DAC-DP interface module

How to make a strain gauge

What you will need

1. Silhouette cameo
2. Computer with Silhouette studio
3. Silhouette pen holder
4. Electroninks Circuitscribe conductive ink pen
5. Common white printer paper
6. Kapton tape 1’’ wide
7. 30ga Magnet wire
8. 2x1 Female pin header/terminals
9. Conductive Wire glue
10. Strain gauge silhouette file
11. Silhouette grid cut file

Procedure

Make the sensor

1. Plugin the silhouette cameo
2. Place the white printer paper onto the cutting mat aligning the top left corner of the paper with the top left corner of the cutting mat.
3. Load the cutting mat into the cameo
4. Set the blade height to “1”
5. Load the blade into the cameo and secure in place
6. Open the grid cutting file on the computer in silhouette studio
7. Click “send to silhouette”
8. The cameo will now cut out the strain gauge backing pieces
9. Remove the blade from the cameo
10. Place the pen in the pen holder and tighten the screws to secure the pen in place
11. Place the pen/pen holder into the cameo and secure
12. Open the “strain gauge” file on the computer in silhouette studio
13. Click “send to silhouette”
14. The cameo will now draw the strain gauges
15. Remove the pen holder and clean any residue that it has collected on the pen tip
16. Repeat steps 10-13 to draw a second layer onto the strain gauges. This will ensure that the ink is properly connected along the trace of the strain gauge.
17. Once it is done, tap the unload button on the cameo.
18. Set the cutting mat and strain gauges off to the side and let the ink dry for at least 30 minutes.
19. Remove the strain gauges - DO NOT try to peel the strain gauges off of the cutting mat, the more that the strain gauge is bent or curled, the more variation or unreliable the strain gauge will be. Instead, Slide a long flat blade (pocket knife or razor blade) under each strain gauge to remove them from the adhesive, keeping the strain gauge as flat as possible

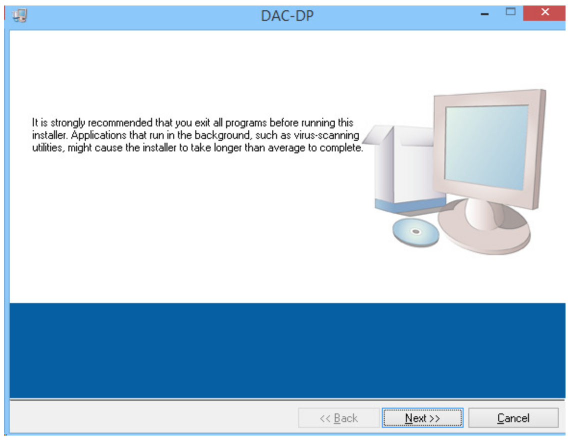
Prepare the wires

20. Cut two lengths of at least 6” of 30ga magnet wire
21. Use a knife to scrape the enamel off of both ends for ¼”
22. Use a crimp tool or pliers to connect a terminal to one end of each wire
23. Place the terminals into the 2 pin connector
24. Place an inch of heatshrink tubing over the pin header/wires
25. Use a heat source to shrink the tubing
26. Flatten the end of the tubing by the wire, this will allow for some flexibility and prevent the wires from breaking from too sharp of a bend.
27. Use a dab of wire glue to adhere the bare end of each wire to each pad of the strain gauge.
28. Wait at least an hour for the glue to cure
29. Cut a length of kapton tape 1.5” long
30. Place the tape centered over the strain gauge and perpendicular to the direction of the wires, being certain to align the top and bottom of the strain gauge with the edges of the tape.
31. Wrap the tape around the back of the strain gauge
32. You are done!

Installation instructions

1. Install the DAC-DP software (might need admin privileges)

- At this point there are 2 options, if you have LabVIEW and NI VISA already installed on the computer, then you can simply run the DAC-DP.exe program from the “Data Acquisition Control and Display Panel” folder on the disc. (does not require admin)
- If LabVIEW and NI VISA is not installed on the computer you are using, then you will need to install the program using the setup.exe (requires admin to install)
- If your computer is setup to autorun removable media, then the setup should start when you insert the disc into the drive, if it does not start by itself, then you will have to open “setup.exe” from the disc’s root directory.



It is strongly recommended that you exit all programs before running this installer. Applications that run in the background, such as virus-scanning utilities, might cause the installer to take longer than average to complete.

- Once the setup opens, keep clicking "next" until the DAC-DP setup is complete and the installer opens up the Arduino setup.
- Continue to step 2.

2. Install the Arduino software/drivers (admin required)

- Install the Arduino IDE. (If you are installing the DAC-DP software using the setup, then it will automatically start the Arduino installer.)
- Continue through the installation until a pop up appears that will ask if you want to install the Arduino USB drivers, click install.
- The computer will ask if you want to restart to apply changes, ignore this for now.
- Plug in your board and wait for Windows to begin its driver installation process.
- The computer should recognize the Arduino and assign a COM port to it.
- In the event that windows cannot automatically install the drivers, follow the below procedure.
- Click on the Start Menu, and open up the Control Panel.
- While in the Control Panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
- Look under Ports (COM & LPT). You should see an open port named "Arduino UNO (COMxx)". If there is no COM & LPT section, look under "Other Devices" for "Unknown Device".
- Right click on the "Arduino UNO (COMxx)" port or "Unknown Device" and choose the "Update Driver Software" option.
- Next, choose the "Browse my computer for Driver software" option.
- Finally, navigate to and select the driver file named **"arduino.inf"**, located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named **"Arduino UNO.inf"**
- Windows will finish up the driver installation from there. "
  (Arduino.cc)

3. Upload the code to the Arduino (only need to do once per Arduino)

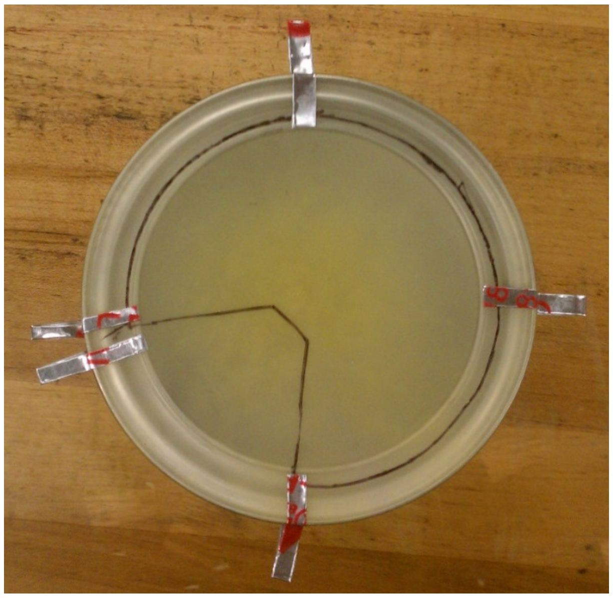
- Once the computer has installed the Arduino IDE, plug in the Arduino, navigate to the folder in the DAC-DP installer disc named xloader and open "**xloader.exe**"

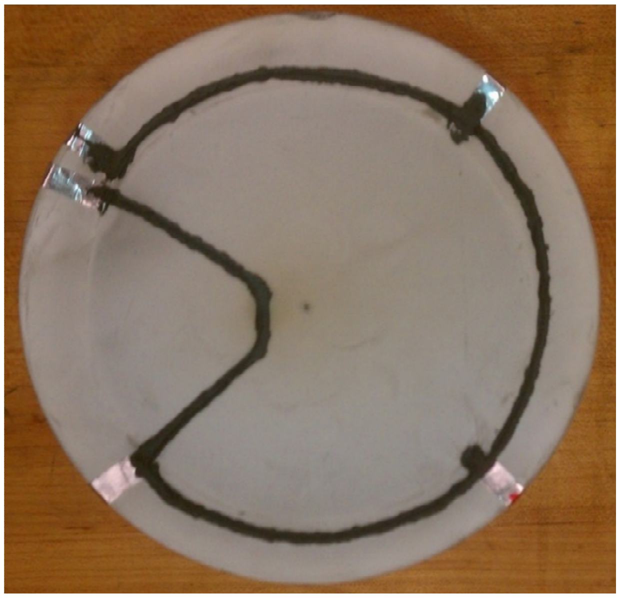
1. For the hex file, click the browse button and navigate to the "Uno_Serial_USB.hex" file located in the xloader folder.
2. Under device choose "UNO(ATmega328)".
3. Select the COM port that the computer assigned to the Arduino. If you don't know what COM port was assigned, open the "devices and printers" from the start menu, the Arduino should be listed in de devices with a "(COM#)" in its name.
4. Leave the Baud rate at 115200
5. Click upload.
6. "Uploading" will appear at the bottom of the xloader screen.
7. Once the "uploading" changes to "13622 bytes uploaded" (usually within 30 seconds) you have successfully set up the Arduino.

Wheatstone bridge Interface instructions

Step 1 - Construct a MEMS Pressure sensor module kit.

The pressure sensor built in this version will be slightly different than the sensor described in the MEMS Pressure sensor module kit. The sensor will be built using the same method as described in the MEMS instructions, but a different design will be used. Start by modifying the stencil to have only one

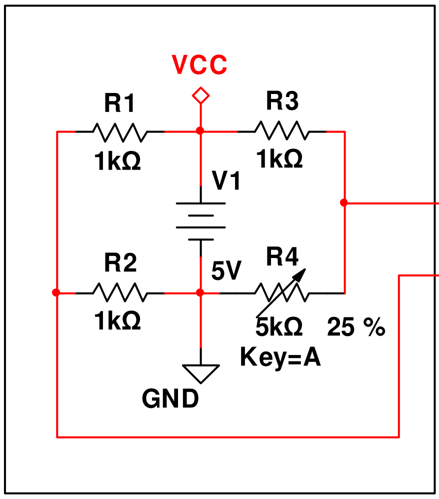




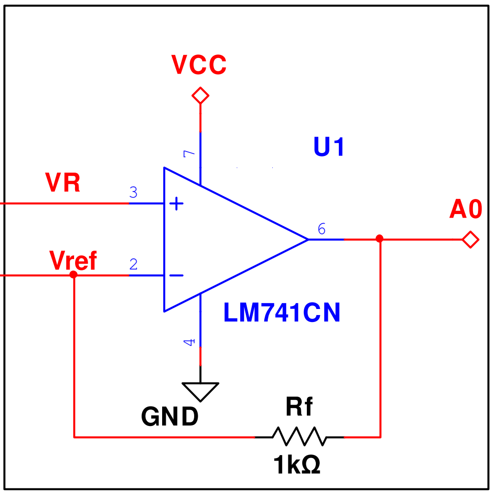
variable resistor and a break between the variable resistor and the static resistor, like the pictures below.

Step 2. – Connect the sensor to the interface module

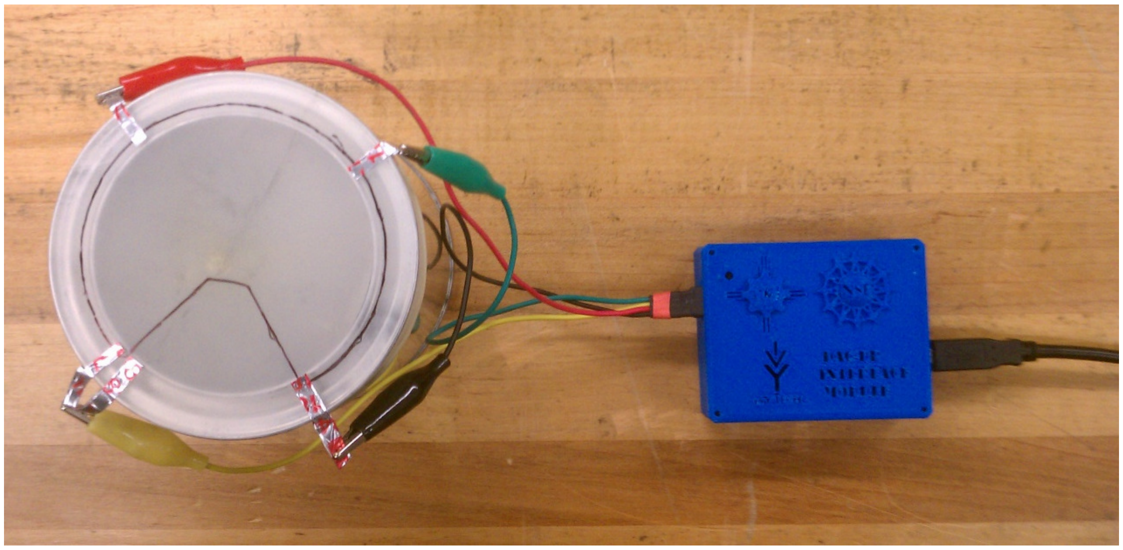
Use alligator clip leads to attach the Pressure sensor to the Interface module as shown below. When connecting to the point where the sensor is broken, make sure both points are connected to the alligator clip.



A simulated wheatstone bridge

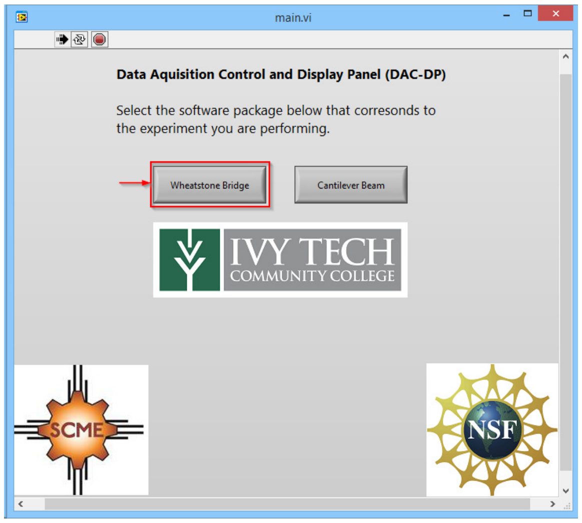


The basic form of the DAC-DP shield



Step 3 – Start the program

Open the “DAC-DP” program. Click the “Wheatstone Bridge” button in the main screen. “In the Device settings” use the dropdown menu select the COM port that the computer assigned to the Interface module. Click the “RUN” ⇨ button to start the program.

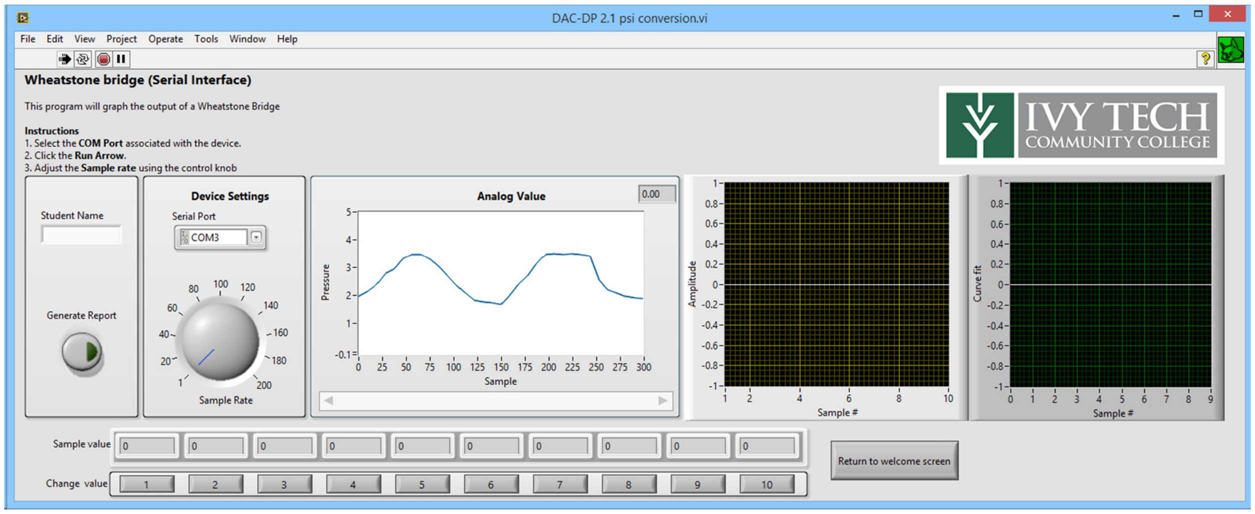


Data Aquisition Control and Display Panel (DAC-DP)

Select the software package below that corresonds to the experiment you are performing.

Step 4 – Acquire results

Once the program starts to run, you should see a line being graphed with a value of roughly 2.5v. Adjust the frequency to a value that is slow enough that the results do not pass too quickly, but fast enough that you are able to view the change in pressure, a good start is around 10Hz. Slowly apply pressure to the center of the sensor. The line on the graph should slowly start to rise. Once you are able to observe a sufficient increase in value, slowly release the pressure applied to the sensor, the output on the graph should decrease as the internal pressure is decreased.



Wheatstone bridge (Serial Interface)

This program will graph the output of a Wheatstone Bridge

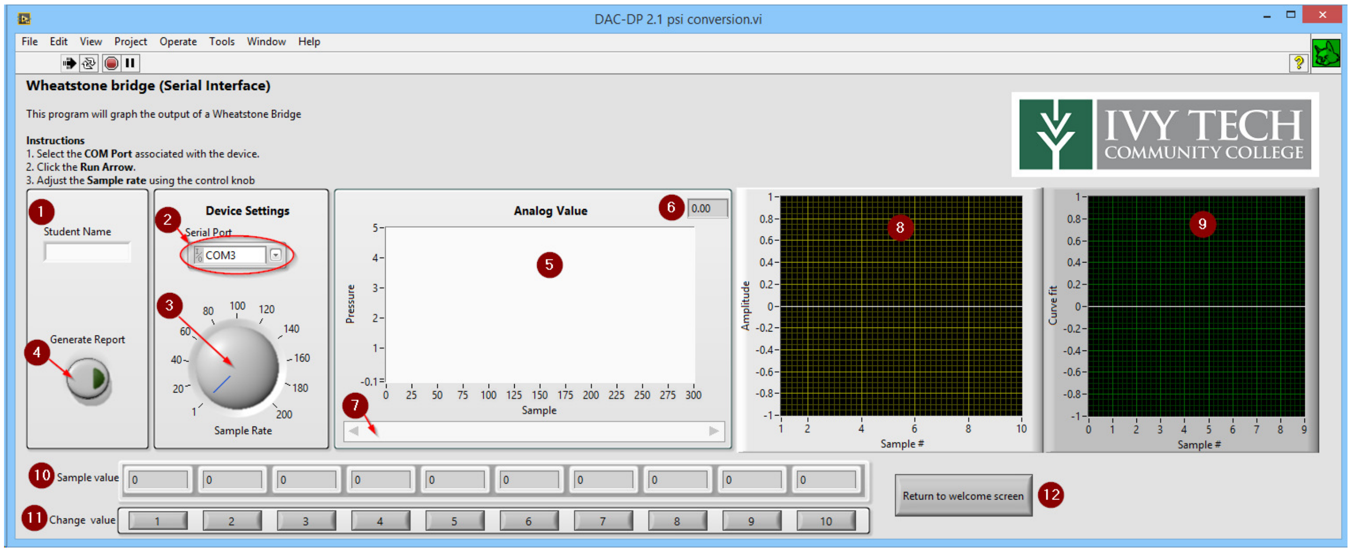
Instructions
1. Select the **COM Port** associated with the device.
2. Click the **Run Arrow**.
3. Adjust the **Sample rate** using the control knob

Troubleshooting –

1. If the program in unable to recognize the interface module, you may need to update the drivers manually through the device manager
2. If the program will run, but the sensor's output on the graph will not change, then double check your connections.

3. If the line being graphed is extremely high (4-4.5v), then it is likely that your Wheatstone bridge is unbalanced, use a multimeter to test the resistance of each bridge section. Each section should be very close in value to every other section, within a %5 difference of each other.

Wheatstone bridge graphing program features



Wheatstone bridge (Serial Interface)

This program will graph the output of a Wheatstone Bridge

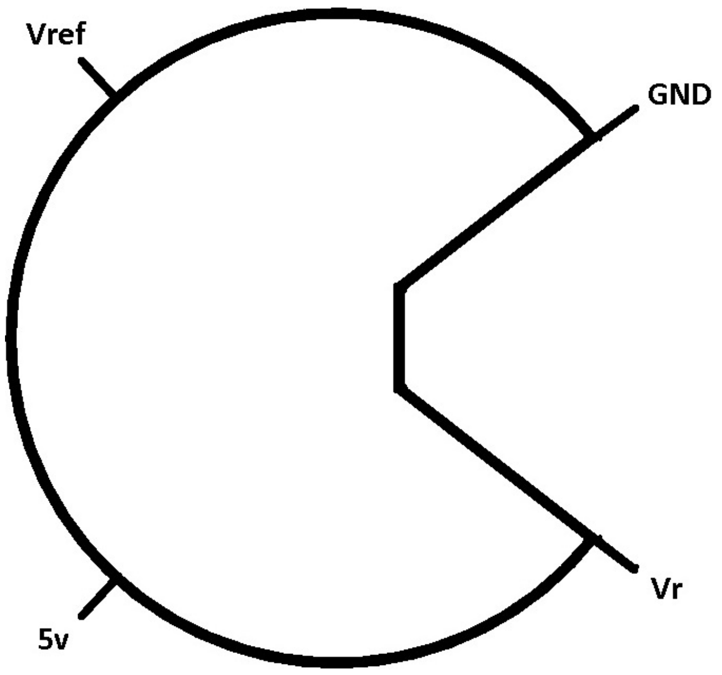
Instructions
1. Select the **COM Port** associated with the device.
2. Click the **Run Arrow**.
3. Adjust the **Sample rate** using the control knob

Serial Port: COM3

Analog Value: 0.00

Sample value: 0 0 0 0 0 0 0 0 0 0

1. **Student's name** – allows the student(s) to sign the report that is generated.
2. **Serial port** – select the correct COM port from the dropdown menu.
3. **Frequency Dial** – Use the dial to change the number samples per second (frequency) that the program will graph.
4. **Generate report** – export all of the data collected, analyzed and curve fit data to an Excel document. (requires excel to be installed)
5. **Graph** – The output of the sensor will be graphed in this area.
6. Most recent sample value on graph in decimal form.
7. **Graph Scroll bar** – Slide scroll bar to view graph history.
8. **Take sample** – By clicking any button, the program will store and graph the current value in to corresponding array.
9. **Sample Value** – displays the numeric value of each sample.
10. **Samples Graph** – Displays the 10 collected values on graph
11. **Curve fit** – Displays a Curve fit to match the collected data values.
12. **Return to welcome screen** – Closes the current window and opens up the welcome screen where either kit can be selected

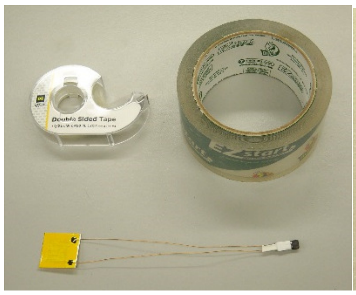


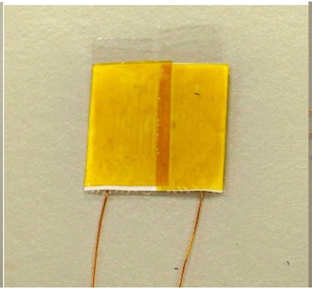
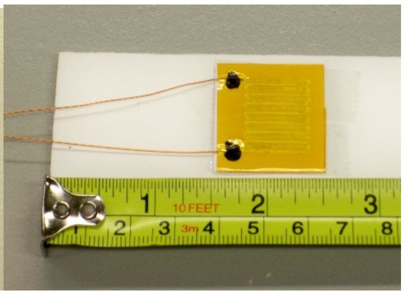
Actual size template
Circle has a 3.45" diameter

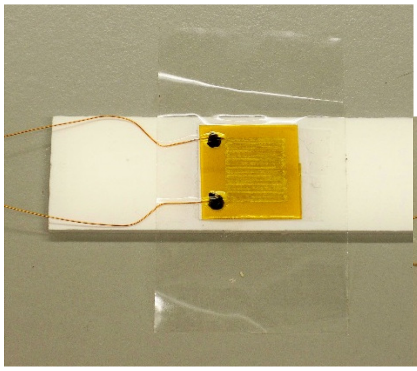
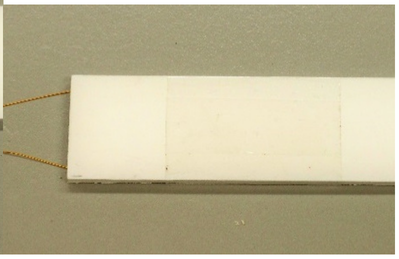
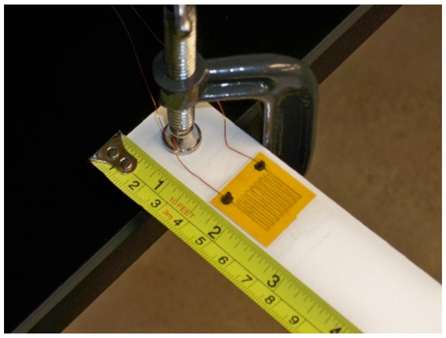
Cantilever Beam instructions

Step 1 – Attaching the strain gauge.

For this step you will need the cantilever beam that you are analyzing, one strain gauge, double sided tape, and packing tape. Start by covering the back of the strain gauge in double sided tape. Place the strain gauge ≈1.5in away from the end of the cantilever beam. You want the center of the strain gauge to be between 1 and 1¼ inch away from the edge of the table that you are attaching the cantilever beam to. Measure and record the resistance value of the strain gauge using a multimeter, you will need this value later. Continue to attach the cantilever beam to the table as described in the SMCE MEMS Cantilever Book 2.

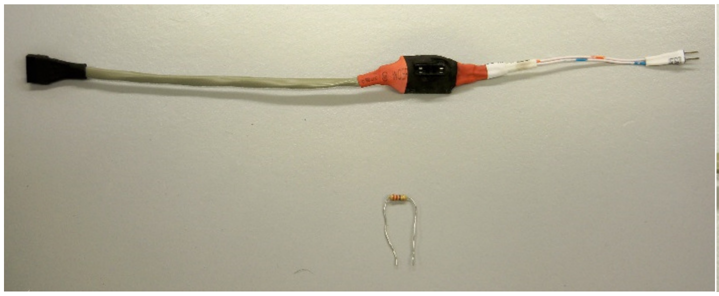
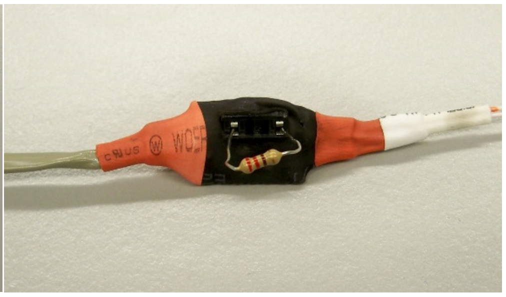


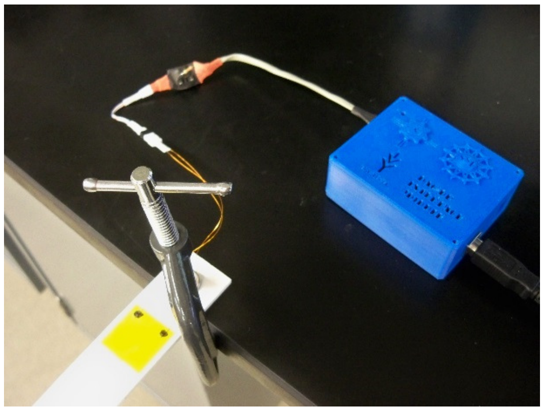



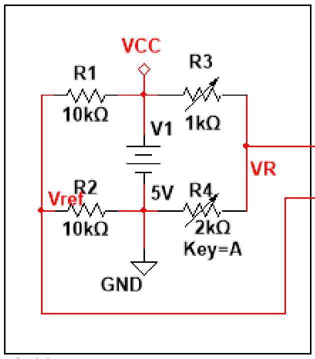




Step 2. – Connecting the sensor – requires updated pictures

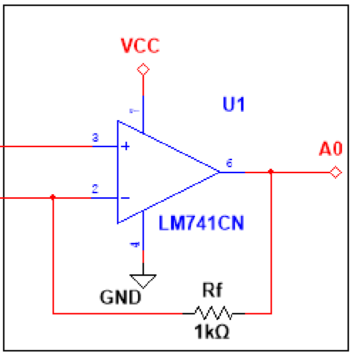
Start by obtaining a resistor that is the closest matching value of the strain gauge to complete the bridge. For example the strain gauge used in this example measures 229.8Ω, so a 220Ω resistor was used. Insert the fixed value resistor into the DAC-DP Cantilever beam Cable. Connect the 4pin end of the cable to the DAC-DP interface module being mindful of the orientation, and the other end to the strain gauge where polarity doesn't matter.





Cable



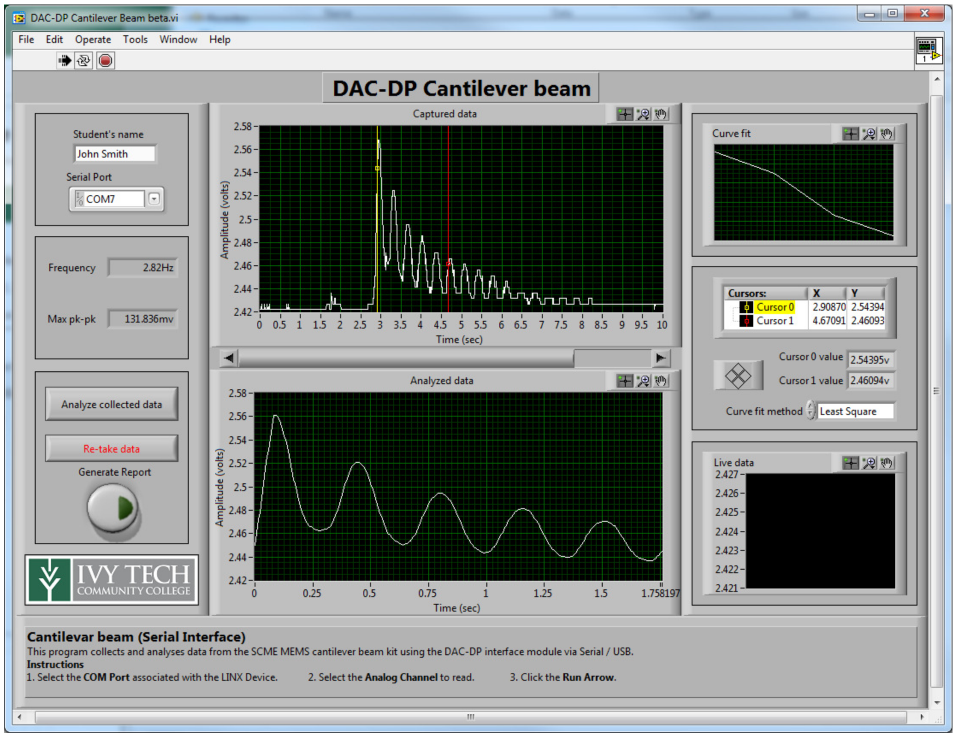
Amplifier Circuit

Step 3 – Software setup

Open the "DAC-DP" program. Click the "Cantilever Beam" button in the main screen. Use the dropdown menu under "Serial Port" to select the COM port that the computer assigned to the Interface module. Click the "RUN" button to start the program.

Step 4 – Acquire results

Once the program starts to run, you should see a line being graphed in the "live data" graph with a value of roughly 2.5v. Lift the end of the cantilever beam up 8-12 inches and release it. Observe the live graph and click "Analyze Data" once the readings start to level off. To get a frequency reading, Use the cursors to select the first 4-5 periods/peaks of the data collected. (optional)Use the cursors to select the entire duration of the cantilever beam's motion and click "Generate Report" (requires excel) for a detailed document that can be further analyzed or saved.



Student's name: John Smith

Serial Port: COM7

Frequency: 2.82Hz

Max pk-pk: 131.836mv

Cursors:	X	Y
Cursor 0	2.90870	2.54394
Cursor 1	4.67091	2.46093

Cursor 0 value: 2.54395v

Cursor 1 value: 2.46094v

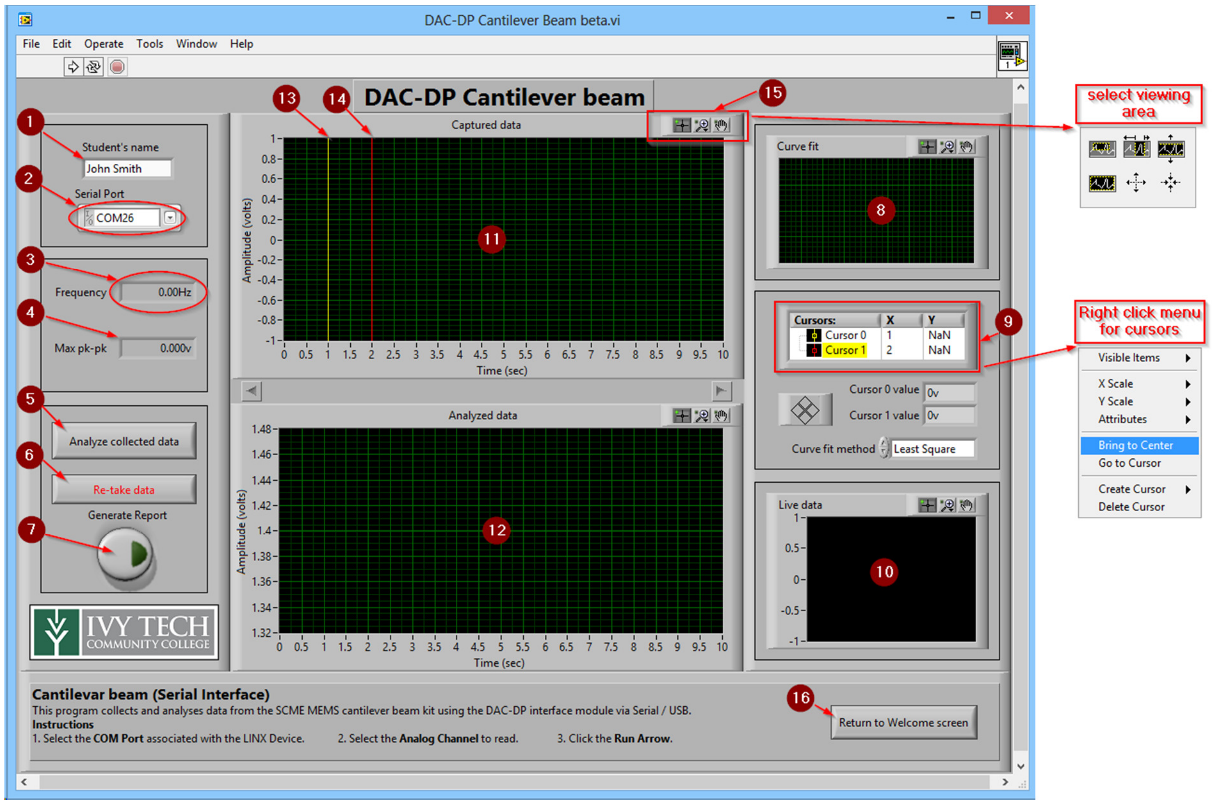
Curve fit method: Least Square

Cantilevar beam (Serial Interface)
This program collects and analyses data from the SCME MEMS cantilever beam kit using the DAC-DP interface module via Serial / USB.
Instructions
1. Select the **COM Port** associated with the LINX Device. 2. Select the **Analog Channel** to read. 3. Click the **Run Arrow.**

Troubleshooting –

1. If the program in unable to recognize the interface module, you may need to update the drivers manually through the device manager
2. If the program will run, but the sensor's output on the graph will not change, then double check your connections.

Cantilever Beam graphing program features



Cantilevar beam (Serial Interface)
This program collects and analyses data from the SCME MEMS cantilever beam kit using the DAC-DP interface module via Serial / USB.
Instructions
1. Select the **COM Port** associated with the LINX Device. 2. Select the **Analog Channel** to read. 3. Click the **Run Arrow.**

1. **Student's name** – allows the student(s) to sign the report that is generated.
2. **Serial port** – Select the correct COM port from the dropdown menu before starting the program.
3. **Frequency** – Displays the frequency of the analyzed data in hertz.
4. **Max pk-pk** – displays the greatest difference in voltage measurement.
5. **Analyze Collected Data** – when clicked, the program will stop collecting data and display the results.
6. **Re-take Data** – If you are not satisfied with the data you collected, click this to forget your old data and start collecting new data.
7. **Generate report** – export all of the data collected, analyzed and curve fit data to an Excel document. (requires excel to be already installed)
8. **Curve fit** – displays an approximate curve to the dampening rate of the cantilever beam.
9. **Cursor management** – displays stats and allows the user to alter the cursor properties.
10. **Live Data** – The output of the sensor will be displayed on this graph as it is being collected.
11. **Captured Data** – Displays all of the readings taken, and allows you to select which data to analyze using the cursors.
12. **Analyzed Data**- displays the selected data after going through a filter that removes noise.
13. **Cursor 0** – click and drag this cursor to select the starting point to analyze the data.
14. **Cursor 1** – click and drag this cursor to select the ending point of the data to analyze.
15. **Viewing area tools** – use these tools to select the viewing area on each of the graphed if desired.

16. **Return to welcome screen** – Closes the current window and opens up the welcome screen where either kit can be selected.

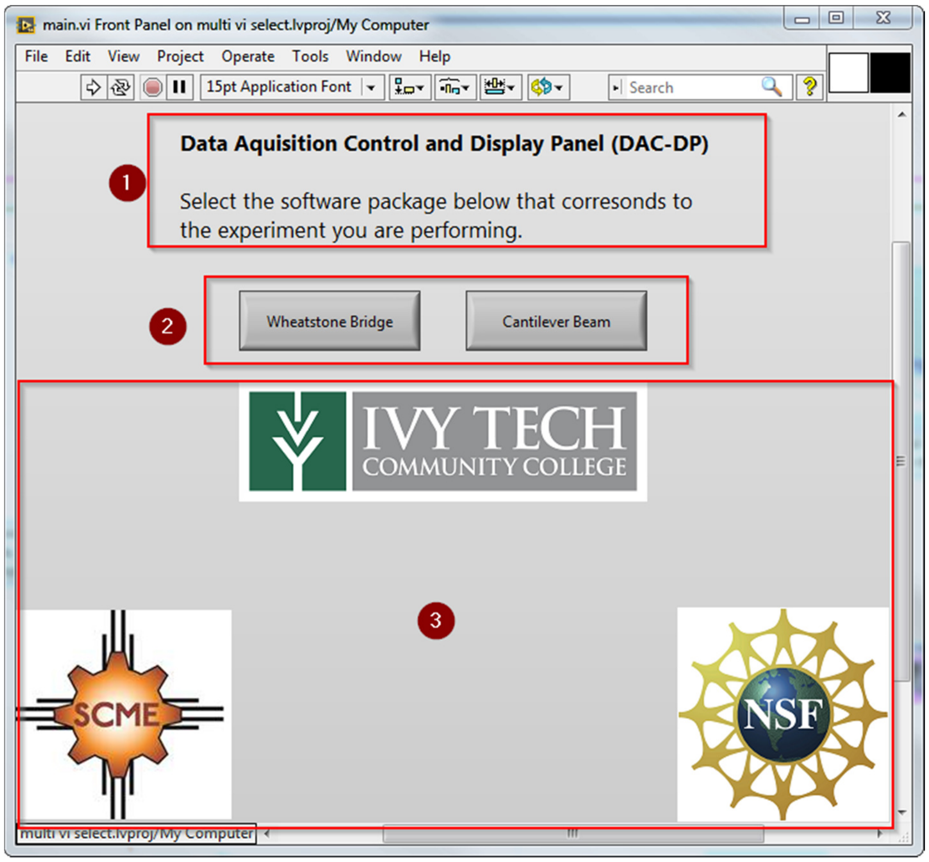
LabVIEW code

The DAC-DP software is written in LabVIEW, with the MakerHub LINX add-on to interface with the Arduino. There are many places to learn about LabVIEW coding. Here are a few links that are helpful when getting started.

- https://www.labviewmakerhub.com/doku.php?id=learn:tutorials:labview:basics
- http://www.ni.com/getting-started/labview-basics/
- https://www.google.com/

Main screen

The main screen allows the program to only need one installer and one shortcut to access both/all programs that correspond to different SCME kits.


main.vi Front Panel on multi vi select.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
1
2
3
Wheatstone Bridge
Cantilever Beam
IVY TECH COMMUNITY COLLEGE
SCME
NSF
multi vi select.lvproj/My Computer

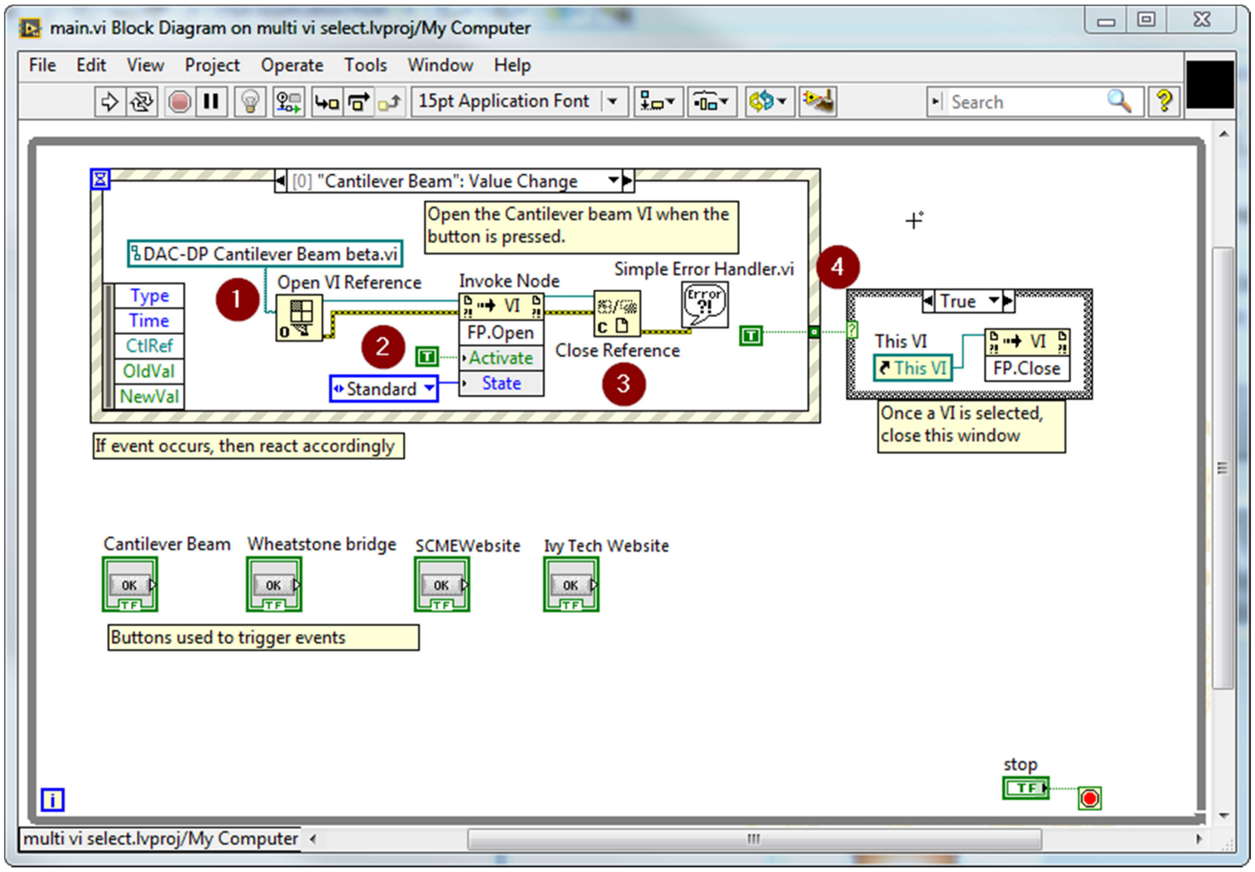
Data Aquisition Control and Display Panel (DAC-DP)

Select the software package below that corresonds to the experiment you are performing.

1. **Program name and quick start info**
2. **Software selection – A**llows the user to choose which program they want to use.
3. **Company info –** The logos of the companies who contributed, sponsored, assisted, etc the development of this kit/software.

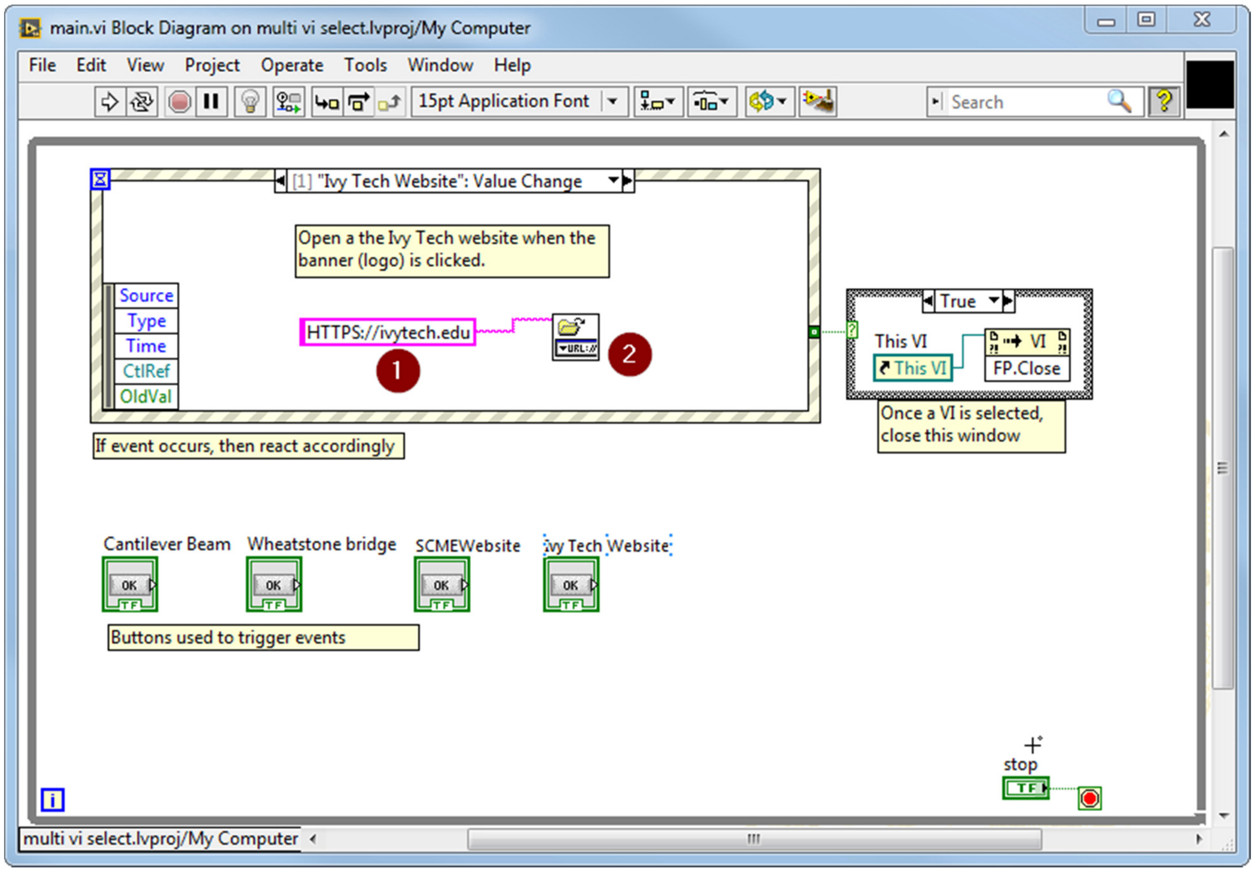
Main Screen Code

The whole program is placed inside a loop so that it will run indefinitely until it is told to stop. The code uses an event structure to detect if any of the 4 buttons are clicked.


main.vi Block Diagram on multi vi select.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
[0] "Cantilever Beam": Value Change
Open the Cantilever beam VI when the button is pressed.
DAC-DP Cantilever Beam beta.vi
Type
Time
CtlRef
OldVal
NewVal
1
Open VI Reference
Invoke Node
VI
FP.Open
Activate
State
2
Standard
Simple Error Handler.vi
Close Reference
3
4
True
This VI
This VI
FP.Close
Once a VI is selected, close this window
If event occurs, then react accordingly
Cantilever Beam
Wheatstone bridge
SCMEWebsite
Ivy Tech Website
OK
Buttons used to trigger events
stop
multi vi select.lvproj/My Computer

If a button is clicked, the event structure will execute the code contained in the corresponding event. If one of the program vi's button is clicked, the code

1. looks for the .vi (program file) in the same directory,
2. uses in invoke node to open the vi,
3. finished activities with the .vi, and checks for errors
4. Then activates a case structure which closes the main screen window.

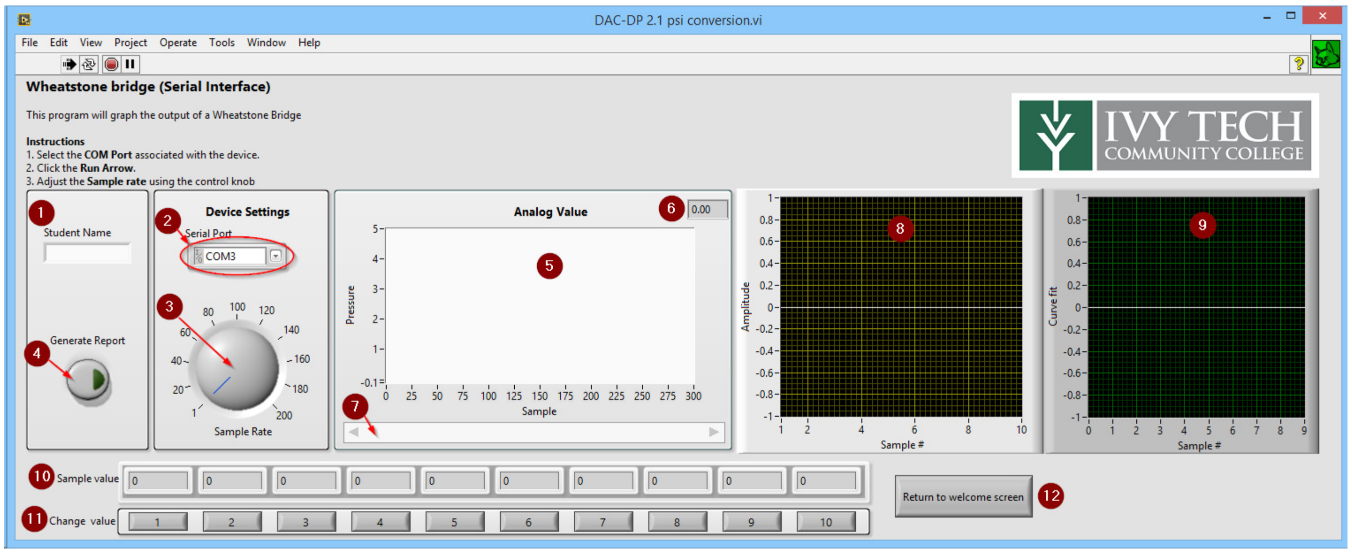

main.vi Block Diagram on multi vi select.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
[1] "Ivy Tech Website": Value Change
Open a the Ivy Tech website when the banner (logo) is clicked.
Source
Type
Time
CtlRef
OldVal
HTTPS://ivytech.edu
1
2
True
This VI
This VI
VI
FP.Close
Once a VI is selected, close this window
If event occurs, then react accordingly
Cantilever Beam
Wheatstone bridge
SCMEWebsite
Ivy Tech Website
Buttons used to trigger events
stop
multi vi select.lvproj/My Computer

If a button is clicked, the event structure will execute the code contained in the corresponding event. Each banner is actually a button with an image used as it's display state. If one of the banners is clicked, the code

1. Uses preset website address
2. And opens it in the default web browser

Wheatstone bridge program

The Wheatstone bridge program is designed to graph and analyze the output from a modified SCME pressure sensor kit pressure sensor.



Wheatstone bridge (Serial Interface)

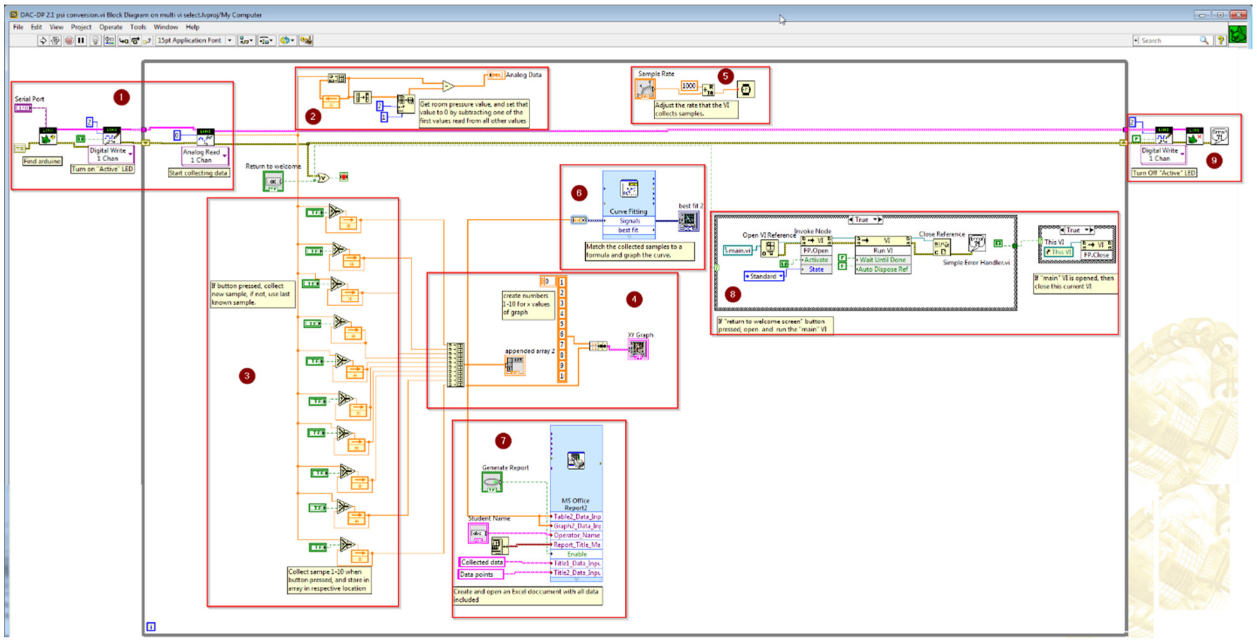
This program will graph the output of a Wheatstone Bridge

Instructions
1. Select the **COM Port** associated with the device.
2. Click the **Run Arrow**.
3. Adjust the **Sample rate** using the control knob

1. **Student's name** – allows the student(s) to sign the report that is generated.
2. **Serial port** – select the correct COM port from the dropdown menu.
3. **Frequency Dial** – Use the dial to change the number samples per second (frequency) that the program will graph.
4. **Generate report** – export all of the data collected, analyzed and curve fit data to an Excel document. (requires excel to be installed)
5. **Graph** – The output of the sensor will be graphed in this area.
6. Most recent sample value on graph in decimal form.
7. **Graph Scroll bar** – Slide scroll bar to view graph history.
8. **Take sample** – By clicking any button, the program will store and graph the current value in to corresponding array.
9. **Sample Value** – displays the numeric value of each sample.
10. **Samples Graph** – Displays the 10 collected values on graph
11. **Curve fit** – Displays a Curve fit to match the collected data values.
12. **Return to welcome screen** – Closes the current window and opens up the welcome screen where either kit can be selected

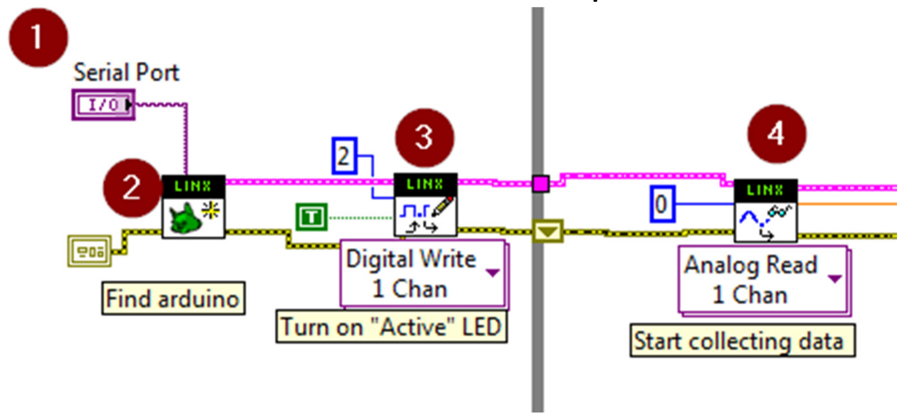
Wheatstone bridge code

The code has several segments that are described below


DAC-DP 2.1 psi conversion.vi Block Diagram on multi vi select.lvproj/My Computer
Serial Port
Find arduino
Digital Write 1 Chan
Turn on "Active" LED
Analog Read 1 Chan
Start collecting data
Analog Data
Get room pressure value, and set that value to 0 by subtracting one of the first values read from all other values
Sample Rate
Adjust the rate that the VI collects samples.
Return to welcome
If button pressed, collect new sample, if not, use last known sample.
Collect sampe 1-10 when button pressed, and store in array in respective location
Curve Fitting
best fit 2
Match the collected samples to a formula and graph the curve.
create numbers 1-10 for x values of graph
appended array 2
XY Graph
Generate Report
MS Office Report2
Student Name
Collected data
Data points
Create and open an Excel doccument with all data included
Open VI Reference
Invoke Node
Close Reference
Run VI
Simple Error Handler.vi
If "return to welcome screen" button pressed, open and run the "main" VI
If "main" VI is opened, then close this current VI
Digital Write 1 Chan
Turn Off "Active" LED

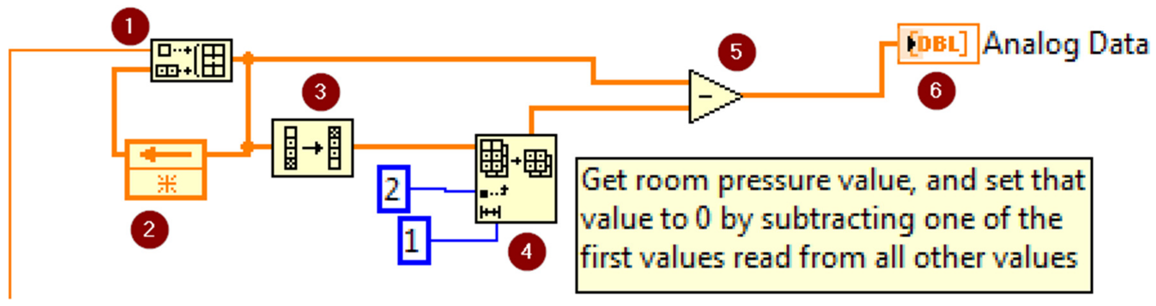
1. **Setup –** Contains the code used for initializing the Arduino from the settings provided.
2. **Graphing –** Calculates and displays the live data
3. **Specific samples –** Lets the user collect a set of 10 samples
4. **Samples graphing –** Takes the 10 samples and graphs them
5. **Rate adjustment –** Allows the user to adjust the rate of which the DAC-DP interface module collects samples.
6. **Curve fit –** Graphs a formula based curve that fits the 10 samples collected
7. **Data Output –** Exports the data to a Microsoft excel document that can be saved or further analyzed.
8. **Return to main screen –** Closes the current program and returns to the main screen.
9. **Closing code –** Performs shutting down code and checks for errors.

1. Setup



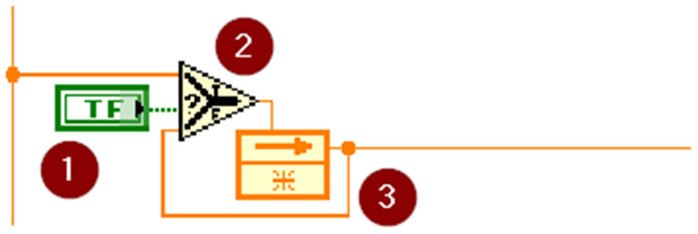
1. **Serial Port –** A drop down selector for the user to choose which installed com port is designated for the DAC-DP interface module
2. **Open LINX Device –** Find and check for proper connection to the Arduino (DAC-DP interface module)
3. **Digital write –** Turns on the “active” LED that is connected to digital pin 2 on the Arduino.
4. **Analog read –** Read the current value of analog pin 0 every loop iteration.

2. Graphing


1
2
3
4
5
6
2
1
DBL
Analog Data
Get room pressure value, and set that value to 0 by subtracting one of the first values read from all other values

1. **Build array –** Places the collected data into an array
2. **Feedback loop –** Adds the previous array onto the end of the of the first current value. This continues adding each collected value on to the array creating a history of values for the graph.
3. **Invert array –** Flips the order that the array is stored at, putting the first value at the beginning and the new values at the end.
4. **Array subset –** Takes the value of the second value collected (first is always 0). This assumes that the first sample is taken at room pressure
5. **Subtract** - Subtracts the numerical value of the first sample from each sample in the array. This calibrates the graph so that 0 is room pressure.
6. **Graph** – Graphs the calibrated array so that the user can observe it.

3. Specific Samples

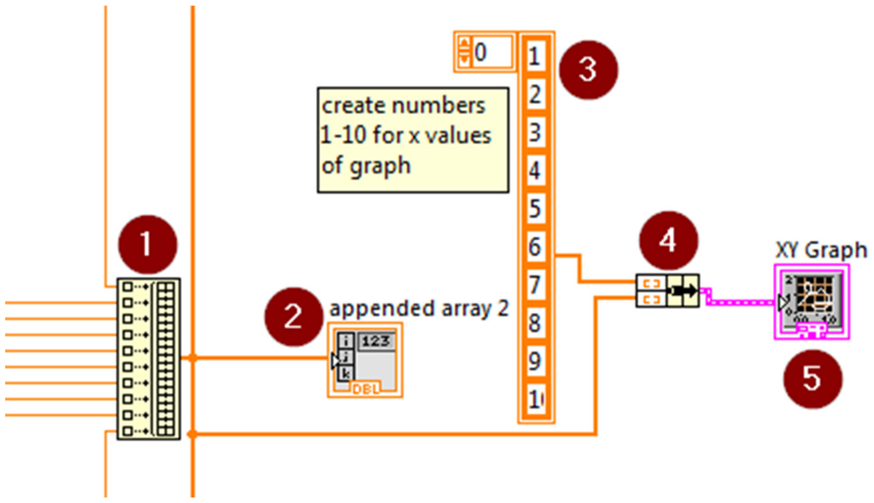


This segment is repeated for 10 different samples. Which then get sent into an array.

1. **Button –** If the button is clicked, then change to output value to the true/false selector.

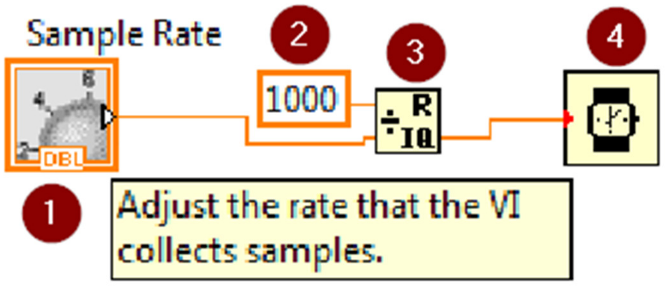
2. **Select –** If button is clicked, save the current value (true input). If button is not clicked use the last known value (false input)
3. **Feedback loop –** Allows the last known value to be sent back to the selector.

4. Samples graphing



1. **Build Array –** Arranges all of the collected samples into a single array.
2. **Array Indicator –** Displays the array on the front panel for the user/
3. **Create Array –** Creates an array of values from 1 to 10 to use for the X values when graphing the collected samples.
4. **Bundle –** Combines the X and Y values into one wire for the XY Graph
5. **XY Graph –** Displays the collected samples on a graph using the collected sample values for Y and the values of 1-10 for X

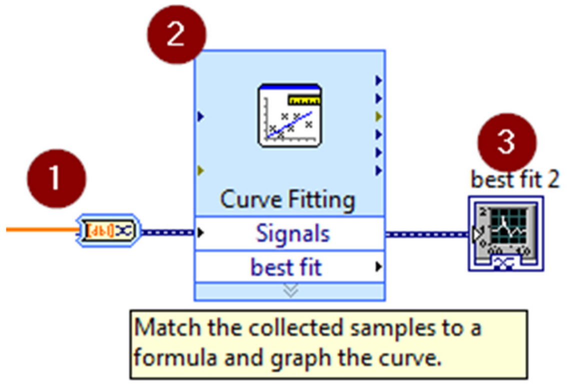
5. Rate adjustment



1. **Dial –** Front panel input for the user to change the value of the number of samples per second to take.
2. **Numeric constant –** Because LabVIEW operates in milliseconds the value of the dial must be divided by 1000.

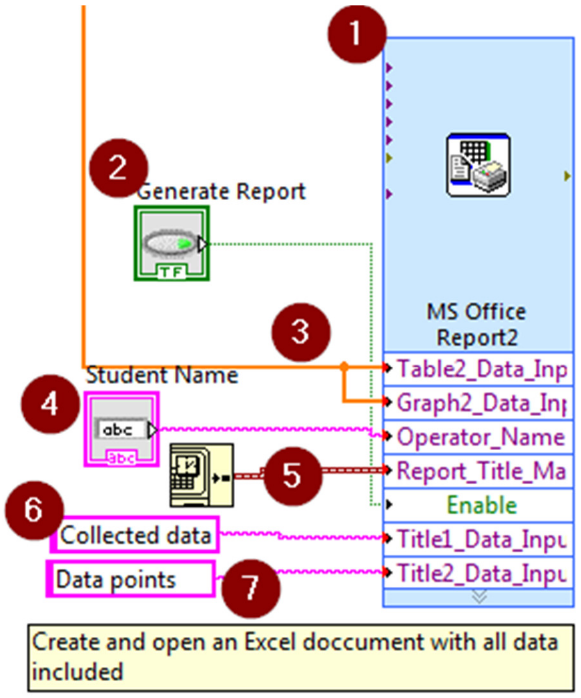
3. **Quotient and Remainder –** Divides the value of the dial by 1000 and outputs the integer quotient which increases as the value of the dial increases.
4. **Wait (ms) –** Tells the program to wait the number of milliseconds inputted before looping the program and taking the next reading.

6. Curve fit


1
2
3
Curve Fitting
Signals
best fit
best fit 2
Match the collected samples to a formula and graph the curve.

1. **Convert to dynamic data –** Converts the array of collected values into a form a data that the curve fitting subvi can use
2. **Curve fitting –** Fits a curve to the collected data points in the array.
3. **Waveform Graph –** Displays the resulting curve on a graph for the user.

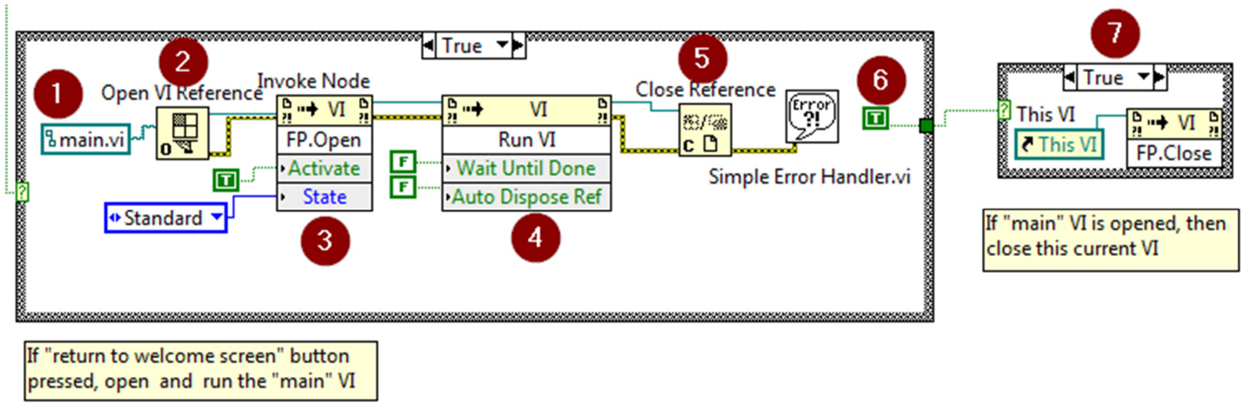
7. Data Output



1. **MS office report –** Creates an excel document based on a template that places inputted data to specific locations in the excel document.
2. **Button –** Tells the MS office report subvi to create a report when the user clicks the button.

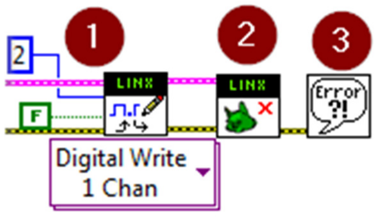
3. **Data input –** Displays the collected data points in a graph and a table.
4. **String input –** Takes the words that the user inputs and includes it in the excel report.
5. **Get date/time in seconds –** Gets the current date and time and “timestamps” the report generated.
6. **String constant –** Places a title above the graph.
7. **String constant –** Places a title above the table.

8. Return to Main Screen



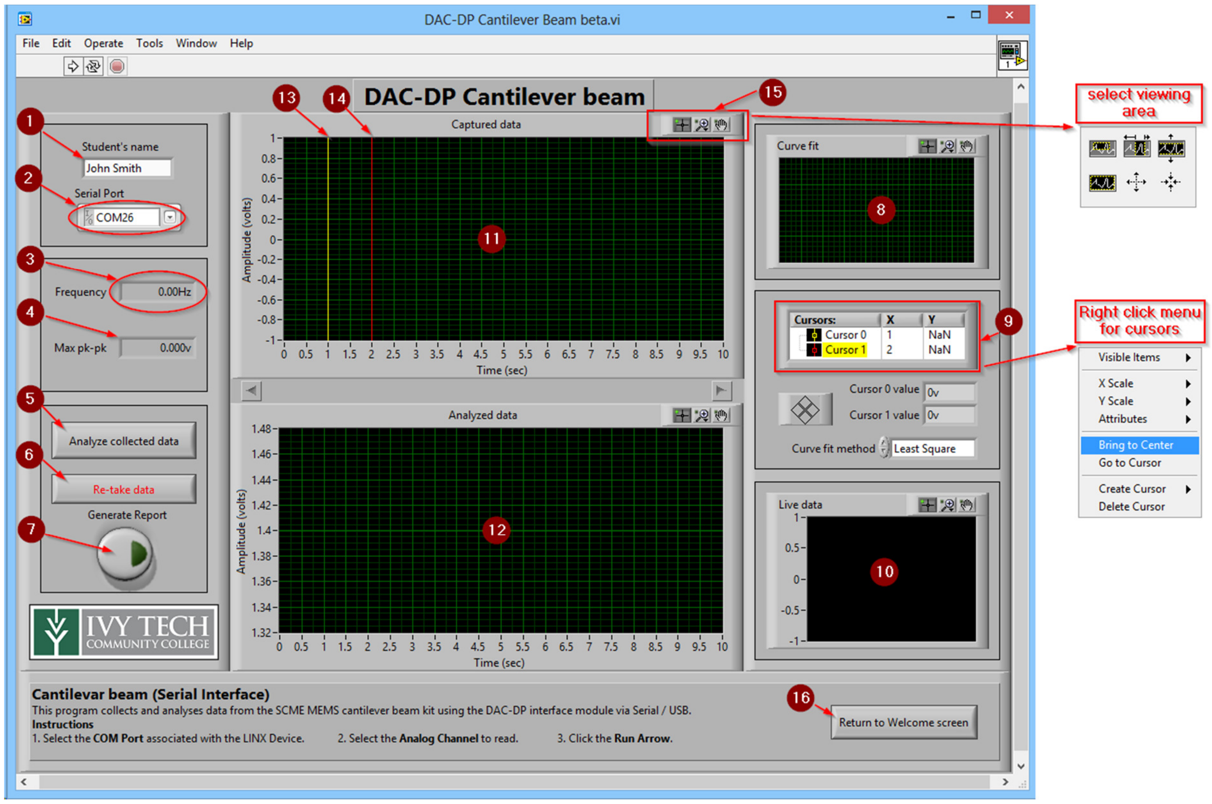
1. **Vi Path –** Identifies the name of the vi to look for in the same directory.
2. **Open VI Reference –** Prepares the vi to receive commands.
3. **Invoke Node –** Tells the program to open the vi.
4. **Invoke node –** Tells the program to automatically run the vi.
5. **Close Reference –** Finishes activities with the .vi, and checks for errors.
6. **Boolean –** Enables the wire to activate the next case structure.
7. **Case structure –** Closes the current vi (program).

9. Closing code



1. **Digital write –** Turns off the “active” LED connected to digital pin 2 on the Arduino.
2. **LINX close –** Closes the connection to the Arduino.
3. **Simple error handler –** Checks for and reports any errors to the user.

Cantilever Beam Program



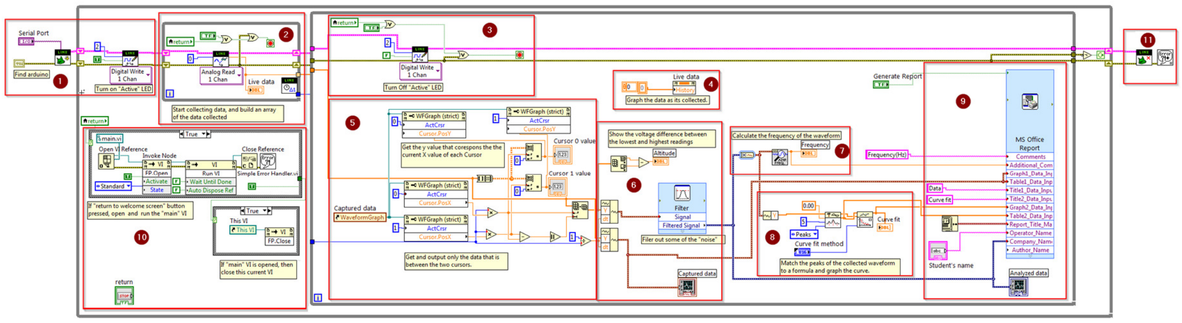
Student's name: John Smith
Serial Port: COM26
Frequency: 0.00Hz
Max pk-pk: 0.000v

Cursors:	X	Y
Cursor 0	1	NaN
Cursor 1	2	NaN

Cursor 0 value: 0v
Cursor 1 value: 0v
Curve fit method: Least Square

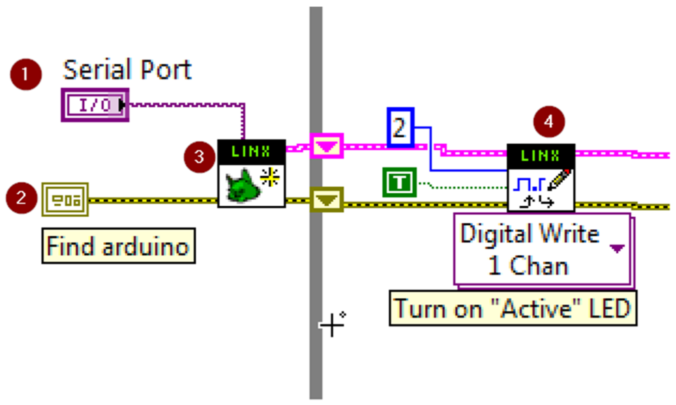
Cantilevar beam (Serial Interface)
This program collects and analyses data from the SCME MEMS cantilever beam kit using the DAC-DP interface module via Serial / USB.
Instructions
1. Select the **COM Port** associated with the LINX Device. 2. Select the **Analog Channel** to read. 3. Click the **Run Arrow.**

1. **Student's name** – Text input that the program uses as the signature for the report.
2. **Serial port** – Selects the correct COM port from the dropdown menu that identifies the Arduino.
3. **Frequency** – Displays the calculated frequency of the analyzed data in hertz.
4. **Max pk-pk** – Displays the greatest difference in voltage measurement.
5. **Analyze Collected Data** – Button that tells the program to end the loop collecting data and start the analysis loop
6. **Re-take Data** – Erases all of the previous collected and analyzed data and starts re-collecting it.
7. **Generate report** – Exports all of the data collected, analyzed and curve fit data to an Excel document.
8. **Curve fit** – Displays an approximate curve to the dampening rate of the cantilever beam.
9. **Cursor management** – Displays stats and allows the user to alter the cursor properties.
10. **Live Data** – The output of the sensor will be displayed on this graph as it is being collected.
11. **Captured Data** – Displays all of the readings taken, and allows you to select which data to analyze using the cursors.
12. **Analyzed Data**- displays the selected data after going through a filter that removes noise.
13. **Cursor 0** – click and drag this cursor to select the starting point to analyze the data.
14. **Cursor 1** – click and drag this cursor to select the ending point of the data to analyze.
15. **Viewing area tools** – use these tools to select the viewing area on each of the graphed if desired.
16. **Return to welcome screen** – Closes the current window and opens up the welcome screen where either kit can be selected.



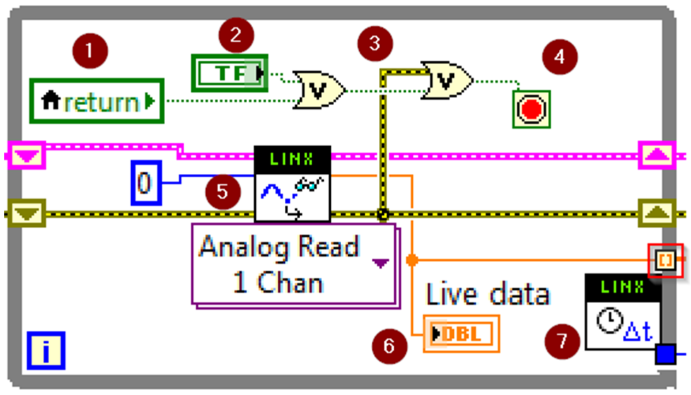
1. **Setup –** Contains the code used for initializing the Arduino from the settings provided.
2. **Collect Data –** Collects the sensor data until user clicks “analyze data” button.
3. **Misc. –** Turns off “active” LED and checks for errors and status.
4. **Live graph reset** – Erases the live graph’s history
5. **Cursor position analysis –** Gets the X and Y values of each cursors current location and makes the cursor box follow the line.
6. **Data analysis –** Analyzes, graphs, and attempts to filter noise out of the selected data
7. **Frequency matching –** Determines the frequency of the analyzed waveform
8. **Curve Fitting -** Calculates and graphs an approximate curve to the dampening rate of the cantilever beam.
9. **Data Output –** Exports the data to a Microsoft excel document that can be saved or further analyzed.
10. **Return to main screen –** Closes the current program and returns to the main screen.
11. **Closing code –** Disconnects from the Arduino and checks for errors.

1. Setup



1. **Serial Port –** A drop down selector for the user to choose which installed com port is designated for the DAC-DP interface module
2. **Error In –** Stats the error report service
3. **Open LINX Device –** Find and check for proper connection to the Arduino (DAC-DP interface module)
4. **Digital write –** Turns on the “active” LED that is connected to digital pin 2 on the Arduino.

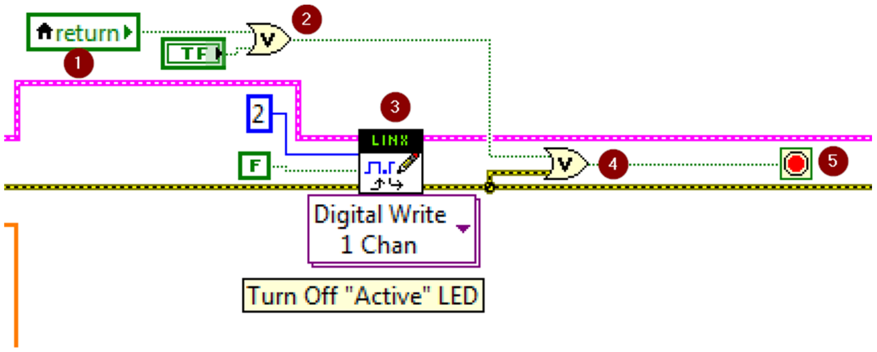
2. Collect Data



1. **Local Variable –** Checks if the "return to main screen" button has been pressed.
2. **Button –** Changes value if the "analyze" button is pressed.
3. **Or –** If any condition (return, analyze, or Error) is true, then output true.
4. **Stop –** Stop the loop if receives a true signal.
5. **Analog read –** Read the current value of analog pin 0 every loop iteration.
6. **Graph –** Graph the data as it is being collected
7. **Loop Frequency –** Determines how fast the program is collecting data points (how many sample per second)

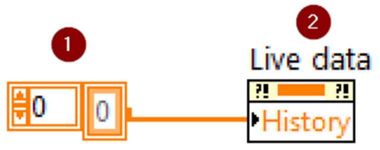
Note – The data wire tunnel leaving the loop is an auto-indexing tunnel. This makes the data "build" an array of values instead of replacing the previous value.

3. Misc.



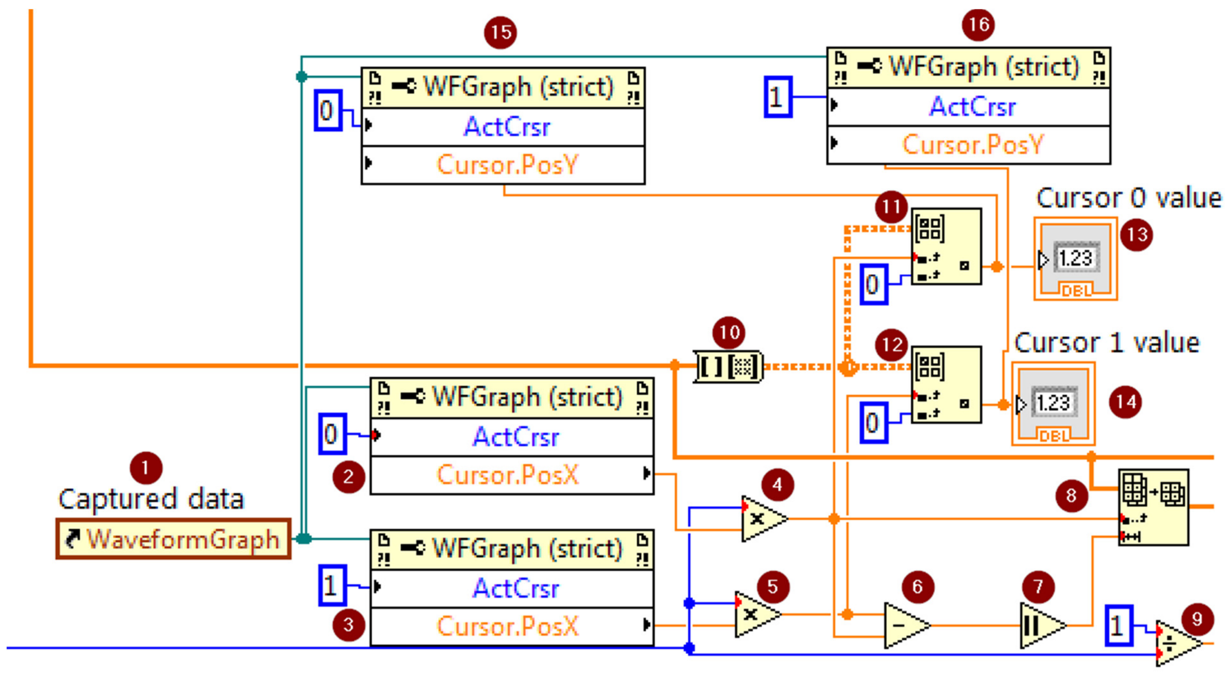
1. **Local Variable –** Checks if the "return to main screen" button has been pressed.
2. **Or –** If any condition (return, Re-collect data) is true, then output true.
3. **Digital write –** Turns off the "active" LED connected to digital pin 2 on the Arduino.
4. **Or –** If any condition (return, Re-collect data, or error) is true, then output true.
5. **Stop –** Stop the loop if receives a true signal.

4. Live graph reset



1. **Array constant** – Creates an empty array to replace the live graph array
2. **Property node (write mode) –** Replaces the current “live Graph” array of data with the array that is wired to the input

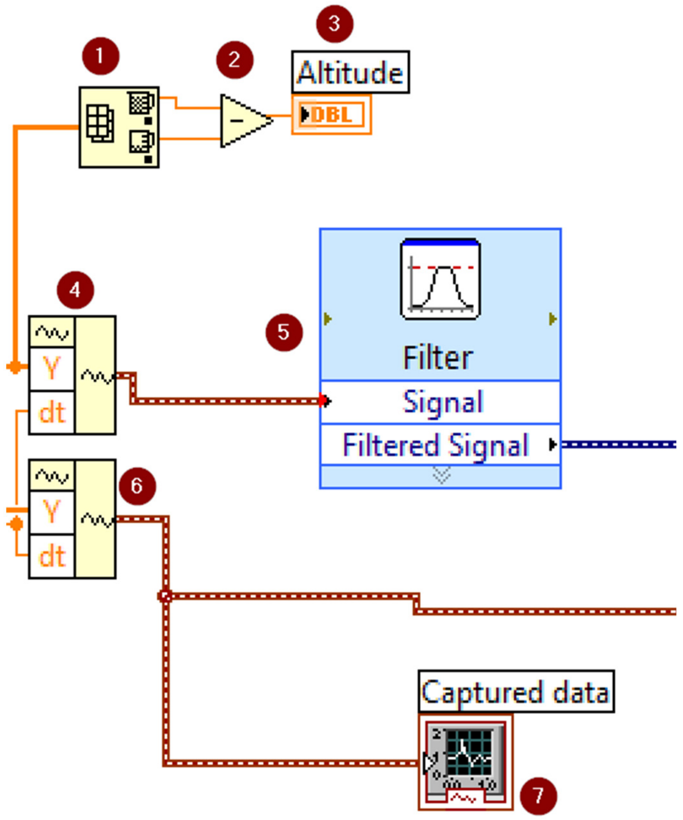
5. Cursor position analysis



1. **Reference –** Opens a reference to the specific graph that will be used.
2. **Property Node –** Determines the X position (in seconds) of cursor 0
3. **Property Node –** Determines the X position (in seconds) of cursor 1
4. **Multiply –** Multiplies the X value (in seconds) of cursor 0 by the sample collection rate to get the corresponding sample number.
5. **Multiply –** Multiplies the X value (in seconds) of cursor 1 by the sample collection rate to get the corresponding sample number.

6. **Subtract –** Subtracts the sample number of cursor 0 from the x value of cursor 1 to determine how many samples will be analyzed
7. **Absolute value –** Because a negative number of values cannot be analyzed, the absolute value of the number is determined for the event where cursor 1 is placed behind cursor 0.
8. **Array subset –** selects the portion of the collected data array starting at the value of cursor 0 and contains the number of samples to be analyzed (length) determined by item 6
9. **Divide –** Divides the loop rate by 1 to get the number of seconds between each sample (dt)
10. **Array to matrix –** Converts the collected data array into a matrix for ease of manipulation
11. **Get Matrix elements –** Gets the Y value of cursor 0 that corresponds the Cursor 0's X value
12. **Get Matrix elements –** Gets the Y value of cursor 1 that corresponds the Cursor 1's X value
13. **Numeric indicator –** Displays the numeric value of cursor 0 on the front panel
14. **Numeric indicator –** Displays the numeric value of cursor 0 on the front panel
15. **Property node (write mode) –** Moves the crosshair of cursor 0 to the Y value that corresponds to the current X value of cursor 0
16. **Property node (write mode) –** Moves the crosshair of cursor 1 to the Y value that corresponds to the current X value of cursor 1

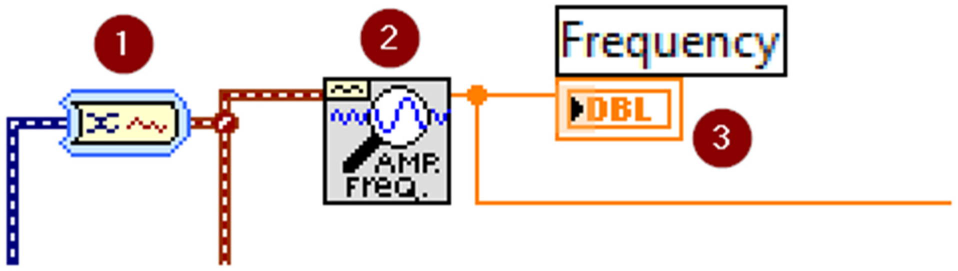
6. Data analysis



1. **Array max & min –** Determines the values of the maximum and minimum points in the array.
2. **Subtract –** Subtracts the min value from the max value to get the voltage difference.
3. **Numeric indicator –** Displays the numeric value the voltage difference on the front panel

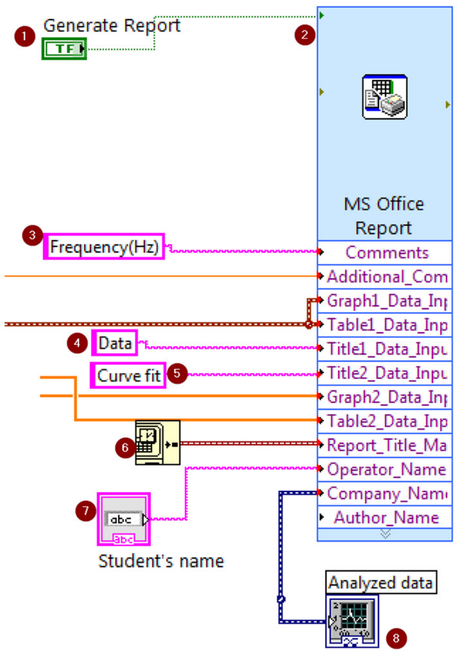
4. **Build waveform –** Builds a waveform using the values between cursor 0 and cursor 1
5. **Filter –** Uses a Smoothing filter with a half width moving average of 10 to filter out most of the "noise" That can happen when collecting data.
6. **Build waveform –** Builds a waveform using all values collected.
7. **Waveform Graph –** Graphs all of the data collected

7. Frequency Matching



1. **Convert from dynamic data –** Converts the filtered signal into a waveform
2. **Extract single tone information –** analyzes and determines the frequency of the waveform
3. **Numeric indicator –** Displays the numeric value of the waveform's frequency on the front panel

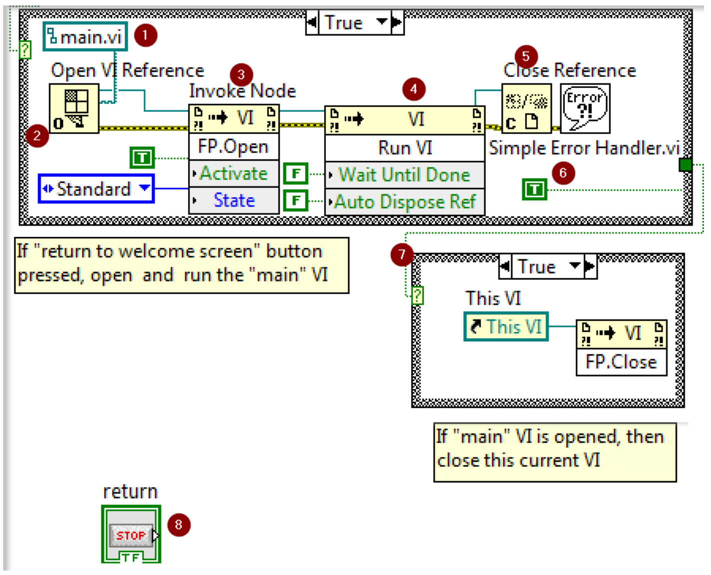
8. Data Output



1. **Button –** Tells the MS office report subvi to create a report when the user clicks the button.
2. **MS office report –** Creates an excel document based on a template that places inputted data to specific locations in the excel document.
3. **String constant –** Places a title above the Frequency cell.

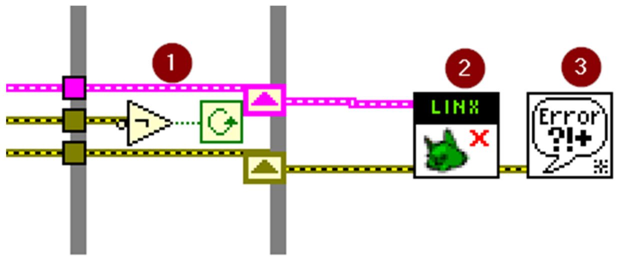
4. **String constant –** Places a title above the graph for all of the collected data.
5. **String constant –** Places a title above the curve fit graph.
6. **Get date/time in seconds –** Gets the current date and time and "timestamps" the report generated.
7. **String input –** Takes the words that the user inputs and includes it in the excel report.
8. **Graph –** Graphs the analyzed data on the front panel.

9. Return to main screen



1. **Vi Path –** Identifies the name of the vi to look for in the same directory.
2. **Open VI Reference –** Prepares the vi to receive commands.
3. **Invoke Node –** Tells the program to open the vi.
4. **Invoke node –** Tells the program to automatically run the vi.
5. **Close Reference –** Finishes activities with the .vi, and checks for errors.
6. **Boolean –** Enables the wire to activate the next case structure.
7. **Case structure –** Uses a vi reference and an invoke node to close the current vi (program).
8. **Button –** A button that is referenced in many places. If clicked, the button will trigger a value change in the referenced, signaling to program to return to the main screen and close the current window.

10. **Closing code**



1. **NOT –** If there is no error, then loop.

2. **LINX close –** Closes the connection to the Arduino.
3. **Simple error handler –** Checks for and reports any errors to the user.