How to use the VernierLib Library

version 073117

One of the best features about Vernier analog sensors is that they have a built-in autoID capability. However, to allow the Arduino to automatically detect which sensor you are using, you must include the VernierLib library in your sketch. This library also makes it easy to read distances with a Vernier Motion Detectort and to control the Vernier Digital Control Unit (DCU), which can be used with LED, lamps, buzzers, DC motors, and stepper motors.

Using the VernierLib Library

Once you have loaded the VernierLib library, there are several new functions you can use in your sketches; but to access the functions in the VernierLib library, you must start each sketch with the following two statements:

```
#include "VernierLib.h"
VernierLib Vernier
```

Vernier.autoID();

This function will check for an analog (BTA) sensor connected to the Analog 1 connector of the Vernier Interface Shield. It is only needed if you plan to read data from a Vernier analog (BTA) sensor. If a sensor is found, it will allow you to read information about it, including:

sensorName, shortName, sensorUnits, sensorNumber

calEquationType (identifies the type of equation used for calibration. For most it is 1 indicating a linear equation. Thermistor temperature sensors and a few others have complex calibration equeations.

Sensor calibration information, including: **slope**, **intercept**, **cFactor**(usde only for non linear calibrations)

voltageID (a few sensors are identified by a resistor included in them and this is the voltage read from a circuit when testing for them.)

page (sensors with memory chips in them can store multiple pages of calibration information. For example, some sensors have different pages for different units.)

To use the information about the sensor in your sketch, you use a statement like:

```
Serial.print(Vernier.sensorName());
```

This will print the name of the sensor on the Serial Monitor.

or:

```
Serial.print(Vernier.slope() );
```

This will print the slope used in the sensor's calibration.

In either case, you are calling a function in the library to have it return the parameter you requested. Note that any time you are getting information from the libary, you have to include "()" at the end of the name.

Vernier.readSensor();

This function reads an analog (BTA) Vernier sensor connected to the Analog 1 connector on the Vernier Interface Shield. Just add a line like:

```
sensorReading = Vernier.readSensor();
```

Here is a very simple sample sketch using the VernierLib library. This sketch is in our examples folder and is named VernierLibDemoSimple.

```
#include "VernierLib.h"
VernierLib Vernier;
float sensorReading;
void setup(void)
{
  Serial.begin(9600);
  Vernier.autoID();// this is the routine to do the autoID
}
void loop()
{
  sensorReading =Vernier.readSensor();
  Serial.print(sensorReading);
  Serial.print(" ");
  Serial.println(Vernier.sensorUnits());
  delay(500);//half a second
}
```

Vernier.readMotionDetector();

This function reads a Motion Detector (MD-BTA) connected to the Digital 1 connector on the Vernier Interface Shield. It returns the distance in centimeters. To use it, just add a line like:

```
distance = Vernier.readMotionDetector();
```

Here is a very simple sample sketch using the VernierLib library. . This sketch is in our examples folder and is named VernierLibDemoMotionDetector. Note thar this sketch does not need to include the AutoID statement, since it is not using an analog (BTA) sensor.

```
#include "VernierLib.h"
VernierLib Vernier;
float distance;
void setup(void)
{
  Serial.begin(9600);
```

Comment [M1]: I find the triple parentheses at the end confusing – the "sensorReading" statement is a much more typical use of this function

Comment [M2]: moved up

Comment [M3]: remove space

Comment [M4]: better example

```
}
void loop(void)
{
  distance =Vernier.readMotionDetector();
  Serial.print(distance);
  Serial.print(" cm");
  delay(100);//delay a tenth of a second
}
```

Vernier.DCU(DCUSetting);

This function sets the output lines on the Vernier Digital Control Unit (DCU) plugged into the Digital 2 port of the Vernier Interface Shield. Use the function with a statement like this:

```
Vernier.DCU(DCUSetting);
```

Where DCUSetting can be any integer from 0 to 15. To keep things simple, we recommend using only the first three DCU lines. You can control them and set them in any pattern using the following values for DCUSetting. Notice that this is the basic binary counting pattern.

Comment [M5]: Misleading - If you declare DCUSetting as a variable, it should be integer, correct?

	DCU Line		
DCUSetting	**D1**	**D2**	**D3**
0	Off	Off	Off
1	On	Off	Off
2	Off	On	Off
3	On	On	Off
4	Off	Off	On
5	On	Off	Of
6	Off	On	On
7	On	On	On

The DCU actually has 6 output lines and controlling lines D4, D5, and D6 is just a little more complicated. Here are the values of DCUSetting you can use and the output patterns:

	DCU Line					
DCUSetting	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**
8	Off	Off	Off	On	Off	Off
9	On	Off	Off	On	Off	Off
10	Off	On	Off	On	Off	Off
11	On	On	Off	On	Off	Off
12	Off	Off	Off	Off	Off	Off
13	Off	Off	Off	Off	Off	On
14	Off	Off	Off	Off	On	Off
15	Off	Off	Off	Off	On	On

Note that not all possible combinations of DCU " lines on" are possible.

See the sample sketches **VernierLibDemoDCUSimple** and **VernierLibDemoDCU** to see how this can be used

Vernier.DCUPWM(PWMSetting);

This function controls the effective voltage on line D4 of the Vernier Digital Control Unit (DCU) plugged into the Digital 2 port of the shield. Use the function with a statement like this:

```
Vernier.DCUPWM(PWMSetting);
```

Comment [M6]: remove space

Where PWMSetting can be any integerfrom 0 to 255. The Arduino will use pulse-width modulation (PWM) to adjust the duty cycle of the signal applied to DCU line D4 to control the voltage. 0 will produce 0 volts and 255 will produce a voltage approximately matching the voltage of the power supply used with the DCU.

See the sample sketch **VernierLibDemoDCUPWM** to see how this function can be used

Vernier.DCUStep(stepCount, stepDirection, stepDelay);

This function controls a stepper motor connected to a Vernier Digital Control Unit (DCU) plugged into the Digital 2 port of the shield. See https://www.vernier.com/engineering/arduino/dcu/stepper-motors/) for information on how to wire the stepper motor. Use the function with a statement like this:

Comment [M7]: I could not find info on wiring a stepper motor here, nor in the DCU manual.

```
Vernier.Step(stepCount, stepDirection, stepDelay);
```

Where:
stepCount can be any positive integer. It is the number of steps that the stepper motor will move.

stepDirection can be 0 or 1. It controls whether the stepper motor rotates clockwise or counter-clockwise.

stepDelay is the number of milliseconds the program delays between stepper motor steps. Note that if you make the delay too small, the stepper motor will not operate properly. 30 ms seems to work fine for most stepper motors we have tested.

See the sample sketch **VernierLibDemoStepper** in the Examples folder to see how this function can be used.

Sample Sketches

There are nine sample sketches included in the Examples folder:

VernierLibDemoSimple:

To use this sketch, use any analog sensor in the Analog 1. Run the sketch and sensor readings (with units) are displayed on the Serial Monitor. Change the "delay" (originally 500 ms) to change how fast the readings are made. Close the Serial Monitor and try the Serial Plotter. Plug in a different analog sensor and press the Reset button on the Vernier interface shield to use a different sensor. Close the Serial

Monitor and try the Serial Plotter to see a graph. Change the "delay" (originally 100 ms) to change how fast the readings are made.

VernierLibDemo:

This is the same as VernierLibDemoSimple, except it has a function added to display all the parameters associated with the sensor, like sensorName, shortName, slope, intercept, etc displayed on the Serial Monitor.

VernierLibDemoMotionDetector:

To use this sketch, connect a Motion Detector (MD-BTA) to the Digital 1 connector on the Vernier Interface Shield. Run the sketch and the Motion Detector's distance readings (in cm) are displayed on the Serial Monitor. Change the "delay" (originally 100 ms) to change how fast the readings are made. Close the Serial Monitor and try the Serial Plotter to see a graph.

VernierLibDemoDCUSimple:

This sketch sends the following values to the Vernier.DCU function: 0, 1, 2, 3, 4, 5, 6, 7. There is a one second delay between values. To try it out, connect a Digital Control Unit (DCU) to the Digital 2 connector on the shield. This sketch is meant to show how the DCU lines are controlled.

VernierLibDemoDCU:

To use this sketch, connect any analog sensor to the Analog 1 port and the Digital Control Unit to the Digital 2 connector on the shield. Run the sketch and sensor readings (with units) are displayed and lines D1, D2, and D3 of the DCU will turn on if the reading of the sensor is above the "threshold" set in the sketch. Change the threshold to change when the action takes place and change DCUsetting to change which lines of the DCU are turned on. This sketch is meant to show how the DCU lines can be used for a feedback and control system.

VernierLibDemoDCUPWM

Connect the Digital Control Unit (DCU) into the Digital 2 port. If you like, connect a small motor, a small lamp, or an LED and a resistor between the D4 connector and the GND connector of the DCU. (If you do not connect anything, you can just watch the red LED next to the D4 line and see how its brightness changes.) Run the sketch and the Pulse-Width Modulation output line, DCU line 4, will cycle through a pattern from low voltage, increasing, and then decreasing, over a few seconds. Then the pattern will repeat. This sketch is meant to show how the PWM line is controlled.

VernierLibDemoDCUStepper

Connect the Digital Control Unit (DCU) into the Digital 2 port. If you have a stepper motor available, connect it to the D1, D2, D3, D4, and GND connector' s of the DCU. Also, connect a power supply to the DCU. (If you do not connect anything, you can just watch the red LEDs of the DCU flash.) Run the sketch and the stepper motor will move stepCount steps clockwise and then the same number of steps counter-clockwise. Experiment with changing all of these parameters: stepCount, stepDirection (0 or 1),

and stepDelay (the number of milliseconds between steps). If you make the stepDelay too small, the stepper motor may not function properly.

VernierLibDemoWithDisplaySimple:

This sketch is used to display sensor information on a 2-line, 16-character display connected to the Digital 2 connector on the shield. Any analog sensor can be used, connected to the Analog 1 connector. To make the sketch easy to understand, only the short name of the sensor, the sensor units, and the sensor reading are displayed.

VernierLibDemoWithDisplay:

This sketch is similar to VernierLibDemoWithDisplaySimple, except many more parameters associated with the sensor are displayed. The display should be connected to the Digital 2 connector and the sensor to the Analog 1 connector on the shield.