Los Caminos de la Alianza: Programación

Este camino cubrirá los fundamentos de la programación en FRC. Se centrará en lenguajes basados en texto (como C ++ o Java / Kotlin), aunque la gran mayoría de los conceptos discutidos tienen sus contrapartes en LabVIEW.

Nivel Uno: Poner a su Robot en Marcha

1. **Escoger un lenguaje de programación: Java, C ++ o LabVIEW**
   - **Java** es un lenguaje textual que se enseña muy comúnmente en las escuelas secundarias y que se usa para los exámenes AP CS. Es un lenguaje de "seguro" en el que se ejecuta en su propio entorno virtual. Por desgracia, este entorno virtual, también conocida como la JVM, significa que los programas Java son notablemente más lento que sus homólogos de menor nivel (como C ++) cuando se utiliza para tareas de alta intensidad. Java se utiliza sobre todo debido a su facilidad de uso y compatibilidad. Los equipos que utilizan Java incluyen 254, 125, 503, 4911 y 1241.
   - **C ++** es un lenguaje de programación textual muy rápido. Se utiliza en la industria para sistemas de tiempo real, debido a su eficiencia, pero la curva de aprendizaje es a menudo mucho más grande que la de Java. C ++ también está plagado de comportamiento inusual, o no definido, que a menudo puede ser difícil para los nuevos equipos de manejar. Muchos equipos de FRC lo utilizan simplemente debido a su velocidad, así como sus extensas bibliotecas matemáticas. Los equipos que utilizan C ++ incluyen 971 y 1678.
   - A diferencia de Java y C ++, LabVIEW es un lenguaje de programación desarrollado por NI. Los lenguajes de programación utilizados para FLL son derivados de LabVIEW, para que los estudiantes procedentes de programas de Lego puedan estar más cómodos de comenzar con esto. Además, NI ofrece amplias herramientas de depuración para LabVIEW. Sin embargo, LabVIEW viene con su propia curva de aprendizaje, debido a sus sistemas de dependencia, interfaz gráfica extraña, de cantidad de información que no sean necesarios, y la lentitud. Los equipos que utilizan LabVIEW incluye a 624.
   - **La elección de idioma siempre depende de lo que es más fácil para su equipo.** Por ejemplo, a menudo puede tener sentido usar C ++, simplemente porque uno de sus mentores de programación sabe C ++. Por otro lado, podría tener sentido usar Java, ya que es más fácil de aprender sin la ayuda de un mentor. No importa cuál sea su decisión, recuerde que la elección del lenguaje de programación es específica para el entorno de trabajo y la gente de su equipo.
2. **La enseñanza del lenguaje de programación**

- Para la enseñanza de programación para FRC, hay dos temas distintos que necesitan ser enseñados. La primera es la semántica y la sintaxis del lenguaje de programación en sí, y la segunda operación de interfaz con los componentes de FRC. Una guía en el aprendizaje del lenguaje C ++ se puede encontrar aquí, de Java aquí y de LabVIEW aquí.

3. **Selección de una clase de robot**
   - Hay cuatro “clases” diferentes que se pueden utilizar, cuando interactúe con el robot. Una comparación de estas clases, y lo que significan se puede encontrar aquí.

4. **Una vez que el equipo ha elegido un lenguaje de programación y una clase de robot, los pasos en WPI son un buen recurso sobre cómo configurar el entorno de desarrollo y conseguir fácilmente código en su robot. Estas guías son de gran valor para la programación FRC.**
   - Java
   - C ++
   - La temporada oficial 2019 utiliza gradleRIO y el editor VSCode con el plugin WPILib. Más información se puede encontrar aquí. GradleRIO, y una guía para su instalación se pueden encontrar aquí. El resto de esta guía será más fácil de seguir con una configuración gradleRIO.

5. **Obtener código en su robot!**
   - Si se utiliza gradleRIO, sólo tiene que escribir `./gradlew desplegar, y su código será en el robot.`
   - ¡Pero espera! Su código no hace nada todavía. Algunos ejemplos simples para el código de la unidad se pueden encontrar aquí para Java, y aquí para C ++. El código en los fragmentos pertenecen, ya sea en Robot.java o Robot.cpp, que debe ser auto-creado con el proyecto / Eclipse Gradle.
6. **Código de mecanismos**
   - Código de unidad simple pueden ser directamente copiado-pegado para Java / C ++ desde aquí.
   - La mayoría de los robots FRC han accionado mecanismos distintos del drivetrain. Esto podría ser cualquier cosa, desde un volante giratorio hasta una catapulta neumática. Todos estos mecanismos deben ser controlables en forma autónoma, o en teleop. A los mecanismos que usan controladores de velocidad sobre PWM, hay una guía para C ++ y Java aquí.
   - Si usa controladores de velocidad a través de CAN, debe seguir la guía aquí aquí para tratarlos como controladores de velocidad PWM o usar la API de Phoenix, cuya documentación está vinculada aquí.

7. **Autónomo**

- Se puede encontrar una guía sobre cómo realizar acciones autónomas en la programación de FRC aquí.
- Equipo de 1619 también ha compilado un código simple de cruzar la línea de auto en Java, que se puede encontrar aquí.

Nivel Dos: arquitectura personalizada

1. **Usando una arquitectura personalizada**
   - Muchas veces, las clases de robots disponibles no son suficientes. Por ejemplo, es posible que desee ejecutar TeleOp periódicamente, y de forma secuencial autónoma. Si este es el caso, lo más probable es el momento de pasar a una arquitectura personalizada.
   - Una arquitectura personalizada está estructurando esencialmente todo el código de una manera personalizada.
   - Algunos ejemplos de arquitecturas personalizadas incluyen el código de 1678 que es aquí. Código de 1678 se construye fuera del código de 971 que es aquí.
   - 254 también tiene una arquitectura personalizada. Su código de 2018 se puede encontrar aquí.
2. **control PID**
   - Control PID permite controlar un mecanismo basado en la posición, en lugar de tensión. El uso de PID, se puede decir un brazo a su vez a 30 grados, en lugar de decirle a la salida una tensión directa. Esto es especialmente útil en autónoma. Ser capaz de decir un robot para conducir 5 metros en lugar de plena potencia durante 0,5 segundos permite una mayor repetibilidad.
   - Algunos documentos útiles para PID son:
     - Blog de Wesley
     - PID del CISM
3. **Motion Magic (CAN solamente)**
   - Si se utiliza un controlador de velocidad TalonSRX, se recomienda utilizar MotionMagic para el control de mecanismos, especialmente algo como un brazo o un ascensor. MotionMagic es esencialmente un bucle 1KHz PID siguiente perfiles de movimiento trapezoidales autogenerado. Si esas palabras no tienen sentido, no se preocupe! Lea lo anterior para obtener información sobre PID y aquí está un documento que explica perfiles de movimiento.
   - La documentación de Motion Magic se encuentra aquí.

Nivel Tres: Advanced Drive Caminos, Control de MP, y la unidad de pruebas

1. **Conducir caminos y seguirlas**
   - A veces, el PID en bruto no es suficiente para controlar el drivetrain de forma autónoma. Por ejemplo, es posible que desee que el robot gire alrededor del

interruptor y recoja un cubo desde atrás. Una forma de hacerlo sería crear una ruta de acceso de unidad. Una ruta de acceso es esencialmente un conjunto de puntos que seguirá el bucle PID de la transmisión, y los puntos conducirán al objetivo final. Jaci de WPILib ha creado una herramienta llamada PathFinder que genera dichas rutas y las guarda en un archivo analizable, que se puede encontrar aquí.

- Una vez que se han generado los puntos, hay una variedad de maneras para seguirlos. Estos van desde el uso de PID de seguir directamente los puntos, hasta la adición de un camino siguiente algoritmo para procesar los puntos antes de dar al bucle PID. Un ejemplo de tal algoritmo de camino siguiente se puede encontrar aquí (Ecuación 5.12).

2. **control basado en modelo**
    - El control basado en modelos es un paso más allá de PID. Permite mantener un modelo matemático del sistema en el código y actualizar el modelo con datos del sensor. Usando un modelo así, uno puede controlar la posición, velocidad, aceleración, etc. de un mecanismo de manera mucho más precisa. Algunos equipos que usan control basado en modelos incluyen 1678 y 971.
    - Recursos útiles para el aprendizaje de control basado en el modelo son:
        - Blog de Wesley
        - Este folleto MIT

3. **Examen de la unidad**
    - Muchas veces, desea probar su código antes de implementarlo en el robot. Esto puede prevenir el desastre. Unidades de prueba es un término para probar partes del código como programas independientes. Por ejemplo, es posible que desee probar la parte del código que ejecuta el elevador, pero no la parte que hace que se enciendan algunas luces. Los mecanismos de prueba para FRC se mejoran en gran medida con el control basado en el modelo, ya que el modelo se puede usar como una simulación del mecanismo, lo que significa que todo el mecanismo se puede probar con una increíble robustez. Algunas bibliotecas de prueba de unidad útiles incluyen:
        - GoogleTest

Apéndice A - Historial de revisiones

Revisión #	Fecha de revisión	Notas de revisión
1.0	de septiembre 2018	Versión inicial