Programming FRC Robots: An Introduction

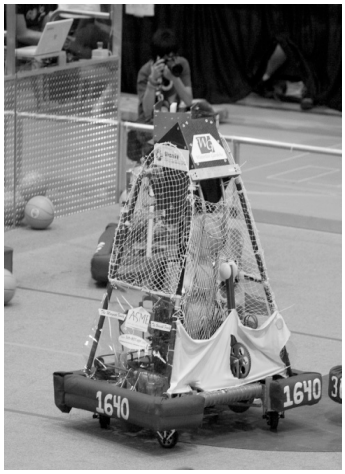
The FRC Robot Framework

Motor Basics

Sensor Basics

Real-Time Systems Programming

Intro to Robot Programming



So, you have a robot with wheels and belts and motors and sensors...

Now you need to write code to control all parts of the robot in an efficient and strategic way in order to compete in the game...

The FRC Robot Framework

- FRC provides a framework for robot code
  - It is required to integrate with the Field Management System (FMS) at competitions
  - Teams hook in their own code in certain places
- Default code and supported libraries are available in Java, C++, and LabVIEW
- WPI library code provides hardware integration
  - For motors, motor controllers, pneumatics, sensors, joysticks, and Kinect sensors

Java Robot Framework

http://www.wbrobotics.com/javadoc/edu/wpi/first/wpilibj/IterativeRobot.html

```
import edu.wpi.first.wpilibj.IterativeRobot;

public class MyRobot extends IterativeRobot {
  public void robotInit() { … }
  public void disabledInit() { … }
  public void disabledPeriodic() { … }
  public void autonomousInit() { … }
  public void autonomousPeriodic() { … }
  public void teleopInit() { … }
  public void teleopPeriodic() { … }
  // <disabled,autonomous,teleop>Continuous as well
  // team-specific methods
}
```

C++ Robot Framework

http://firstforge.wpi.edu/sf/go/doc1197

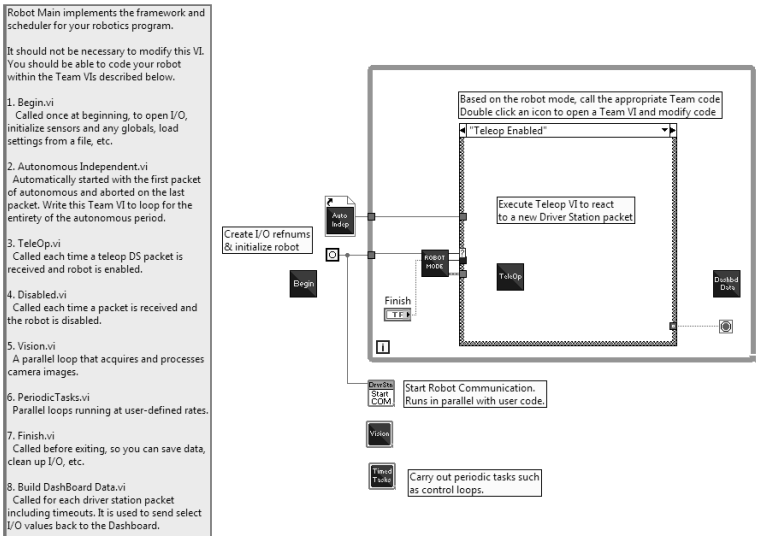
```
class My2011Robot : public IterativeRobot

{  // data member declarations …
   public:
   MyRobot(void) { // constructor }
   void RobotInit() { // initialization }
   void DisabledInit() { … }
   void AutonomousInit() { … }
   void TeleopInit() { … }
   void DisabledPeriodic() { … }
   void AutonomousPeriodic() { … }
   void TeleopPeriodic() { … }
   // team-specific methods …

};

START_ROBOT_CLASS(My2011Robot);
```


LabVIEW Robot Framework



Motor Basics

- Motors need control signals to run
  - Can be simple on/off using a spike (relay) controller
- Motor speed controllers give percentage of power
  - Two types in FRC support libraries:
    - Victors
    - Jaguars
  - Code refers to the speed controller, not the motor
  - Speed controller sends signals to the motor

Motor Basics: Java Examples

```
import edu.wpi.first.wpilibj.Jaguar;

// creating a Jaguar reference
Jaguar motor1 = new Jaguar(1); // default digital module, channel 1
// setting the desired speed (range of -1.0 to 1.0)
motor1.set(0.5);

import edu.wpi.first.wpilibj.Victor;

// creating a Victor reference
Victor motor2 = new Victor(2); // default digital module, channel 2
// setting the desired speed (range of -1.0 to 1.0)
motor2.set(0.25);

import edu.wpi.first.wpilibj.Relay;

// creating a Relay reference, allowing forward direction only
// default digital module, channel 1
Relay motor3 = new Relay(1, kForward);
// setting the motor to on and off
motor3.set(kOn);
motor3.set(kOff);
```

Motor Basics: C++ Examples

```
#include “WPILib.h”;

// creating a Jaguar reference
Jaguar *motor1 = new Jaguar(1); // default digital module, channel 1
// setting the desired speed (range of -1.0 to 1.0)
motor1->Set(0.5);

// creating a Victor reference
Victor *motor2 = new Victor(2); // default digital module, channel 2
// setting the desired speed (range of -1.0 to 1.0)
motor2->Set(0.25);

// creating a Relay reference, allowing forward direction only
// default digital module, channel 1
Relay *motor3 = new Relay(1, kForward);
// setting the motor to on and off
motor3->Set(kOn);
motor3->Set(kOff);
```


Motor Basics: LabVIEW Examples

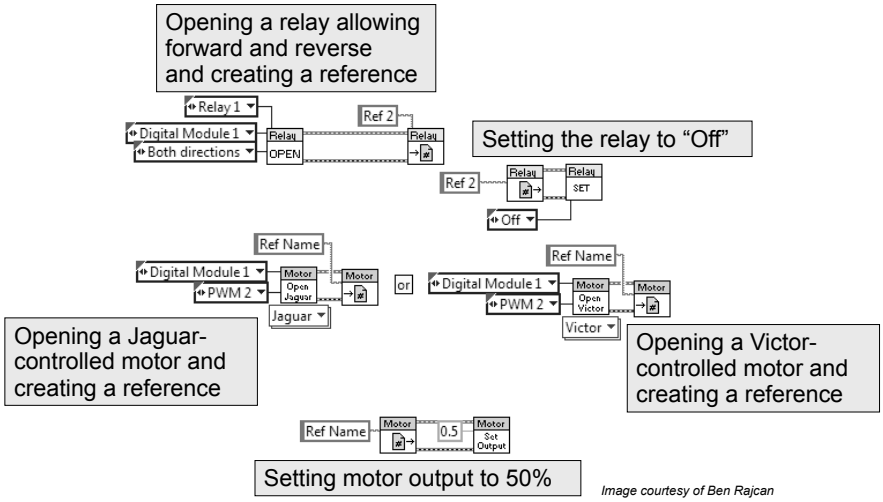


Image courtesy of Ben Rajcan

Sensor Basics

- Sensors provide data about a robot and its environment
  - The angle of wheels, the distance from an object, etc.
- Sensors can provide analog or digital data
  - **Analog** sensors provide varying voltages
  - **Digital** sensors provide on/off data
- Support libraries are provided for both types

Sensor Basics: Java Examples

```
import edu.wpi.first.wpilibj.AnalogChannel;

// Create a reference to an analog sensor
// default analog module, channel 1
AnalogChannel sensor1 = new AnalogChannel(1);

// Get the average voltage from the analog sensor
double voltage = sensor1.getAverageVoltage();

import edu.wpi.first.wpilibj.DigitalInput;

// Create a reference to a digital sensor
// default digital module, channel 1
DigitalInput sensor2 = new DigitalInput(1);

// Get the value from the sensor
boolean value = sensor2.get();
```

Sensor Basics: C++ Examples

```
#include <AnalogChannel.h>

// Create a reference to an analog sensor
// default analog module, channel 1
AnalogChannel *sensor1 = new AnalogChannel(1);

// Get the average voltage from the analog sensor
float voltage = sensor1->GetAverageVoltage();

#include <DigitalInput.h>

// Create a reference to a digital sensor
// default digital module, channel 1
DigitalInput *sensor2 = new DigitalInput(1);

// Get the value from the sensor
UINT32 value = sensor2->Get();
```


Sensor Basics: LabVIEW Examples

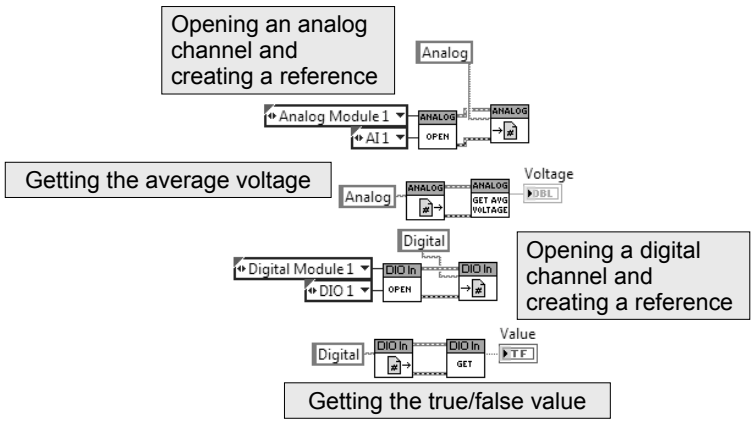
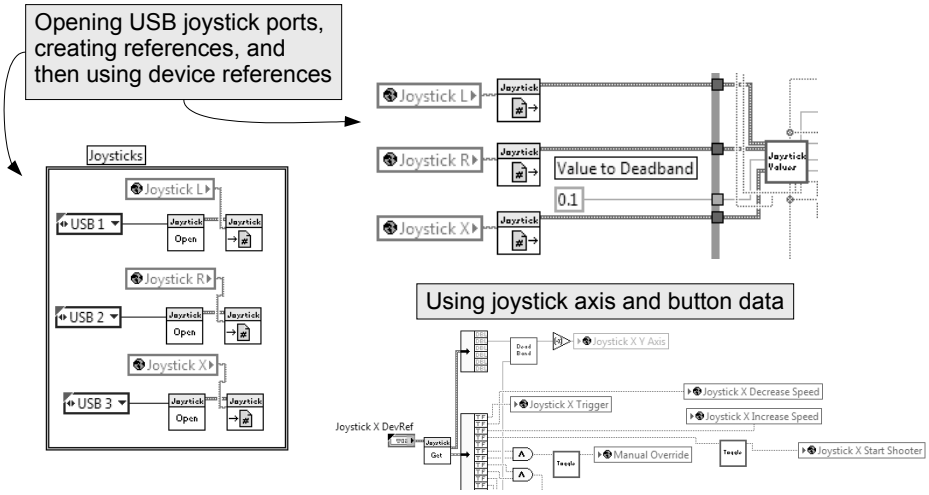


Image courtesy of Ben Rajcan

Joystick Basics

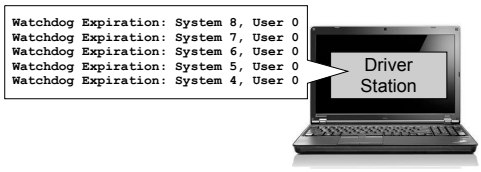


Real-Time Systems Programming

- Real-time system – hardware/software system with real-time constraints
  - e.g. deadline for system response after an event
- Robots are real-time systems
  - Driver controls need to be carried out in real-time
  - Interacting with the world has to be in real-time
- All parts of a real-time system must be time-aware or actively scheduled

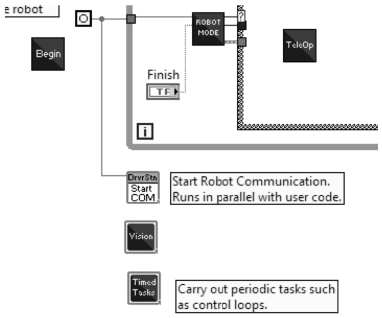
Real-Time Constaint: Safety

- The robot framework has built-in safety features to ensure safe robot operation
  - e.g. motors will be turned off if no signals are received within 100ms (default value)
- *Best practice*: Make sure to send each motor a signal at least every 50ms
  - Even if it is to set it to 0 power



"Parallel" Processes

- Processes happen "**in parallel**" in real-time systems
  - Actually, each gets a slice of CPU time one at a time
  - LabVIEW handles this "threading" automatically



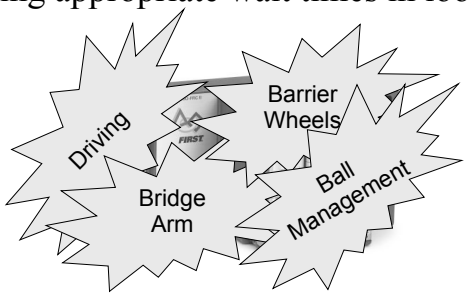
Memo

I don't have personal experience with the Java and C++ FRC frameworks, but it looks like you have to create threads explicitly.

- JCBC

How NOT to Freeze Your Robot

- Threading of multiple processes needs the processes to "play nicely" by having "idle time" built in
  - Each needs to allow other processes time to run
  - Putting appropriate wait times in loops is important



Summary

- FRC framework is required for all competition bots
- Default code and WPI support library is available
  - Java, C++, and LabVIEW are supported
  - Other languages possible, but not supported
- Real-time systems programming involves:
  - Handling real-time constraints (safety monitors)
  - Handling multiple "parallel" processes and making sure they all share CPU time nicely