Programowanie 101

Ten zasób będzie obejmował podstawy programowania w zawodach FIRST® Robotics Competition. Obejmuje C++, Java/Kotlin i LabVIEW.

Poziom pierwszy: Uruchomienie robota

1. Wybór języka programowania: Java, C++ lub LabVIEW

- **Jawa** to język tekstowy, który jest powszechnie nauczany w szkołach średnich i używany do egzaminów AP CS. Jest to "bezpieczny" język, który działa we własnym środowisku wirtualnym. Chociaż na ogół nie ma to wpływu na PIERWSZY Zawody w robotyce, to wirtualne środowisko, znane również jako JVM, oznacza, że programy Java są zauważalnie wolniejsze niż języki kompilowane, gdy są używane do zadań wymagających dużej mocy obliczeniowej. Java jest często wybierana ze względu na łatwość użycia i wzajemną kompatybilność. Zespoły korzystające z języka Java to m.in. 254, 125, 503, 4911, i 1241.
- **C++** to szybki tekstowy język programowania. Jest używany w przemyśle w systemach czasu rzeczywistego ze względu na swoją moc i wydajność, ale krzywa uczenia się jest znacznie bardziej stroma niż w Javie. C++ wyewoluował z języka programowania C, a połączenie historycznych i nowoczesnych funkcji czasami prowadzi do mylącej składni i/lub nieoczekiwanych zachowań. Zespoły FIRST Robotics Competition korzystają z niego przede wszystkim ze względu na jego szybkość, elastyczność i obszerne biblioteki matematyczne. Zespoły korzystające z języka C++ to 971 i 1678.
- **LabVIEW** to graficzny język programowania przepływu danych opracowany przez National

  Przyrządy (NI) do użytku przez inżynierów i techników. LEGO WeDo i Języki Mindstorms używane w FIRST LEGO League Jr i FIRST LEGO League są pochodnymi LabVIEW, więc uczniowie wywodzący się z tych programów mogą uznać go za znajomy. Na diagramie LabVIEW bardzo łatwo jest skorzystać z zaawansowanych funkcji obliczeniowych, takich jak równoległe uruchamianie fragmentów kodu. Chociaż takie funkcje są potężne, często wprowadzają nowe problemy, z którymi trzeba sobie poradzić. NI zapewnia jednak rozbudowane narzędzia do debugowania. Środowisko i język LabVIEW mają swoją własną

krzywą uczenia się i unikalne wyzwania. Zespoły FIRST Robotics Competition korzystają z niego przede wszystkim ze względu na uproszczoną składnię graficzną i obszerne biblioteki inżynierskie. Zespoły korzystające z LabVIEW to 33, 359, 624, 1986 i 2468.

- **Wybór języka zawsze zależy od tego, co jest najłatwiejsze dla *Twojego* zespołu.** Na przykład, często wybór języka może mieć sens, ponieważ mentor programowania jest ekspertem w tym języku. Z drugiej strony, sensowny może być wybór na podstawie łatwości nauki danego języka. Bez względu na to, jaką decyzję podejmiesz, pamiętaj, że wybór języka programowania jest specyficzny dla środowiska pracy i osób w Twoim zespole. Wszystkie Języki są sprawne, dobrze obsługiwane i wystarczająco wydajne, aby można je było używać w FIRST Robotics Competition.

2. Nauczanie języka programowania

- Podczas nauczania programowania dla PIERWSZY Zawody w robotyce, istnieją dwa odrębne przedmioty, których należy nauczać. Pierwszym z nich jest semantyka i składnia samego języka programowania, a drugim jest powiązanie z PIERWSZY Elementy Zawodów Robotycznych. Przewodnik po nauce języka C++ można znaleźć tuJawa jej___e, oraz LabVIEW tu.

3. Wybór kodu startowego

- W przypadku Javy i C++ istnieją cztery różne "klasy", których można używać podczas interakcji z robotem. Porównanie tych klas i ich znaczenie można znaleźć tutaj.
- W przypadku LabVIEW szablony początkowe zawierają mieszankę mechanizmów czasowych, iteracyjnych i spawn/abort. Szablony są opisane tutaj.

4. ***Gdy Twój zespół wybierze język programowania i zacznie kodować, witryna FIRST Robotics Competition Docs jest dobrym źródłem informacji o tym, jak skonfigurować środowisko programistyczne i wprowadzić kod do robota. Przewodniki te są nieocenione w programowaniu FIRST Robotics Competition.***
   - Wprowadzenie
   - C++\Java
   - Widok LabVIEW

5. ***Wprowadzanie kodu do robota!***
   - Java i C++

- Jeśli używasz gradleRIO, po prostu wpisz "./gradlew deploy" w VS Code Konsola, a twój kod będzie na robocie.
- Ale zaraz! Twój kod jeszcze nic nie robi. Kilka prostych przykładów kodu dysku można znaleźć tutaj. Kod we fragmentach kodu należy do Robot.java lub Robot.cpp, które powinny zostać automatycznie utworzone za pomocą projektu gradle/Eclipse.
- Widok LabVIEW
- W przypadku programowania należy uruchomić ze źródła, jak pokazano tutaj.
- Po zakończeniu wdrożysz skompilowany plik wykonywalny, jak pokazano tutaj.

6. Kod mechanizmów

- W przypadku języków Java i C++ prosty kod dysku można skopiować i wkleić stąd.
- W przypadku LabVIEW prosty kod dysku znajduje się w szablonie w TeleOp VI.
- Większość robotów FIRST Robotics Competition ma inne mechanizmy uruchamiające niż układ napędowy. Może to być wszystko, od obracającego się koła zamachowego po pneumatyczną katapultę. Wszystkie te mechanizmy powinny być sterowane w trybie autonomicznym lub teleoperacyjnym. Aby sterować mechanizmami za pomocą kontrolerów prędkości przez PWM, przewodnik dla C++ i Javy jest dostępny tutaj, a dla LabVIEW tutaj.
- Jeśli używasz kontrolerów prędkości przez CAN, musisz albo postępować zgodnie z instrukcją tutaj , aby traktować je jako kontrolery prędkości PWM, albo użyć API Phoenix, którego dokumentacja jest podlinkowana tutaj.

7. Autonomiczny

- Przewodnik po tym, jak wykonywać autonomiczne działania w programowaniu w Javie i C++, można znaleźć tutaj.
- Zespół 1619 skompilował również prosty kod do przekroczenia linii automatycznej w Javie, który można znaleźć tutaj.
- Szablony LabVIEW zawierają autonomiczny kod do poruszania robotem w miejscu. Możesz modyfikować wartości mocy silnika i taktowanie, aby wykonać wiele zadań. Oto przykład autonomicznego kodu 2468 z 2018 roku.

Poziom drugi: Niestandardowa architektura i sterowanie silnikiem w pętli zamkniętej

1. Korzystanie z niestandardowej architektury

- W wielu przypadkach dostępne klasy robotów nie wystarczają. Na przykład możesz chcieć okresowo uruchamiać teleop i autonomicznie sekwencyjnie. W takim przypadku prawdopodobnie nadszedł czas, aby przejść na architekturę niestandardową.
- Architektura niestandardowa polega zasadniczo na strukturyzowaniu całego kodu w niestandardowy sposób.
- Niektóre przykłady architektur niestandardowych obejmują kod 1678, który znajduje się tutaj.

  Kod 1678 opiera się na kodzie 971, który jest tutaj.

- 254 ma również niestandardową architekturę. Ich kod z 2019 roku można

  znaleźć tu. - 33, 624, i 1986oraz 2468

2. Kontrola PID

- Regulacja PID umożliwia sterowanie mechanizmem w oparciu o położenie, a nie napięcie. Używając PID, możesz powiedzieć ramieniu, aby obróciło się do 30 stopni, zamiast mówić mu, aby bezpośrednio wyprowadzało napięcie. Jest to szczególnie przydatne w autonomicznych. Możliwość nakazania robotowi przejechania 5 metrów zamiast pełnej mocy przez 0,5 sekundy pozwala na zwiększenie powtarzalności.
- Niektóre przydatne dokumenty dotyczące PID to:
- Blog Wesleya
- PID CSIM dla opornych

3. Magia ruchu (tylko CAN)

- Jeśli używasz regulatora prędkości TalonSRX, zaleca się użycie MotionMagic do sterowania mechanizmami, zwłaszcza czymś takim jak ramię lub winda. MotionMagic to zasadniczo pętla PID o częstotliwości 1 kHz podążająca za automatycznie generowanymi profilami ruchu trapezowego. Jeśli te słowa nie mają sensu, nie martw się! Zapoznaj się z powyższymi informacjami, aby uzyskać informacje na temat PID i tutaj'__s Dokument objaśniający profile ruchu. -
  Dokumentacja Motion Magic znajduje się jej e.

Poziom trzeci: zaawansowane ścieżki napędu, sterowanie wielofunkcyjne i testowanie jednostkowe

1. Prowadź ścieżki i podążaj za nimi

- Czasami surowy PID nie wystarcza do autonomicznego sterowania układem napędowym. Na przykład możesz chcieć, aby robot obszedł przełącznik i podniósł kostkę od tyłu. Czystym sposobem na to byłoby utworzenie ścieżki dysku. Ścieżka napędu to zasadniczo zestaw punktów, za którymi będzie podążać pętla PID układu napędowego, a punkty te doprowadzą do ostatecznego celu. PathWeaver to narzędzie graficzne, które korzysta z biblioteki o nazwie PathFinder, która generuje takie ścieżki i zapisuje je w pliku do analizy. Szczegółowe instrukcje dotyczące korzystania z PathWeaver można znaleźć jej e.
- Po wygenerowaniu punktów istnieje wiele sposobów podążania za nimi. Obejmują one zarówno użycie PID do bezpośredniego podążania za punktami, jak i dodanie algorytmu podążania za ścieżką w celu przetworzenia punktów przed przekazaniem ich do pętli PID. Przykład takiej ścieżki podążającej za algorytmem można znaleźć jej e (równanie 5.12). Inne popularne podejścia do podążania ścieżką to Adaptacyjny Czysty pościg Contro l. 254 ma przydatną implementację adaptacyjnego czystego pościgu, którą można znaleźć jej e.

2. Sterowanie oparte na modelu

- Sterowanie oparte na modelu to krok poza PID. Pozwala na zachowanie modelu matematycznego systemu w kodzie, a także aktualizację modelu o dane z czujników. Korzystając z takiego modelu, można znacznie dokładniej kontrolować położenie mechanizmu, prędkość, przyspieszenie itp. Niektóre zespoły, które korzystają z kontroli opartej na modelach, to 1678 i 971.
- Przydatne zasoby do uczenia się sterowania opartego na modelu to:
- Blog Wesleya
- Ta ulotka MIT

3. Testowanie jednostkowe

- Często chcesz przetestować swój kod przed wdrożeniem go na robocie. Może to zapobiec katastrofie. Testowanie jednostkowe to termin określający testowanie fragmentów kodu jako samodzielnych programów. Na przykład możesz chcieć przetestować część kodu, która uruchamia windę, ale nie część, która powoduje kilku kontrolek. Mechanizmy testowania *PIERWSZY*

  Zawody w robotyce są znacznie wzbogacone o sterowanie oparte na modelu, ponieważ model może być używany jako symulacja mechanizmu, co oznacza, że cały mechanizm może być testowany z niesamowitą

wytrzymałością. Niektóre przydatne biblioteki testów jednostkowych obejmują:

- GoogleTest (Test) Google

O firmie Compass Alliance

Compass Alliance zostało założone przez 10 zespołów z całego świata, a ich misją jest pomaganie zespołom FIRST Robotics Competition w utrzymaniu i rozwoju. Rosnące repozytorium zasobów i call center działające 24 godziny na dobę, 7 dni w tygodniu zapewniają każdemu, niezależnie od poziomu umiejętności, narzędzia do nauczenia się czegoś nowego lub dowiedzenia się więcej z dowolnego miejsca na świecie. Zdalne drużyny, które nie mają mentorów, mogą zapisać się do Tag Teamu, który będzie ich zdalnym przewodnikiem przez cały sezon, a Centra Pomocy wskazują, gdzie uzyskać dostęp do lokalnych usług oferowanych przez inne drużyny FIRST. Hear For You zapewnia zasoby i narzędzia, które pomagają zespołom i wolontariuszom rozwijać dobre samopoczucie psychiczne w swoich zespołach i podczas wydarzeń. Możesz dowiedzieć się więcej o The Compass Alliance, znaleźć wysokiej jakości pomoc i zaangażować się na www.thecompassalliance.org

O tym zasobie

Ten zasób został przygotowany przez The Compass Alliance, przy wsparciu i przeglądzie FIRST.

Jeśli masz pytania dotyczące tego zasobu, skontaktuj się z thecompassalliance@gmail.com lub

firstroboticscompetition@firstinspires.org.

Historia zmian

Rewizja #	Data aktualizacji	Uwagi do wersji
1.0	Grudzień 2018 r.	Pierwsze wydanie