Programmering 101

Denne ressource vil dække det grundlæggende i programmering i FIRST® Robotics Competition.

Det dækker C++, Java/Kotlin og LabVIEW.

Niveau et: Få din robot til at køre

1. Valg af et programmeringssprog: Java, C++ eller LabVIEW

- **Java** er et tekstsprog, der almindeligvis undervises i på gymnasier og bruges til AP CS-eksamenerne. Det er et "sikkert" sprog, der kører i sit eget virtuelle miljø. Selvom det generelt ikke påvirker FIRST Robotics Competition, betyder dette virtuelle miljø, også kendt som JVM, at Java-programmer er mærkbart langsommere end kompilerede sprog, når de bruges til beregningsintensive opgaver. Java vælges ofte på grund af dets brugervenlighed og krydskompatibilitet. Teams, der bruger Java, omfatter . 254,125,503,4911 og 1241.
- **C++** er et hurtigt tekstprogrammeringssprog. Det bruges i industrien til realtidssystemer på grund af dets kraft og effektivitet, men indlæringskurven er meget stejlere end Java. C++ udviklede sig fra programmeringssproget C, og blandingen af historiske og moderne funktioner fører nogle gange til forvirrende syntaks og/eller uventet adfærd. FIRST Robotics Competition-hold bruger det primært på grund af dets hastighed, fleksibilitet og dets omfattende matematiske biblioteker. Teams, der bruger C++, omfatter 971 og 1678.
- **LabVIEW** er et grafisk dataflow-programmeringssprog udviklet af National Instrumenter (NI) til brug for ingeniører og teknikere. LEGO WeDo og Mindstorms-sprog, der bruges til FIRST LEGO League Jr og FIRST LEGO League, er afledte af LabVIEW; Så studerende, der kommer fra disse programmer, kan finde det bekendt. I et LabVIEW-diagram er det meget nemt at drage fordel af avancerede computerfunktioner, såsom at køre kodestykker parallelt. Selvom de er effektive, introducerer sådanne funktioner ofte nye problemer at håndtere. NI tilbyder dog omfattende fejlfindingsværktøjer. LabVIEW-miljøet og sproget kommer med sin egen indlæringskurve og unikke udfordringer. FIRST Robotics Competition-hold bruger det primært på grund af dets forenklede grafiske syntaks

og omfattende tekniske biblioteker. Teams, der bruger LabVIEW, omfatter 33, 359, 624, 1986 og 2468.

- **Valget af sprog afhænger altid af, hvad der er nemmest for** dit **team.** For eksempel kan det ofte give mening at vælge et sprog, fordi en programmeringsmentor er ekspert i det sprog. På den anden side kan det give mening at vælge ud fra, hvor let det er at lære et givent sprog. Uanset hvad du beslutter, skal du huske, at valget af programmeringssprog er specifikt for arbejdsmiljøet og medarbejderne i dit team. Alle sprog er dygtige, godt understøttet og tilstrækkeligt kraftfulde til brug af FIRST Robotics Competition.

2. Undervisning i programmeringssproget

- Når du underviser i programmering for FØRSTE Robotkonkurrence, der er to forskellige emner, der skal undervises i. Den første er semantik og syntaks for selve programmeringssproget, og den anden er grænseflade med FØRSTE Robotteknologi Konkurrence komponenter. En guide til at lære C++-sproget kan findes HerJava hendese, og LabVIEW Her.

3. Vælg din startkode

- For Java og C++ er der fire forskellige "klasser", der kan bruges, når du interagerer med robotten. En sammenligning af disse klasser, og hvad de betyder, kan findes her.
- For LabVIEW inkluderer startskabelonerne en blanding af tidsindstillede, iterative og spawn/abort-mekanismer. Skabelonerne er beskrevet her.

4. ***Når dit team har valgt et programmeringssprog og er begyndt at kode, er FIRST Robotics Competition Docs-webstedet en god ressource til, hvordan du opsætter dit udviklingsmiljø og får kode på din robot. Disse vejledninger er uvurderlige for programmering af FIRST Robotics Competition.***
   - Introduktion
   - C++\Java
   - LabVIEW

5. ***Få kode på din robot!***
   - Java og C++
   - Hvis du bruger gradleRIO, skal du blot skrive "./gradlew deploy" i VS-koden

Konsol, og din kode vil være på robotten.

- Men vent! Din kode gør ikke noget endnu. Nogle enkle eksempler på drevkode kan findes her. Koden i uddragene hører til i enten Robot.java eller Robot.cpp, som skal oprettes automatisk med gradle/Eclipse-projektet.
- LabVIEW
- Til udvikling skal du køre fra kilden som vist her.
- Når du er færdig, skal du implementere en bygget eksekverbar fil som vist her.

6. Kode for mekanismer

- For Java og C++ kan simpel drevkode kopieres og indsættes herfra.
- For LabVIEW er den simple drevkode i skabelonen i TeleOp VI.
- De fleste FIRST Robotics Competition-robotter har andre aktiveringsmekanismer end drivlinjen. Det kan være alt fra et roterende svinghjul til en pneumatisk katapult. Alle disse mekanismer bør kunne kontrolleres autonomt eller teleop. For at styre mekanismer ved hjælp af hastighedsregulatorer over PWM er der en guide til C++ og Java her, og til LabVIEW, her.
- Hvis du bruger hastighedsregulatorer over CAN, skal du enten følge vejledningen her for at behandle dem som PWM-hastighedsregulatorer, eller bruge Phoenix API, hvis dokumentation er linket her.

7. Autonom

- En guide til, hvordan man udfører autonome handlinger i Java- og C++-programmering, kan findes her.
- Team 1619 har også samlet nogle simple koder til at krydse autolinjen i Java, som kan findes her.
- LabVIEW-skabeloner indeholder autonom kode til at ryste robotten på plads. Du kan ændre motoreffektværdier og timing for at udføre mange opgaver. Her er et eksempel på 2468's autonome kode fra 2018.

Niveau to: Brugerdefineret arkitektur og motorstyring med lukket kredsløb

1. Brug af en brugerdefineret arkitektur

- Mange gange er de tilgængelige robotklasser ikke nok. Du kan f.eks. køre teleop med jævne mellemrum og autonomt sekventielt. Hvis dette er tilfældet, er det sandsynligvis tid til at flytte til en brugerdefineret arkitektur.
- En brugerdefineret arkitektur strukturerer i bund og grund al koden på en tilpasset måde.
- Nogle eksempler på brugerdefinerede arkitekturer inkluderer 1678's kode, som er her.

  1678's kode bygger på 971's kode, som er her.

- 254 har også en brugerdefineret arkitektur. Deres 2019-kode kan findes Her. -

  33, 624, og 1986og 2468

2. PID-kontrol

- PID Control giver dig mulighed for at styre en mekanisme baseret på position i stedet for spænding. Ved hjælp af PID kan du bede en arm om at dreje til 30 grader i stedet for at bede den om at udsende en spænding direkte. Dette er især nyttigt i autonome. At kunne bede en robot om at køre 5 meter i stedet for fuld effekt i 0,5 sekunder giver mulighed for forbedret repeterbarhed.
- Nogle nyttige dokumenter til PID er:
- Wesleys blog
- CSIM's PID til dummies

3. Motion Magic (kun CAN)

- Hvis du bruger en TalonSRX-hastighedsregulator, anbefales det at bruge MotionMagic til at styre mekanismer, især noget som en arm eller en elevator. MotionMagic er i bund og grund en 1KHz PID-sløjfe, der følger autogenererede trapezformede bevægelsesprofiler. Hvis disse ord ikke giver mening, så fortvivl ikke! Se ovenstående for oplysninger om PID, og her's et dokument, der forklarer bevægelsesprofiler. - Dokumentationen til Motion Magic er fundet hendes e.

Niveau tre: Avancerede drevstier, MP-kontrol og enhedstest

1. Kør stier og følg dem

- Nogle gange er rå PID ikke nok til at styre drivlinjen autonomt. For eksempel vil du måske have robotten til at gå rundt om kontakten og samle en terning op bagfra. En ren måde at gøre dette på ville være at skabe en køresti. En kørebane er i bund og grund et sæt punkter, som drivlinjens PID-sløjfe vil følge, og punkterne vil føre til det endelige mål. PathWeaver er et grafisk værktøj, der

bruger et bibliotek kaldet PathFinder, som genererer sådanne stier og gemmer dem i en parsbar fil. Detaljerede instruktioner om brug af PathWeaver findes hendese.

- Når pointene er blevet genereret, er der en række forskellige måder at følge dem på. Disse spænder fra at bruge PID til direkte at følge punkterne, til at tilføje en stifølgende algoritme til at behandle punkterne, før de gives til PID-løkken. Et eksempel på en sådan stifølgende algoritme kan findes hendes e (EQN 5.12). Andre populære tilgange til stifølgende omfatter Adaptiv ren forfølgelseskontrol l. 254 har en praktisk implementering af adaptiv ren forfølgelse, som kan findes hendese.

2. Modelbaseret kontrol

- Modelbaseret kontrol er et skridt ud over PID. Det giver mulighed for at opbevare en matematisk model af systemet i koden og opdatere modellen med sensordata. Ved hjælp af en sådan model kan man kontrollere en mekanismes position, hastighed, acceleration osv. meget mere præcist. Nogle teams, der bruger modelbaseret kontrol, omfatter 1678 og 971.
- Nyttige ressourcer til indlæring af modelbaseret styring er:
- Wesleys blog
- Dette MIT-uddelingsark

3. Afprøvning af enheder

- Ofte vil du teste din kode, før du implementerer den på robotten. Dette kan forhindre katastrofe. Enhedstest er en betegnelse for test af dele af koden som selvstændige programmer. Du kan f.eks. teste den del af koden, der kører elevatoren, men ikke den del, der får et par lys til at blinke. Afprøvningsmekanismer for *FØRSTE* Robotkonkurrencen forbedres betydeligt med modelbaseret styring, da modellen kan bruges som en simulering af mekanismen, hvilket betyder, at hele mekanismen kan testes med utrolig robusthed. Nogle nyttige enhedstestbiblioteker omfatter:
- GoogleTest

Om Compass Alliance

Compass Alliance blev grundlagt af 10 hold fra hele verden med missionen om at hjælpe FIRST Robotics Competition-hold med at opretholde og vokse. Et voksende ressourcelager og 24/7 callcenter giver alle på ethvert færdighedsniveau værktøjerne til at lære noget nyt eller lære mere fra hvor som helst i verden. Fjernhold, der mangler mentorer, kan tilmelde sig et Tag Team for at være deres fjernguide i løbet af sæsonen og hjælpe hubs med at finde ud af, hvor de kan få adgang til lokale tjenester, som andre FIRST-teams tilbyder. Hør for dig giver ressourcer og værktøjer til at hjælpe teams og frivillige med at udvikle mental velvære på deres teams og ved arrangementer. Du kan lære mere om The Compass Alliance, finde kvalitetshjælp og blive involveret på www.thecompassalliance.org

Om denne ressource

Denne ressource er udarbejdet af The Compass Alliance med støtte og overblik fra FIRST. Hvis du har spørgsmål om denne ressource, bedes du kontakte thecompassalliance@gmail.com eller firstroboticscompetition@firstinspires.org.

Revisions historie

Revision #	Dato for revision	Bemærkninger til revision
1.0	Dec. 2018	Første udgivelse