

NI C Series CompactRIO Modules

Third-Party CompactRIO Modules

cRIO

2009 Control System Workshop

Edina Robotics FIRST Team 1316
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

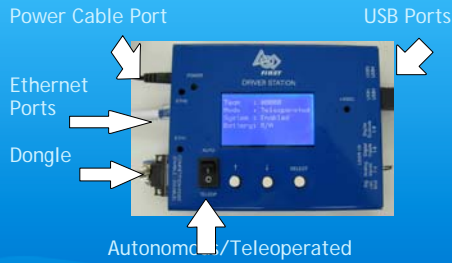
Digital Side Car (DSC)

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

Power Distribution Block

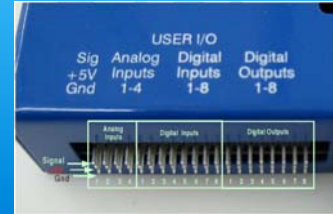
- Power center
- Replaces old power distribution block and fuse blocks
- Contains room for circuit breakers
- Everything that requires power is connected to the Power Distribution Block

Driver's Station



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

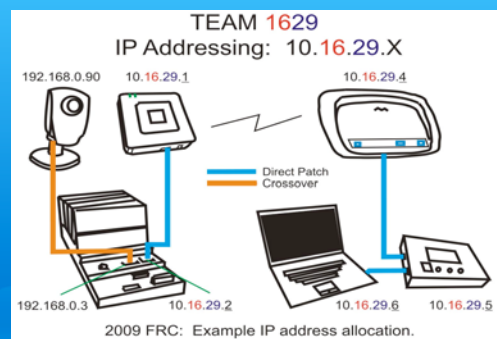
- WRT600N Dual band N Router attaches to Driver's Station
 - Broadcasts both 5.8Ghz (N) and 2.4Ghz (G and B)
- WGA600N Dual band gaming bridge attaches to cRIO



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



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 *lefttrap; // left ramp up&down
}
```

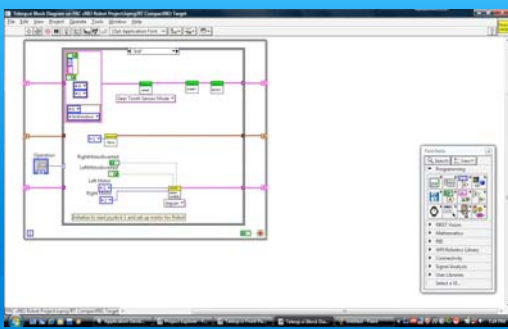
Operator Control

```
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();
  }
}
```

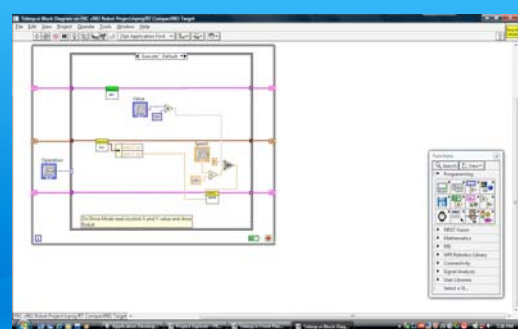
LabVIEW

- 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



Interface



WindRiver vs LabVIEW

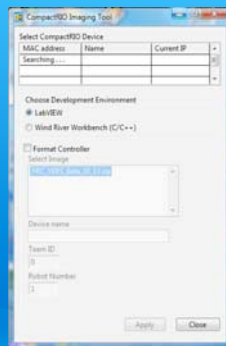
- WindRiver
 - Easier to adapt to from C and MPLab
 - More simplistic manipulation of code and data
 - Debugging hasn't changed much from MPLab
 - Lacks built-in functionality of VIs in LabVIEW
 - Compiler errors are often vague or imprecise

WindRiver vs LabVIEW

- LabVIEW
 - GUI is often more inviting to newer programmers
 - Specifically designed for machine/hardware manipulation
 - WPILib 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?