Programmation 101

Cette ressource couvrira les principes fondamentaux de la programmation dans le cadre de la compétition de robotique FIRST®. Il couvre C++, Java/Kotlin et LabVIEW.

Premier niveau : Mettez votre robot en état de marche

1. Choisir un langage de programmation : Java, C++ ou LabVIEW

Java est un langage textuel largement enseigné dans les lycées et utilisé dans les examens AP CS. Il s'agit d'un langage « sécurisé » qui fonctionne dans son propre environnement virtuel. Bien que cela n'affecte pas la compétition de robotique FIRST dans son ensemble, cet environnement virtuel, également connu sous le nom de JVM, signifie que les programmes Java sont nettement plus lents que les langages compilés lorsqu'ils sont utilisés pour des tâches nécessitant beaucoup de calcul. Java est souvent choisi pour sa facilité d'utilisation et sa compatibilité croisée. Les équipes qui utilisent Java sont les suivantes : 254, 125, 503, 4911 et 1241.

C++ est un langage de programmation textuel rapide. Il est utilisé dans l'industrie pour les systèmes en temps réel en raison de sa puissance et de son efficacité, mais la courbe d'apprentissage est beaucoup plus raide que celle de Java. Le C++ a évolué à partir du langage de programmation C, et le mélange de fonctionnalités historiques et modernes conduit parfois à une syntaxe déroutante et/ou à un comportement inattendu. Les équipes de la compétition de robotique FIRST utilisent ce langage principalement en raison de sa vitesse, de sa flexibilité et de ses bibliothèques mathématiques étendues. Les équipes qui utilisent C++ incluent 971 et 1678 .

LabVIEW est un langage de programmation de flux de données graphiques développé par National. Instruments (NI) à l'usage des ingénieurs et des techniciens. Les langages Mindstorms utilisés pour LEGO WeDo et FIRST LEGO League Jr et FIRST LEGO League sont des dérivés de LabVIEW ; par conséquent, les étudiants de ces programmes peuvent trouver LabVIEW familier. Dans un diagramme LabVIEW, il est très facile de tirer parti des fonctionnalités de calcul avancées, telles que l'exécution de morceaux de code en parallèle.

Bien que puissantes, ces fonctionnalités posent souvent de nouveaux problèmes à gérer. Cependant, NI fournit des outils de débogage complets. L'environnement et le langage LabVIEW s'accompagnent de leur propre ensemble de courbes d'apprentissage et de défis uniques. Les équipes de la compétition FIRST Robotics utilisent principalement ce langage en raison de sa syntaxe graphique simplifiée et de ses bibliothèques d'ingénierie étendues. Les équipes qui utilisent LabVIEW comprennent 33, 359, 624, 1986 et 2468.

La langue que vous choisissez dépend toujours de ce qui est le plus facile pour votre équipe. Par exemple, il peut souvent être judicieux de choisir un langage parce qu'un consultant en programmation est un expert de ce langage. D'autre part, il peut également être judicieux de choisir en fonction de la facilité d'apprentissage d'une langue particulière. Quoi que vous décidiez, n'oubliez pas que le choix du langage de programmation est spécifique à l'environnement de travail et aux personnes de votre équipe. Tous les langages sont performants, bien pris en charge et suffisamment puissants pour être utilisés dans la compétition de robotique FIRST.

2. Enseignement du langage de programmation

- Lors de l'enseignement de la programmation pour la compétition de robotique FIRST, il y a deux matières différentes qui doivent être enseignées. Le premier est la sémantique et la syntaxe propres au langage de programmation, et le second est l'interfaçage avec les composants de la compétition de robotique FIRST. Guide d'apprentissage du langage C++ donc, à Java donc et LabVIEW donc Atteindre.

3. Choisissez votre code de démarrage

- Pour Java et C++, il existe quatre « classes » différentes qui peuvent être utilisées lors de l'interfaçage avec le Bot. Une comparaison de ces classes et de ce qu'elles signifient peut être trouvée ici.
- Pour LabVIEW, les modèles de démarrage incluent un mélange de mécanismes planifiés, itératifs et de génération/annulation. Les modèles sont décrits ici.

4. Une fois que votre équipe a choisi un langage de programmation et commencé à coder, le site FIRST Robotics Competition Docs est une bonne ressource sur la façon de configurer votre environnement de développement et de télécharger du code sur votre

robot. Ces guides sont inestimables pour la programmation de la compétition de robotique FIRST.

- Commencer
- C++\Java
- LabVIEW

5. Téléchargez le code sur votre robot !

- Java et C++

  Si vous utilisez -gradleRIO, tapez simplement « ./gradlew deploy » dans le VS Code. La console et votre code seront sur le robot.

  - Mais attendez ! Votre code ne fait encore rien. Vous trouverez ici quelques exemples simples de code de pilote. Le code des snippets appartient à Robot.java ou Robot.cpp, qui doivent être générés automatiquement avec le projet gradle/Eclipse.

- LabVIEW

  - Pour le développement, vous devez l'exécuter à partir des sources comme indiqué ici.

  - Une fois cela fait, vous allez déployer un exécutable qui a été créé comme indiqué ici.

6. Codes des mécanismes

- Pour Java et C++, le code simple du pilote peut être copié et collé ici.
- Pour LabVIEW, le code du pilote simple est disponible dans le modèle dans TeleOp VI.
- La plupart des robots de compétition FIRST Robotics ont des mécanismes qui sont actionnés à l'extérieur de la transmission. Il peut s'agir d'un volant d'inertie rotatif ou d'une catapulte pneumatique. Tous ces mécanismes doivent pouvoir être contrôlés de manière autonome ou téléoptique. Pour contrôler les mécanismes à l'aide de contrôleurs de vitesse sur PWM, il existe un guide pour C++ et Java, et un guide pour LabVIEW.

- Si vous utilisez des contrôleurs de vitesse sur CAN, vous devez soit suivre le guide ici , soit utiliser l'API Phoenix, dont la documentation est liée ici, pour les traiter comme des contrôleurs de vitesse PWM.

7. Autonome

- Un guide sur la façon d'effectuer des actions autonomes en programmation Java et C++ peut être trouvé ici.
- L'équipe 1619 a également compilé du code simple pour traverser la ligne automatique en Java, que vous pouvez trouver ici.
- Les modèles LabVIEW contiennent du code autonome pour secouer le robot en place. Vous pouvez modifier les valeurs de puissance du moteur et le calage pour effectuer de nombreuses tâches. Voici un exemple du code autonome de 2468 de 2018.

Deuxième niveau : architecture personnalisée et contrôle moteur en boucle fermée

1. Utilisation d'une architecture personnalisée

- Souvent, les classes de robots existantes ne suffisent pas. Par exemple, vous pouvez souhaiter téléopérer périodiquement et exécuter de manière séquentielle de manière autonome. Si c'est le cas, il est probablement temps de passer à une architecture personnalisée.
- L'architecture personnalisée est essentiellement la configuration de tout le code de manière personnalisée.
- Quelques exemples d'architecture spécifique incluent le code de 1678, qui est inclus ici. Le code de 1678 est basé sur le code de 971 ici.
- La 254 possède également une architecture particulière. Leur code 2019 peut être trouvé ici.
- 33, 624 et 1986 et 2468

2. Contrôle PID

- Le contrôle PID vous permet de contrôler un mécanisme en fonction de la position plutôt que de la tension. À l'aide du PID, vous pouvez dire à un bras de pivoter de 30 degrés au lieu de lui dire directement de produire une tension. Ceci est particulièrement utile dans les systèmes autonomes. Le fait de pouvoir dire à

un robot de parcourir 5 mètres au lieu de conduire à pleine puissance pendant 0,5 seconde offre une meilleure répétabilité.

- Voici quelques documents utiles pour le PID :
  - Le blog de Wesley
  - Le PID du CSIM pour les nuls

3. Magie du mouvement (CAN uniquement)

- Si vous utilisez un variateur TalonSRX, il est recommandé d'utiliser MotionMagic, en particulier pour contrôler des mécanismes tels que des leviers ou des ascenseurs. MotionMagic est essentiellement une boucle PID de 1 kHz qui suit des profils de mouvement trapézoïdal générés automatiquement. Si ces mots n'ont pas de sens, ne vous inquiétez pas ! Voir ci-dessus pour plus d'informations sur le PID et ici Il existe un document qui décrit les profils de transaction.

- La documentation de Motion Magic se trouve ici.

Troisième niveau : trajectoires de conduite avancées, contrôle MP et tests unitaires

1. Conduisez sur les routes et suivez-les

- Parfois, le PID brut n'est pas suffisant pour contrôler de manière autonome la transmission. Par exemple, vous voudrez peut-être que le robot contourne l'interrupteur et prenne un cube à l'arrière. Une façon astucieuse de le faire est de créer un chemin de conduite. La trajectoire d'entraînement est essentiellement une série de points que la boucle PID de la transmission suivra, et les points mèneront à la destination finale. PathWeaver est un outil graphique qui utilise une bibliothèque appelée PathFinder qui crée de tels chemins et les enregistre dans un fichier analysable. Vous trouverez des instructions détaillées sur l'utilisation de PathWeaver ici.

- Une fois les points créés, il existe différentes façons d'en garder une trace. Celles-ci vont de l'utilisation du PID pour suivre directement les points à l'ajout d'un algorithme de suivi de chemin pour traiter les points avant de les donner à la boucle PID. Un exemple d'un tel algorithme de suivi de chemin peut être trouvé ici (éq. 5.12). D'autres approches populaires du suivi routier incluent le contrôle adaptatif du suivi pur. Une implémentation pratique du suivi pur adaptatif est disponible sur le 254 et peut être trouvée ici.

2. Contrôle basé sur le modèle

- Le contrôle basé sur un modèle va au-delà du PID. Il permet de conserver un modèle mathématique du système dans le code et de mettre à jour le modèle avec les données des capteurs. En utilisant un tel modèle, la position, la vitesse, l'accélération, etc., d'un mécanisme peuvent être

contrôlées avec beaucoup plus de précision. Parmi les équipes qui utilisent le contrôle basé sur un modèle, citons 1678 et 971.

- Voici quelques ressources utiles pour l'apprentissage du contrôle basé sur un modèle :
  - Le blog de Wesley
  - Cette brochure du MIT

3. Tests unitaires

- La plupart du temps, vous souhaitez tester votre code avant de le déployer sur le robot. Cela pourrait éviter une catastrophe. Les tests unitaires sont un terme utilisé pour tester des parties de code en tant que programmes autonomes. Par exemple, vous voudrez peut-être tester la partie du code qui fait fonctionner l'ascenseur, mais pas la partie qui fait clignoter quelques lumières. Le test des mécanismes pour la compétition de robotique FIRST est considérablement amélioré par le contrôle basé sur le modèle, car le modèle peut être utilisé comme une simulation du mécanisme, ce qui signifie que l'ensemble du mécanisme peut être testé avec une robustesse incroyable. Voici quelques bibliothèques de tests unitaires utiles :
- GoogleTest

À propos de Compass Alliance

L'Alliance Compass a été fondée par 10 équipes du monde entier pour aider les équipes de la compétition de robotique FIRST à survivre et à se développer. Un référentiel de ressources en pleine expansion et un centre d'appels 24h/24 et 7j/7 fournissent à toute personne, quel que soit son niveau de compétence, les outils nécessaires pour apprendre quelque chose de nouveau ou en apprendre davantage de n'importe où dans le monde. Les équipes à distance sans mentor peuvent s'inscrire à une équipe qui les guidera à distance tout au long de la saison, et les centres d'aide déterminent où accéder aux services locaux proposés par d'autres équipes FIRST. Hear For You fournit des ressources et des outils pour aider les équipes et les bénévoles à améliorer leur santé mentale au sein de leurs équipes et lors d'événements. Vous pouvez en savoir plus sur The Compass Alliance, trouver de l'aide de qualité et vous impliquer à www.thecompassalliance.org.

À propos de cette ressource

Cette ressource a été produite par la Compass Alliance avec le soutien et la vue d'ensemble de FIRST. Si vous avez des questions sur cette ressource, veuillez communiquer avec thecompassalliance@gmail.com ou firstroboticscompetition@firstinspires.org.

Historique des révisions

Révision #	Date de révision	Notes de révision
1.0	Casser. 2018	Version initiale