



cRIO

2009 Control System Workshop

Edina Robotics FIRST Team 1816
The Green Machine
Eyes on the Future

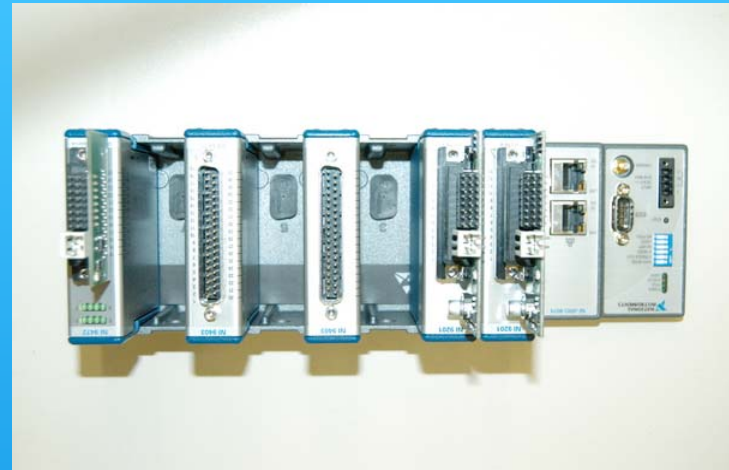
Wiring and Mounting

The electrical and mechanical
components

CompactRIO

And Modules

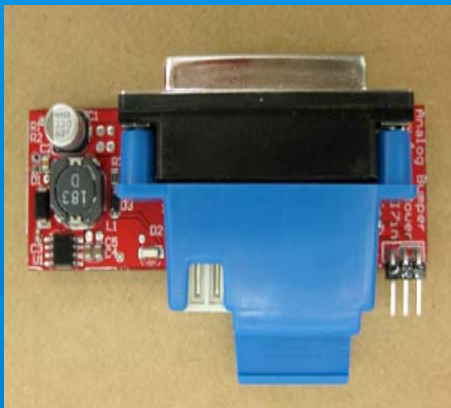
- What is the cRIO?
 - Compact Realtime Input Output
 - Replaces robot controller
- i/o Modules
 - NI 9201- analog input (8)
 - analog bumper
 - NI 9403- digital i/o (32)
 - digital sidecar
 - NI 9472- digital i/o (8)
 - pneumatic bumper



Bumpers for the cRio

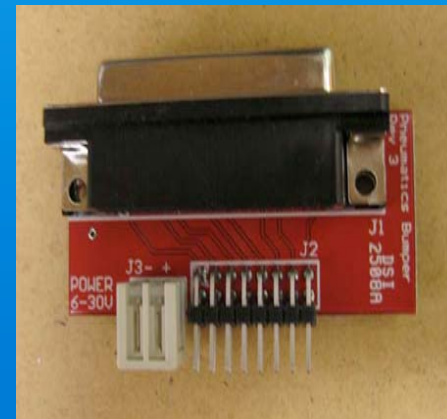
Analog bumper

- Center for all sensors and any none digital componets



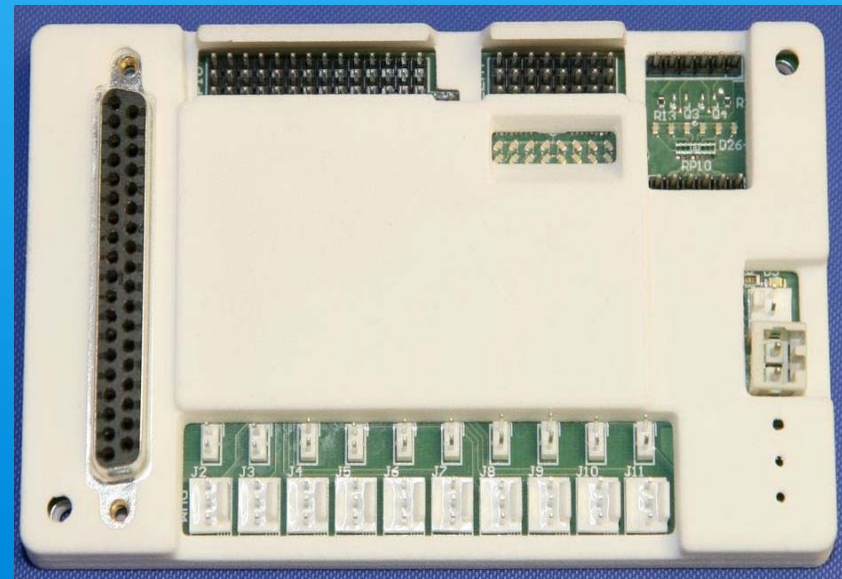
Pneumatic bumper

- Center for all pneumatic componets
- Only used when solenoids are not wired through spikes



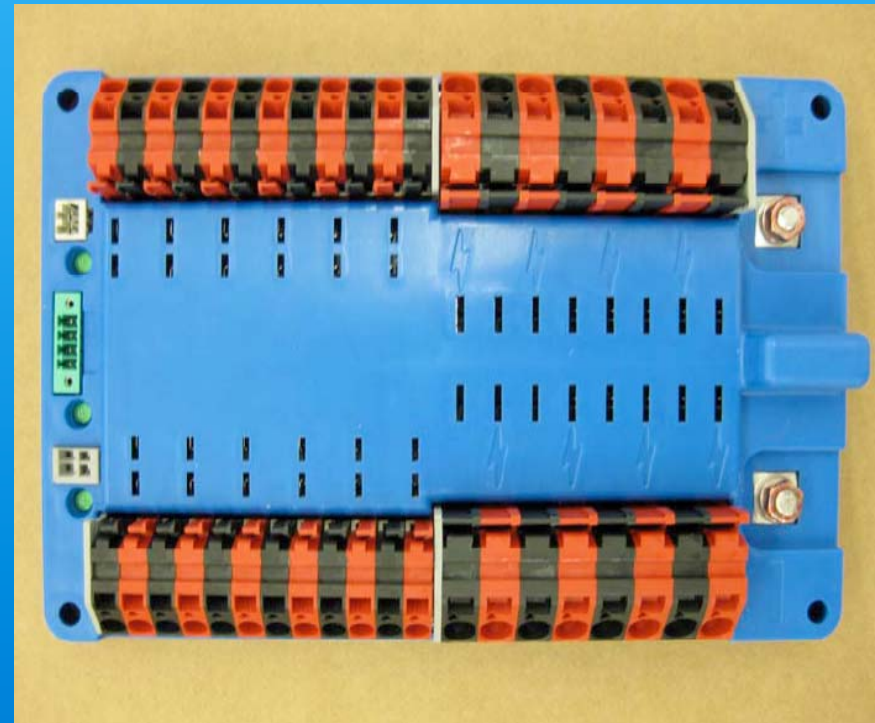
Digital Side Car (DSC)

- All digital components are connected here
- Examples:
 - Speed Controllers
 - Spikes
 - Geartooth



Power Distribution Block

- Power center
- Replaces old power distribution block and fuse blocks
- Contains room for circuit breakers
- Everything that requires power is connected to the Power Distribution Block



Driver's Station

Power Cable Port

USB Ports

Ethernet
Ports

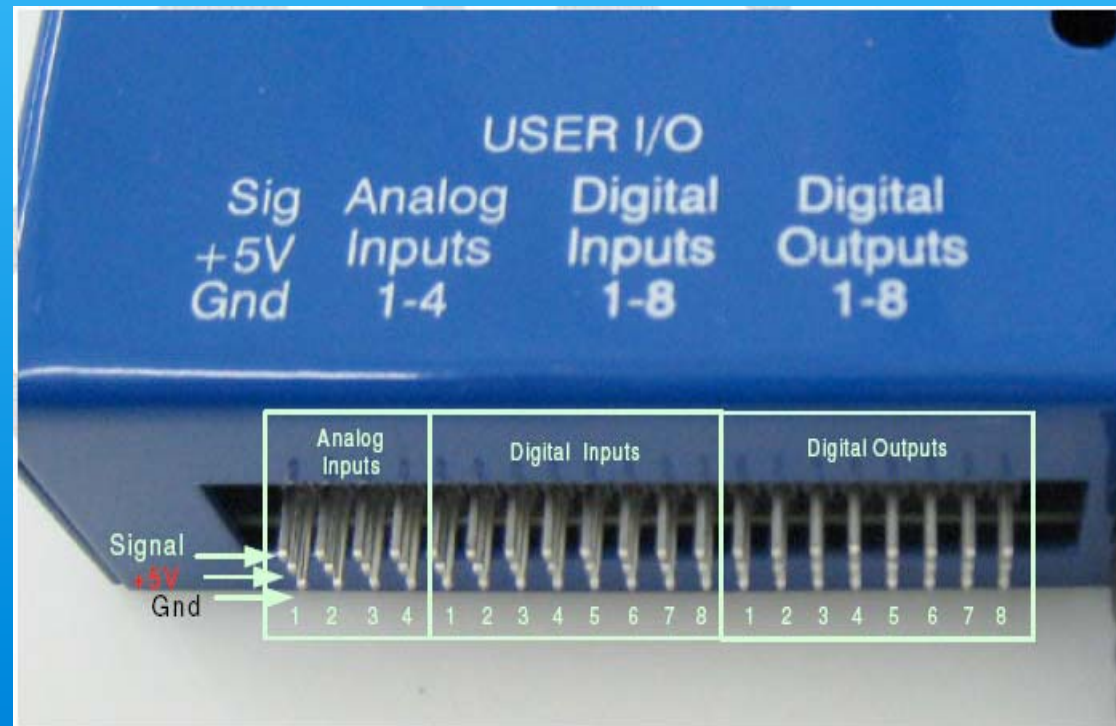
Dongle

Autonomous/Teleoperated



Analog and Digital Outputs/Inputs

- Analog Inputs
 - ex. dials
- Digital Inputs
 - ex. jumpers and switches
- Digital Outputs
 - ex. lights



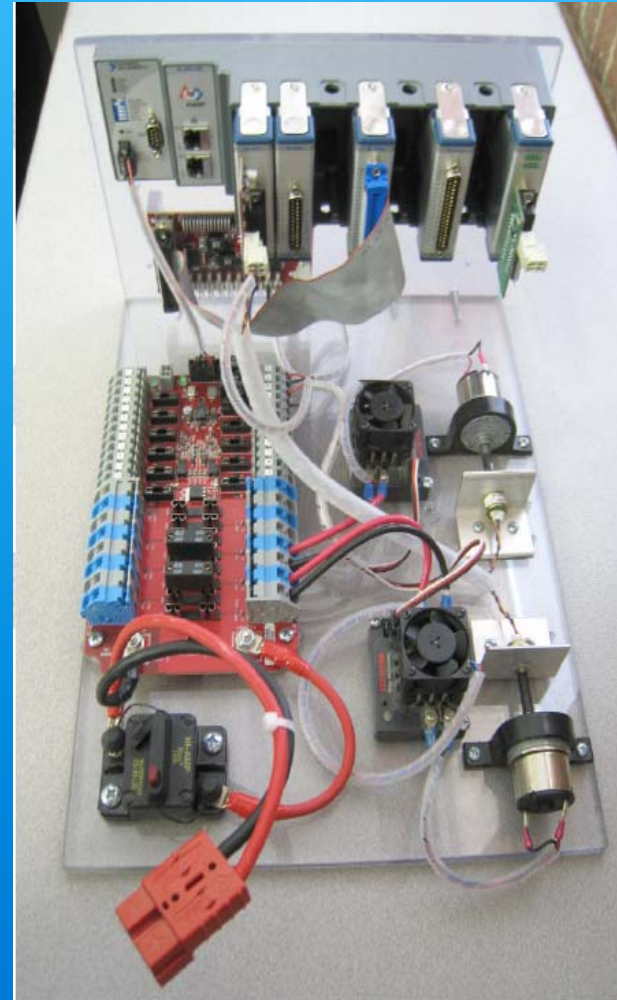
Camera: Axis 206W

- Utilized by the updated WPILib
- Built-in VIs in LabVIEW set up to implement camera control
- Cannot manipulate itself



Miscellaneous Tips

- Mounting
 - where are the mounting holes?
 - mount cRIO vertically to prevent dust accumulation
 - mount for easy access



Miscellaneous Tips

- The key to "Wago"-ing
 - Insert the Wago tool at the correct angle and lift upwards
 - Compartment below should open for wire insertion
 - Might feel like you are breaking the power distribution block
- Ferrule Pins
 - handy for connections
 - no stray wires
 - easy to crimp



Weights of Components

CompactRIO	2 lbs
Analog Input Module and bumper	.4 lbs
Solenoid Module and bumper	.4 lbs
Digital I/O Module	.3 lbs
Digital Sidecar	.25 lbs
Power Distribution Board	1.6 lbs
Total	5 lbs

2-3 lbs more than previous years' control system

Putting it all together

Data distribution

Power distribution

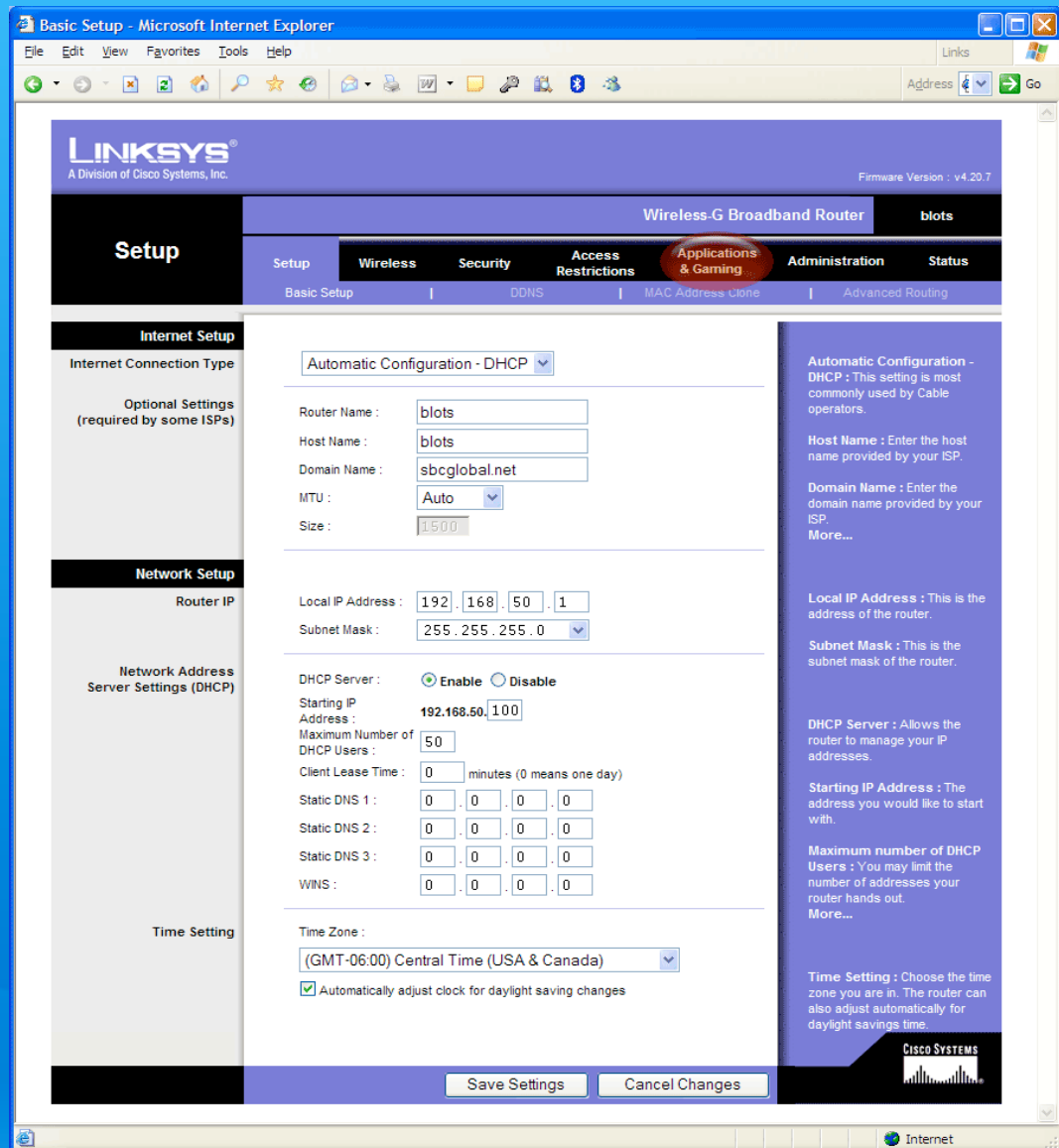


Programming

LabVIEW and C/C++

Wireless Radio

- Uses Linksys hardware available at local retailers
- Standard Linksys maintenance system



Wireless Radio

- WRT600N Dual band N Router attaches to Driver's Station
 - Broadcasts both 5.8Ghz (N) and 2.4Ghz (G and B)
- WGA600N Dual band gaming bridge attaches to cRIO



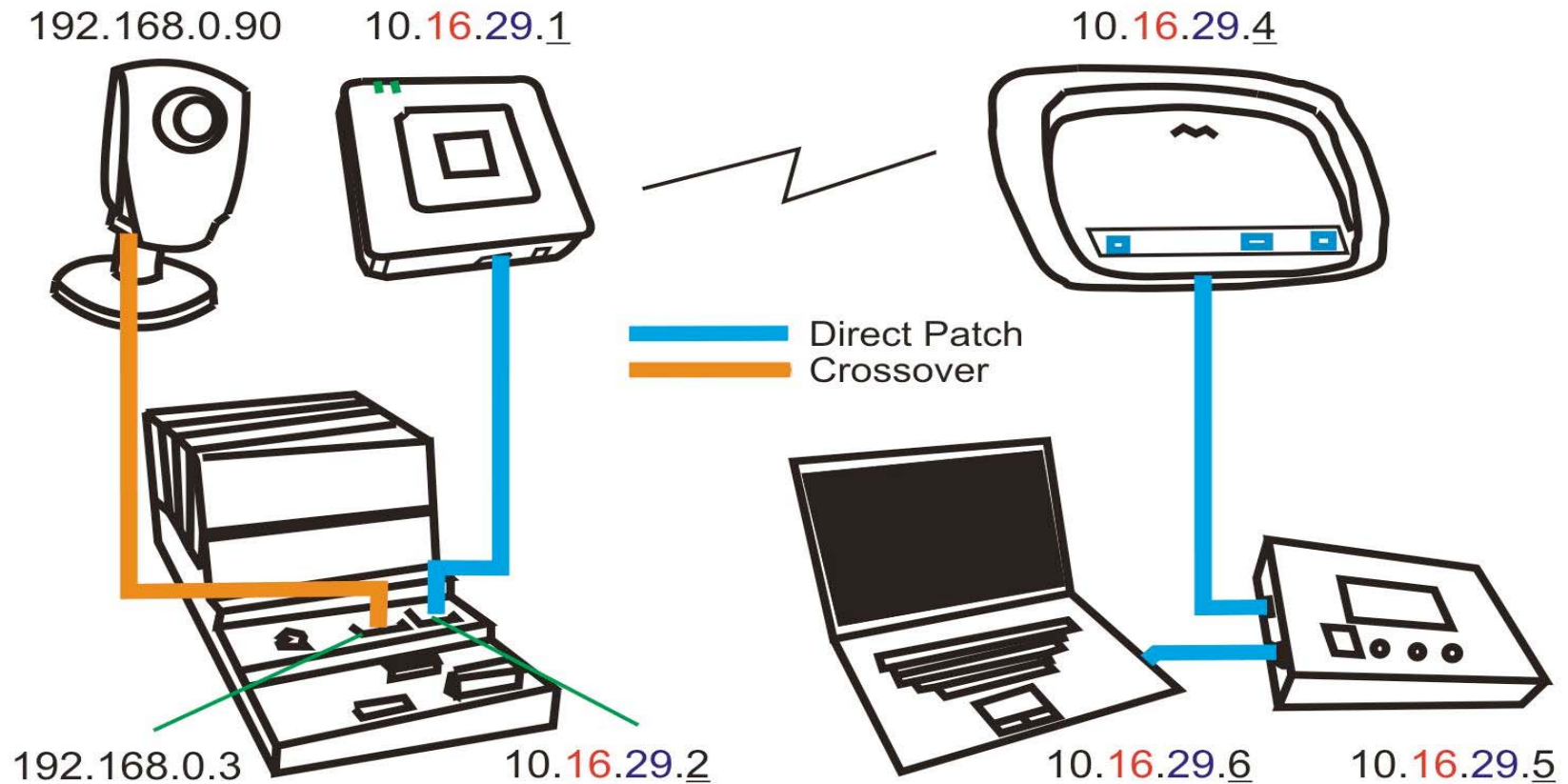
Wireless Radio

- Must install router and bridge discs to laptop in order to configure
- cRIO default IP is 10.0.0.2
- Change laptop IP to 10.0.0.6 for default
- Once the team number is changed, cRIO changes to 10.18.16.2 and laptop must change to 10.18.16.6

Wireless Radio

TEAM 1629

IP Addressing: 10.16.29.X



2009 FRC: Example IP address allocation.

WindRiver

- C/C++ IDE with a FIRST plug-in for use with the cRIO
- Updated version of WPILib with new imaging library and classes for use with C++
- Able to debug to the console with either a serial or wireless connection

Initialization

```
class DefaultRobot : public SimpleRobot
{
  RobotDrive *myRobot; // robot drive system
  DigitalInput *armUpperLimit; // arm upper limit switch
  DigitalInput *armLowerLimit; // arm lower limit switch
  Joystick *rightStick; // joystick 1 (arcade stick or right tank
stick)
  Joystick *leftStick; // joystick 2 (tank left stick)
  Joystick *armStick; // joystick 3 to control arm
  DriverStation *ds; // driver station object
  Gyro *gyro; // gyro sensor
  GearTooth *geartooth; // geartooth sensor
  Compressor *compressor;
  Relay *ramprelease; // solenoid for ramp release
  Relay *rightramp; // right ramp up&down
  Relay *leftramp; // left ramp up&down
```

Operator Control

```
void OperatorControl(void)
{
    Victor armMotor(5);    // create arm motor instance
    float gyro_angle;
    while (IsOperatorControl())
    {
        GetWatchdog().Feed();
        // determine if tank or arcade mode; default with no jumper is for tank drive
        if (ds->GetDigitalIn(ARCADE_MODE) == 0) {
            myRobot->TankDrive(leftStick, rightStick); // drive with tank style
        } else{
            myRobot->ArcadeDrive(rightStick);
        // drive with arcade style (use right stick)
        }
        // Control the movement of the arm using the joystick
        // Use the "Y" value of the arm joystick to control the movement of the arm
        float armStickDirection = armStick->GetY();
    }
}
```

LabVIEW

- GUI-centered IDE
- Blocks of code revolve around “VIs”
- New WPILib methods and classes included as VIs
- Dynamic user interface during runtime
- Core package of FLL RoboLab

Interface

The screenshot displays the LabVIEW software interface for a robot's teleop interface. The main window is titled "Teleop.vi Block Diagram on FRC cRIO Robot Project.lvproj/RT CompactRIO Target". The block diagram is organized into several functional areas:

- Initialization:** An "Init" sub-diagram at the top contains a "0" constant, a "Gear Tooth Sensor Mode" dropdown menu, and three "Counter" blocks labeled "OPEN", "START", and "RESET".
- Control Inputs:** A "Joystick" block is connected to an "Open" block. Below it, there are "RightMotorInverted", "LeftMotorInverted", "Left Motor", and "Right Motor" blocks, each with a numeric control (1 or 2).
- Motor Control:** A "DRIVE" block is connected to the motor outputs. It includes a "Jaguar" dropdown menu and a "OPEN 2WHEEL" label.
- Operation:** An "Operation" sub-diagram on the left contains an "Enum" block.
- Initialization Note:** A text box at the bottom of the diagram reads "Initialize to read joystick 1 and set up motor for Robot".

On the right side of the interface, a "Functions" palette is open, showing a search bar and a list of categories: Programming, FIRST Vision, Mathematics, PID, WPI Robotics Library, Connectivity, Signal Analysis, User Libraries, and Select a VI... The Windows taskbar at the bottom shows the time as 7:24 PM and lists several open applications, including "Application Devel...", "Project Explorer - F...", "Teleop.vi Front Pa...", "Teleop.vi Block Dia...", and "Untitled - Paint".

Interface

The screenshot displays the LabVIEW interface for a robot drive system. The main window is titled "Teleop.vi Block Diagram on FRC cRIO Robot Project.lvproj/RT CompactRIO Target". The block diagram is enclosed in a "Default" execution environment. It features several key components: a "GET" block, a "Value" block containing the number "123" and a "500" constant, a "Joystick" block with "axis 2 (y)" and "axis 1 (x)" outputs, a "Speed" block with "123" and "0" values, and a "DRIVE DRIVE" block. A text box at the bottom of the diagram reads "On Drive Mode read Joystick X and Y value and drive Robot". A "Functions" palette is open on the right side, showing various programming and robotics-related functions. The Windows taskbar at the bottom shows the application development environment, project explorer, and the front panel and block diagram windows.

WindRiver vs LabVIEW

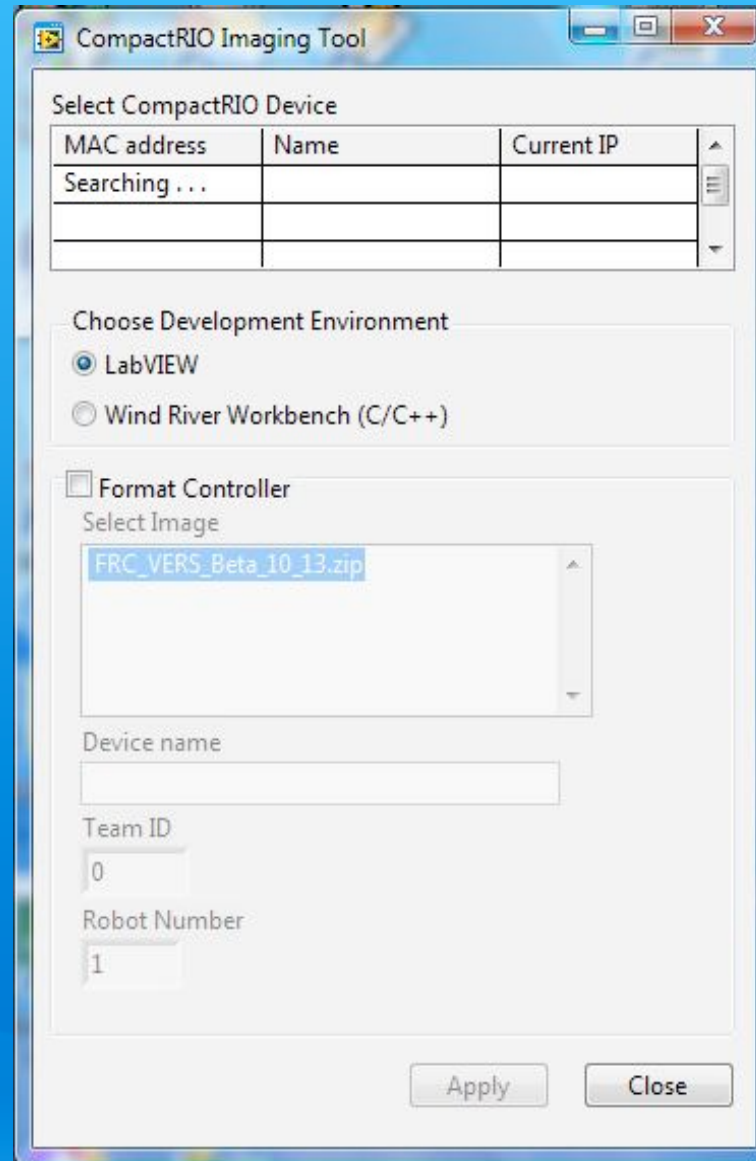
- WindRiver
 - Easier to adapt to from C and MPLab
 - More simplistic manipulation of code and data
 - Debugging hasn't changed much from MPLab
 - Lacks built-in functionality of VIs in LabVIEW
 - Compiler errors are often vague or imprecise

WindRiver vs LabVIEW

- LabVIEW
 - GUI is often more inviting to newer programmers
 - Specifically designed for machine/hardware manipulation
 - WPI Lib API is built-in via VIs / "Safety net"
 - Can be difficult to build complex program in a new system
 - Outputting data can be cumbersome
 - "Wiring" and maintaining format can be tedious

cRIO Imaging Tool

- Allows user to switch kernel between C++ and LabVIEW





Any Questions?
Comments? Concerns?