně 45° směrem k podlaze. Vytvořený program byl nejprve simulován a následně bezdrátově nahrán přímo do řídicí jednotky robotu prostřednictvím vytvořeného Wi-Fi hotspotu.



Obrázek 4.8: Hlavní řídicí struktura programu (Main program)

Na obrázku 4.8 je vidět základní struktura hlavního programu. Celý program je navržen tak, aby fungoval sekvenčně v jasně daných krocích. Program začíná v inicializačním bloku *Init*, který bezpečně připraví robota na jízdu. Tranzice s hodnotou *true* následně zajišťuje okamžitý přechod do pracovního bloku *Krok1 (Jizda)*.

Aby robot věděl, jak dlouho má čáru sledovat, je pod krokem jízdy umístěna kontrolní podmínka *Jizda==1*. Ta zajišťuje běh programu ve smyčce až do doby, dokud není splněna podmínka pro ukončení úlohy.



Obrázek 4.9: Blokové schéma regulační smyčky uvnitř podprogramu Jízda

Detailní pohled na obsah řídicího podprogramu je znázorněn na obrázku 4.9. Řešení úlohy lze logicky rozdělit na dvě hlavní části: zpracování obrazu (získání dat) a samotnou regulaci pohybu.

Zpracování obrazových dat a výpočet odchylky

Zdrojem dat je funkční blok *Camera*, který nepřetržitě snímá prostor před robotem. Na jeho výstup je napojen blok *Line detector* (Detektor čáry). Tento matematický filtr analyzuje pixely v obrazu a hledá nepřerušený pruh kontrastní barvy. Výstupem tohoto bloku je aktuální horizontální pozice těžiště čáry v obrazu (v pixelech).

Aby robot věděl, kam má přesně zatáčet, je nutné vypočítat takzvanou regulační odchylku. Program nejprve přečte celkové horizontální rozlišení kamery (blok *Image Information*) a pomocí matematického bloku pro dělení (hodnotou 2) vypočítá přesný střed

obrazu. Tato vypočtená hodnota představuje **žádanou hodnotu** w – tedy ideální stav, kdy se čára nachází přesně uprostřed zorného pole kamery.

Následně je do bloku odčítání přivedena žádaná hodnota středu a skutečná pozice čáry detekovaná filtrem. Rozdíl těchto dvou hodnot představuje **regulační odchylku** e. Pokud je čára přesně uprostřed, odchylka je nulová. Pokud čára uhýbá doleva odchylka nabývá záporných hodnot, pokud doprava odchylka nabývá kladných hodnot.

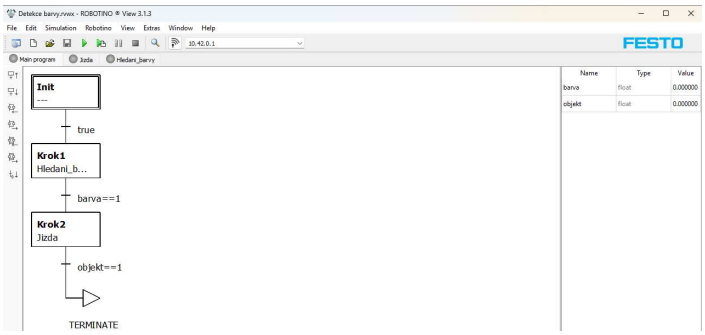
Proporcionální regulátor a řízení podvozku

Vypočítaná regulační odchylka je dále vedena do matematického bloku násobení, který v tomto obvodu plní funkci **proporcionálního (P) regulátoru**. Odchylka je zde násobena nastavenou konstantou zesílení ($K_\mathrm{p} = 0{,}1$). Velikost této konstanty byla stanovena experimentálně během testování. Zajišťuje, že úhel zatáčení je přímo úměrný tomu, jak moc se čára odchyluje od středu – čím větší je chyba, tím ostřeji robot zatáčí.

Výstup z regulátoru určuje úhlovou rychlost rotace robotu (osa ω). Dopředná rychlost v ose x je naproti tomu nastavena jako konstantní pomocí bloku konstanty na bezpečnou hodnotu 75 mm/s.

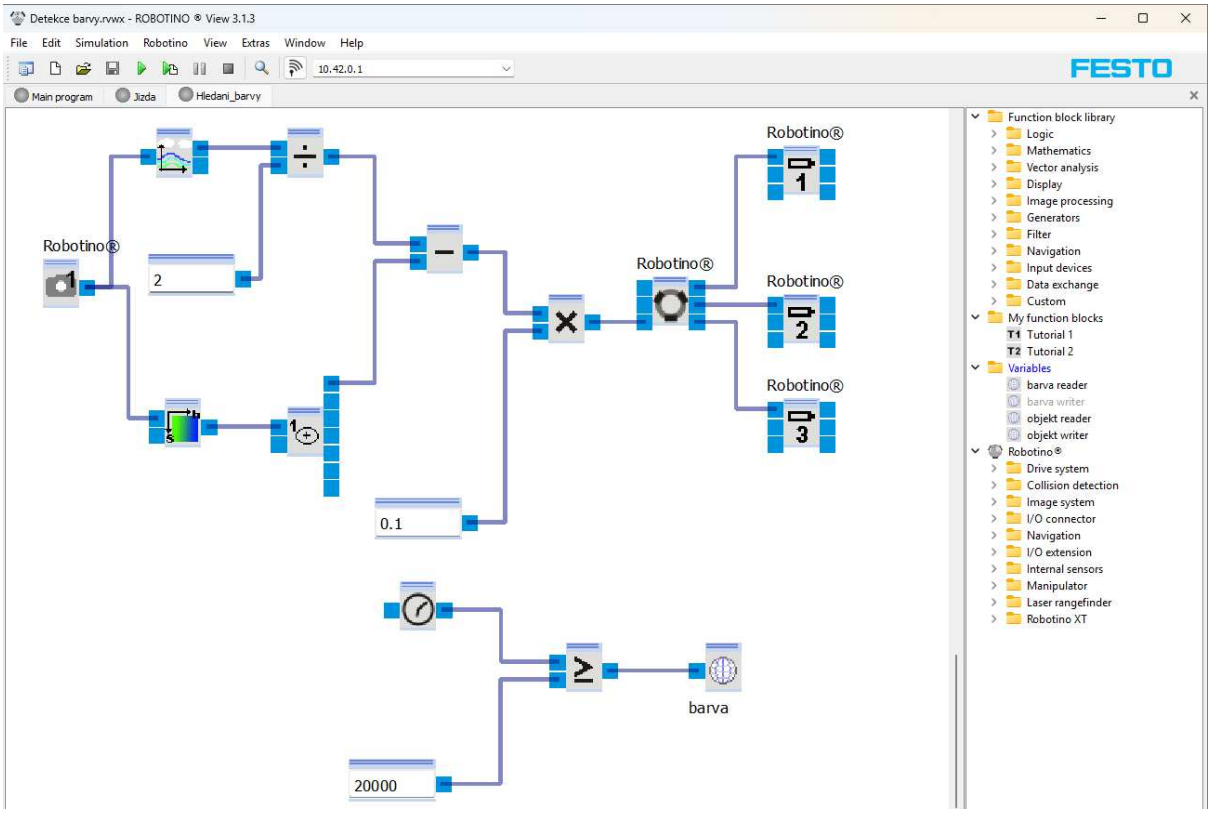
4.4.2 Hledání objektu podle barvy a přiblížení

Druhá autonomní úloha se zaměřila na detekci a následování objektu specifické barvy (tzv. color tracking). Logika celého procesu vyžadovala, aby se robot nejprve otáčel kolem své osy, dokud v zorném poli kamery neidentifikuje předem definovanou barvu. Jakmile byl cíl nalezen, robot zastavil rotační pohyb, směrově se na objekt vycentroval a plynule se k němu začal přibližovat. Vzdálenost k objektu byla kontinuálně měřena infračerveným senzorem, což umožnilo robotu autonomně zastavit v předem definované bezpečné vzdálenosti (když výstup ze senzoru klesl pod absolutní hodnotu 0,1 V).



Obrázek 4.10: Hlavní řídicí program (Main program) pro detekci barvy

Struktura hlavního programu (obr. 4.10) je opět založena na sekvenčním řízení. Po bloku *Init* následuje krok *Krok1 (Hledani_barvy)*, kde probíhá rotační vyhledávání. Aby mohl program plynule přejít k další fázi, byly vytvořeny globální proměnné, z nichž stěžejní je pro tento krok proměnná *barva*. Tranzice pod prvním krokem obsahuje podmínku *barva==1*. Na rozdíl od okamžité reakce na první detekci je však zápis do této proměnné v podprogramu řízen časovačem (Timer). Robotu je tak poskytnut pevně stanovený časový úsek (například 20 sekund, odpovídající konstantě 20000), během kterého má prostor požadovaný barevný odstín v prostoru nalézt a směrově se na něj vycentrovat. Jakmile tento nastavený čas uplyne, proměnná *barva* nabude hodnoty 1 a program přejde do kroku *Krok2 (Jizda)*, kde dochází k samotnému přiblížení k objektu. Celý proces je následně ukončen pomocí tranzice vyhodnocující proměnnou *objekt*, která se aktivuje po dosažení minimální bezpečné vzdálenosti z čelního senzoru.



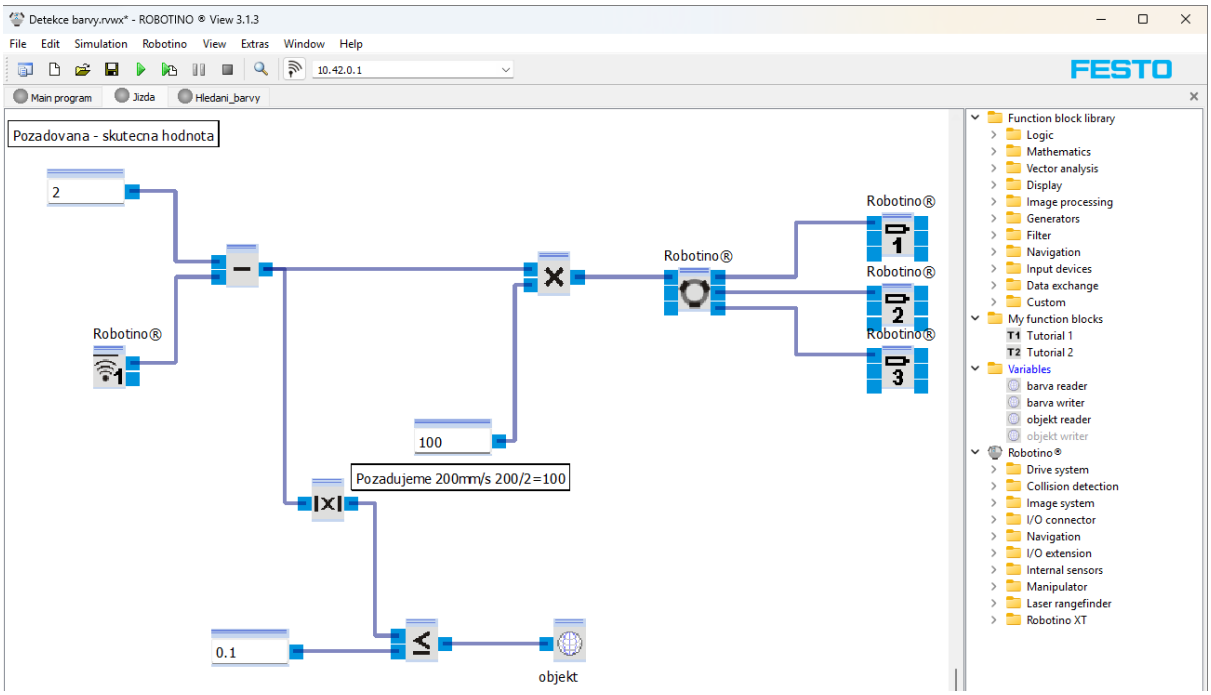
Obrázek 4.11: Podprogram pro hledání barvy a centrování

Obsah kroku *Hledani_barvy* je ukázán na obrázku 4.11. Jádrem zpracování obrazu je blok *Camera*, jehož výstup je přiveden do bloku *Color range finder* (vyhledávač barevného

rozsahu). V tomto bloku je uložena přesná definice sledované barvy. Navazující blok *Segment tracker* identifikuje pixely odpovídající této barvě a určuje jeho těžiště.

Regulační smyčka pro vycentrování na objekt využívá opět princip výpočtu odchylky. Z bloku *Image information* je načteno celkové rozlišení kamery, které je vyděleno konstantou 2 pro získání ideálního středu obrazu (žádaná hodnota).

Od tohoto středu se odečítá aktuální poloha detekovaného objektu. Výsledná regulační odchylka je násobena konstantou a přivedena do bloku *Omnidrive* jako požadavek rotační rychlosti (ω).



Obrázek 4.12: Podprogram pro autonomní jízdu k objektu a zastavení

Jakmile je robot směrově vycentrován, aktivuje se podprogram *Jizda* (obr. 4.12). Zde je rychlost generována na základě dat z infračerveného senzoru vzdálenosti (*Distance sensor 1*). Tento senzor neměří vzdálenost přímo v metrech, ale vrací napětí (typicky v rozsahu 0–2,5 V), přičemž vyšší napětí značí bližší překážku.

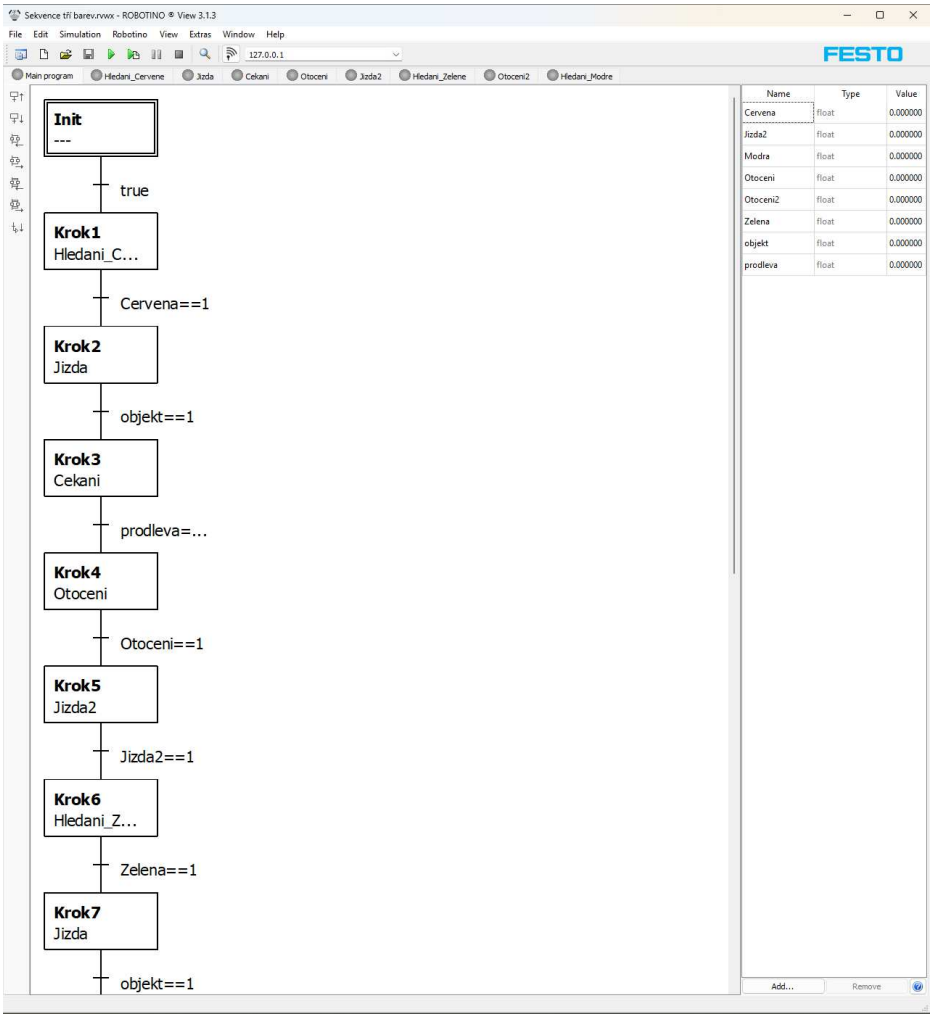
Výstup ze senzoru je odečten od konstanty určující bezpečnou vzdálenost a následně vynásoben hodnotou určující rychlost pohybu vpřed. Tato hodnota vstupuje do bloku *Omnidrive*. Pro bezpečné zastavení je využito bloku *Absolute value* napojeného na výstup odčítání vzdálenosti. Získaná absolutní hodnota je komparována v bloku menší nebo rovno ($\leq$) s nastavenou mezí (0,1). Jakmile hodnota klesne pod tuto mez – tedy robot se bezpečně přiblíží – je zapsána jednička do proměnné *objekt*, čímž se cyklus ukončí.

4.4.3 Vyhledávání sekvence tří barevných objektů

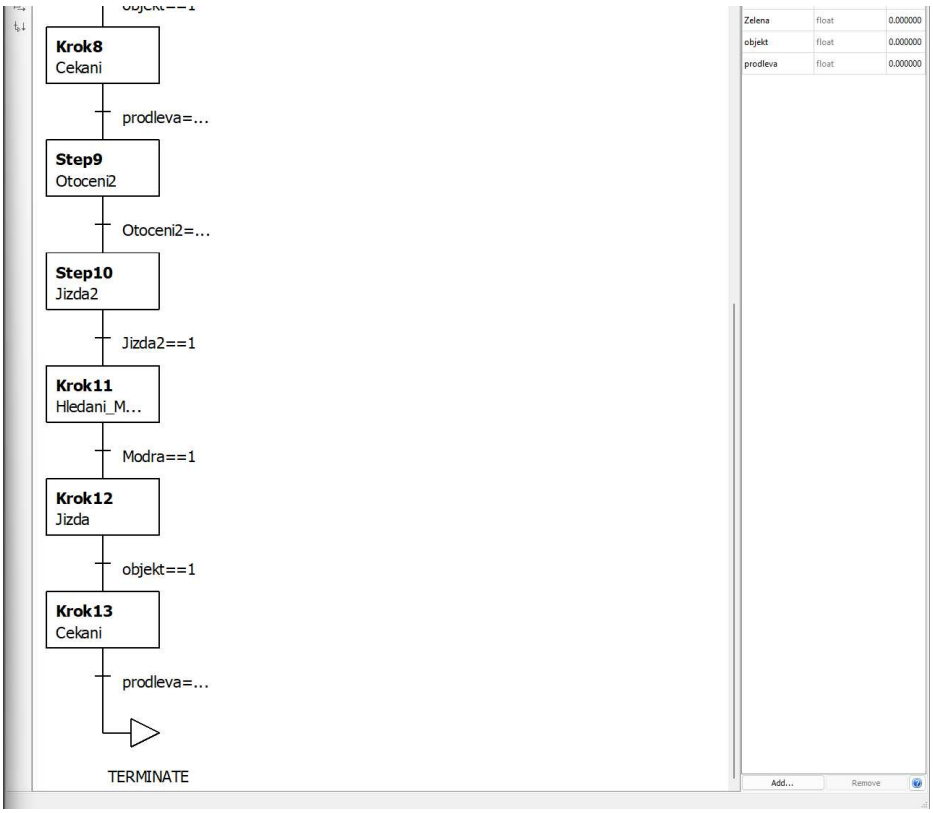
Třetí autonomní úloha měla experimentální charakter a byla primárně navržena pro testování robotu Robotino ve virtuálním prostředí RobotinoSIM. Cílem bylo naprogramovat robota tak, aby v prostoru postupně nalezl tři objekty odlišných barev (červená, zelená, modrá) v přesně stanoveném pořadí. Po dosažení každého objektu musel robot provést definovanou sekvenci úhybných manévrů, aby si uvolnil zorné pole pro vyhledávání dalšího cíle.

Hlavní řídicí struktura programu

Vzhledem ke komplexnosti úlohy byl hlavní program (SFC diagram) rozdělen do velkého množství na sebe navazujících kroků. Pro přehlednost je zobrazen na dvou následujících snímcích (obr. 4.13 a 4.14).



Obrázek 4.13: Hlavní program pro hledání sekvence barev (první část)

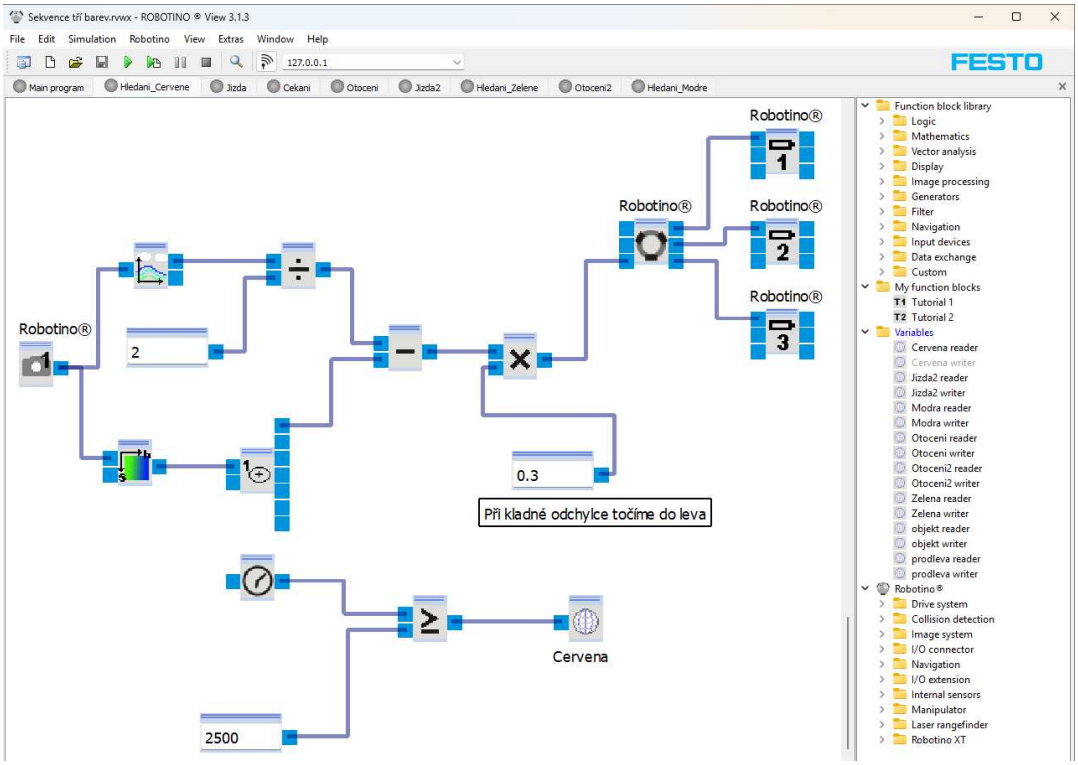


Obrázek 4.14: Hlavní program pro hledání sekvence barev (druhá část)

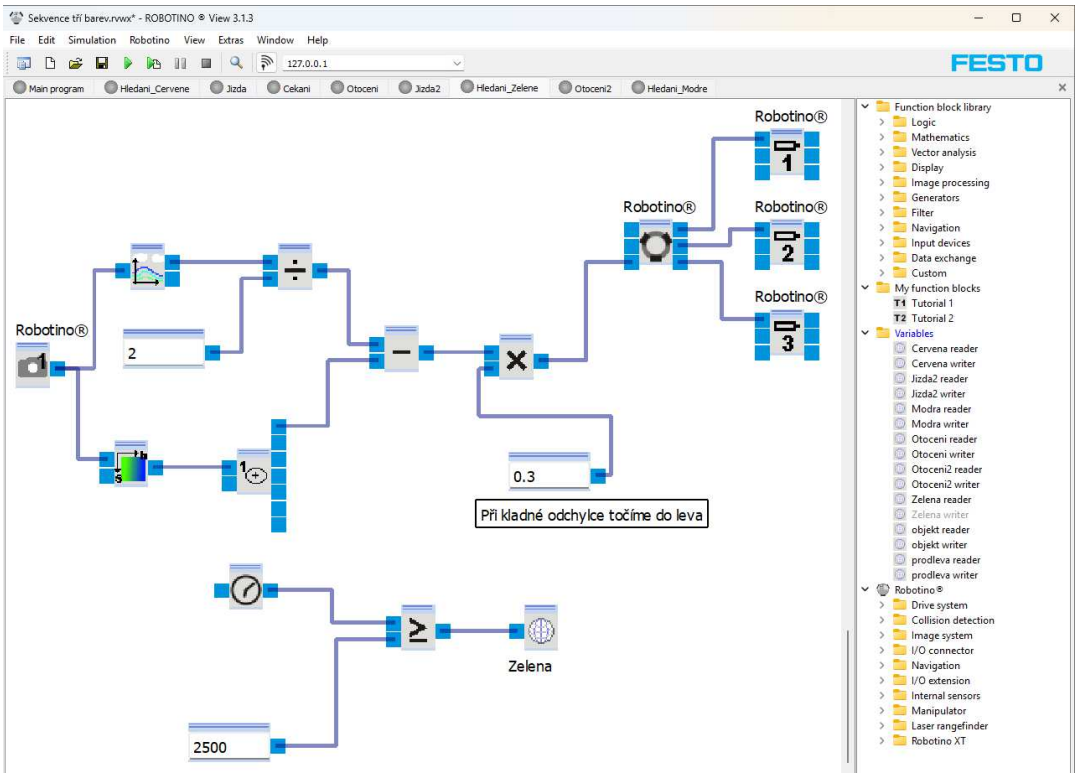
Algoritmus funguje v sekvenčních cyklech. Robot nejprve pomocí palubní kamery vyhledá první definovanou barvu (*Krok1*), autonomně se k ní přiblíží na požadovanou vzdálenost (*Krok2*) a následně po stanovenou dobu čeká (*Krok3*). Poté provede experimentální úhybný manévr, skládající se z přesného otočení a popojetí (*Krok4* a *Krok5*), čímž si záměrně uvolní zorné pole pro další fázi. Tento celý blok se následně opakuje pro druhou a třetí barvu. Přechody mezi jednotlivými stavy jsou striktně řízeny sadou uživatelských proměnných (např. *Cervena, objekt, prodleva, Otoceni*), které fungují jako logické přepínače a zajišťují plynulou návaznost celého procesu bez rizika zacyklení.

Detekce barev

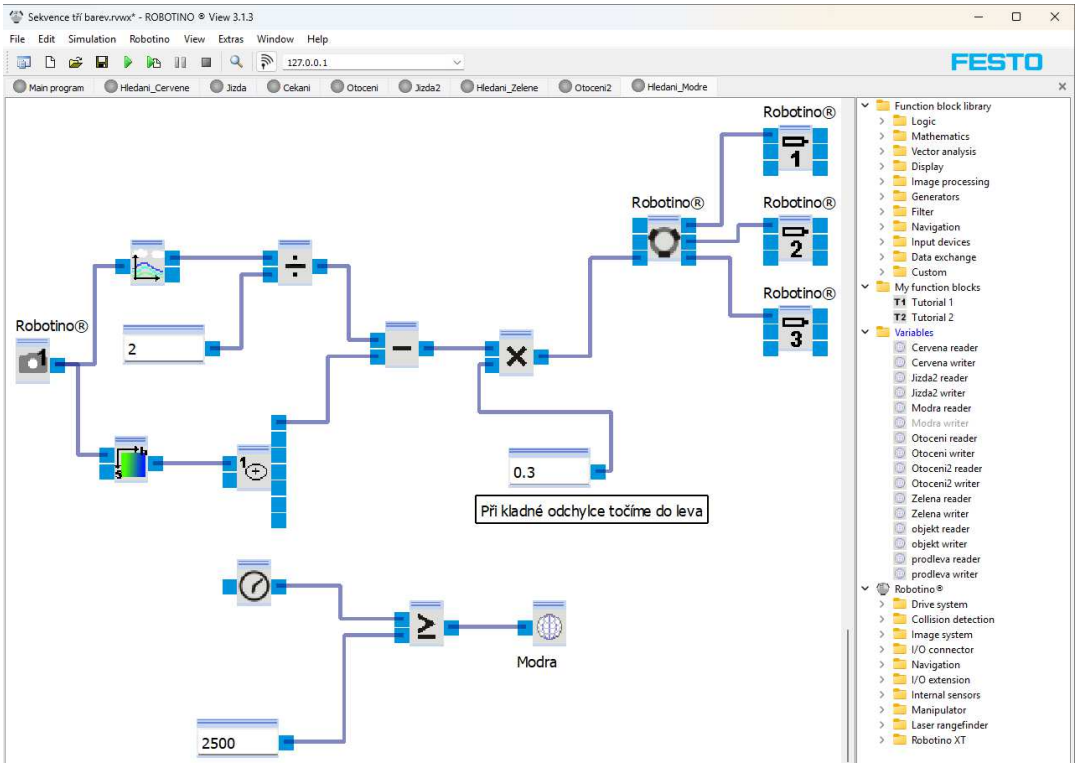
Logika pro detekci barev do značné míry vychází z předchozí úlohy, proto zde není znovu detailně rozebírána matematická podstata výpočtu směrové odchylky. Zásadním rozdílem a vylepšením je však rozdělení procesu hledání do tří samostatných podprogramů. V každém z nich je funkční blok *Color range finder* nezávisle zkalibrován na jinou cílovou barvu. Toto rozdělení zajišťuje, že systém v daný okamžik analyzuje a filtruje obrazová data pouze pro jeden konkrétní barevný odstín.



Obrázek 4.15: Podprogram pro vyhledání červené barvy



Obrázek 4.16: Podprogram pro vyhledání zelené barvy

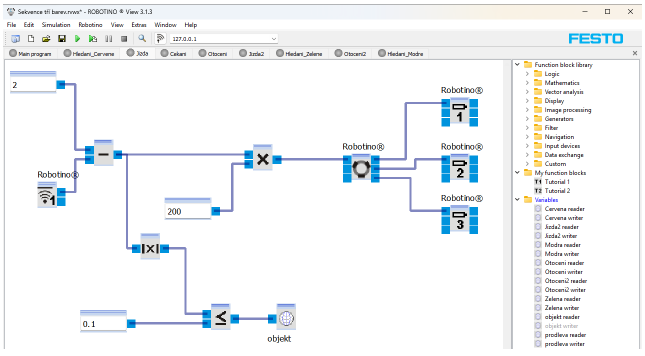


Obrázek 4.17: Podprogram pro vyhledání modré barvy

Jak je vidět z obrázků 4.15, 4.16 a 4.17, po uplynutí vyhledávacího času (2500 ms), během kterého se robot otáčí a skenuje prostor, je do příslušné stavové proměnné odeslána logická jednička, čímž je inicializován pohyb směrem k detekovanému objektu.

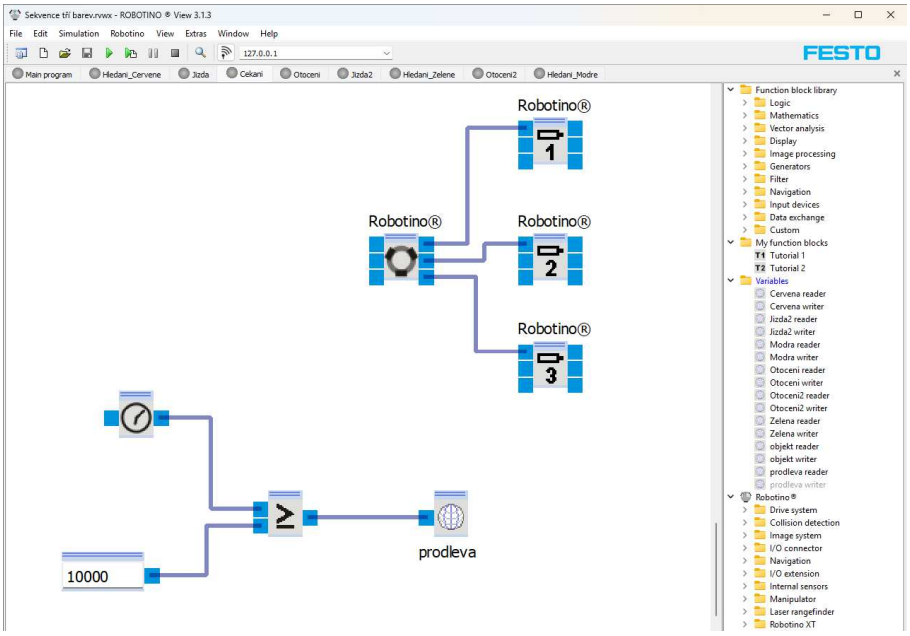
Přiblížení a fáze přeskupení (Repositioning)

Samotná jízda k nalezené barvě je řešena pomocí podprogramu *Jizda* (obr. 4.18). Dopředná rychlost je nastavena na 200 mm/s a pohyb je ukončen v momentě, kdy čelní senzor detekuje překážku v bezpečné vzdálenosti (absolutní hodnota $\leq 0,1$).



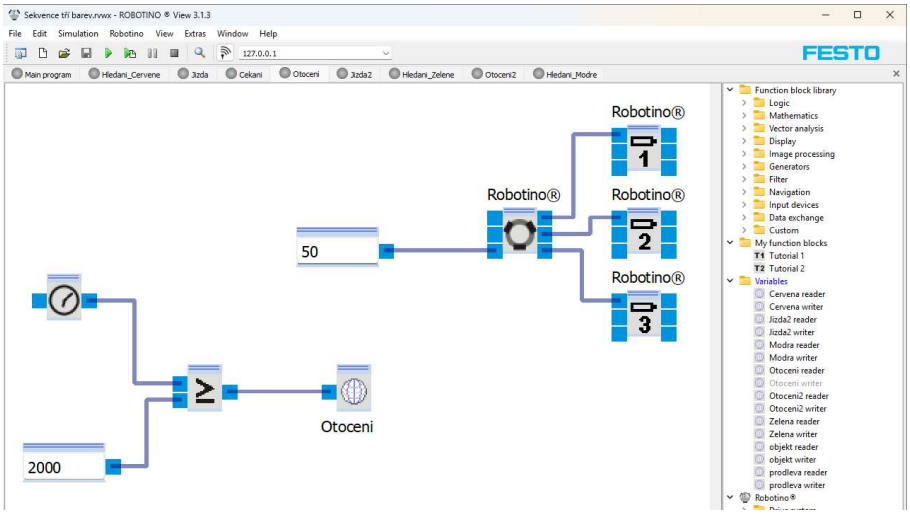
Obrázek 4.18: Podprogram pro jízdu k nalezenému objektu

Aby robot nenarazil do objektu, který právě našel, a mohl pokračovat v hledání dalšího cíle, byla navržena experimentální fáze přeskupení. Nejprve je spuštěn blok *Cekani*, který robotu nařídí setrvat na místě po dobu 10 sekund(obr. 4.19). Následně probíhá takzvaný slepý pohyb (bez zpětné vazby ze senzorů), jehož účelem je pouze změna výchozí pozice pro další hledání.

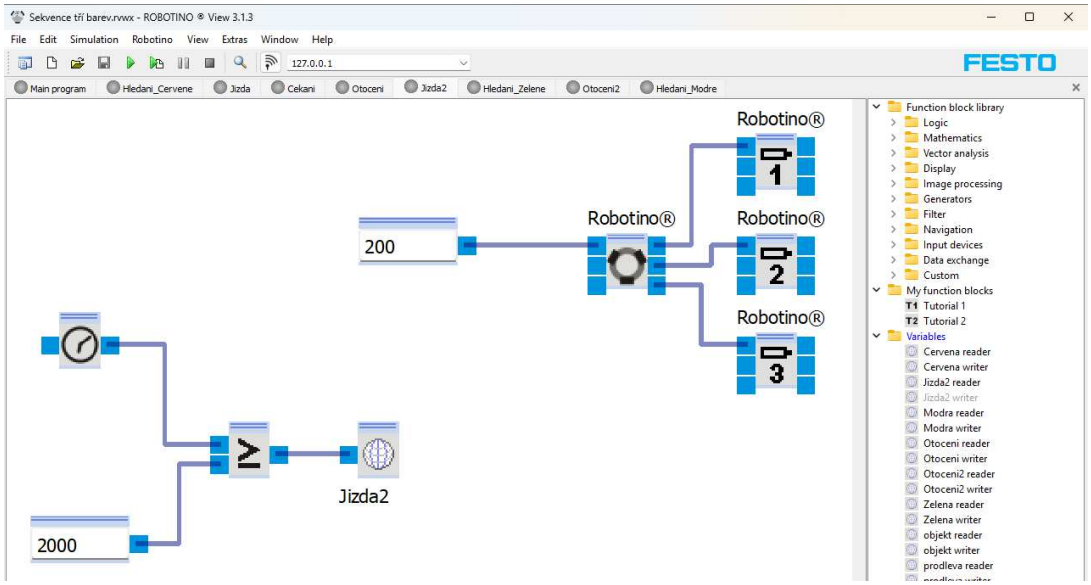


Obrázek 4.19: Podprogram pro prodlevu (čekání) mezi akcemi

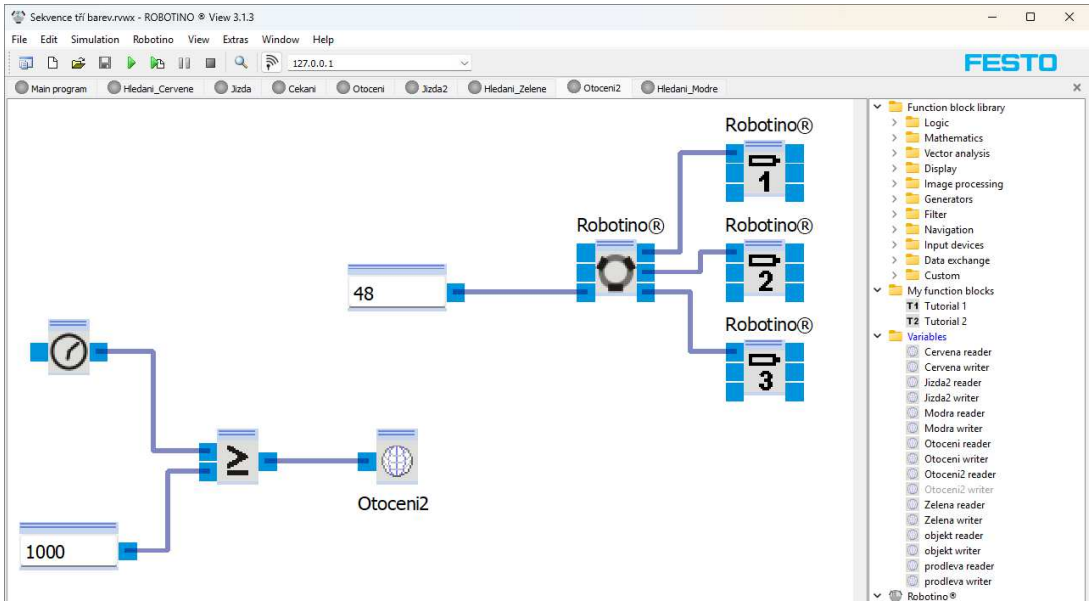
Tento manévr se skládá z časově omezené rotace – *Otoceni* a *Otoceni2* (obr. 4.20 a 4.22) a následné slepé dopředné jízdy po stanovenou dobu – *Jizda2* (obr. 4.21).



Obrázek 4.20: Podprogram pro první fázi rotace (změna směru)



Obrázek 4.21: Podprogram pro slepý přesun na novou pozici



Obrázek 4.22: Podprogram pro druhou fázi rotace

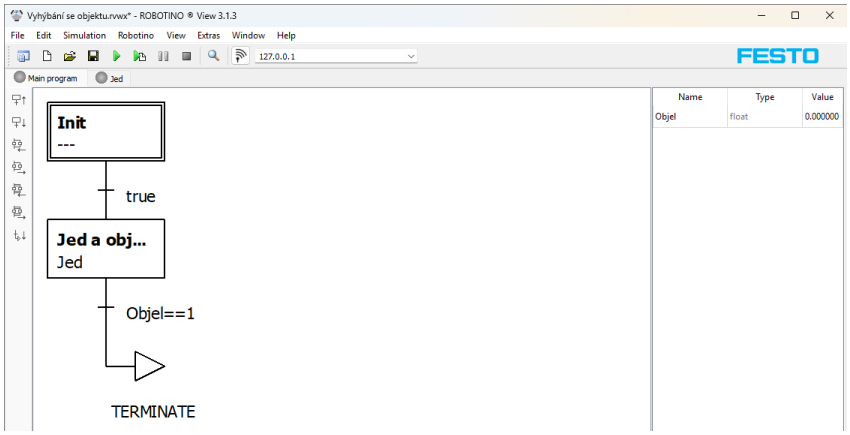
Po dokončení tohoto manévru se celý stavový automat posune do další fáze a začne vyhledávat následující barvu v sekvenci. Tento komplexní experimentální program úspěšně ověřil možnosti větvení a zacyklení programu v simulovaném prostředí RobotinoSIM.

Průběh této sekvenční úlohy, včetně plynulých přechodů mezi jednotlivými barvami, je detailně zachycen v následující části videodokumentace (ukázka začíná v čase 3:15):

`https://www.youtube.com/watch?v=zD8QVryrceg&t=195s`

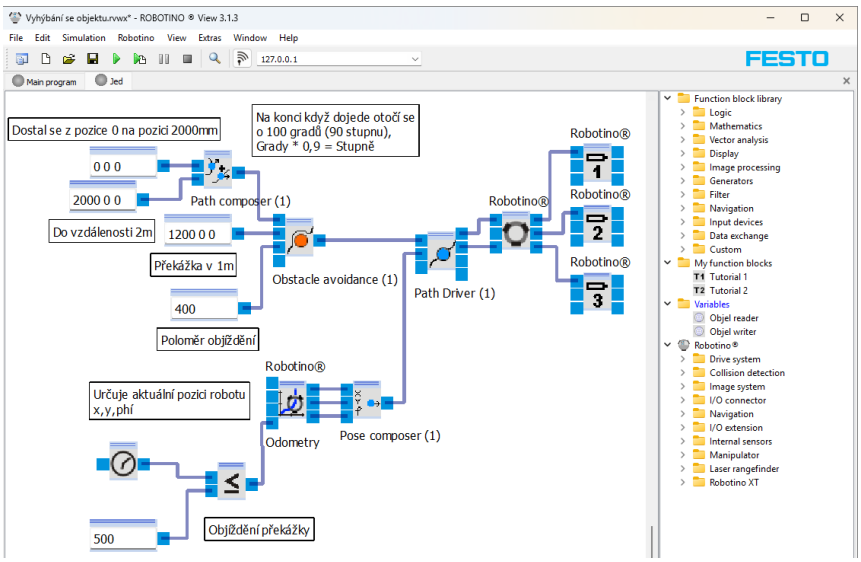
4.4.4 Experimentální testování manévru pro vyhýbání se překážce

Tato část kapitoly popisuje ověření funkčnosti navigačního programu pro vyhnutí se překážce. Na rozdíl od předchozích komplexních úloh zde nebyla implementována žádná složitá logika ani doplňkové stavy. Cílem bylo v simulačním prostředí RobotinoSIM otestovat, jakým způsobem blok *Obstacle avoidance* deformuje přímou trasu robotu.



Obrázek 4.23: Struktura experimentálního programu pro ověření manévru

Hlavní program (obr. 4.23) byl sestaven v nejjednodušší možné podobě. Po inicializaci systému přechází program do jediného pracovního kroku, který zajišťuje pohyb. Jak bylo zmíněno dříve, program záměrně neobsahuje zápis do ukončovací proměnné *Objel*, což umožňovalo sledovat manévr v simulaci opakovaně.



Obrázek 4.24: Blokové schéma testovaného manévru

Vlastní provedení manévru, které je dokumentováno ve videu, probíhalo následovně (viz schéma na obr. 4.24):

- **Přímý směr:** Robotu byla pomocí bloku *Path composer* zadána přímá trasa o délce 2000 mm (2 metry).
- **Vložení manévru:** Do této trasy byl „vřazen“ blok *Obstacle avoidance*. Experiment byl nastaven tak, že ve vzdálenosti 1,2 metru (1200 mm) se nachází virtuální překážka.
- **Parametry vybočení:** Poloměr, s jakým má robot překážku objet, byl stanoven na 400 mm.

Během testu robot vyjel z výchozí pozice 0 mm přímo vpřed. Jakmile se přiblížil k definované hranici překážky, navigační systém automaticky přepočítal akční veličiny pro motory tak, aby robot plynulým obloukem objekt objel a následně se vrátil do původní osy jízdy.

4.4.5 Kroužení kolem detekovaného objektu (Orbiting)

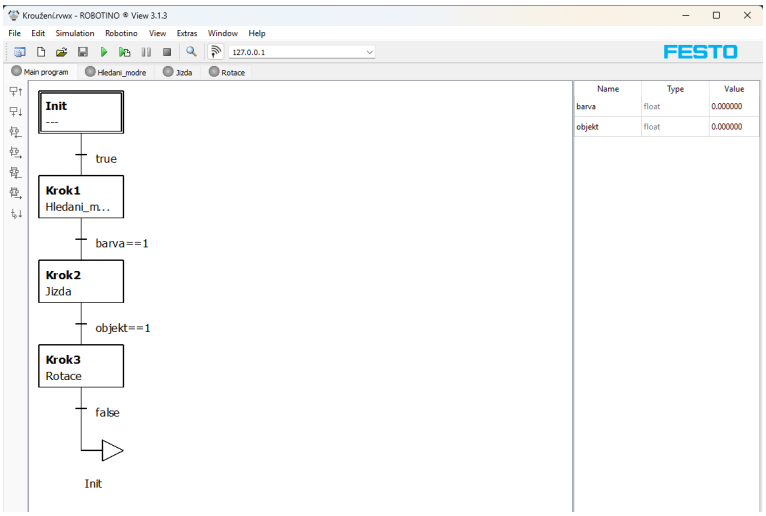
Závěrečná a technicky nejnáročnější autonomní úloha spojuje všechny dosud otestované principy do jednoho komplexního celku. Cílem tohoto programu bylo vyhledat specifický objekt (v tomto případě modré barvy), bezpečně se k němu přiblížit a následně přejít do režimu takzvaného orbitálního pohybu – tedy plynulého kroužení či kopírování hrany překážky (například stolu) při zachování konstantní vzdálenosti a správného natočení podvozku. Tato úloha naplno demonstruje výhody všesměrové kinematiky (omni-drive).

Řídicí struktura programu

Struktura celého programu je zachycena na obrázku 4.25. Program sekvenčně prochází fázemi *Hledani_modre* a *Jizda*, které jsou podmíněny nalezením barvy a dosažením bezpečné vzdálenosti (*barva==1*, respektive *objekt==1*).

Zajímavostí tohoto stavového automatu je výstupní tranzice pod posledním pracovním krokem *Rotace*. Její hodnota je pevně definována jako *false* (logická nepravda). Z hlediska řídicí logiky to znamená, že podmínka pro opuštění tohoto stavu není nikdy splněna a program zůstane v daném kroku trvale uzamčen. Robot by tak kolem objektu kroužil nepřetržitě. Vzhledem k absenci korektní ukončovací podmínky a z důvodu zajištění

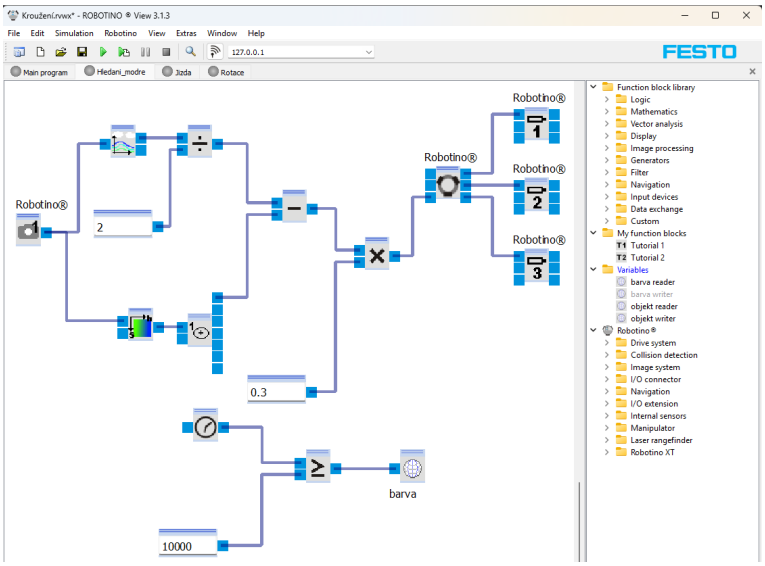
bezpečnosti hardwaru byl tento specifický program otestován výhradně v simulačním prostředí RobotinoSIM.



Obrázek 4.25: Hlavní program pro vyhledání objektu a přechod do orbitálního pohybu

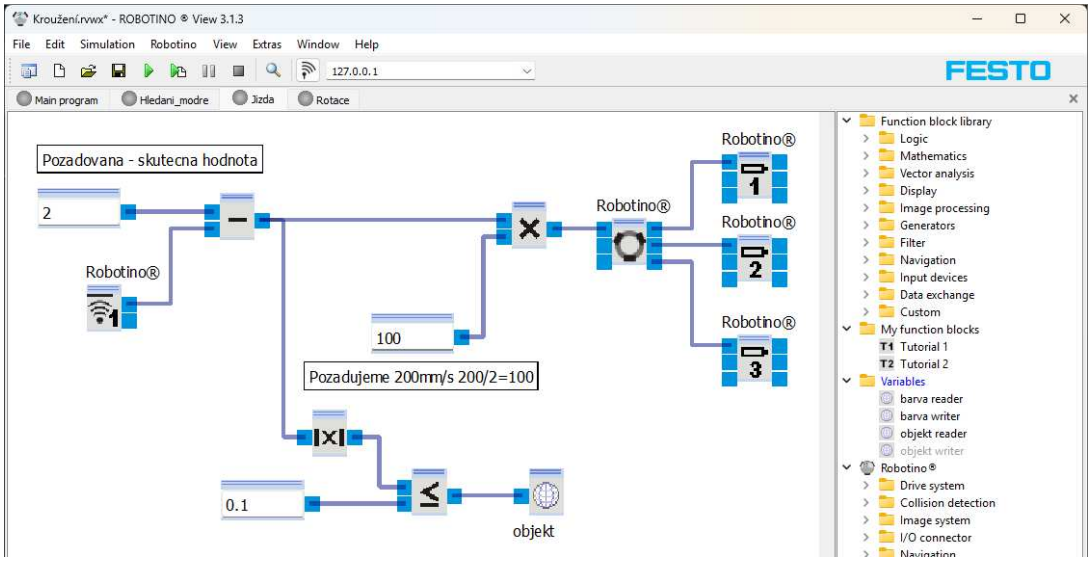
Přípravné fáze: Vyhledání a přiblížení

První dvě fáze využívají již dříve popsané modely řízení. Blok pro hledání barvy využívá kameru k lokalizaci modrého objektu a na základě odchylky od středu obrazu generuje rotační pohyb k vycentrování robotu(obr. 4.26). Součástí je i časovač (10 sekund), který poskytuje dostatečný prostor pro proskenování okolí.



Obrázek 4.26: Podprogram pro směrové vycentrování na modrý objekt

Jakmile je objekt vycentrován, přebírá řízení podprogram *Jizda* (obr. 4.27). Ten pomocí čelního infračerveného senzoru reguluje dopřednou rychlost. Žádaná hodnota ze senzoru je odečtena od konstanty definující odstup a výsledek přímo úměrně určuje rychlost přibližování. Když absolutní hodnota regulační odchylky klesne pod 0,1, robot zastaví a spustí finální fázi.



Obrázek 4.27: Podprogram pro přiblížení se do bezpečné vzdálenosti

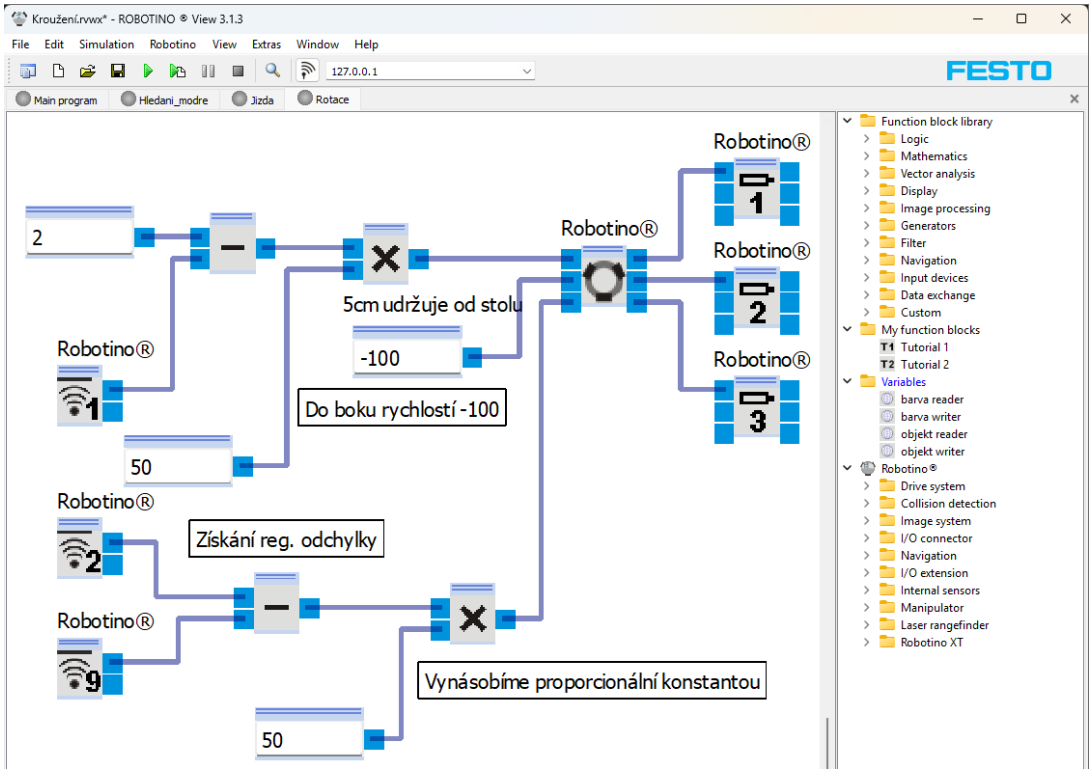
Algoritmus orbitálního pohybu (Rotace)

Klíčovou částí celé úlohy je samotný pohyb kolem objektu, jehož vnitřní logika je detailně zobrazena na obrázku 4.28. Aby robot dokázal plynule kroužit a neustále udržoval čelní stranu orientovanou směrem k překážce, musí systém nezávisle a současně řídit všechny tři složky rychlosti (v_x, v_y, ω):

1. **Udržování odstupu (osa x):** Horní větev obvodu sleduje vzdálenost od stolu pomocí senzoru 1. Žádaná hodnota je porovnávána s aktuální naměřenou vzdáleností. Výsledná chyba se násobí proporcionální konstantou 50 a je přivedena na první vstup bloku *Omnidrive*. Tím je zajištěno, že robot udržuje konstantní odstup od hrany stolu (5 cm) – pokud se přiblíží, mírně zacouvá, pokud se vzdálí, přitáhne se.

2. **Boční pohyb (osa y):** Aby robot objekt objížděl, je do druhého vstupu (*Omnidrive* - osa y) přivedena konstantní hodnota -100 mm/s. Tím je generován takzvaný krabí chod (lineární pohyb do boku) bez toho, aby se robot natáčel.

3. **Úhlové zarovnání (ω):** Spodní část obvodu představuje efektivní způsob úhlové stabilizace. Program neustále odečítá hodnoty ze dvou různých senzorů vzdálenosti (Senzor 2 a Senzor 9), které jsou umístěny asymetricky na čelní straně robotu. Rozdíl těchto dvou hodnot tvoří regulační odchylku, která je vynásobena konstantou 50 a vstupuje do řízení rotace. Pokud je jedna strana robotu blíže ke stolu než druhá, vznikne nenulová odchylka a robot se automaticky pootočí tak, aby se obě vzdálenosti srovnaly. Tím je garantováno dokonalé zarovnání čelní plochy robotu s objížděnou hranou.



Obrázek 4.28: Regulační smyčky zajišťující orbitální pohyb a zarovnání k hraně

4.4.6 Využití Robotino Factory pro telemetrii

Ačkoliv byla autonomní navigace pro výše zmíněné úlohy řízena zcela nezávisle lokálními programy v RobotinoView, prostředí Robotino Factory nepřestalo plnit svou funkci. Díky aktivnímu síťovému spojení v režimech *Master/Slave* robot neustále odesílal svá telemetrická data, především pak odometrii a aktuální skeny z LiDARu Hokuyo. Fyzický pohyb robotu prostorem učebny se tak v reálném čase propisoval do dříve vytvořené

mapy. Software Robotino Factory tak byl efektivně využit jako monitorovací rozhraní, jež umožňovalo vizuálně kontrolovat skutečnou trajektorii a přesnost robotu ze vzdáleného pracoviště.

Všechny praktické úlohy a autonomní manévry, které jsou detailně popsány v následujících kapitolách 4.4.1 až 4.4.5, jsou v jejich reálném provozu zachyceny v přiloženém videu na adrese: `https://www.youtube.com/watch?v=zD8QVryrceg`.

4.5 Výuková videodokumentace

Závěrečným bodem praktické části bylo vytvoření edukačního videa, které má po konzultaci s vedoucím práce sloužit jako výuková pomůcka pro další ročníky. Scénář byl sestaven z poznatků nabytých během postupného oživování systému. Samotný záznam byl pořízen ve vysokém rozlišení pomocí mobilního telefonu Samsung Galaxy S25 Ultra.

Obrázek 4.29: Samsung S25 Ultra ((Samsung Electronics, 2025))

Struktura videa chronologicky provádí diváka celým procesem práce s platformou, od počátečního zapnutí až po pokročilé autonomní funkce. Pro zajištění maximální přehlednosti a didaktické hodnoty byl vytvořen detailní scénář, který video rozděluje do šesti hlavních scén a jedné bonusové výukové sekce:

Scéna 1: Úvod a představení

Video začíná širokým záběrem na stojícího robota v prostředí školní učebny. Následují dynamické detaily na klíčový hardware – všesměrová kola (omni-wheels), infračervené senzory po obvodu, palubní kameru a řídicí jednotku. V komentáři je představen cíl práce: kompletní softwarové oživení platformy Robotino 3, zmapování učebny a zprovoznění autonomní navigace.

Scéna 2: Software a zprovoznění

Divák je vizuálně seznámen s procesem aktualizace operačního systému na bázi Li-

nuxu, včetně praktické ukázky zapojení instalačního USB disku (vytvořeného pomocí nástroje Rufus dle pokynů společnosti Festo). Následně jsou formou záznamu obrazovky představena dvě hlavní pracovní prostředí: *Robotino View* pro tvorbu logiky a *Robotino SIM* pro virtuální testování.

Scéna 3: Skenování prostoru a vymezení hranic

Tato pasáž ukazuje proces tvorby mapy. Reálné záběry zachycují manuální posouvání robota po učebně za účelem detailního skenování prostoru pomocí senzorů. Výsledná surová mapa z programu *Robotino Factory* je ponechána v autentické podobě. Následuje ukázka práce v softwaru *Adobe Photoshop*, kde jsou do mapy ručně dokresleny vnější černé hranice. Tento krok zajišťuje, že se robot při autonomní navigaci nepokusí opustit vyhrazenou oblast.

Scéna 4: Praktické úlohy a interakce

Tato nejobsáhlejší část vizualizuje pět vytvořených autonomních úloh, které byly nahrány do řídicí jednotky prostřednictvím Wi-Fi hotspotu:

- **Sledování čáry (Task 1):** Ukázka analýzy obrazu v reálném čase. Robot kopíruje černou pásku na podlaze, přičemž úhel kamery musel být mechanicky nastaven přibližně na 45°.
- **Detekce barvy (Task 2):** Kombinace počítačového vidění a řízení. Robot vyhledá předmět (např. červený kužel), vypočítá vzdálenost a bezpečně u něj zastaví.
- **Sekvence tří barev (Task 3):** Robot postupně vyhledává tři rozdílné barvy, u každé zastaví, provede úhybný manévr a pokračuje v hledání další v pořadí.
- **Vyhýbání se objektu (Task 4):** Demonstrace automatického objíždění překážky v časovém limitu.
- **Kroužení (Task 5):** Využití všesměrového podvozku k plynulému orbitálnímu pohybu kolem objektu při zachování konstantního odstupu.

Scéna 6: Závěr

Celkové zhodnocení projektu, potvrzení plné funkčnosti zaktualizovaného a zmapovaného systému a rozloučení s divákem.

Bonusová sekce: Programování krok za krokem

Do videa byl vložen bonusový videotutoriál. Formou záznamu obrazovky s českým

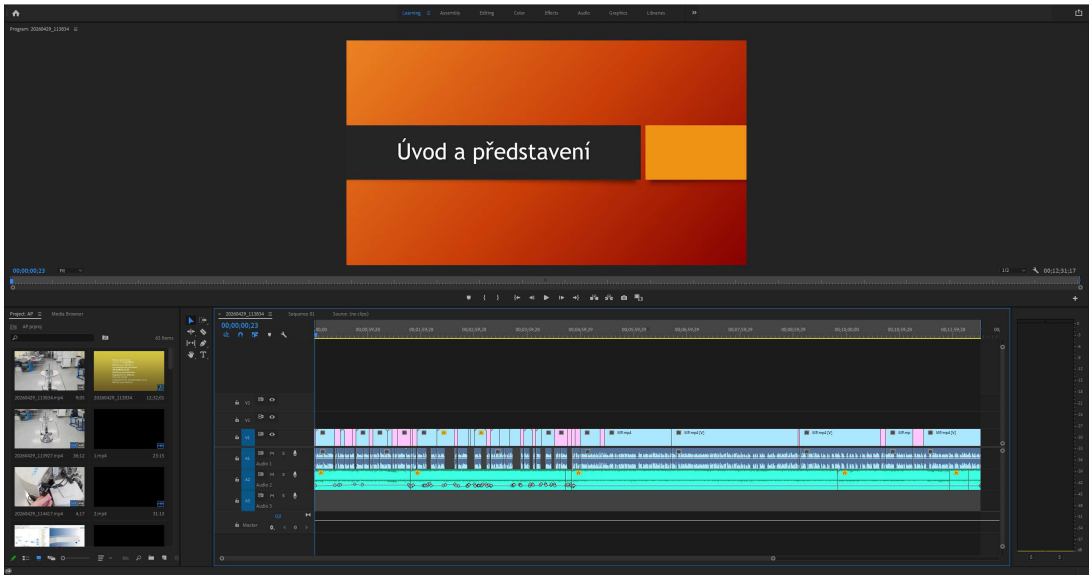
komentářem podrobně vysvětluje tvorbu programu pro detekci barvy v prostředí *Robotino View*. Divák je proveden zakládáním proměnných, propojováním bloků motorů s regulátory a nastavením podmínek zastavení podle infračervených senzorů vzdálenosti.

4.5.1 Postprodukce v prostředí Adobe Premiere Pro

Střih a postprodukce videodokumentace probíhaly v profesionálním programu Adobe Premiere Pro. Prvním krokem byla selekce a synchronizace surových záběrů z oživování robotu a záznamů obrazovky. Následně byl samostatně nahrán čistý hlasový doprovod (dabing).

V hlavní fázi střihu byl kladen důraz na plynulé propojení teoretických výkladů s praktickými ukázkami pohybu robotu. Zde byla využita technika vícestopé editace, která umožnila přesné časování komentáře k dění na obrazovce. Pro zvýšení atraktivity a udržení pozornosti diváka byla do pozadí vložena hudba. Klíčovým úkolem bylo precizní vyvážení audio stop (tzv. audio mixing), aby doprovodná hudba nepřekrývala mluvené slovo a zároveň nepůsobila rušivě.

Finální úpravy zahrnovaly aplikaci barevných korekcí pro sjednocení záběrů z různých světelných podmínek učebny a vložení grafických prvků, jako jsou například popisky funkčních bloků. Výsledkem je ucelené výukové video, které splňuje nároky na moderní edukační materiál.



Obrázek 4.30: Finální edit v Adobe Premiere Pro

Kapitola 5

Závěr

Cílem této absolventské práce bylo kompletní zprovoznění výukového robotu Robotino třetí generace a návrh jeho autonomní navigace v reálném prostředí školní učebny. Všechny stanovené cíle byly úspěšně splněny.

V rámci praktické realizace byl nejprve aktualizován operační systém řídicí jednotky na verzi Ubuntu 20.04 a byl otestován program pro ruční řízení robotu. Následně proběhlo zmapování učebny pomocí palubního laserového skeneru Hokuyo. Vzhledem k přirozeným limitům senzorů při detekci skla a nízkých překážek byla mapa postprocesingově upravena v grafickém editoru přidáním virtuálních zdí.

Kvůli zjištěným softwarovým omezením programu Robotino Factory při globální navigaci bylo toto prostředí využito primárně pro telemetrii. Těžiště práce se přesunulo k tvorbě lokálních programů v prostředí RobotinoView, kde bylo úspěšně sestaveno pět úloh.

Největším přínosem práce je její didaktický přesah. Veškeré nabyté zkušenosti z oživování a programování systému byly transformovány do strukturované výukové videodokumentace. Vytvořené video slouží nejen jako přehled dosažených výsledků, ale i jako praktický návod pro budoucí studenty pracující s touto platformou.

Pro další rozvoj projektu se v budoucnu nabízí například hardwarové rozšíření robotu o manipulátor (chapač), díky kterému by platforma mohla objekty i aktivně uchopovat a přemisťovat.

Pro ucelený přehled o dosažených výsledcích a funkčnosti celého systému doporučuji zhlédnout výslednou videodokumentaci projektu:

`https://www.youtube.com/watch?v=zD8QVryrceg`

Literatura

Akeo (2024), *Rufus - Snadné vytváření bootovacích USB disků* [online]. [cit. 2026-05-06], ⟨https://rufus.ie/cs/⟩.

Canonical Ltd. (2020), *Ubuntu 20.04 LTS (Focal Fossa)* [online]. [cit. 2026-05-06], ⟨https://releases.ubuntu.com/focal/⟩.

Correll, N., Hayes, B., Heckman, C. a Roncone, A. (2022), *Introduction to Autonomous Robots*, Cambridge, Massachusetts: The MIT Press. ISBN 978-0-262-04782-1.

Festo Didactic (2024*a*), *Restoring the operating system* [online]. [cit. 2026-05-06], ⟨https://wiki.openrobotino.org/Robotino3_usb_restore.html⟩.

Festo Didactic (2024*b*), *Robotino OS Images* [online]. [cit. 2026-05-06], ⟨https://wiki.openrobotino.org/Robotino4_images.html⟩.

Festo Didactic (2024*c*), *Robotino® – Wiki* [online]. [cit. 2026-05-06], ⟨https://wiki.openrobotino.org/⟩.

Samsung Electronics (2025), *Galaxy S25 Ultra* [online]. [cit. 2026-05-13], ⟨https://www.samsung.com/cz/smartphones/galaxy-s25-ultra/buy/⟩.

Schenk, C. (2024), *MiKTeX* [online]. [cit. 2026-05-06], ⟨https://miktex.org/⟩.

Thrun, S., Burgard, W. a Fox, D. (2005), *Probabilistic Robotics*, Cambridge, Massachusetts: The MIT Press. ISBN 978-0-262-20162-9.

van der Zander, B. a Sundermeyer, J. (2024), *TeXstudio – A LaTeX editor* [online]. [cit. 2026-05-06], ⟨https://www.texstudio.org/⟩.

Wikipedia contributors (2026), '*Intel Atom* [online]'. [cit. 2026-05-13], ⟨https://en.wikipedia.org/wiki/Intel_Atom⟩.

Příloha A

Obsah přiloženého DVD

K této práci je přiloženo DVD s následující adresářovou strukturou.

- `Absolventská práce v systému LaTeX`
- `Videodokumentace`
- `RobotinoView - úlohy`
- `Cuchna_AP_2025_2026.pdf` – absolventská práce ve formátu PDF

Příloha B

Použitý software

Adobe Podcast (Enhance Speech) ⟨https://podcast.adobe.com/en⟩

Adobe Photoshop ⟨https://www.adobe.com/cz/products/photoshop.html⟩

Adobe Premiere Pro ⟨https://www.adobe.com/cz/products/premiere.html⟩

MiKTeX ⟨https://miktex.org/⟩

Robotino Factory ⟨https://wiki.openrobotino.org/⟩

Robotino SIM ⟨https://wiki.openrobotino.org/⟩

Robotino View ⟨https://wiki.openrobotino.org/⟩

TeXstudio ⟨https://www.texstudio.org/⟩

Software z výše uvedeného seznamu je buď volně dostupný, nebo jeho licenci toho času vlastní Vyšší odborná škola, Střední škola, Centrum odborné přípravy, Sezimovo Ústí, Budějovická 421, kde autor toho času studoval a vytvořil tuto práci. Výjimku tvoří grafický a střihový software společnosti Adobe (Adobe Photoshop a Adobe Premiere Pro), který byl pro účely tohoto projektu legálně využit v rámci oficiální sedmidenní bezplatné zkušební verze (tzv. Free Trial).

Příloha C

Časový plán absolventské práce

Činnost	Časová náročnost	Termín ukončení	Splněno
Popište hardware a software mobilního robotu Robotino.	2 týdny	08.11.2025	
Popište programy RobotinoView, RobotinoSIM a Robotino Factory.	2 týdny	22.11.2025	
Vytvořte program pro ruční řízení robotu Robotino.	2 týdny	06.12.2025	
Autonomně ovládejte robotu Robotino v daném prostoru.	2 týdny	20.12.2025	
Aktualizujte operační systém robotu Robotino.	2 týdny	03.01.2026	
Vytvořte mapu školní učebny v RobotinoFactory.	2 týdny	17.01.2026	
Vytvořte programy pro řízení robotu Robotino v učebně na základě její mapy.	3 týdny	07.02.2026	
Vytvořte scénář pro natočení videa o robotu Robotino.	2 týdny	21.02.2026	
Video natočte a sestříhejte.	1 měsíc	21.03.2026	
AP: kapitola Úvod	2 týdny	04.04.2026	
AP: kompletní text	5 týdnů	11.05.2026	