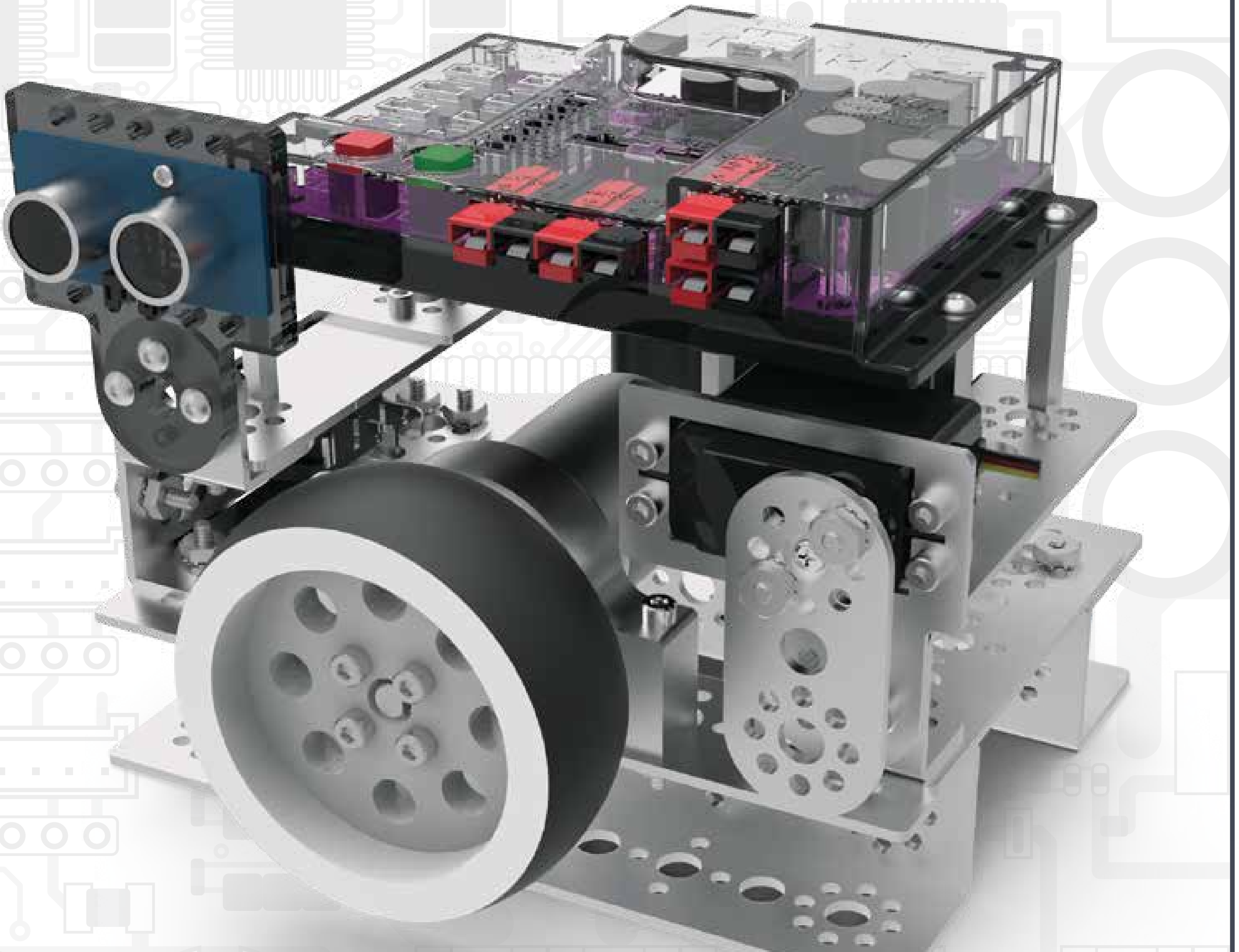
TETRIX® PRIZM™ Workshop Guide



43527

Who Are We and Why Are We Here?

Who is Pitsco?

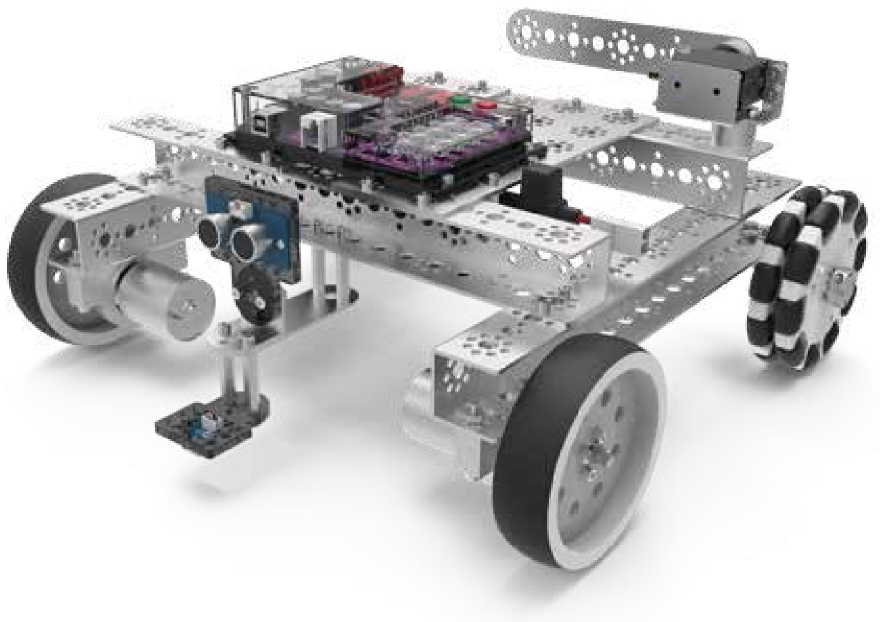
Pitsco's unwavering focus on innovative educational solutions and unparalleled customer service began when the company was founded in 1971 by three teachers. Using product flyers promoting just a handful of kits and related curriculum in the beginning, Pitsco expanded rapidly from a humble dream to a multifaceted corporation with thousands of products, more than 200 employees, and a simple vision: **Leading education that positively affects learners.**

What do we do?

The company helps students excel with a variety of science, technology, engineering, and math (STEM) classroom solutions that are equally robust and engaging. Our age-appropriate, student-centered K-12 learning solutions in STEM and acceleration comprise standards-based, relevant, hands-on activities delivered via a student-focused learning process. At Pitsco, every product we engineer, every activity we write, every curriculum we develop, and every solution we design is provided for the purpose of helping students use their hands to engage their minds to learn, grow, and succeed – in the classroom and in life.

What is TETRIX®?

Introduced in 2008, TETRIX® is the metal building system from Pitsco, currently comprising TETRIX MAX and TETRIX PRIME. As a great platform for robotics, it has become the building system of choice for many robotics competitions as well as the preferred choice in classrooms across the country and around the world. In 2016, Pitsco Education introduced the TETRIX PRIZM™ Robotics Controller, a proprietary brain for MAX robots geared at bringing coding to life for students.



Notes

Workshop Agenda

- Communicate logistics from classroom implementation
- Introduction to TETRIX MAX, PRIZM controller, and programming language
- Hands-on building experience
- Hands-on programming experience
- Discuss STEM connections and educational value for students
- Present classroom activity options
- Have fun!

Logistics for Classroom Implementation

Physical location

- Room to move – The room should have building space/tables as well as an area to demo the robot after it is built. Keep in mind that space will depend on the number of students as well as the platform.

Necessary material

- The recommended ratio of participants to sets would be two participants for each set. If computers are needed, the same ratio would apply – two participants per computer. If computers are used, keep in mind that software must be preloaded and ready to use. Any extra activity-specific material must be on hand: measuring devices, tape, play/field elements, mats, and so on.

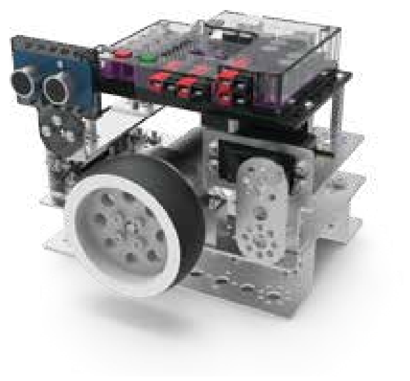
Front-end prep

- Any necessary software must be preloaded and ready to use, which might require collaboration with on-site IT staff.
- Ensure all sets have been inventoried and sorted and are complete and ready to go.
- Batteries must be accounted for (for remote controls as well as robots) and charged if needed.
- Inquire about or confirm any special needs or requirements for participants.

Notes

TETRIX® PRIZM™ Test Bed Activities

After completing the construction of the PRIZM™ Test Bed, it is time to have some fun. We will work through a series of activities where we program the PRIZM controller to control a variety of motors and sensors plugged into the test bed.



Types of Activities
Each of the five activities is designed to introduce you to the *Arduino Software (IDE)* and how it works with the PRIZM controller and select basic hardware. Success with these five activities demonstrates how easy it is to get started building and programming robots using the TETRIX® solution.

Activity 1: Hello World!
- Complete an intro activity for beginner to blink onboard LED.

Activity 2: Moving Your DC Motors
- Keep things simple – use PRIZM to control one TETRIX DC Motor.

Activity 3: Moving Your Servo Motors
- Rotate a servo motor – work with positioning.

Activity 4: Introduction to the Line Finder Sensor
- Introduce sensors and how to work with the Line Finder Sensor using PRIZM.

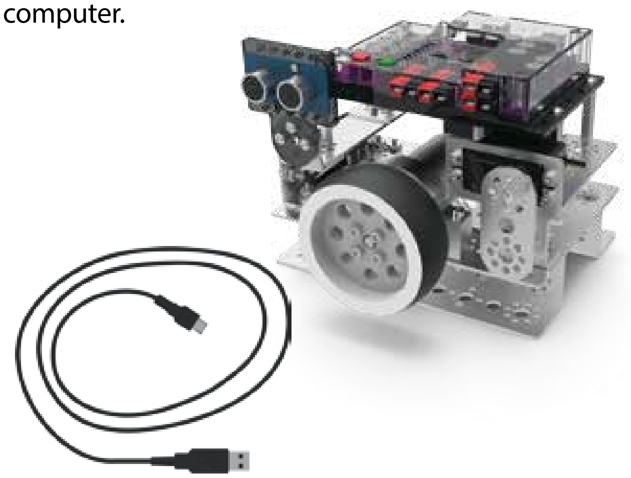
Activity 5: Introduction to the Ultrasonic Sensor
- Work with the Ultrasonic Sensor using PRIZM.

Notes

Activity 1: Hello World!

Let's begin with a very simple sketch that will blink the onboard PRIZM red LED. This activity will be the PRIZM equivalent of a Hello World! program, which is usually the intro activity for any beginning programmer. The sketch we will create is one of the simplest and most basic PRIZM functions and requires only the PRIZM, a power source, and a USB connection to the computer.

Notes

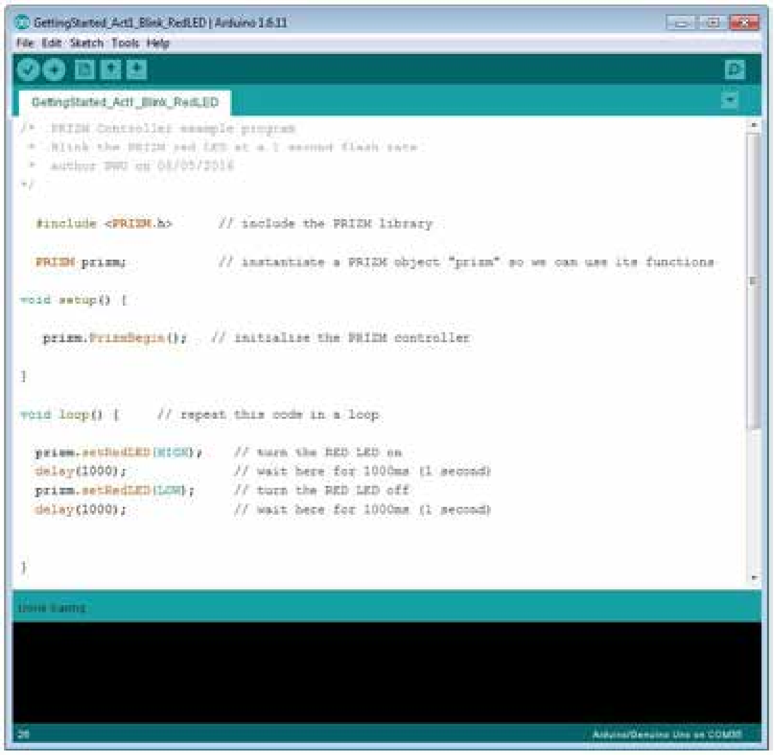


Parts Needed:

- PRIZM Test Bed
- USB cable
- Computer

Opening the Sketch

Let's start by looking at our first example sketch. Open the sketch by selecting **File > Examples > TETRIX_PRIZM > GettingStarted_Act1_Blink_RedLED**. A new sketch window will open titled GettingStarted_Act1_Blink_RedLED (Figure 1).


GettingStarted_Act1_Blink_RedLED | Arduino 1.6.11
File Edit Sketch Tools Help
GettingStarted_Act1_Blink_RedLED
Done Saving.
Arduino/Genuino Uno on COM8

```
/*  PRIZM Controller example program
 *  Blink the PRIZM red LED at a 1 second flash rate
 *  author PWU on 08/05/2016
*/

  #include <PRIZM.h>      // include the PRIZM library

  PRIZM prizm;            // instantiate a PRIZM object "prizm" so we can use its functions

void setup() {

   prizm.PrizmBegin();   // initialize the PRIZM controller

}

void loop() {     // repeat this code in a loop

  prizm.setRedLED(HIGH);     // turn the RED LED on
  delay(1000);               // wait here for 1000ms (1 second)
  prizm.setRedLED(LOW);      // turn the RED LED off
  delay(1000);               // wait here for 1000ms (1 second)


}
```

Figure 1

Building the Knowledge Base
Before we can upload the sketch to the PRIZM, we need to make sure the PRIZM has power, is connected to the computer, and is detected by the computer (Figure 2). When the PRIZM is connected as shown, power on the PRIZM with the on/off switch. You will know the PRIZM has power by the glowing blue light. To see if the PRIZM is detected by the computer, check the port.

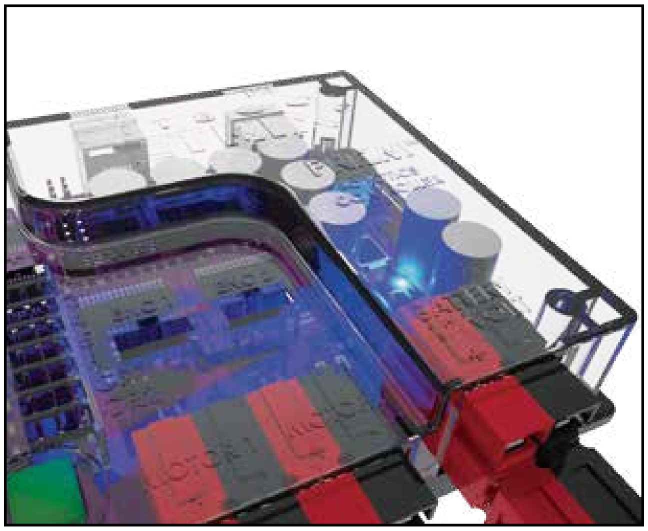


Figure 2

Executing the Code
To upload the sketch to the PRIZM, click **Upload** (Figure 3).

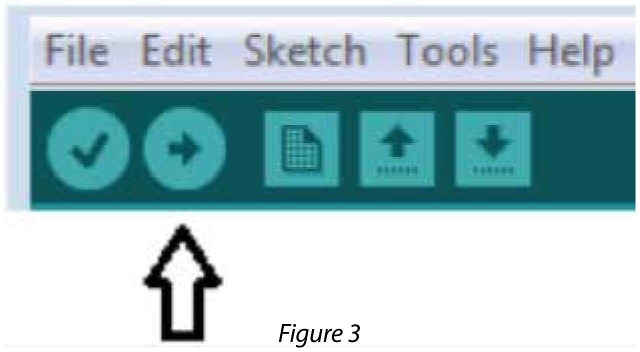


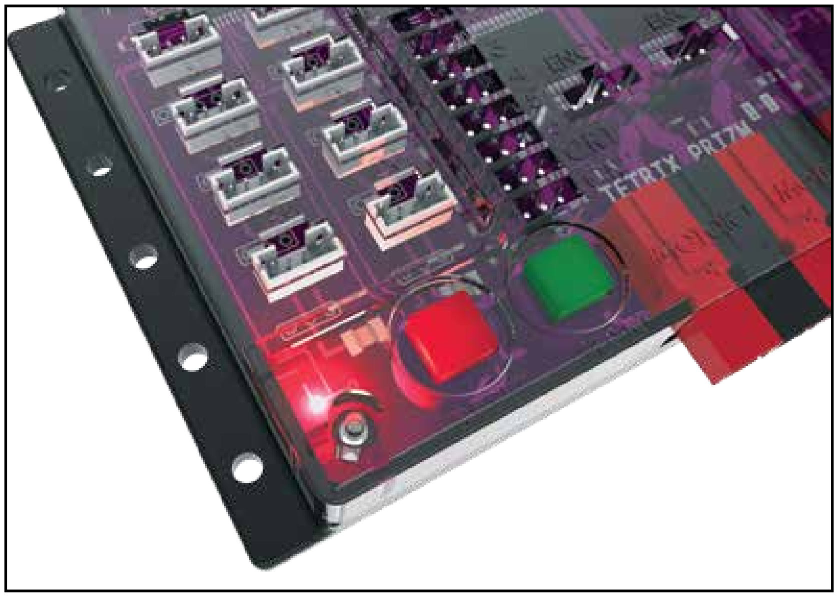
Figure 3

As the data uploads, the yellow LEDs to the side of the USB connection will flash. When the upload is finished, there will be a solid green LED beside the red Reset button. The green LED means the code is ready to execute. Press Start to execute the code. The red LED next to the Reset button will blink off and on in one-second intervals. To stop the program, press the Reset button.

Congratulations! You have successfully uploaded your first sketch to the PRIZM and demonstrated the results to the world.

Notes

Moving Forward
For now, let's change some parameters in our code and see how it affects the behaviour of the red LED. According to the comments in the example, the delay function defines the duration the LED is on or off. This is a parameter we can change in our code. Experiment with changing those values to create new blinking behaviors for the LED. Try making the LED blink faster or slower.



Real-World Connection
Many things within the electronic world around us blink, such as caution lights for road construction warnings, waffle irons (the blinking light turns solid when the waffles are ready), or notifications on our phones indicating that we have incoming calls. The rate at which these items blink – or when they blink – is controlled by electronics. This control can be from simple timing circuits – or it can be from computers or other devices with microprocessor chips.

STEM Extensions
Science

- Electricity terms (voltage, current, and resistance)
- How an LED works
- What determines the colour of an LED

Technology

- How computers work
- How computers are programmed

Engineering

- Problem-solving process

Math

- Frequency
- Duration
- Sequence

Notes

Hacking the Code Activity

With the example as a reference, try creating the blinking LED in a new sketch. Instead of just blinking the red LED, try to blink the green LED too. Flashing or blinking lights have a rich tradition as a method of signaling or long-distance communication. You could challenge yourself to communicate "Hello World!" in blinking Morse code.

To start a new sketch, select **File > New**.

When creating your own sketch, there is a built-in software tool to help ensure your code is free of syntax errors. You can check your syntax by clicking **Verify** (Figure 4). This will cause the code to compile but not upload. You will be prompted to save your sketch before verification.

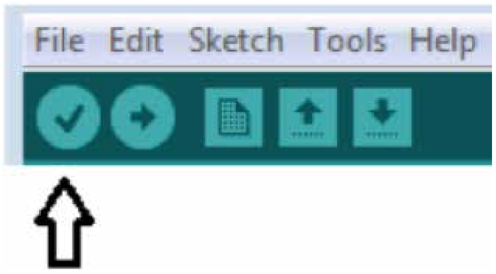
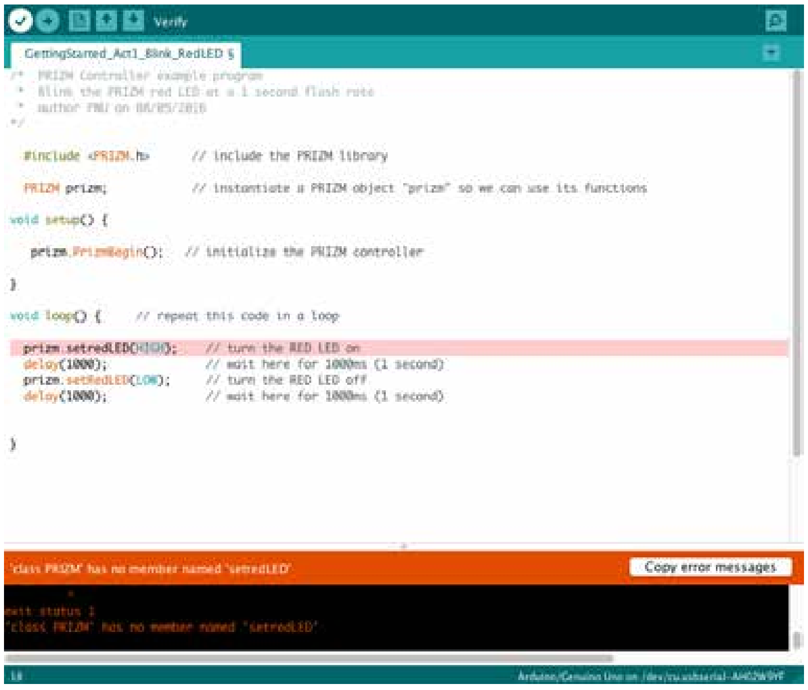


Figure 4

If there are errors in the code, they will be displayed in the compiler error window at the bottom of the sketch window (Figure 5).

Errors will need to be corrected before code can be uploaded to the PRIZM controller. If there are no errors, the compiler will complete and indicate that it is done compiling, and you can upload your code.



```
/*  PRIZM Controller example program
 *  Blink the PRIZM red LED at a 1 second flash rate
 *  author PWU on 08/05/2016
*/

  #include <PRIZM.h>      // include the PRIZM library

  PRIZM prizm;            // instantiate a PRIZM object "prizm" so we can use its functions

void setup() {

   prizm.PrizmBegin();   // initialize the PRIZM controller

}

void loop() {     // repeat this code in a loop

  prizm.setredLED(HIGH);     // turn the RED LED on
  delay(1000);               // wait here for 1000ms (1 second)
  prizm.setRedLED(LOW);      // turn the RED LED off
  delay(1000);               // wait here for 1000ms (1 second)


}
```

```
'class PRIZM' has no member named 'setredLED'
```

Copy error messages

```
exit status 1
'class PRIZM' has no member named 'setredLED'
```

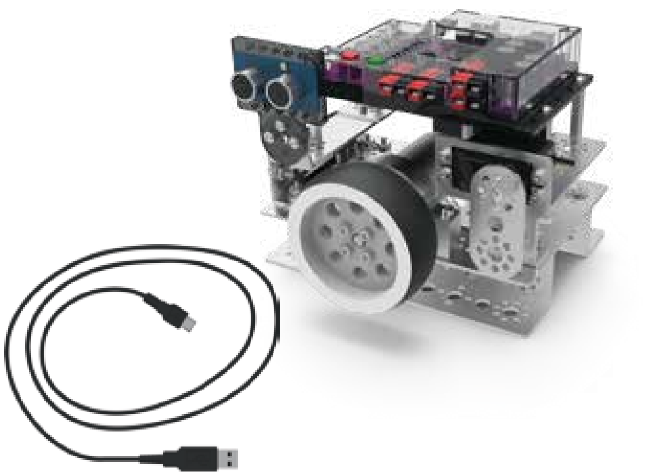
18 Arduino/Genuino Uno on /dev/cu.usbserial-AH02W9YF

Figure 5

Notes

Activity 2: Moving Your DC Motors

For our second activity we want to keep things simple but add an element of motion. We will create a sketch that will rotate a TETRIX DC Motor.

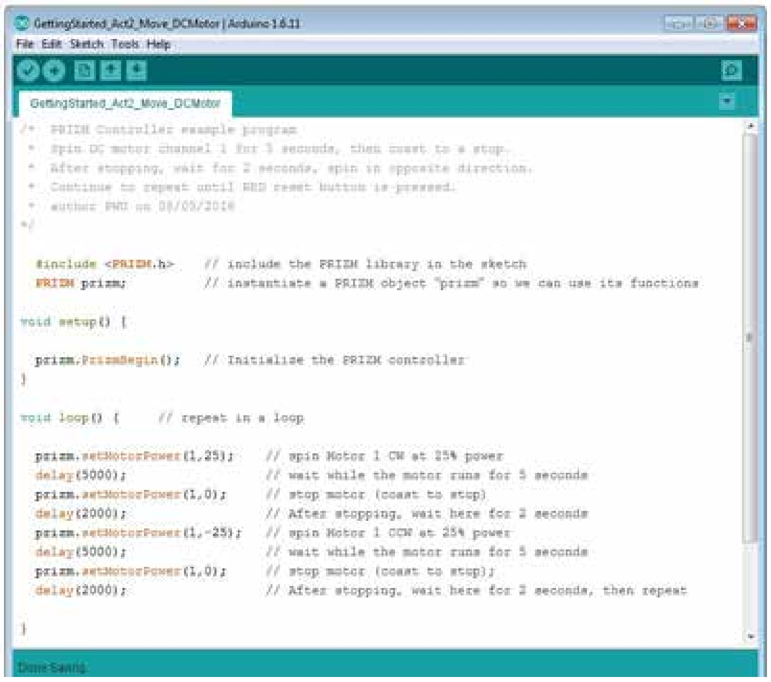


Parts Needed:

- PRIZM Test Bed
- USB cable
- Computer

Opening the Sketch

Before we open our next example sketch, be sure to save any sketch you want to reference later. Open the sketch by selecting **File > Examples > TETRIX_PRIZM > GettingStarted_Act2_Move_DCMotor.** A new sketch window will open titled GettingStarted_Act2_Move_DCMotor (Figure 6).


GettingStarted_Act2_Move_DCMotor | Arduino 1.6.11
File Edit Sketch Tools Help
GettingStarted_Act2_Move_DCMotor
Done Saving.

```
/*  PRIZM Controller example program
 *  Spin DC motor channel 1 for 5 seconds, then coast to a stop.
 *  After stopping, wait for 2 seconds, spin in opposite direction.
 *  Continue to repeat until RED reset button is pressed.
 *  author PWU on 08/05/2016
*/

  #include <PRIZM.h>    // include the PRIZM library in the sketch
  PRIZM prizm;          // instantiate a PRIZM object "prizm" so we can use its functions

void setup() {

  prizm.PrizmBegin();   // Initialize the PRIZM controller
}

void loop() {     // repeat in a loop

  prizm.setMotorPower(1,25);    // spin Motor 1 CW at 25% power
  delay(5000);                  // wait while the motor runs for 5 seconds
  prizm.setMotorPower(1,0);     // stop motor (coast to stop)
  delay(2000);                  // After stopping, wait here for 2 seconds
  prizm.setMotorPower(1,-25);   // spin Motor 1 CCW at 25% power
  delay(5000);                  // wait while the motor runs for 5 seconds
  prizm.setMotorPower(1,0);     // stop motor (coast to stop);
  delay(2000);                  // After stopping, wait here for 2 seconds, then repeat

}
```

Figure 6

Notes

Building the Knowledge Base
In this second sketch we want to take a closer look at the sketch syntax, specifically comments. Comments are lines in the program that are used to inform yourself or others about the way the program works. They are for informational purposes only and do not affect the code.

There are two types of comments: single line and multiline. Single-line comments are preceded by // and anything after // is a comment to the end of the line. Multiline comments are preceded by /* and can be several lines long but must be closed with */.

When you look at this second sketch, the comments explain how the sketch author intended this program to work. The intent of this sketch is to spin the DC motor channel 1 for five seconds and then coast to a stop. After two seconds of coasting, the motor will spin in the opposite direction. This behavior will continue until the red Reset button is pressed.

Executing the Code
Before we can upload the sketch to the PRIZM, remember to check our connections.

Upload the sketch. The green LED will light up, indicating that the code is ready to execute. When this has happened, press the green Start button on the PRIZM controller.

Observe the direction and duration of the motor rotation. Based on the sketch comments, did the behavior match expectations?

Press the red Reset button when you are ready to stop the motor.

Moving Forward
The function used in this sketch is prizm.setMotorPower. It has two parameters: motor channel and motor power. In the example, prizm.setMotorPower(1,25) means motor 1 will spin at 25% power clockwise. The first value in parentheses defines the motor channel, and the second value in parentheses defines power percentage and direction.

We can alter the direction and stopping behavior by changing the second value in parentheses. If the second value is negative, the motor rotates counterclockwise, as shown in the sketch. If we change the value to 125 instead of 0, the stopping behavior will change from coast to brake.

Practice changing the parameters in the sketch. We can change the motor power, motor direction, stopping behavior, and delay between functions. Observe the effect these changes have on the motor.

Notes

Real-World Connection
Controlling motors is not a new thing. Controlling them by an onboard computer in an electric car (such as a Tesla) at 70 mph down the road or during a sharp turn on a curvy highway – that is new! For these cars that are driven by electric motors, the speed and power levels of all the drive motors must be coordinated to make the turn in the highway as the driver is turning the steering wheel. All this has been programmed into the brains of these cars to make it simple and easy for the driver.

STEM Extensions
Science

- How DC motors work
- Angular velocity

Technology

- Relationship between power, voltage, and current
- Torque

Engineering

- Determining load and torque

Math

- Pulse width modulation (PWM)
- Revolutions per minute (rpm)

Hacking the Code Activity
With the example as a reference, try creating a new sketch to move your DC motor. Remember what we learned from our first activity and think of creative ways to include blinking LEDs with your rotating motor.

Notes

Activity 3: Moving Your Servo Motors

Our third activity will explore another element of motion with servo motors. We will create a sketch to rotate a servo motor.

Servo motors offer the benefit of being able to move to a set position regardless of the start position within a limited range of motion. Servo motors have an approximate range of motion from 0 to 180 degrees. For example, we can tell a servo to go to position 45 degrees regardless of where it starts. If it starts at 0 degrees, it will move clockwise to 45 degrees. If it starts at 120 degrees, it will move counterclockwise to 45 degrees.

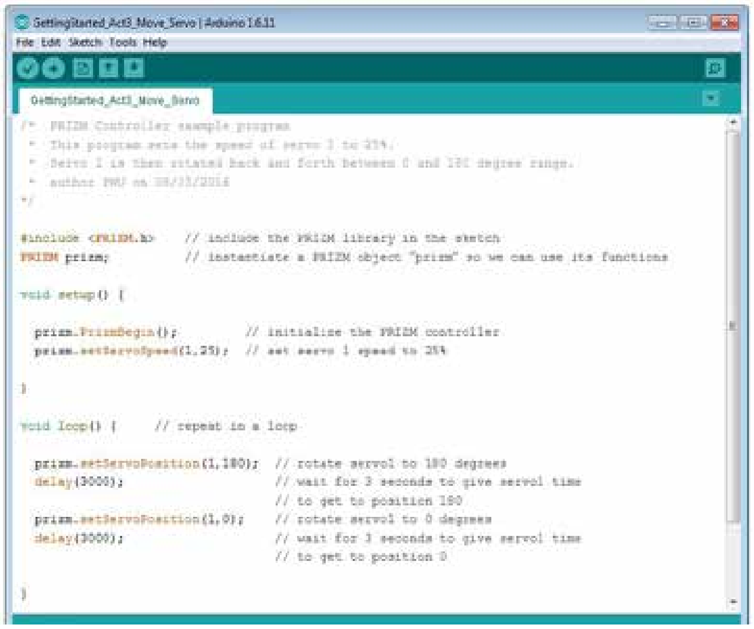
Parts Needed:

- PRIZM Test Bed
- USB cable
- Computer

Opening the Sketch

Before we open our next example sketch, be sure to save any sketch you want to reference later.

Open the sketch by selecting **File > Examples > TETRIX_PRIZM > GettingStarted_Act3_Move_Servo**. A new sketch window will open titled GettingStarted_Act3_Move_Servo (Figure 7).


GettingStarted_Act3_Move_Servo | Arduino 1.6.11
File Edit Sketch Tools Help
GettingStarted_Act3_Move_Servo

```
/*  PRIZM Controller example program
 *  This program sets the speed of servo 1 to 25%.
 *  Servo 1 is then rotated back and forth between 0 and 180 degree range.
 *  author PWU on 08/05/2016
*/

#include <PRIZM.h>    // include the PRIZM library in the sketch
PRIZM prizm;          // instantiate a PRIZM object "prizm" so we can use its functions

void setup() {

  prizm.PrizmBegin();         // initialize the PRIZM controller
  prizm.setServoSpeed(1,25);  // set servo 1 speed to 25%

}

void loop() {     // repeat in a loop

  prizm.setServoPosition(1,180);  // rotate servo1 to 180 degrees
  delay(3000);                    // wait for 3 seconds to give servo1 time
                                  // to get to position 180
  prizm.setServoPosition(1,0);    // rotate servo1 to 0 degrees
  delay(3000);                    // wait for 3 seconds to give servo1 time
                                  // to get to position 0

}
```

Figure 7

Notes

Building the Knowledge Base
In this third sketch we want to continue looking at sketch syntax, specifically the include statement and the object declaration. The include statement is used to add the functionality of the PRIZM software library into the sketch. The object declaration is a technique used when writing programs to make them easier to manage as they grow in size and complexity.

The PRIZM software library is a collection of special mini programs each with its own distinct function name. These mini programs are designed to make writing sketches for PRIZM easy and intuitive. By using the include statement, we add the functionality of the library to our sketch. The include statement for PRIZM is #include <PRIZM.h>.

The object declaration is an important statement when using the PRIZM controller along with the PRIZM software library. In order to use the functions contained in the PRIZM software library, we must first declare a library object name that is then inserted as a "prefix" before each library function. The object declaration we use is PRIZM prizm;. We use this statement just after the include statement.

In all of our sketch examples, we use an include statement and object declaration to add the functionality of the PRIZM software library. Include statements and object declarations are common for most forms of C-based language.

Executing the Code
Before we can upload the sketch to the PRIZM, remember to check our connections.

Upload the sketch. The green LED will light up, indicating the code is ready to execute. When this has happened, press the green Start button on the PRIZM controller.

Observe the direction and duration of the servo motor rotation. Based on the sketch comments, did the behavior match expectations?

Press the red Reset button when you are ready to stop the motor.

Notes

Moving Forward
In this sketch we introduce two new PRIZM functions, prizm.setServoSpeed and prizm.setServoPosition. Both functions have two parameters, but they are different.

The two parameters of the prizm.setServoSpeed function are servo channel and servo speed. In the example, prizm.setServoSpeed(1,25) means servo 1 will spin at 25% power while it rotates to the position commanded by the prizm.setServoPosition function. This function needs to be called only once at the beginning of the program.

The two parameters of the prizm.setServoPosition function are servo channel and target position. In the example, prizm.setServoPosition(1,180) means servo 1 will rotate to the target position of 180 degrees.

In the sketch both of these functions work together to tell the servo motor not only the target position but also the speed to use while moving to the target position. We can alter the position and speed of the servo by changing the values of both functions.

Practice changing the parameters in the sketch. Observe the effect these changes have on the servo motor.

Real-World Connection
Historically, servo motors were used mostly with a remote-controlled (R/C) transmitter for R/C model cars to control steering or R/C model airplanes to control flaps and rudders. Robots can use servos controlled by R/C transmitters – but they can also use servos controlled by PRIZM to operate robotic arms, grippers, tilting and rotating mounts for cameras, or many other applications in which precise movement is needed.

STEM Extensions
Science

- Levers
- Centripetal force

Technology

- Mechanical linkages
- Transmission of force

Engineering

- Applying torque

Math

- Radian vs Cartesian measurements
- Arc length

Hacking the Code Activity
With the example as a reference, try creating a new sketch to move your servo motor. Remember what we learned from our previous activities and think of creative ways to combine the functions you have learned.

Notes

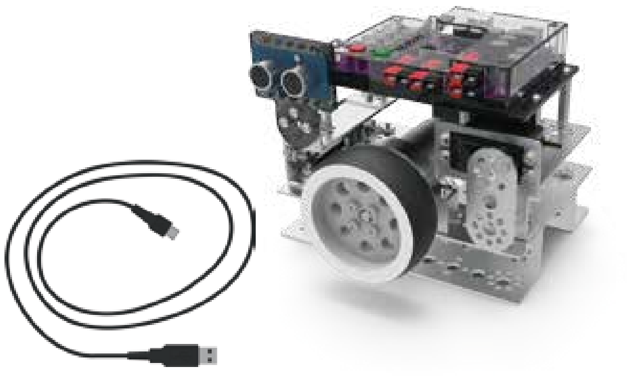
Activity 4: Introduction to the Line Finder Sensor

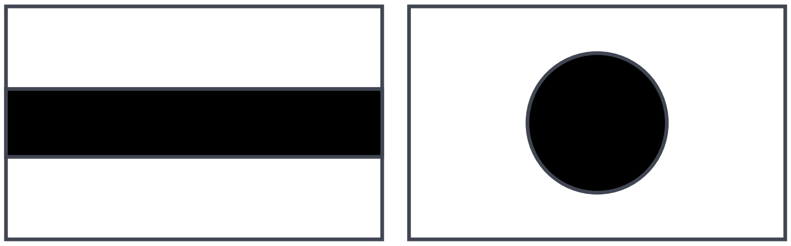
For the fourth activity we will change focus from motor outputs to sensor inputs by introducing and exploring sensors. In this example we will connect a Line Finder Sensor to digital sensor port D3, and we will create a sketch to read a digital input from the Line Finder Sensor.

Sensors enable us to gather information from the world around us. The type of information depends on the type of sensor. The Line Finder Sensor uses reflected infrared light to distinguish between light and dark surfaces.

Parts Needed:

- PRIZM Test Bed
- USB cable
- Computer



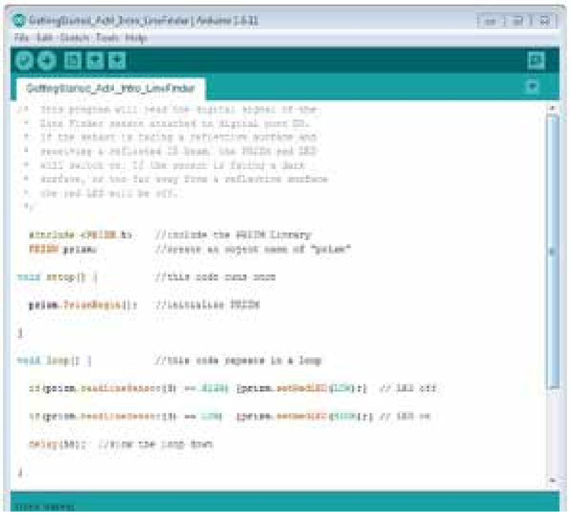


Contrasting light and dark surface

Opening the Sketch

Before we open our next example sketch, be sure to save any sketch you want to reference later.

Open the sketch by selecting **File > Examples > TETRIX_PRIZM > GettingStarted_Act4_Intro_LineFinder**. A new sketch window will open titled GettingStarted_Act4_Intro_LineFinder (Figure 8).


GettingStarted_Act4_Intro_LineFinder | Arduino 1.6.11
File Edit Sketch Tools Help
GettingStarted_Act4_Intro_LineFinder

```
/* This program will read the digital signal of the
 * Line Finder sensor attached to digital port D3.
 * If the sensor is facing a reflective surface and
 * receiving a reflected IR beam, the PRIZM red LED
 * will switch on. If the sensor is facing a dark
 * surface, or too far away from a reflective surface
 * the red LED will be off.
 */

 #include <PRIZM.h>    //include the PRIZM library
 PRIZM prizm;          //create an object name of "prizm"

void setup() {         //this code runs once

 prizm.PrizmBegin();   //initialize PRIZM

}

void loop() {          //this code repeats in a loop

 if(prizm.readLineSensor(3) == HIGH) {prizm.setRedLED(LOW);}  // LED off

 if(prizm.readLineSensor(3) == LOW)  {prizm.setRedLED(HIGH);} // LED on

 delay(50);  //slow the loop down

}
```

Done Saving.

Figure 8

Notes

Building the Knowledge Base
For the fourth sketch we want to take a closer look at two of the fundamental structure elements that make up a sketch. In every sketch you write, there will be a setup() and a loop() function.

The setup() function follows right after the include statement and object declaration as part of the beginning of our code. The setup() function contains items that need to be run only once as part of the sketch. Many functions can go here, but we always use at least the PRIZM initialization statement, prizm.PrizmBegin(). The main purpose of this function is to configure the Start button.

The loop() function does exactly what its name suggests. Everything contained within the brackets of the loop will repeat consecutively until the sketch is ended with a command or the Reset button. The loop() contains the main body of our code.

The contents of the setup() and the loop() are contained between curly braces. A left curly brace { begins a group of statements, and a right curly brace } ends a group of statements. They can be thought of as bookends, and the code in between is said to be a block of code.

Executing the Code
Before we can upload the sketch to the PRIZM, remember to check our connections.

Upload the sketch. The green LED will light up indicating the code is ready to execute. When this has happened, press the green Start button on the PRIZM controller.

Hold the sensor over the contrasting surface. As the sensor moves from light to dark, observe the red LED on the PRIZM. When the sensor is over a line, non-reflective, or dark surface, the red LED will be off. When the sensor is over a white or reflective surface, the red LED will be on.

Press the red Reset button when you are ready to stop the sensor.

Notes

Moving Forward
This sketch introduces a program structure, a new function, and a comparison statement. The program structure is an "if" statement, the new function is prizm.readLineSensor, and the comparison statement is == (equal to).

The basic "if" statement enables us to test for a certain condition. If this condition is met, then the program can perform an action. If the statement in the parentheses is true, the statements within brackets are run; if the statement is not true, the program will skip over the code.

The function prizm.readLineSensor reads the state of the Line Finder Sensor and returns a value of "1" (HIGH) or "0" (LOW). A value of "1" is returned when the Line Finder Sensor detects a dark line or a non-reflective surface; a value of "0" is returned when the Line Finder Sensor detects a white or reflective surface.

The comparison statement == (equal to) defines a type of test.

When these three elements are combined in the sketch, we create a test condition based on the input of the Line Finder Sensor that turns the red LED on or off. In simple terms, if the Line Finder Sensor detects a line or a non-reflective surface, then it turns the red LED off. If the Line Finder Sensor detects a white or reflective surface, it turns the LED on.

Experiment with the Line Finder Sensor on different surfaces and different heights to see how the sensor reacts.

Real-World Connection
Finding our way in this world can be challenging. Telling a robot how to find its way can be challenging as well. One way that robots within a warehouse can find their way to the right location is by having them follow lines. But to follow lines, they have to detect the lines. One way to accomplish this is by using a sensor that detects dark and light surfaces. This, through the computer code, provides information to the robot about where the line is. Other code controls the DC motors to change course if the robot strays from the line.

STEM Extensions
Science

- Light – reflection and absorption
- Electromagnetic spectrum

Technology

- Digital vs analog
- Calibration

Engineering

- Determining an edge location

Math

- Data analysis

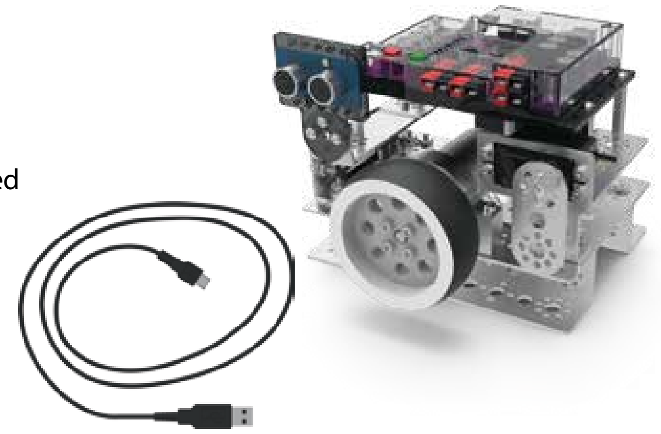
Hacking the Code Activity
With the example as a reference, try creating a new sketch to use your Line Finder Sensor. Remember what we learned from our previous activities, and think of additional creative actions to perform based on the condition of the Line Finder Sensor.

Notes

Activity 5: Introduction to the Ultrasonic Sensor

For the final getting started activity, we will finish up our exploration of sensors by creating a sketch using the Ultrasonic Sensor. In this activity we will connect an Ultrasonic Sensor to digital sensor port D3 and display the distance to an object we place in front of it using the serial monitor window.

Like all sensors, the Ultrasonic Sensor enables us to gather information. For the Ultrasonic Sensor, the information gathered communicates distance. The sensor works by sending a sonic pulse burst and then waiting on its return to the sensor as it is reflected off an object in range. The reflected sonic pulse time period is measured to determine the distance to the object. The sensor has a measuring range of approximately 3-400 centimeters.



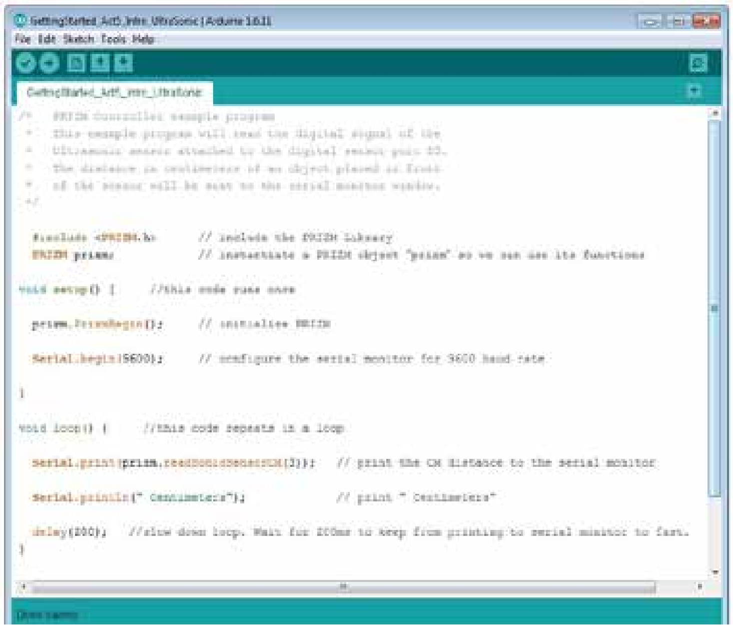
Parts Needed:

- PRIZM Test Bed
- USB cable
- Computer

Opening the Sketch

Before we open our next example sketch, be sure to save any sketch you want to reference later.

Open the sketch by selecting **File > Examples > TETRIX_PRIZM > GettingStarted_Act5_Intro_UltraSonic**. A new sketch window will open titled GettingStarted_Act5_Intro_UltraSonic (Figure 9).


GettingStarted_Act5_Intro_UltraSonic | Arduino 1.6.11
File Edit Sketch Tools Help
GettingStarted_Act5_Intro_UltraSonic
Done Saving.

```
/*   PRIZM Controller example program
 *   This example program will read the digital signal of the
 *   Ultrasonic sensor attached to the digital sensor port D3.
 *   The distance in centimeters of an object placed in front
 *   of the sensor will be sent to the serial monitor window.
 */

  #include <PRIZM.h>      // include the PRIZM library
  PRIZM prizm;            // instantiate a PRIZM object "prizm" so we can use its functions

void setup() {     //this code runs once

  prizm.PrizmBegin();     // initialize PRIZM

  Serial.begin(9600);     // configure the serial monitor for 9600 baud rate

}

void loop() {     //this code repeats in a loop

  Serial.print(prizm.readSonicSensorCM(3));   // print the CM distance to the serial monitor

  Serial.println(" Centimeters");             // print " Centimeters"

  delay(200);   //slow down loop. Wait for 200ms to keep from printing to serial monitor to fast.
}
```

Figure 9

Notes

Building the Knowledge Base
For our fifth sketch of the getting started activities, we want to look at a useful tool for viewing data as a dynamic text output. The serial monitor displays serial data being sent from the PRIZM via the USB connection.

The serial monitor's value as a tool lies in its ability to display data in real time that enables you to make better-informed design decisions about robot builds and programming. It can be used to display data from sensors connected to PRIZM or to examine any program data collected by the PRIZM – for example, encoder count data, DC motor current data, or servo position data.

Executing the Code
Before we can upload the sketch to the PRIZM, remember to check our connections.

Upload the sketch. Before we execute the sketch, we need to open the serial monitor from the sketch window. To open the serial monitor, click the magnifying glass in the top-right corner of the sketch window (Figure 10).

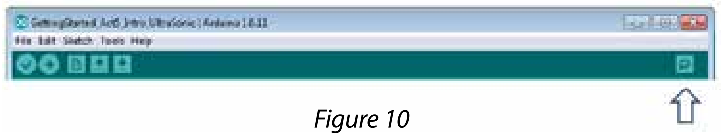


Figure 10

The serial monitor will open in a separate window (Figure 11).

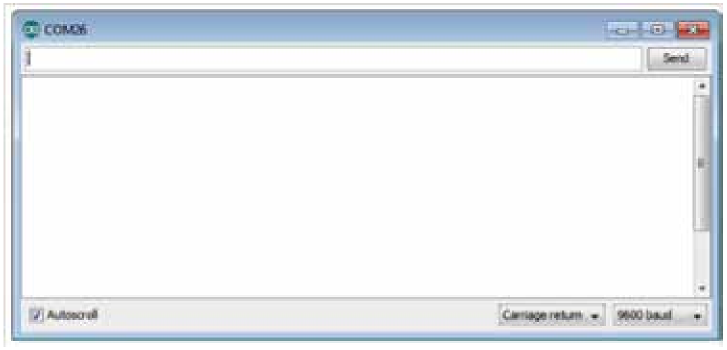


Figure 11

With the sensor lying flat on the desk pointed up, press the green Start button to execute the code.

Hold an object above the sensor at varying distances and observe the serial monitor to see the real-time data.

Press the red Reset button when you are ready to stop the sensor.

Notes

Moving Forward
This sketch introduces several new functions: Serial.begin(), Serial.print(), prizm.readSonicSensorCM(), and Serial.println().

Serial.begin() enables the use of serial communication within the sketch. As an initialization function that needs to be run only once, it belongs in the setup of the sketch. An important part of Serial.begin() is the speed of communication in bits per second, which is called baud rate. The default setting for baud rate in the serial window is 9600, so that is what we use in Serial.begin().

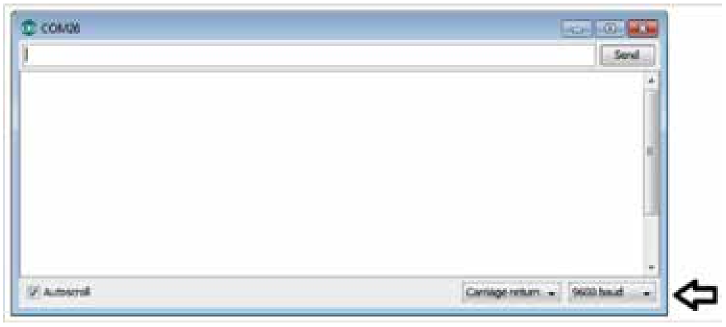
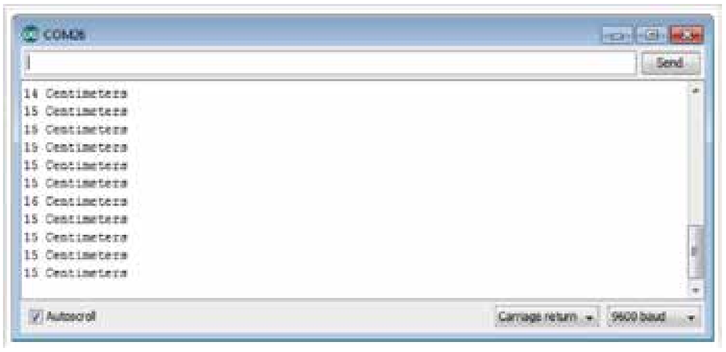


Figure 12

Serial.print() prints data to the serial port as readable text. This can take the form of dynamic information from another device or static information from the programmer.

prizm.readSonicSensorCM() reads the state of the Ultrasonic Sensor and returns a digital value within the designated measurement range. For our sensor, that range is between approximately 3-400 centimeters. This value should reflect the distance the Ultrasonic Sensor is from a detectable object.

Serial.println() prints data to the serial port as readable text followed by a built-in command for a new line.



```
14 Centimeters
15 Centimeters
15 Centimeters
15 Centimeters
15 Centimeters
15 Centimeters
16 Centimeters
15 Centimeters
15 Centimeters
15 Centimeters
15 Centimeters
```

Figure 13

These four functions might seem complicated, but they actually work together simply. In this sketch, Serial.begin(9600) enables and defines the speed of communication in the setup. Serial.print() tells what type of data to print. prizm.readSonicSensorCM() provides the type of data to print because it is within the parentheses of Serial.print(). And Serial.println(" Centimeters") clarifies the type of data being printed – in this case, with the modifier " Centimeters."

Notes

Real-World Connection
When you were very, very young, you were probably scanned by an ultrasonic sensor. If you do not remember this, it is because it was prior to you being born. One of the great applications of ultrasonic technology is within the medical field. Doctors can use the reflection of sound waves back to sensor waves to see inside a living person. Doctors can see a baby's size and development – and determine when they think he or she might be born!

STEM Extensions
Science

- Sound wave terminology (frequency, amplitude, crest, trough)
- Reflection of sound waves

Technology

- Measuring frequency
- Sound digitisation

Engineering

- Applications of sonic measurements

Math

- Inverse square law

Hacking the Code Activity
With the example as a reference, try creating a new sketch to use the Ultrasonic Sensor with the serial monitor. Remember what we learned from our previous activities and experiment with different objects in front of the Ultrasonic Sensor to see if they are detectable and if the distances can be measured accurately.

Try changing the code for prizm.readSonicSensorCM() to prizm.readSonicSensorIN() to display the distance from an object in inches. Also be sure to change the Serial.println from " Centimeters" to "Inches" so that the unit is correctly labeled in the serial monitor window. Understanding how to use the Ultrasonic Sensor will give your robot vision to be able to steer around objects and obstacles.

Notes

Standards Addressed

Common Core State Standards

CCSS.ELA-LITERACY.RST.9-10.1
Cite specific textual evidence to support analysis of science and technical texts, attending to the precise details of explanations or descriptions.

CCSS.ELA-LITERACY.RST.9-10.2
Determine the central ideas or conclusions of a text; trace the text's explanation or depiction of a complex process, phenomenon, or concept; provide an accurate summary of the text.

CCSS.ELA-LITERACY.RST.9-10.3
Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text.

CCSS.ELA-LITERACY.RST.9-10.4
Determine the meaning of symbols, key terms, and other domain-specific words and phrases as they are used in a specific scientific or technical context relevant *to grades 9-10 texts and topics*.

CCSS.ELA-LITERACY.RST.9-10.5
Analyze the structure of the relationships among concepts in a text, including relationships among key terms (e.g., *force, friction, reaction force, energy*).

CCSS.ELA-LITERACY.SL.9-10.1.A
Come to discussions prepared, having read and researched material under study; explicitly draw on that preparation by referring to evidence from texts and other research on the topic or issue to stimulate a thoughtful, well-reasoned exchange of ideas.

CCSS.ELA-LITERACY.SL.9-10.1.B
Work with peers to set rules for collegial discussions and decision-making (e.g., informal consensus, taking votes on key issues, presentation of alternate views), clear goals and deadlines, and individual roles as needed.

CCSS.ELA-LITERACY.SL.9-10.1.C
Propel conversations by posing and responding to questions that relate the current discussion to broader themes or larger ideas; actively incorporate others into the discussion; and clarify, verify, or challenge ideas and conclusions.

CCSS.MATH.CONTENT.HSN.Q.A.1
Use units as a way to understand problems and to guide the solution of multi-step problems; choose and interpret units consistently in formulas; choose and interpret the scale and the origin in graphs and data displays.

CCSS.MATH.CONTENT.HSN.Q.A.3
Choose a level of accuracy appropriate to limitations on measurement when reporting quantities.

CCSS.MATH.CONTENT.HSA.SSE.A.1.A
Interpret parts of an expression, such as terms, factors, and coefficients.

CCSS.MATH.CONTENT.HSA.CED.A.4
Rearrange formulas to highlight a quantity of interest, using the same reasoning as in solving equations.

CCSS.MATH.CONTENT.HSF.IF.A.2
Use function notation, evaluate functions for inputs in their domains, and interpret statements that use function notation in terms of a context.

CCSS.MATH.CONTENT.HSF.BF.A.1.A
Determine an explicit expression, a recursive process, or steps for calculation from a context.

CCSS.MATH.CONTENT.HSF.BF.A.1.B
Combine standard function types using arithmetic operations.

CCSS.MATH.CONTENT.HSF.LE.A.1.B
Recognize situations in which one quantity changes at a constant rate per unit interval relative to another.

CCSS.MATH.CONTENT.HSF.LE.B.5
Interpret the parameters in a linear or exponential function in terms of a context.

CCSS.MATH.CONTENT.HSS.IC.A.2
Decide if a specified model is consistent with results from a given data-generating process, e.g., using simulation.

Next Generation Science Standards

NGSS.HS-ETS1-2
Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.

NGSS.HS-ETS1-3
Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics, as well as possible social, cultural, and environmental impacts.

NGSS.HS-PS2-3
Apply scientific and engineering ideas to design, evaluate, and refine a device that minimises the force on a macroscopic object during a collision.

NGSS.HS-PS3-3
Design, build, and refine a device that works within given constraints to convert one form of energy into another form of energy.

International Technology and Engineering Educators Association

ITEEA 2.N
Systems thinking involves considering how every part relates to others.

ITEEA 2.Q
Malfunctions of any part of a system may affect the function and quality of the system.

ITEEA 2.R
Requirements are the parameters placed on the development of a product or system.

ITEEA 2.S
Trade-off is a decision process recognizing the need for careful compromises among competing factors.

ITEEA 2.V
Controls are mechanisms or particular steps that people perform using information about the system that causes systems to change.

ITEEA 8.E
Design is a creative planning process that leads to useful products and systems.

ITEEA 8.F
There is no perfect design.

ITEEA 8.G
Requirements for a design are made up of criteria and constraints.

ITEEA 9.F
Design involves a set of steps, which can be performed in different sequences and repeated as needed.

ITEEA 9.G
Brainstorming is a group problem-solving design process in which each person in the group presents his or her ideas in an open forum.

ITEEA 9.H
Modeling, testing, evaluating, and modifying are used to transform ideas into practical solutions.

ITEEA 10.F
Troubleshooting is a problem-solving method used to identify the cause of a malfunction in a technological system.

ITEEA 10.H
Some technological problems are best solved through experimentation.

ITEEA 11.I
Specify criteria and constraints for the design.

ITEEA 11.K
Test and evaluate the design in relation to pre-established requirements, such as criteria and constraints, and refine as needed.

Computer Science Teachers Association

CT.2.1
Use the basic steps in algorithmic problem-solving to design solutions.

CT.2.3
Define an algorithm as a sequence of instructions that can be processed by a computer.

CT.2.4
Evaluate ways that different algorithms may be used to solve the same problem.

CT.2.12
Use abstraction to decompose a problem into sub problems.

CT.2.14
Examine connections between elements of mathematics and computer science including binary numbers, logic, sets and functions.

CL.2.3
Collaborate with peers, experts, and others using collaborative practices such as pair programming, working in project teams, and participating in group active learning activities.

CL.2.4
Exhibit dispositions necessary for collaboration: providing useful feedback, integrating feedback, understanding and accepting multiple perspectives, socialisation.

CPP.2.4
Demonstrate an understanding of algorithms and their practical application.

CPP.2.5
Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions.

CPP.2.8
Demonstrate dispositions amenable to open-ended problem solving and programming.

CPP.2.9
Collect and analyse data that is output from multiple runs of a computer program.

Notes

Notes

Notes

Conference Presentation Evaluation

Topic: ____________________________

1. Which of the following best describes your role?

___ Classroom teacher
___ Building administrator
___ District administrator
___ College professor
___ Informal education (e.g., library, museum)
___ Content specialist (e.g., reading teacher, gifted and talented coordinator)
Other ____________________________

2. Which of the following best describes your school?

___ Public	___ Private
___ Charter	___ University

Other ____________________________

3. Which grade level(s) of students do you work with?

___ K-2	___ Postsecondary
___ 3-5	___ After school
___ 6-8	___ Summer camps
___ 9-12	___ I don't teach.

4. Which of these subject areas do you teach?

___ Science
___ Technology
___ Engineering
___ Math
___ Robotics competitions
___ Robotics in the classroom

5. How many years have you been teaching?

___ 1st year	___ 11-20 years
___ 2-5 years	___ 21+ years
___ 6-10 years	___ I don't teach.

6. Please provide your email address:

7. How strongly do you agree with the following statements?
(1 = strongly disagree, 3 = neutral, 5 = strongly agree)

This session was selected for immediate classroom use.
1 2 3 4 5

This session was selected to improve my personal pedagogical knowledge/skill.
1 2 3 4 5

This session met my needs.
1 2 3 4 5

The information in this session was clear and well organized.
1 2 3 4 5

Safe practices were employed during the session.
1 2 3 4 5

8. Did the session cover the content that you expected it to? If not, what was different?

9. How likely are you to recommend the session or product to a friend or colleague? (10 is extremely likely, and 0 is not at all likely.)

This session?	0	1	2	3	4	5	6	7	8	9	10
The product?	0	1	2	3	4	5	6	7	8	9	10

10. Do you have any input for the presenter or future presentations?

11. Have you ever used the product that was demonstrated in the workshop?

Yes No

12. What ideas do you have about how you might use this product in your classroom?

13. What is your favorite takeaway from this workshop?

14. Please share any other comments you might have regarding the workshop and your overall experience.

Use the back of this sheet if needed.

Add your contact information to the back of this sheet to be entered in the drawing.

Name ____

School ____

Address ____

City _________________________________ ST ___________

Zip _________________________________ Country ___________

Phone ____

Visit TETRIXrobotics.com for more information about TETRIX PRIME and TETRIX MAX.