Getting Started with Sensors

Contents

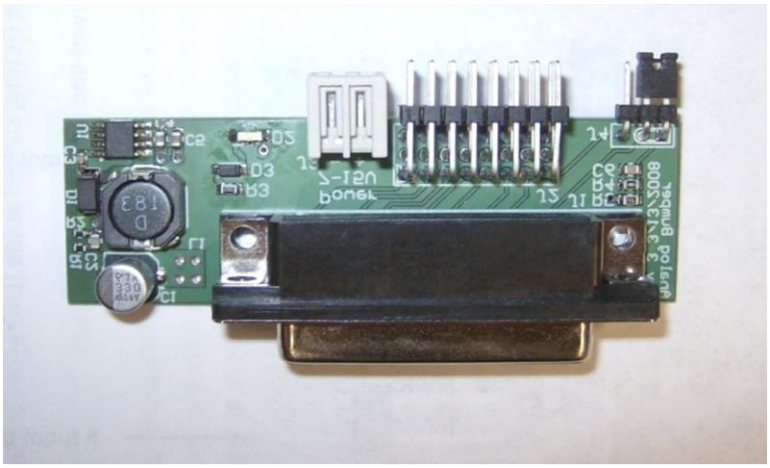
Introduction

In this tutorial, we will explore how to acquire data from sensors using a cRIO-FRC II and LabVIEW 2013 (installed from the FIRST Robotics Software 2014 installation disc). We will give detailed instructions on using an accelerometer followed by a brief overview of other sensor types including a gyro, an ultrasonic sensor, a counter, and an encoder.

> **Note:** This tutorial was created using beta software and hardware. The final product may differ slightly from the products shown. The bare boards used in this tutorial will have enclosures in the final version.

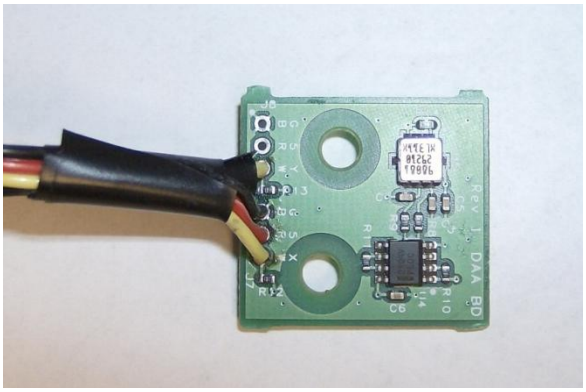
Configuring the Hardware

Sensors are electrical devices that return a voltage corresponding to a physical property. In the case of an accelerometer, the voltage it returns is proportional to the acceleration the sensor is experiencing. The physical connections for the gyro and ultrasonic sensors are the same as the accelerometer. You will be using the analog breakout to interface all of your analog sensors with the NI 9201 module in your cRIO-FRC chassis.

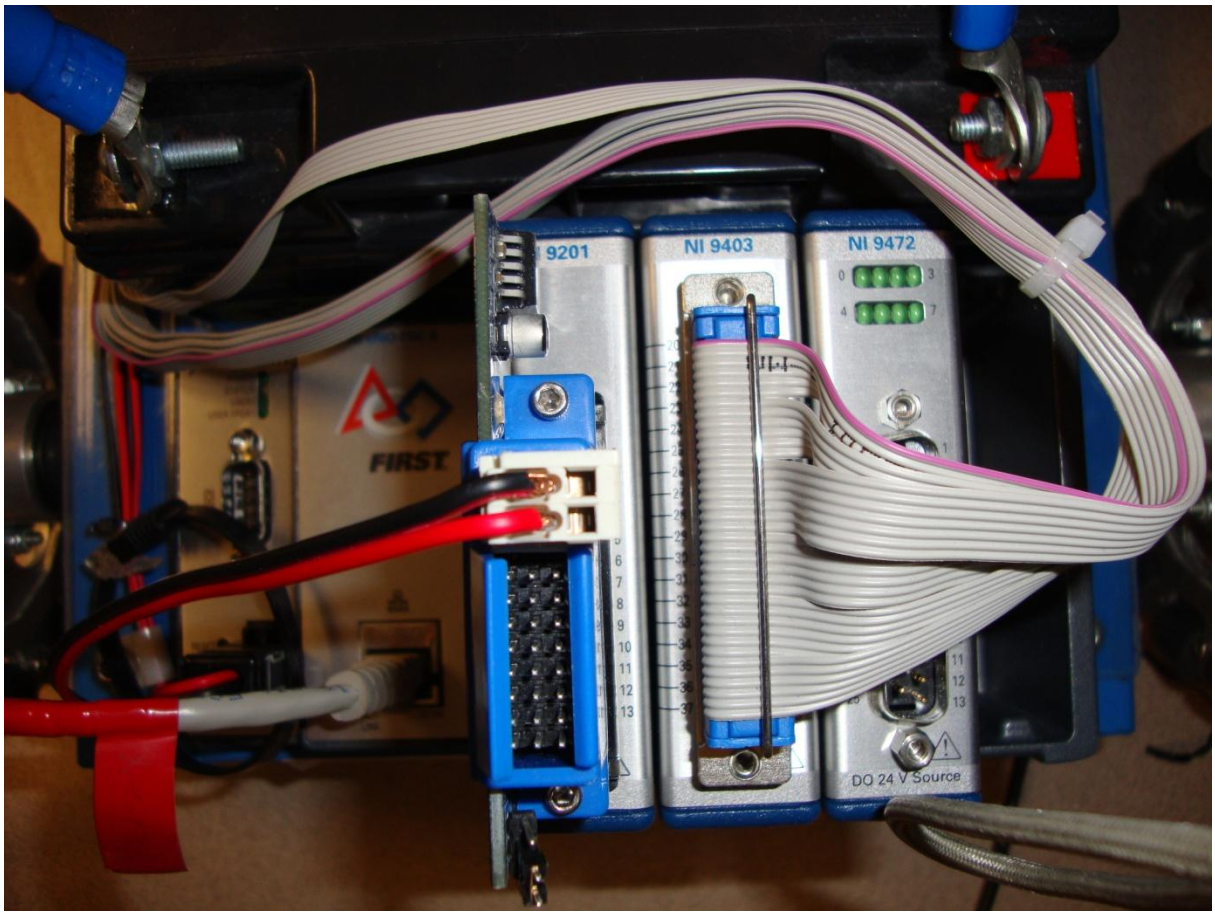


Connect the analog breakout to the NI 9201 in Slot 1. The 25-pin D-sub connector will connect the analog breakout to the NI 9201, while the 24 exposed pins (three for each channel) and the white 2-pin connector will connect to your sensors and power supply, respectively. The three pins on the edge of the breakout are use to designate if the analog breakout is to use the 12 volt connector or analog input 8 as its power source—you can connect power to either one, but make sure you set the jumper correctly. In this demonstration we will be using the 12 volt connector so we will jumper the select common pin and the select 12V pin.

Once the analog breakout is powered you are ready to connect your sensors. We will be using an accelerometer similar to the one in the 2014 FRC kit of parts.



In general sensors require power (often called excitation) and output a voltage that is proportional to a physical phenomon. This sensor is a two axis accelerometer meaning that it can measure acceleration in two directions—this is achieved by putting two separate accelerometers on the same sensor with different orientations. While most sensors would have three wires: power, ground, and signal out, this sensor has four wires since it is really two accelerometers in one. Both accelerometers share the power and ground wires and each has their own signal out wires. In order to measure the acceleration in both directions we would need to use two analog input channels. However, in this demo we will only be using a single analog input to measure acceleration in one direction so we will not be connecting the second signal out wire.

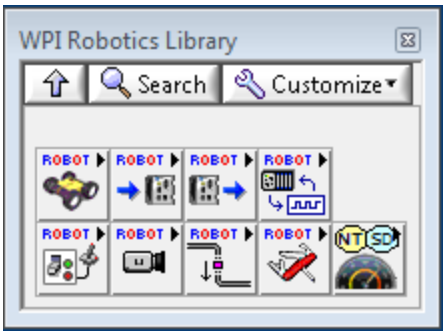


Within each set of three exposed pins for each analog input channel, the pin farthest from the board is used for the analog input signal, the center pin is used for the +5V power supply (for excitation), and the pin closest to the board is the ground pin. Connect to accelerometer to the breakout board according to this pin configuration. This is the last connection required to take measurements from the sensor so now we will examine the software side of things.

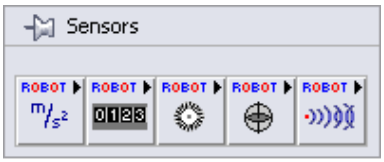
Exploring the Sensor API

In this tutorial we will assume that you are already familiar with opening a new project and VI in LabVIEW FRC Edition 2014. If you would like details on how to open a new project, please refer to [FRC 2014] FRC Robot Framework Tutorial.

Once you have a new VI open, access the Functions palette by right-clicking on the white space in the Block Diagram. Here you will find the WPI Robotics Library palette where you can navigate to the Sensors palette.

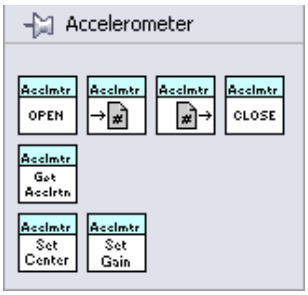


Click on the Sensors icon in order to see the different sensor palettes available to you. Each palette contains VIs that allow you to gather data from different types of sensors.



For example, the VIs in the Accelerometer palette allow you to measure an acceleration value as well as set different properties for the measurement. These properties, SetCenterVolt and SetGain, can change the how acceleration measurements are taken. The SetCenterVolt VI will calibrate the accelerometer's zero point and the SetGain VI will set how sensitive your graph will be to a change in voltage.

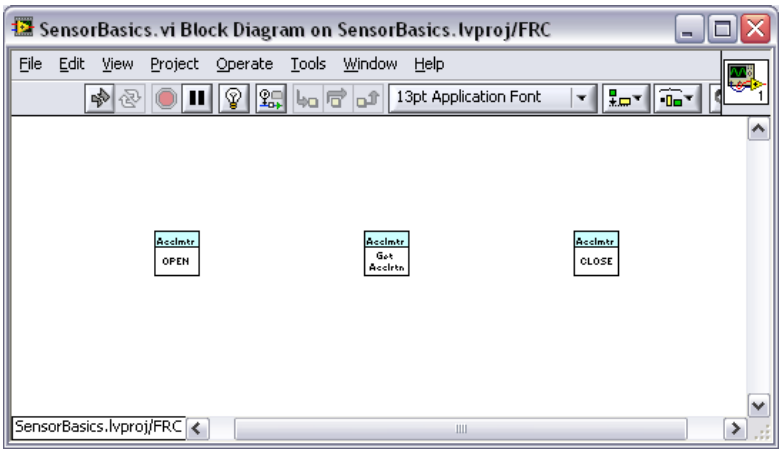
To display the Accelerometer VIs, select the Accelerometer palette.



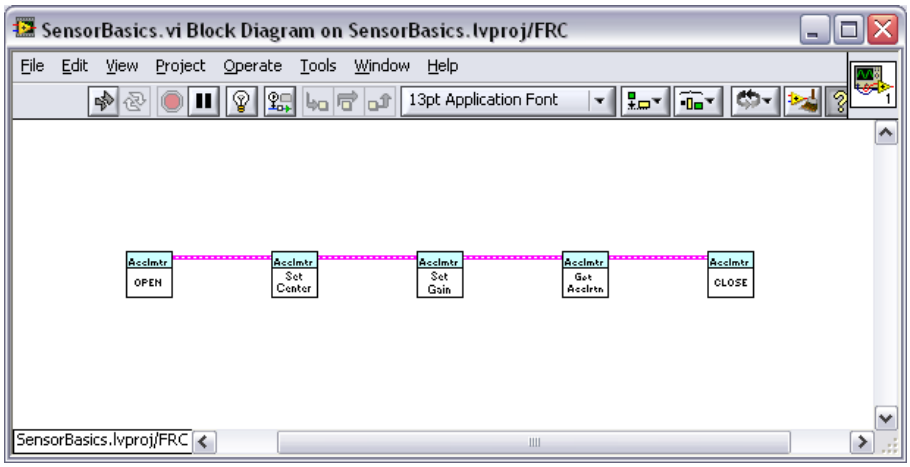
The Accelerometer VIs use the same **Open»Get»Close** paradigm that many of the other Robotics APIs use.

Developing an Example

To begin writing your code, drag and drop the Open, GetAcceleration, and Close VIs onto your Block Diagram. The Open VI opens communication between the cRIO and the accelerometer. The GetAcceleration VI returns the voltage data from the accelerometer. The Close VI closes the communication between the cRIO and the accelerometer.



Next we need to set the center voltage and the gain for the accelerometer in order to calibrate the accelerometer and receive useful data. On your Block Diagram, make room between the Open VI and the GetAcceleration VI and then right-click on a blank space of the Block Diagram to open the Functions Palette. Navigate to the Accelerometer palette and place a SetCenterVoltage VI and a SetGain VI in between the Open VI and GetAcceleration VI. You can now wire together each of the VIs as shown below.

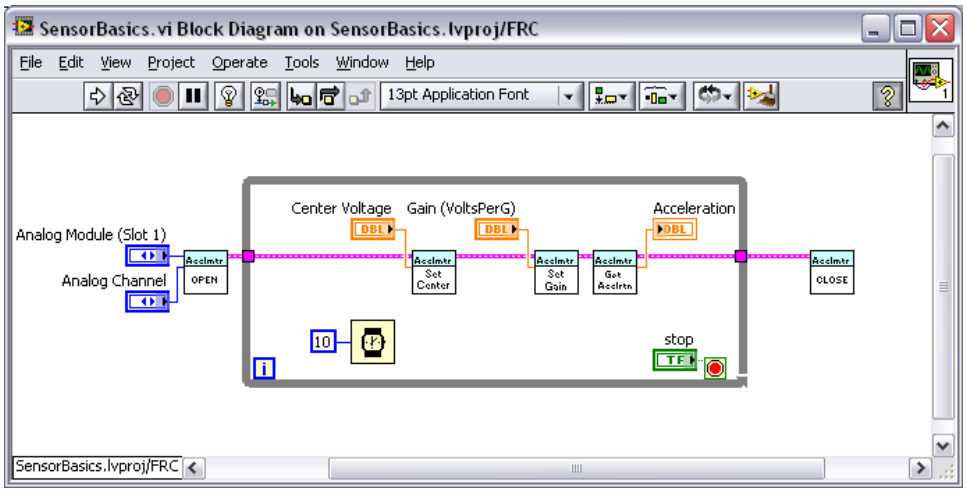


To be able to set the Center Voltage for the accelerometer, right-click on the Center Voltage input terminal of the SetCenter VI and navigate to **Create»Control** – this will place the corresponding control on the Front Panel. Follow the same steps to create a control for the Gain input terminal of the SetGain VI. To display the data collected by the sensor on the Front Panel, right-click on the Acceleration output of the GetAcceleration VI and navigate to **Create»Indicator**.

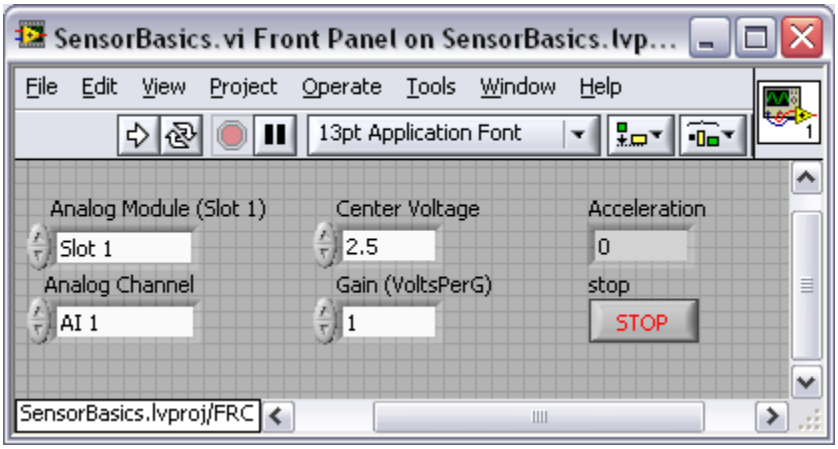
Next we will direct the VIs to the appropriate module and channel to which the accelerometer is connected. Right-click on the Analog Module (Slot 1) input terminal on the Open VI and select **Create»Control**. Follow the same steps to create a control for the Analog Channel input terminal. This will place controls on the Front Panel that will allow you to select the module and analog channel for your accelerometer.

Next, add a While Loop around the SetCenter, SetGain and GetAcceleration VIs so the VI continually updates with new data. To do this, right-click on the Block Diagram to bring up the Functions palette and navigate to the **Functions»Programming»Structures** palette and select a While Loop. Click to the top-left of the SetCenter VI and drag the loop to include all VIs except the Open and Close VIs. References for the sensors only need to be open and closed once, so they do not need to be inside the loop.

Right-click on the condition terminal in the bottom-right of the While Loop and select **Create»Control** to place a stop button on the Front Panel. Finally, add a Wait VI to the While Loop so that the code executes every 10 milliseconds. The Wait function is located in the **Functions»Programming»Timing** palette. Right-click on the Milliseconds to Wait input terminal and select **Create»Constant**. Set this constant to 10.



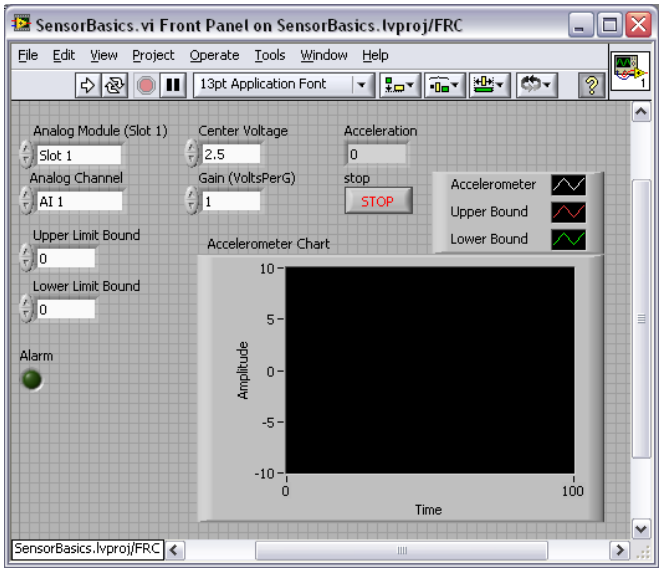
Switch to the Front Panel and set the controls (Analog Module (Slot 1), Analog Channel, Center Voltage, Gain (VoltsPerG)) accordingly. You can switch to the Front Panel by going to the Window menu and selecting Show Front Panel. You can also move and resize the items on the Front Panel as you see fit. Now you are ready to collect data from your accelerometer. Run the program to see the data from the accelerometer displayed on your Front Panel.



Customizing a LabVIEW Application

LabVIEW allows you to customize a program for your own personal needs. For instance, you can graph your data on the Front Panel and set different parameters for your application. For this example we will graph the sensor data on the Front Panel as well as turn an LED on or off depending on upper and lower limits that we determine.

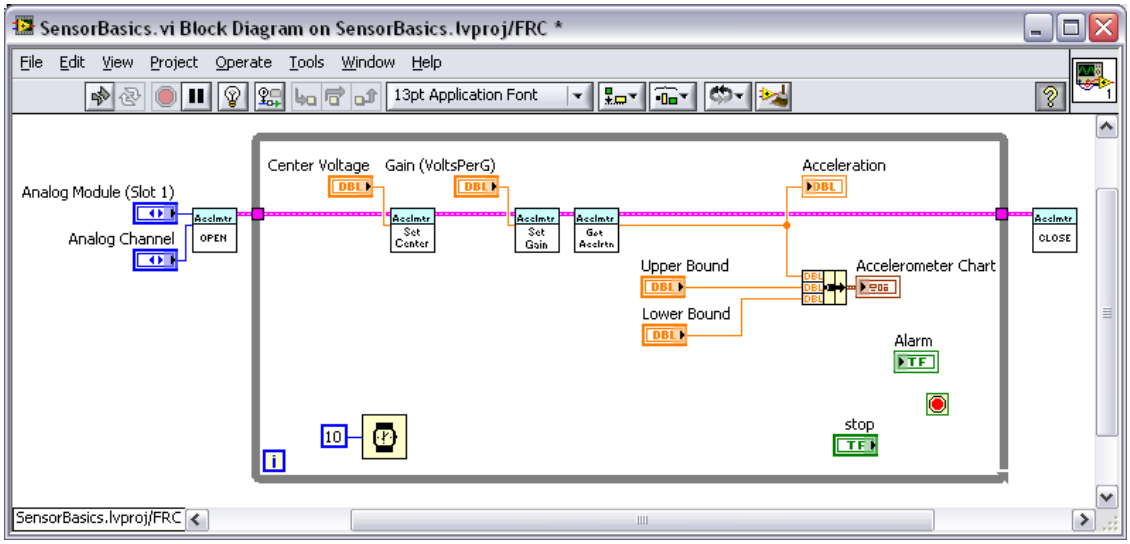
Start by switching to the Front Panel if it is not already showing. You should already see your Stop button, Analog Module (Slot 1) control, Analog Channel control, Center Voltage control, Gain control, and Accelerometer indicator. We will now add two numeric controls for the upper and lower limits, a round LED, and a chart to the Front Panel. Right-click on the Front Panel to bring up the Controls palette. Navigate to the **Controls»Modern»Graph** palette and place a Waveform Chart on the Front Panel. Next, bring up the Controls palette again and navigate to the **Controls»Modern»Numeric** and place a Numeric Control on the Front Panel. Repeat this step to add a second Numeric Control. Change the names of the controls to Upper Limit Bound and Lower Limit Bound. Finally, navigate to the **Controls»Modern»Boolean** palette and place a Round LED on the Front Panel. Change the name of the Boolean to Alarm. Feel free to move and resize and rename the controls on the Front Panel as you wish. Your Front Panel should resemble the figure below.



Switch back to the Block Diagram (**Window»Show Block Diagram**) where you should see the different controls you added to your Front Panel. Resize the While Loop to make room for additional code and move all the new controls inside of the While Loop.

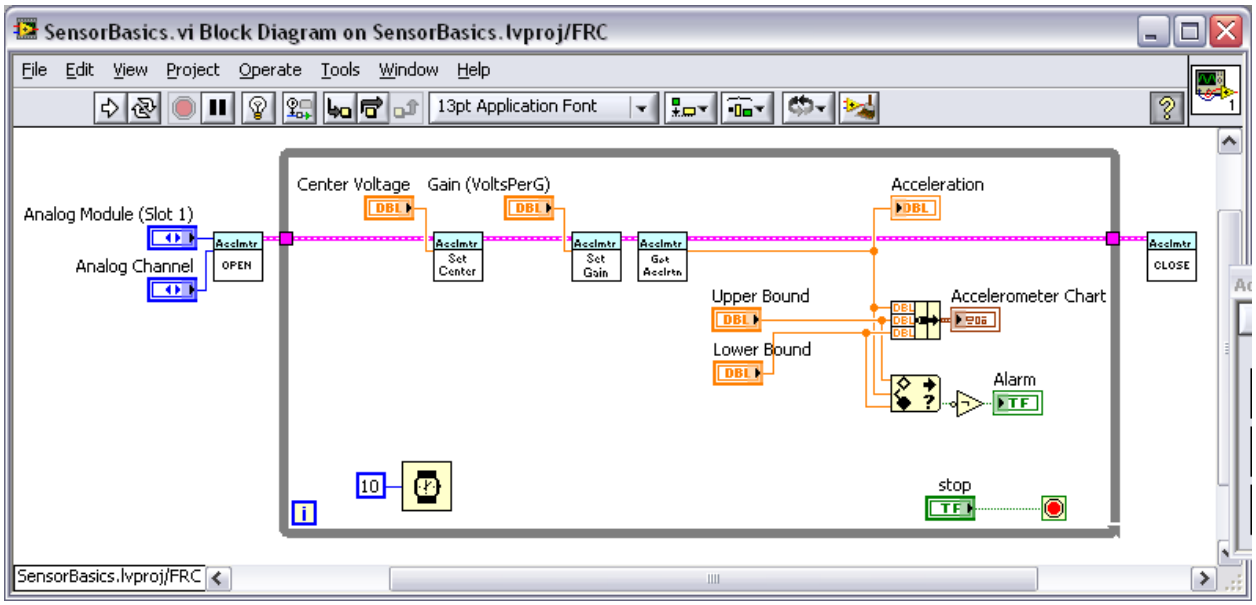
We want our new Chart to display not only the accelerometer data, but also the upper and lower bounds that we set on the Front Panel. To do this, use a Bundle function to bundle each of these values into a cluster to be passed into the Waveform Chart. Right-click on the Block Diagram to bring up the Functions palette and then navigate to the **Functions»Programming»Cluster, Class, & Variant** palette and place a Bundle function on the Block Diagram to the right of your Upper and Lower Bound controls. Hover your mouse over the bottom of the Bundle function and expand the Bundle to three inputs. Next wire the Acceleration output of the GetAcceleration VI to the first input, the Upper Limit Bound to the

second input, and the Lower Limit Bound to the third slot input. Finally, wire the output of the Bundle to the Waveform Chart.



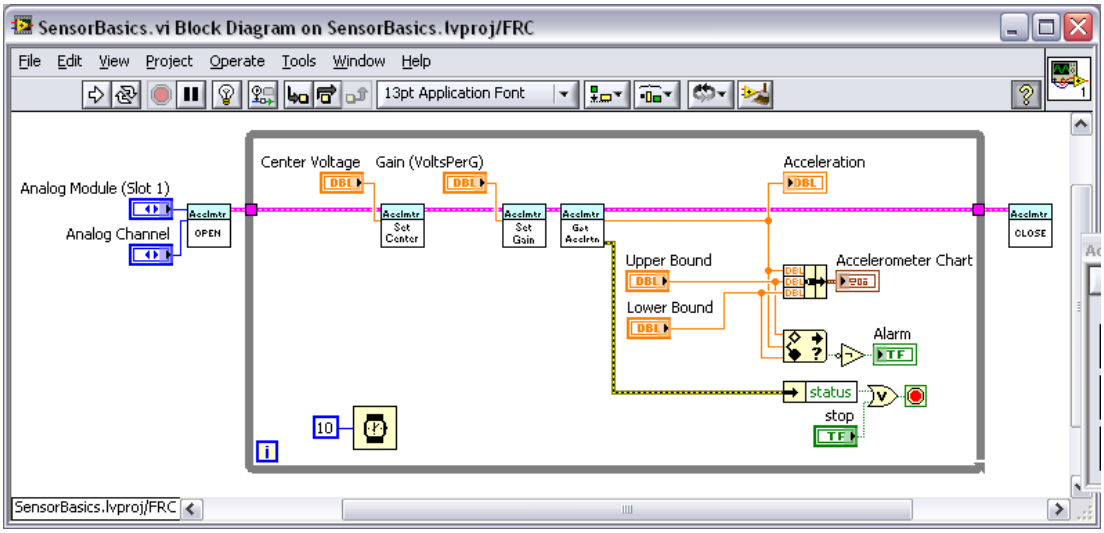
The last thing we want to do is set our alarm to true if the value of the data from our sensor is below the lower bound or above the upper bound. We can use comparison VIs to determine if either of these events happen. Right-click on the Block Diagram and navigate to the **Functions»Programming»Comparison** palette and place an In Range and Coerce VI on the Block Diagram.

Wire the Acceleration output from the GetAcceleration VI to the x input of the In Range and Coerce VI. Next, wire the Upper Limit Bound to the upper limit input and the Lower Limit Bound to the lower limit input of the In Range and Coerce VI. This will send a True output if Acceleration value is within this range. We want to turn the Alarm LED on when the Acceleration value is outside of the specified range. To accomplish this, we need to use a Not function to compute the logical negation of the input. Right-click on the Block Diagram and navigate to the **Functions»Programming»Boolean** palette and place a Not function to the right of the In Range and Coerce VI. Wire the In Range? output of the In Range and Coerce VI to the input of the Not function, and then the output of the Not function to the input of the Alarm LED.

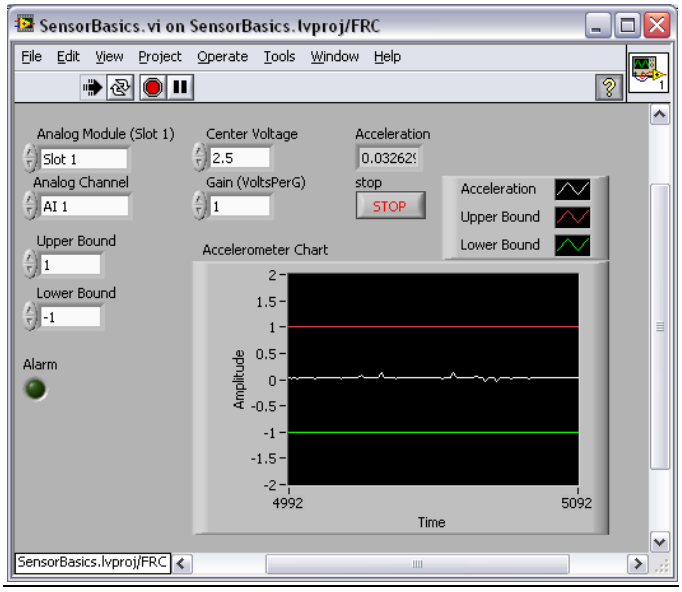


We want our program to stop when we click on the Stop button, but we also want it to stop if an error occurs during the program. The Error Out output for each of the Accelerometer VIs is a cluster of three different values: Status, Code, and Source. The Status is a boolean value that will signal True when an error has occurred. To get access to the Status value, we need to "unbundle" the Error Out cluster. Right-click on the Block Diagram and navigate to the **Functions»Programming»Cluster, Class, & Variant** palette and place an Unbundle By Name function on the Block Diagram. Wire the Error Out output of the GetAcceleration VI to the input of the Unbundle By Name. By default, the Status value should appear in the Unbundle By Name. If Status does not appear, left-click on the Unbundle By Name and select Status from the pop-up menu.

Since we want to stop the While Loop when the Stop button or the Status value is true, we need to use an Or function to evaluate the signals. Right-click on the Block Diagram and navigate to the **Functions»Programming»Boolean** palette and place an Or function to the right of the Unbundle By Name. Disconnect the Stop button from the condition terminal by clicking on the wire connecting the two and pressing the Delete key or right-clicking on the wire and selecting Delete Wire Branch. Then wire the output of the Unbundle By Name and the Stop button to the inputs of the Or function and the output of the Or function to the input of the condition terminal.



You now have a complete program that will read values from your accelerometer and display that information to a graph on your Front Panel and alert you if the value exceeds the specified range. Run the VI from within your project that targets the cRIO-FRC. You should see the values on your chart and indicator change depending on how much you shake the accelerometer.



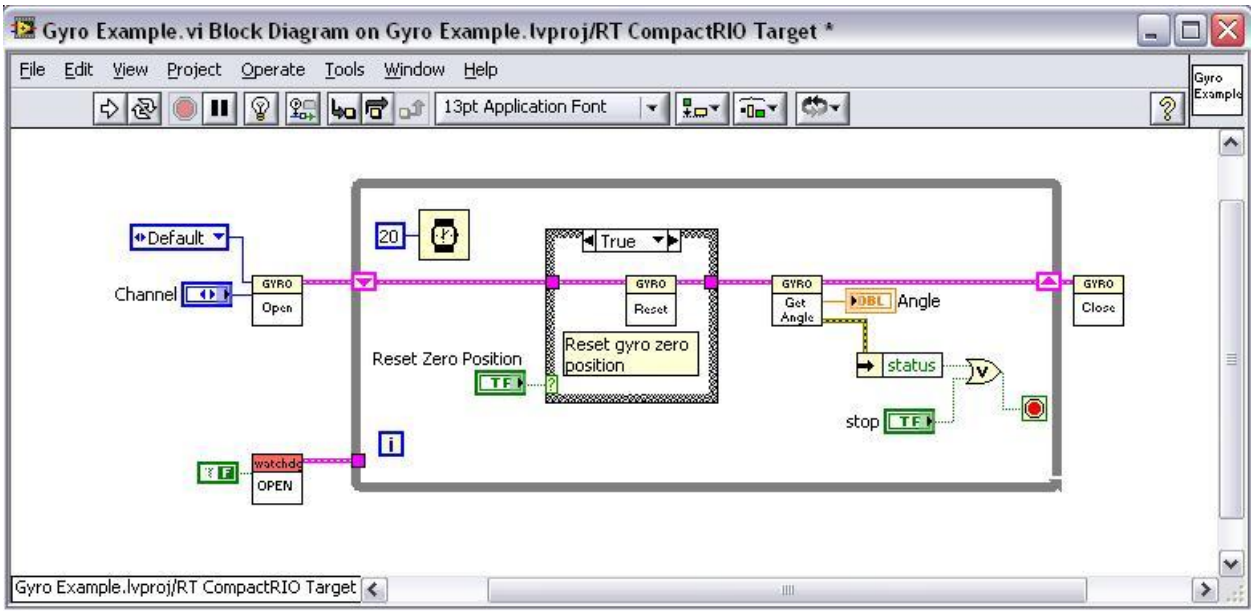
Using Other Types of Sensors

In addition to accelerometers, WPI Robotics Library provides an API for other types of sensors. For each sensor there is a palette of VIs that make extensive use of the **Open»Get»Close** and **Open»Set»Close** paradigm. This allows you to make use of your experience with different sensor typess.

For accelerometers, gyroscopes, and ultrasonic distance sensors, changing between measurement types can be as simple as plugging a new sensor into the analog breakout board and using the corresponding

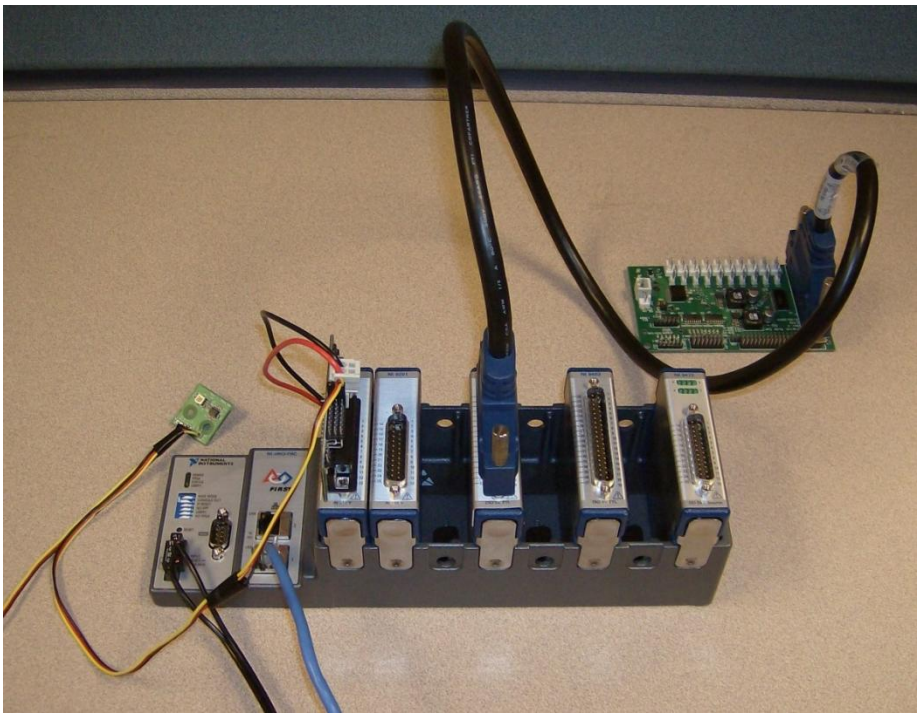
sensor API. Each of these sensors use an analog signal that returns a voltage proportional to a physical value.

Examples for these types of sensors, such as the gyroscope example below, can be found by going to Getting Started window and clicking the Find Examples link in the bottom right corner. You can access the Getting Started window from any LabVIEW window by going to **View»Getting Started Window…**. Each example will show how to use the sensor's corresponding API in the WPI Robotics Library.

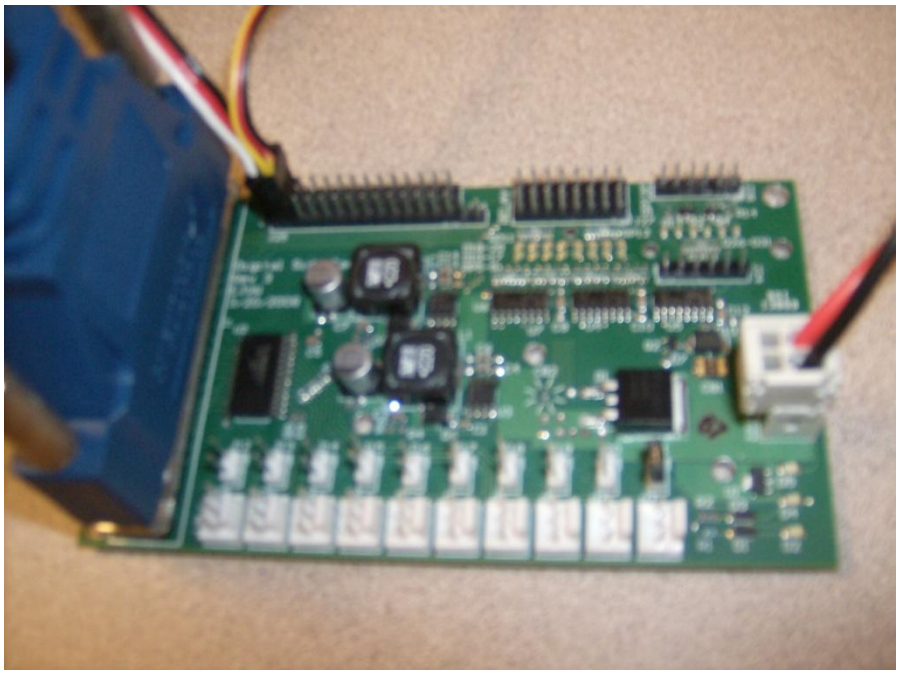


For the remaining sensor types, such as encoder and counters, the programming follows a similar procedure, but the hardware setup differs.

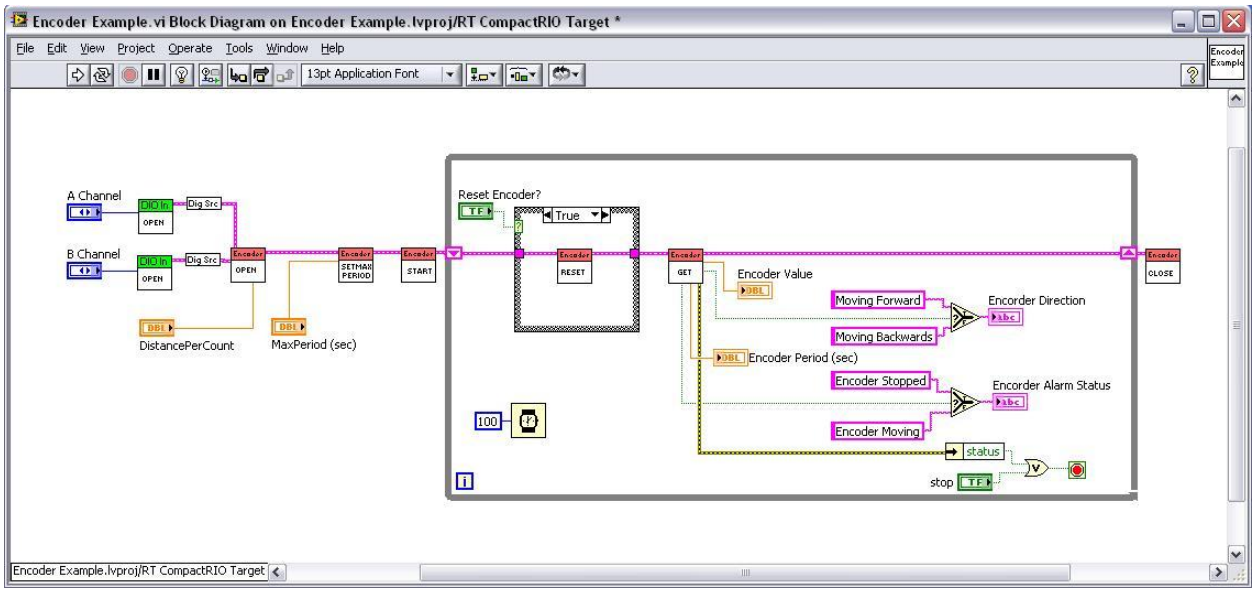
Encoders and counters are digital sensors. Rather than returning a voltage proportional to a physical value they monitor rising and/or falling edges of digital signals to either count or measure position. Since these are digital sensors they will need to connect to the NI 9403 digital module and use the digital sidecar rather than the analog breakout board.



The digital side car has many more connections than the analog bumper, but we will focus on the necessary connections for making counter and encoder measurements. The digital side car also requires power at the +12 V pins like the analog breakout board. You can connect to the internal counters using the GPIO (General Purpose Input Output) pins which allow you to make either counter or encoder measurements.



Once you have connected the encoder to the appropriate channel you can use the encoder palette to make an angular position measurement. Examples for these sensors, such as the encoder example below, can also be found by going to Getting Started windows and clicking the Find Examples link in the bottom right corner.



Conclusion

This concludes the overview of the FRC Robotics Sensor API. We have taken a detailed look into how to make analog sensor measurements using an accelerometer and the WPI Robotics Library Sensor API. Please refer to the FIRST Robotics Competition VIs Help file and other tutorials for more detailed information about this API.