

# LabVIEW for FRC

## Session 2: FRC Robot Control using LabVIEW

Brian Moorhead – FRC 1250

[moorhead.brian@gmail.com](mailto:moorhead.brian@gmail.com)

# Objectives

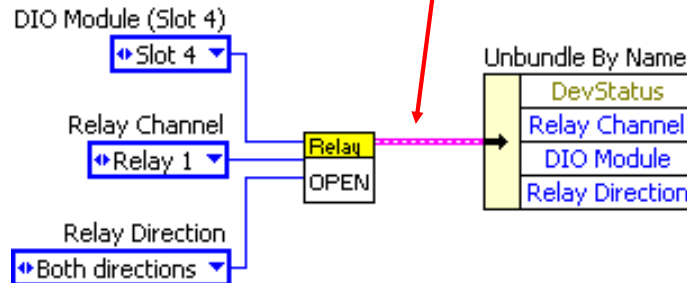
- Introduce LabVIEW users to the FRC program architecture
- Introduce LabVIEW users to the Team Ford FIRST Default Code

# WPI Robotics Library Basics

- FRC program architecture
- Driver station
  - Joystick
  - Digital inputs
  - Analog inputs
- Robot Drive
- Actuators
  - Motor control
  - Servo
  - Relay
  - Solenoid
  - Compressor
- IO
  - Digital input
  - Analog input
- Utilities
  - Watchdog

# FRC Program Architecture

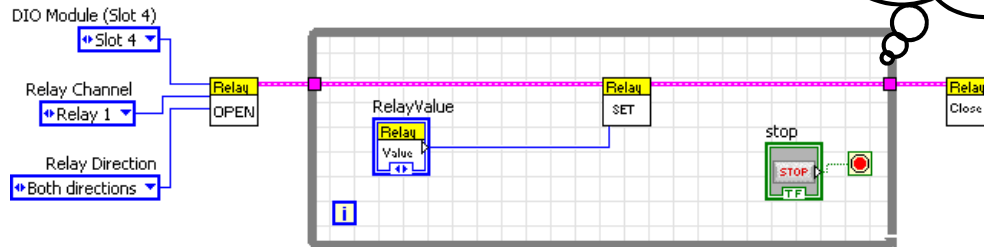
- Everything is a **Device**
- Devices use clusters called **DevRef** to pass data around



# FRC Program Architecture

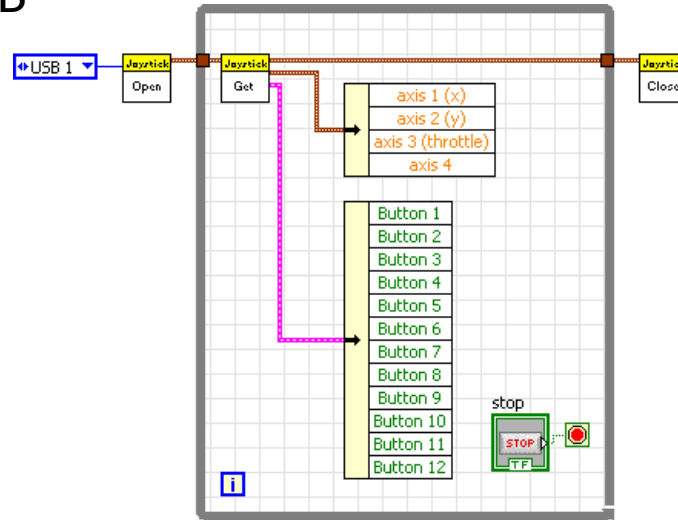
- Use the **Open** block to create and define a new device
- Use the **Get** and **Set** blocks to use the device
- Finally, use the **Close** block to dispose of the reference when you're done

Use a **While** loop to keep it running



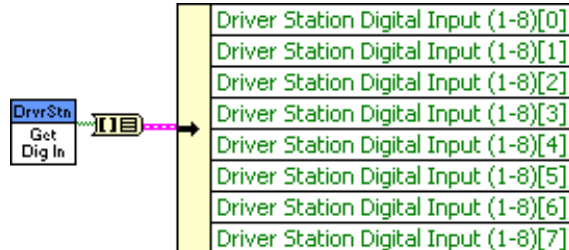
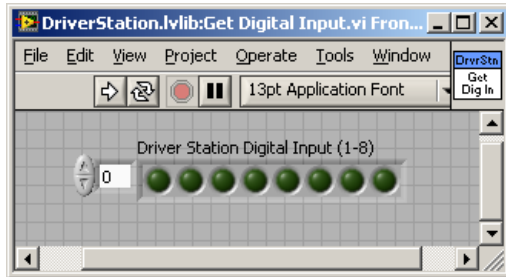
# Driver Station: Joystick

- Initialize by selecting USB channel
- Unbundle **Axes** and **Buttons** clusters to get driver inputs
- There's also a **Get Axis** VI if you only need one axis...



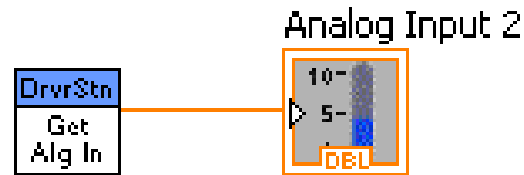
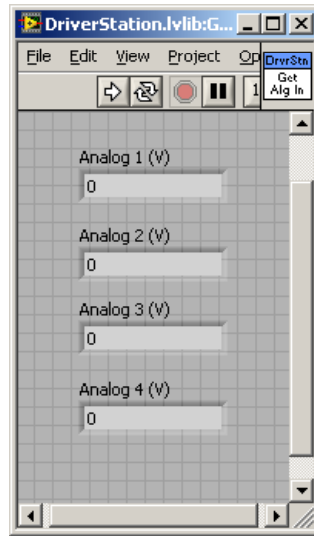
# Driver Station: Get Dig In

- Doesn't require a DevRef because there's only ever one driver station (ok, seems I told a fib earlier)
- Returns an array containing the 8 Driver Station digital inputs (zero-indexed)
- Use the **Array To Cluster** and **Unbundle By Name** functions to get an individual input



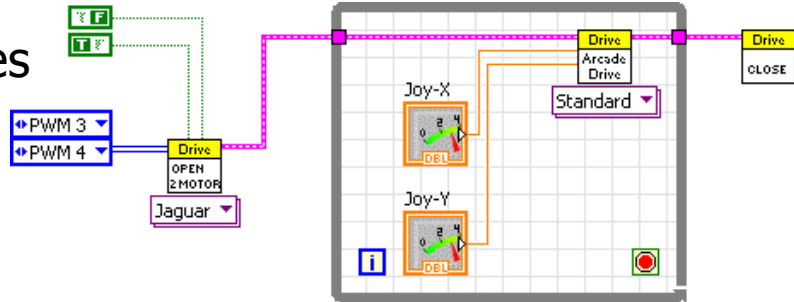
# Driver Station: Get Alg In

- Also doesn't require a DevRef
- Returns the four Driver Station analog input channels



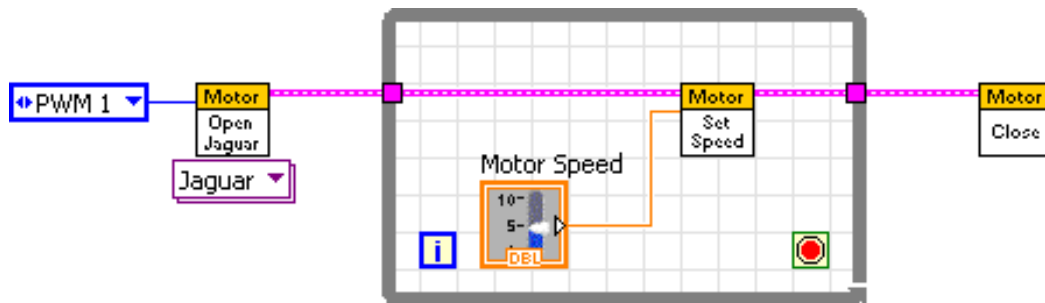
# Robot Drive

- Controls two PWM outputs based on -1.0 to 1.0 inputs (usually from joystick)
- **Arcade Drive, Tank Drive, and Holonomic Drive** blocks available
- Feature for inverting one of the PWM drives in software
- Other cool features



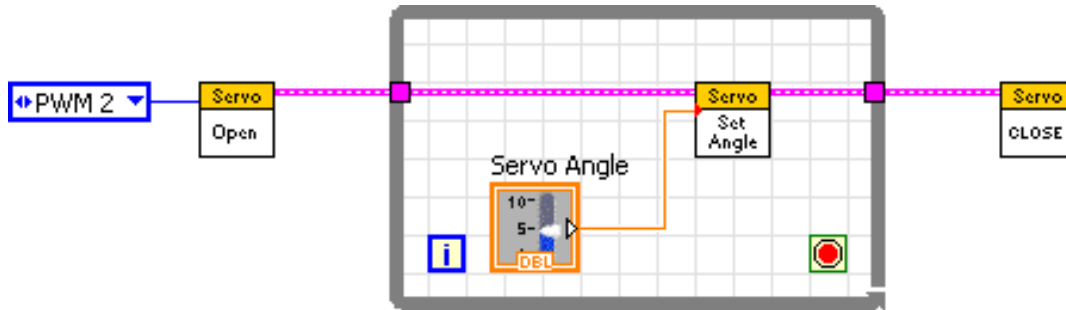
# Actuators: Motor Control

- Used to control a Victor or Jaguar speed controller
- Speed command range at  $-1.0$  to  $1.0$
- There's also a **Get Speed** block available, but don't let the name fool you... it returns to commanded speed, not the actual speed



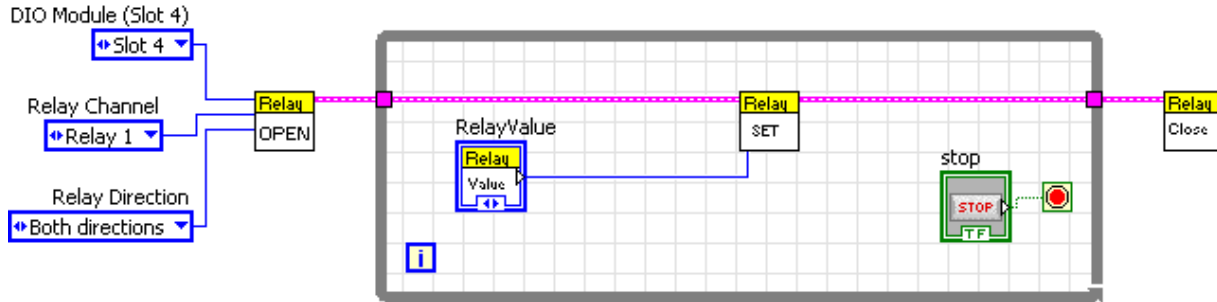
# Actuators: Servo

- Used to control a Servo (duh!)
- Allows user to specify an Angle in degrees or a position from  $-1.0$  to  $1.0$
- Like the motor control, there are also **Get Angle** and **Get Position** blocks



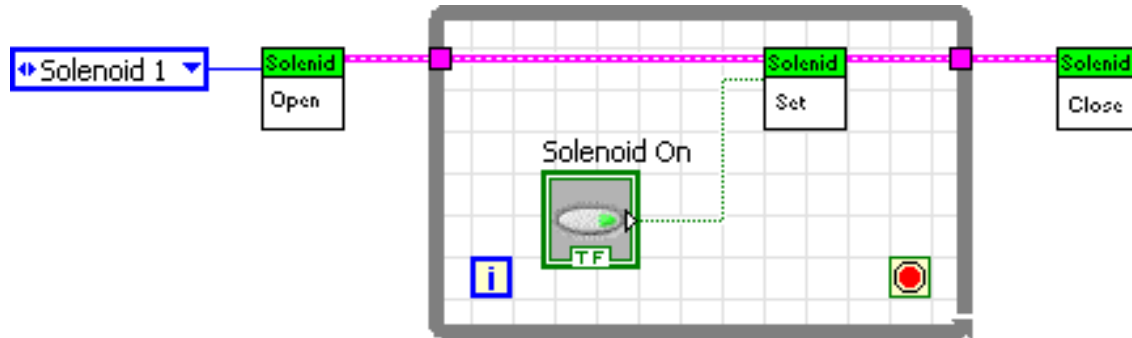
# Actuators: Relay

- Used to control a Spike relay
- Can be set to **Off**, **On** (for forward only mode), **Forward** or **Reverse**
- There's also a **Get** block



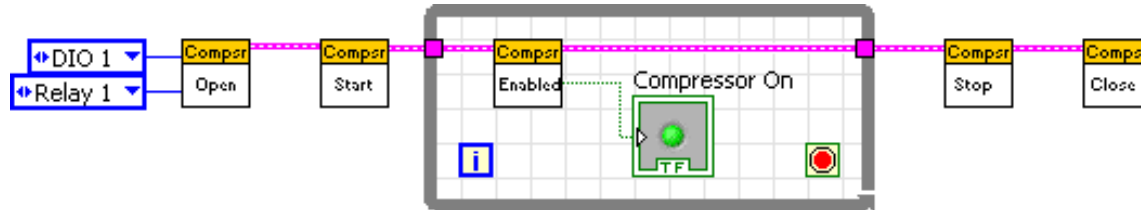
# Actuators: Solenoid

- Used to control a solenoid
- Can be set to **False** or **True**
- There's also a **Get** block



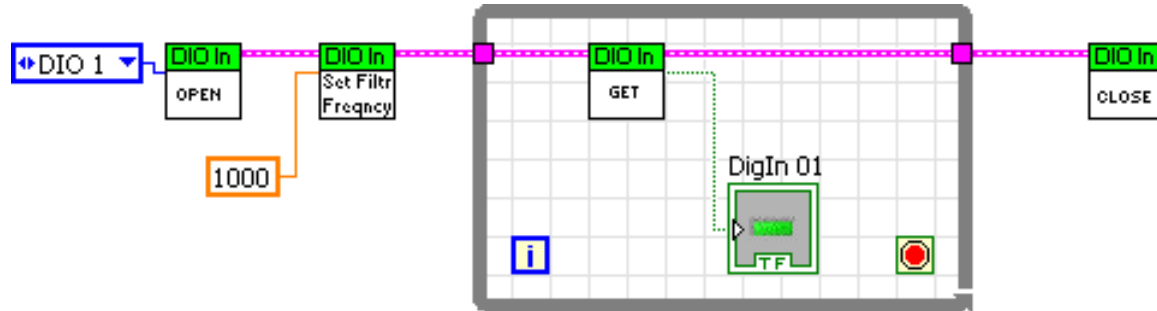
# Actuators: Compressor

- Required inputs to configure
  - Digital input for pressure switch
  - Relay output for compressor Spike
- **Start** and **Stop** determine if compressor is allowed to run
- Example starts the compressor and turns on as necessary based on pressure switch



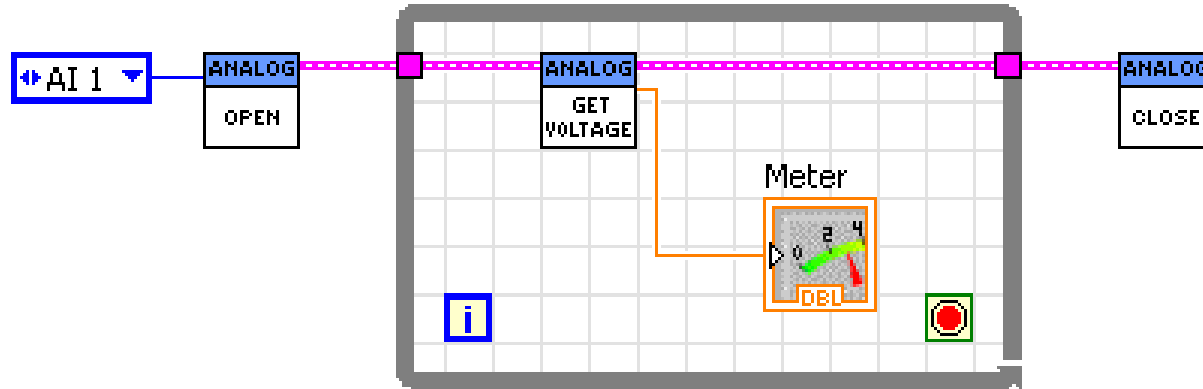
# IO: Digital Input

- Useful for implementing limit switches
- Optional filtering available to avoid issues with switch bounce



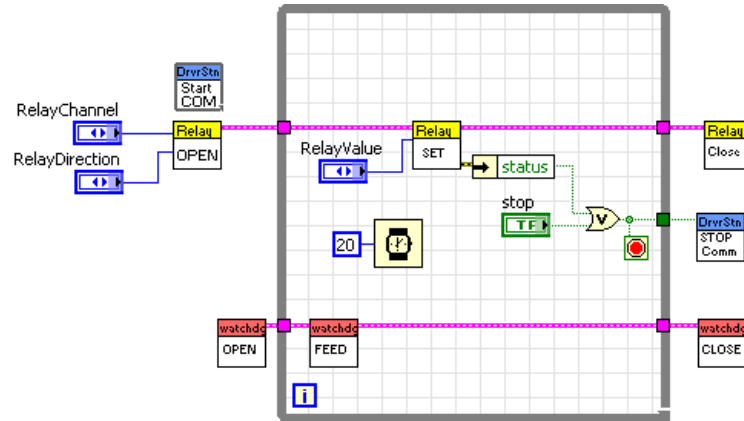
# IO: Analog Input

- Useful for implementing position sensors
- Accumulator and averaging blocks also available



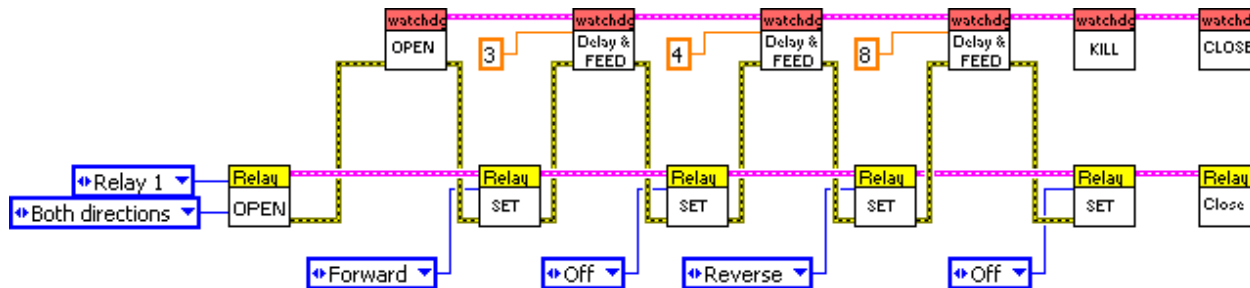
# Utilities: Watchdog

- The watchdog keeps an eye on the clock to make sure your functions don't take too long
- The watchdog must be fed periodically, otherwise it will halt the robot code, disabling the robot



# Utilities: Watchdog

- Use the **Watchdog Delay & Feed** block in autonomous mode to implement simple timed actions
- Maintains the present robot state for the specified number of seconds while keeping the watchdog fed
- Use the **error in** and **error out** terminals to determine order of execution



# Features Not Covered

- Driver station
  - Set user data
  - Set digital output
- IO
  - Digital output
  - PWM
  - Analog trigger
- Camera
- Utilities
  - Interrupts
  - DMA
- Others

# LabVIEW for FRC

End of Session 2

Brian Moorhead – FRC 1250  
moorhead.brian@gmail.com