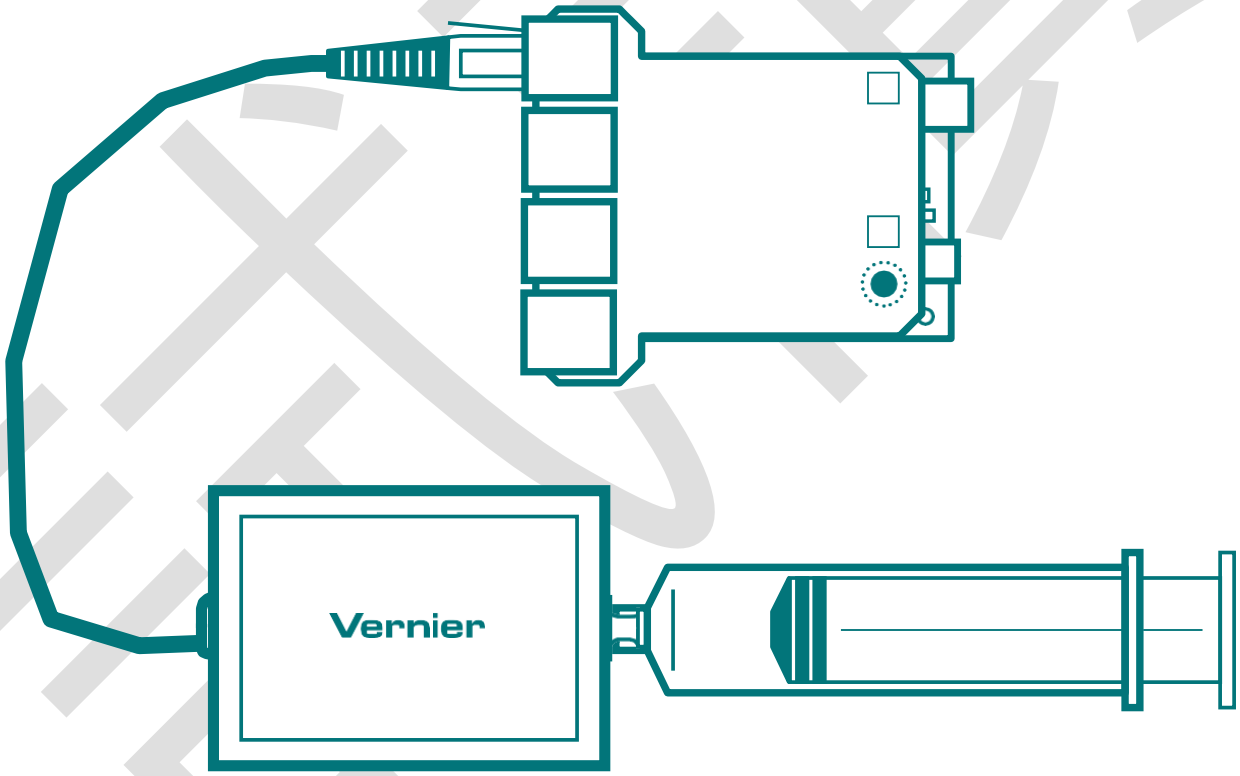
Vernier Coding Activities with Arduino®



VCA-AS-E

Vernier

Vernier Software & Technology • 888-VERNIER (888-837-6437)
info@vernier.com • vernier.com

Vernier Coding Activities with Arduino®

Analog Sensors

Vernier Software & Technology
13979 SW Millikan Way • Beaverton, OR 97005-2886
Toll Free (888) 837-6437 • (503) 277-2299 • Fax (503) 277-2440
info@vernier.com • www.vernier.com

Contents

Instructor Guide

Student Activities

Instructor Guide

Teaching Coding with Vernier and Arduino

INTRODUCTION

These activities are designed to provide an introduction to coding as well a lesson in sensor technology using Vernier sensors and Arduino microcontrollers. Students get excited when they see coding come to life through hands-on technology. Integrating Vernier sensor technology with Arduino connects the physical world to the computer-centric activity of learning to code.

Arduino has developed both its hardware and software as open source. The SparkFun RedBoard is equivalent (although not identical) to the Arduino Uno, and when using the Arduino software, it should be treated as an Uno.

In this series of activities, students learn about sensor technology and coding with our fun, interactive approach. Teaching students about the underlying physics in our technology opens the door for them to explore and become interested in how technology works.

Students need access to the following equipment for these activities:

- SparkFun RedBoard (or equivalent) with USB cable and power supply
- Vernier Analog Protoboard Adapter or Vernier Interface Shield
- Vernier Gas Pressure Sensor
- Chromebook or computer with Arduino software

Figure 1 Connecting a Vernier Gas Pressure Sensor to a Protoboard Adapter (left) and Interface Shield (right)

ACTIVITIES

There are eight activities in this series. Students begin with the basics of coding with an Arduino and quickly move to integrating our sensors into code. Your students will also be introduced to fundamental coding structures and learn how to access the VernierLib library.

1	Introduction to Arduino Programming	Students learn how to connect Arduino to their computer or Chromebook and modify the Blink program. Success in this activity enhances student confidence in the activities that follow.
2	Using Vernier Sensors with Arduino	Students are introduced to methods for connecting Vernier sensors to Arduino and observe the output from the sensor in the Serial Monitor.
3	Calibrating a Sensor	Students learn how Vernier sensors convert electrical signals into sensor readings.
4	Displaying Data	Students gain practical experience formatting output in the Serial Plotter and the Serial Monitor.
5	Functions: Part 1	Students explore integrating functions into their code to simplify the structure and make it easier to interpret.
6	Functions: Part 2	Students learn how to pass data from a function to another location in their code. They explore the difference between local and global variables.
7	Output and Logic Statements	Students learn how to designate and control output devices (LEDs) based on sensor values.
8	Using the VernierLib Library	Students are exposed to the tools that are incorporated in the VernierLib library and how to access these tools.

COMPUTER SCIENCE TEACHERS ASSOCIATION (CSTA) STANDARDS

CSTA Standard		Activity
3A-AP-13	Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.	2, 3, 5, 6
3A-AP-15	Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.	5, 6
3A-AP-16	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.	2, 4, 7
3A-AP-17	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.	1, 2, 3, 4, 5, 6, 7, 8
3A-AP-21	Evaluate and refine computational artifacts to make them more usable and accessible.	2, 3, 4, 6, 7, 8
3A-DA-11	Create interactive data visualizations using software tools to help others better understand real-world phenomena.	2, 3, 4
3B-AP-13	Illustrate the flow of execution of a recursive algorithm.	7
3B-AP-16	Demonstrate code reuse by creating programming solutions using libraries and APIs.	8
3B-DA-05	Use data analysis tools and techniques to identify patterns in data representing complex systems.	3, 4
3B-DA-06	Select data collection tools and techniques to generate data sets that support a claim or communicate information.	3

Using Vernier Sensors with Arduino

INTRODUCTION

Have you collected data with sensors for a science class? Most people collect data and accept without question that sensors and software work together to provide relatively accurate readings.

This activity is designed to remove some of the mystery of how sensors work.

OBJECTIVES

- Understand how sensors work
- Measure the output from a sensor using an Arduino microcontroller
- Convert the output of the Arduino into a voltage

MATERIALS

SparkFun RedBoard (or equivalent) with USB cable and power supply
Vernier Analog Protoboard Adapter or Vernier Interface Shield
Vernier Gas Pressure Sensor
Chromebook or computer with Arduino software

BACKGROUND

There are two ways to connect Vernier sensors to an Arduino board. One method requires simple wiring and a protoboard adapter, while the other involves using a shield that sits on top of the Arduino board and makes the necessary connections (see Figure 1).

Figure 1 Vernier Protoboard Adapter on a protoboard (l) and the Vernier Interface Shield (r)

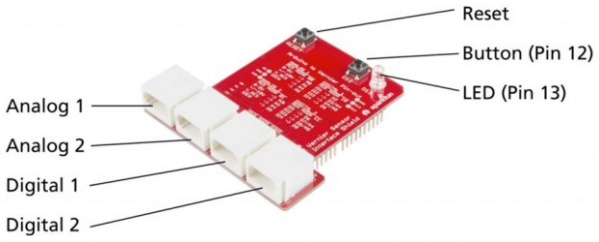


Figure 2 The Vernier Interface Shield pin headers connect to the RedBoard and make the connection between the sensor pins and the RedBoard.

The Vernier Interface Shield has four ports (two for analog devices and two for digital), a general purpose button (tied to digital pin 12), a Reset button, and an LED. The Reset button and LED are linked to the same components on the RedBoard. When the shield is mounted on the Arduino, the header pins connect the two together (see Figure 3).

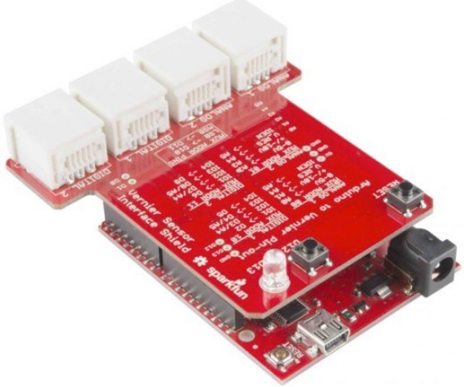


Figure 3 The shield has header pins that connect to the RedBoard.

PROCEDURE

Step 1: Connect hardware and software

a. Open the Arduino IDE software and connect your Arduino.

b. Select the board and COM port from the Tools menu (refer to the "Introduction to Arduino Programming" activity for more detailed directions).

c. Connect the Vernier Interface Shield or the Protoboard Adapter to the Arduino. If you are using the Protoboard Adapter, the following connections are required (see Figure 4):
   - GND (Vernier BTA pin 2) to Arduino pin GND (ground)
   - 5V (Vernier BTA pin 5) to Arduino pin 5V (power)
   - SIG1 (Vernier BTA pin 6) to Arduino pin A0

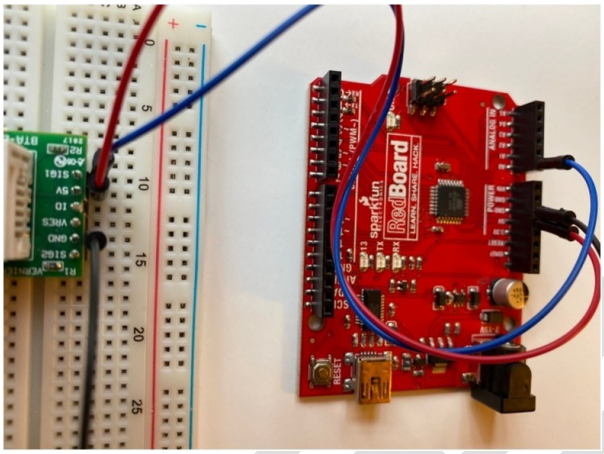


Figure 4 Connecting the Protoboard Adapter pins to the Arduino: 5V, ground, and Signal 1

d. Set the plunger on the syringe to about 10 mL, and then attach it to the Gas Pressure Sensor (see Figure 5).

Figure 5

e. Plug your sensor into the Vernier Interface Shield or Protoboard Adapter. If you are using the shield, plug the sensor into the Analog 1 port.

f. To open the sketch for this activity, select File>Examples > 01.Basics > AnalogReadSerial.

g. Verify and upload the AnalogReadSerial sketch by clicking Upload, . A message at the bottom of the screen will indicate the sketch has compiled and has been uploaded, and the program should run.

h. To monitor the output from the sensor, click the Serial Monitor icon, , or select Tools>Serial Monitor. The Serial Monitor window will open and display the sensorValue. Notice how the sensorValue changes when you push or pull on the syringe plunger.

SOFTWARE TIP

The Arduino IDE comes with an extensive collection of Example programs, one of which will be used in this activity. View a complete list of available programs by selecting File>Examples in the software.

Step 2: Review the AnalogReadSerial sketch

The AnalogReadSerial sketch is not specific to Vernier analog sensors; it can be used with any device that outputs a signal. The Vernier Pressure Sensor outputs a signal in the form of a voltage. The Arduino microcontroller converts the (analog) voltage reading to a digital reading between 0 and 1023 and prints the reading in the Serial Monitor. This is referred to as an analog-to-digital converter (ADC).

The value read by the Arduino board is often referred to as the "count" from the ADC. (Calculate the value of 2^{10} to see where the range of values has its "roots.") The count is proportional to the voltage signal from the sensor (Vernier sensors typically range from 0 to 5 volts).

```
/*
AnalogReadSerial
   Reads an analog input on pin 0, prints the result to the serial monitor.
   Graphical representation is available using serial plotter (Tools > Serial
     Plotter menu)
   Attach the center pin of a potentiometer to pin A0, and the outside pins
     to +5V and ground.

   This example code is in the public domain.
*/

// the setup routine runs once when you press reset:
void setup() {
   // initialize serial communication at 9600 bits per second:
   Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
   // read the input on analog pin 0:
   int sensorValue = analogRead(A0);
   // print out the value you read:
   Serial.println(sensorValue);
   delay(1); // delay in between reads for stability
}
```

Step 3: Modify the sketch

Follow these instructions to modify the sketch so the Serial Monitor displays the voltage rather than the count.

a. Modify the sketch to slow the sampling rate to two times per second. **Note**: The sampling rate is controlled by the delay statement at the end of the sketch. The number in the delay statement is measured in milliseconds.

b. Modify the sketch to calculate the voltage read by the microcontroller.
   i. Declare a variable named "sensorVoltage" by typing the following code within the curly brackets of the loop() function:

      ```
      float sensorVoltage; // declare a variable to store the voltage
      ```

   ii. Convert the "count" to a voltage by inserting the following line of code after the `int sensorValue = analogRead(A0)` line.

      ```
      sensorVoltage = sensorValue *5.0 / 1023;
      ```


iii. Now, add comments after your new line to explain what the code accomplishes. An example of how this could look follows, but you can use your own words.

```
sensorVoltage = sensorValue *5.0 / 1023; // calculates the sensor
voltage from the "count"
```

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  float sensorVoltage; // declare a variable to store the voltage
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  sensorVoltage = sensorValue *5.0 / 1023; // calculates the sensor voltage from the "count"
  // print out the value you read:
  Serial.println(sensorValue);
  delay(500);        // delay in between reads for stability
}
```

c. Change your sketch to print the voltage instead of the sensorValue in the Serial Monitor.

d. Confirm that the sketch operates as expected.

e. Save this program with a unique name. **Note**: If you are using the Arduino IDE (rather than the online editor), the software will save it in a folder with the same name.

EXPLORE

The line that calculates the sensorVoltage takes the count and multiplies it by 5 volts and divides it by 1023. The order of operations and the type of data (integer and floating point) cause the calculation to occur in specific ways. Conduct the following investigations and try to understand the differences:

- Multiply the sensorValue by 5 (vs. 5.0). This will result in an integer value. Can you cause this number to change?
- Change the order of operations in the calculation by first dividing by 1023 and then multiplying by 5.0. The first calculation (sensorValue/1023) gives an integer value of a number much closer to zero than one. What is the result of this order of operations?

After your investigations, record your findings:

OPTIONAL EXTENSIONS

Complete the following extensions if they are assigned by your instructor.

1. The "delay(1);" causes the program to pause for one millisecond before repeating the loop() function. Change the sampling rate so that it samples once every second.
2. Modify the AnalogReadSerial sketch to sample at slower rates as it runs.

Displaying Data

INTRODUCTION

In Activity 3, "Calibrating a Sensor," you displayed data using the Serial Monitor. In this activity, you will see how the Serial Plotter can be used to easily view data in a graph. You will also experiment with how you can use code to format the data in the Serial Monitor to make it more useful.

OBJECTIVES

- Understand formatting output to the Serial Plotter (computer only)
- Understand formatting output to the Serial Monitor

MATERIALS

SparkFun RedBoard (or equivalent) with USB cable and power supply
Vernier Analog Protoboard Adapter or Vernier Interface Shield
Vernier Gas Pressure Sensor and syringe
Chromebook or computer with Arduino software

PROCEDURE

Part A: Serial Plotter

The Serial Plotter is a great way to quickly display data from your sketch. Follow the steps below to see how simple and powerful this tool can be. **Note**: The Serial Plotter is only available on the Arduino IDE (computer software). If using the Web Editor, take a look at Figure 1 to see an example of the Serial Plotter graph and then skip to Part B.

Step 1: Connect hardware and software

a. Open the Arduino IDE software and connect your Arduino and sensor.

b. Select the board and COM port from the Tools menu.

c. Open the sketch you saved at the end of Activity 3.

d. Upload the program.

Step 2: Explore the Serial Plotter

a. Select Tools>Serial Plotter. You should see a graph appear; the graph displays the sensor reading over time.

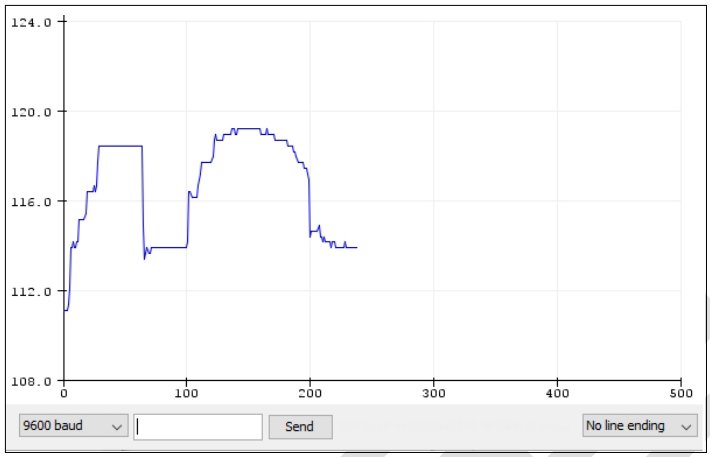


Figure 1 Example of a Serial Plotter graph

b. Push and pull on the syringe and notice how the line responds to changes as you adjust the pressure with the syringe.

c. Look at the sketch and count the number of actual lines of code necessary to create this graph. It is quite remarkable to be able to graph sensor data with so little code.

Part B: Serial Monitor

Step 1: Explore the Serial Monitor

a. Close the Serial Plotter (if it is open).

b. To open the Serial Monitor, click the Serial Monitor icon, , or select Tools>Serial Monitor. Note that you can only have one of these tools open at a time, not both.

c. Push and pull on the syringe and observe how the values change as you adjust the pressure.

Notice how the simple number in the Serial Monitor is not very helpful after having seen a graph of the data. Let's add some complexity to the sketch to make the display in the Serial Monitor more useful.

Step 2: Create string variables

Strings are variables that contain letters or words. Create three string variables that will be used to make a header in the Serial Monitor: Gas Pressure Sensor, Pressure, and kPa. To create the variables, insert the following text after the `Serial.begin (9600);` line of code in the setup() function:

```
char sensor [ ] ="Gas Pressure Sensor";
char measurement [ ] = "Pressure";
char units [ ] =" (kPa)";
```


SOFTWARE TIP

Notice that some of the code changes color as you type. These commands are recognized by the software as having additional associated information.

- `char` indicates the variable is a 'character' or "string".
- `float` indicates the variable is a number that has a decimal point.
- `int` indicates a variable that is an integer number.

You can find additional information about these (and other) types of variables at **https://www.arduino.cc/reference/en**

Arduino's built-in Serial Monitor has a large family of functions that allow you to format the Serial Monitor output. In this activity, you will focus on the following:

- `Serial.print();  // prints the item in the parentheses`
- `Serial.println();  // prints the item in the parentheses and a carriage return`
- `Serial.print("\t");  // prints a tab`

REVIEW

Go to **https://www.arduino.cc/reference/en/language/functions/communication/serial/print** and review how the data in the parentheses are handled. Answer the following questions.

1. What is the default number of decimal places the Serial Monitor provides? _____
2. Do words and letters need to be placed within quotation marks? _____

Step 3: Create a header

a. Use the string variables you created earlier in the setup () function of your sketch to create a "header" that prints once and includes the following:
   - Sensor name
   - On a new line, print the heading "Pressure", followed by " (kPa)".
   - On the next line (and subsequent lines), the sensor reading should print out, one reading per line.

b. Compile and upload your sketch to verify it works as expected. Modify the formatting until you are satisfied with its appearance. Save your file as you refine your sketch.

Step 4: Modify the sketch to print a timestamp for each data point

a. In the space immediately above setup() function, insert the following line of code to declare a "global" variable (you will learn more about global variables in later activities):

```
int readingNumber; // global variable to count number of samples taken
```

b. In the loop () function, insert the following code immediately after the first curly bracket: "{":

```
// calculates the time for each reading
float timeBetweenReadings = 500; // in ms

float sensorTime = timeBetweenReadings/1000 * readingNumber; // calculates
  time
readingNumber++; // increments readingNumber by 1 for next reading
```

c. Modify the Serial.print statements so that the heading of the data includes "time (s)" and the data includes the timestamp on the same line followed by a carriage return for the next reading.

d. Upload and observe the output from your new sketch in the Serial Monitor.

e. Save your sketch.

SOFTWARE TIP

The Serial Monitor has many useful functions in addition to the functions you used in this activity. For a complete list and explanations, visit
https://www.arduino.cc/reference/en/language/functions/communication/serial

OPTIONAL EXTENSIONS

Complete the following extensions if they are assigned by your instructor.

1. Try out the Serial Plotter with the formatting for Serial Monitor in place. What is unusual about the graph plotted? Can you fix the problem?
2. Modify the sketch so that it displays and graphs pressure in millimeters of mercury instead of kilopascal.

