ZX Distance and Gesture Sensor SMD Hookup Guide

Introduction

The ZX Distance and Gesture Sensor is a collaboration product with XYZ Interactive. The innovative people at XYZ Interactive have created a unique technology that allows for simple infrared (IR) beams to be used to detect an object's location in two dimensions.

The ZX Sensor is a touchless sensor that is capable of looking for simple gestures in the air above the sensor (e.g. swipe left or swipe right). Additionally, the sensor can also recognize the distance of an object away from the sensor at distances up to about 12 inches (30 cm), referred to as the "Z" axis, and the location of the object from side to side across the sensor in about a 6 inch (15 cm) span, referred to as the "X" axis.

ZX Distance and Gesture Sensor
◉ SEN-13162

Covered in This Tutorial

We can use I^{2}C or UART to communicate with the ZX Sensor. In this tutorial, we will show you how to connect the sensor to an Arduino or Arduino-compatible board as well as a computer so you can start creating gestures to handle all our your daily tasks or add some interactive flair to your project.

Materials Used

In addition to the sensor itself, you will need a few extra components to follow along with the Arduino examples:

ZX Gesture and Distance Sensor Hookup Guide SparkFun Wish List	
	ZX Distance and Gesture Sensor SEN-13162 The ZX Distance and Gesture Sensor is a touchless sensor that is ca…
	SparkFun RedBoard - Programmed with Arduino DEV-13975 At SparkFun we use many Arduinos and we're always looking for the…
	SparkFun USB Mini-B Cable - 6 Foot CAB-11301 This is a USB 2.0 type A to Mini-B 5-pin cable. You know, the mini-B…
	Breadboard - Mini Modular (White) PRT-12043 This white Mini Breadboard is a great way to prototype your small proj…
	Jumper Wires Premium 6" M/M Pack of 10 PRT-08431 This is a SparkFun exclusive! These are 155mm long jumpers with m…
	Break Away Headers - Straight PRT-00116 A row of headers - break to fit. 40 pins that can be cut to any size. Us…

If you would like to try the ZX Sensor on a Windows-based PC, you will need an FTDI Breakout:

SparkFun FTDI Basic Breakout - 5V
◉ DEV-09716

Recommended Reading

There are a few concepts that you should be familiar with before getting started with the ZX Sensor. Consider reading some of these tutorials before continuing:

- What is an Arduino? – Two of the examples use an Arduino to control the ZX Sensor
- I²C – I²C is the one of the protocols used by the ZX Sensor
- Serial Communication – We use serial communications to program the Arduino, view debugging information, and transmit data from the ZX Sensor
- How to Use a Breadboard – The breadboard ties the Arduino to the ZX Sensor

- How to Install FTDI Drivers – If you are programming an Arduino or using the ZX Sensor demo app, chances are you will need to use an FTDI

Board Overview

The ZX Sensor works by bouncing infrared (IR) beams of light from the two LEDs on either side off of an object above the sensor. The bounced light returns to the receiver in the center of the sensor, and a microcontroller on the back of the sensor interprets the data. We can read the results using an I²C or UART connection.

Pin Descriptions

The ZX Sensor gives us two ports to connect to: I²C and UART. You can see both ports are broken out to the 0.1" thru holes. See the table below for a list of each pin and its function.

Pin Label	Description
GRN	Not used
TXO	UART transmit out from the ZX Sensor
RXI	UART receive. Not used at this time.
VCC	3.3 - 5 V power supply
GND	Connect to ground
BLK	Not used, but connected to GND
DR	Data Ready. High when there is data to be read via I²C
CL	I²C clock
DA	I²C data

Setting the Jumpers

The ZX Sensor has a couple of jumpers on the back of the board that can be opened or closed with a soldering iron.

I2C Pullups

The ZX Sensor, by default, comes with 4.7 kΩ pull-up resistors on the SDA and SCL I²C lines. Remove the solder on this jumper using solder wick to disconnect the pull-ups.

I2C Addr

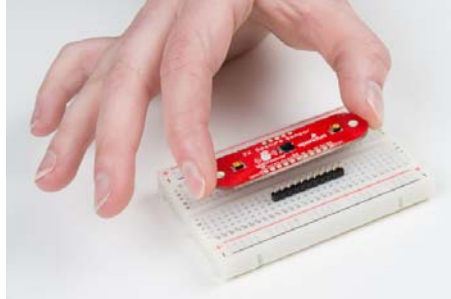By default, this jumper is open. Close it to change the I²C address of the sensor.

Jumper	I²C Address

Open	0x10
Closed	0x11

Hardware Hookup

Add Headers

Solder a row of male headers to the nine headers holes on the board.
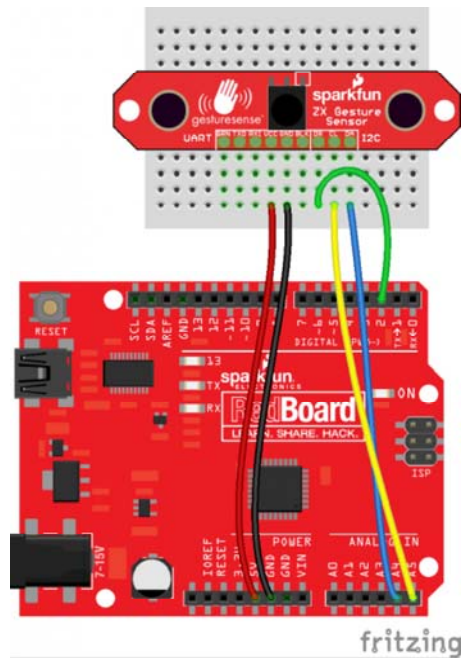

To keep the board from tilting while soldering, place the unused break away headers sideways under the board.

> **Heads up!** Do not solder headers to the row of holes at the top of the board. Those are for programming the PIC micrcontroller.

Connect the Breakout Board

For the Arduino examples, we will be using I²C. Connect the breakout board to the following RedBoard pins:

ZX Sensor	RedBoard
VCC	5V
GND	GND
DR	2
CL	A5
DA	A4

Note that we connect the DR pin, but we will only use it in the Arduino: Gesture Example. DR stands for "Data Ready," which is active high whenever data is ready to be read from the ZX Sensor. We can attach this to an Arduino interrupt so we don't have to continuously poll the sensor.
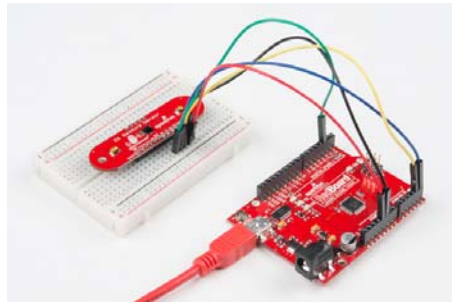
Arduino Library Installation

All of the hard work for the ZX Sensor is being accomplished in the microcontroller on the sensor itself. All we need to do is read the results! We have created an Arduino library to make that even easier for you. Click the button to download the latest version of the ZX Sensor Arduino Library. You can also find the latest files in the GitHub repository.

DOWNLOAD THE ZX SENSOR ARDUINO LIBRARY!

Unzip the downloaded file. Follow this guide on installing Arduino libraries to install the files as an Arduino library.

https://github.com/sparkfun/SparkFun_ZX_Distance_and_Gesture_Sensor_Arduino_Library/archive/master.zip

Arduino: ZX Example

Load the ZX Demo

Open up the Arduino program and select File → Examples → SparkFun_ZX_Distance_and_Gesture_Sensor → I2C_ZX_Demo.
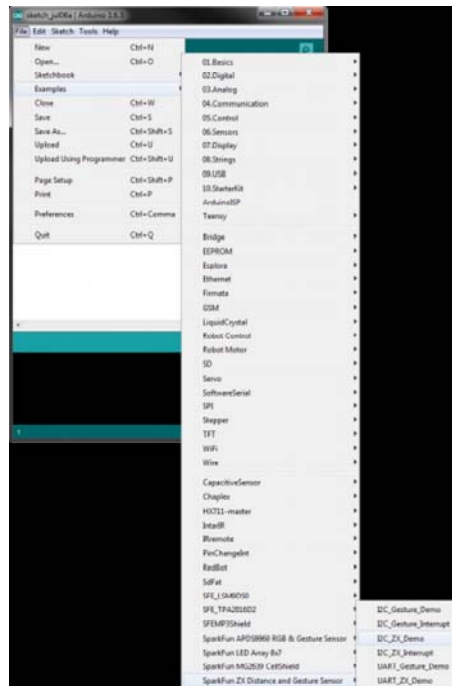
Attach a USB mini cable from your computer to the RedBoard. If you have not previously done so, install the FTDI drivers.

For reference, here is the I2C_ZX_Demo.ino sketch.

```
/*****************************************************
*******
    I2C_ZX_Demo.ino
    XYZ Interactive ZX Sensor
    Shawn Hymel @ SparkFun Electronics
    May 6, 2015
    https://github.com/sparkfun/SparkFun_ZX_Distance_and_Gestu
re_Sensor_Arduino_Library

    Tests the ZX sensor's ability to read ZX data over I2C. Th
is demo
    configures the ZX sensor and periodically polls for Z-axi
s and X-axis data.

    Hardware Connections:

     Arduino Pin  ZX Sensor Board  Function
     ----------------------------------------
     5V            VCC             Power
     GND           GND             Ground
     A4            DA              I2C Data
     A5            CL              I2C Clock

    Resources:
    Include Wire.h and ZX_Sensor.h

    Development environment specifics:
    Written in Arduino 1.6.3
    Tested with a SparkFun RedBoard

    This code is beerware; if you see me (or any other SparkFu
n
    employee) at the local, and you've found our code helpfu
l, please
    buy us a round!

    Distributed as-is; no warranty is given.
    *****************************************************
******/

    #include <Wire.h>
    #include <ZX_Sensor.h>

    // Constants
    const int ZX_ADDR = 0x10;  // ZX Sensor I2C address

    // Global Variables
    ZX_Sensor zx_sensor = ZX_Sensor(ZX_ADDR);
    uint8_t x_pos;
    uint8_t z_pos;

    void setup() {

      uint8_t ver;

      // Initialize Serial port
      Serial.begin(9600);
      Serial.println();
      Serial.println("--------------------------------");
      Serial.println("SparkFun/GestureSense - I2C ZX Demo");
      Serial.println("--------------------------------");

      // Initialize ZX Sensor (configure I2C and read model I
D)
```

```
    if ( zx_sensor.init() ) {
      Serial.println("ZX Sensor initialization complete");
    } else {
      Serial.println("Something went wrong during ZX Sensor
init!");
    }

    // Read the model version number and ensure the library
will work
    ver = zx_sensor.getModelVersion();
    if ( ver == ZX_ERROR ) {
      Serial.println("Error reading model version number");
    } else {
      Serial.print("Model version: ");
      Serial.println(ver);
    }
    if ( ver != ZX_MODEL_VER ) {
      Serial.print("Model version needs to be ");
      Serial.print(ZX_MODEL_VER);
      Serial.print(" to work with this library. Stopping.");
      while(1);
    }

    // Read the register map version and ensure the library
will work
    ver = zx_sensor.getRegMapVersion();
    if ( ver == ZX_ERROR ) {
      Serial.println("Error reading register map version num
ber");
    } else {
      Serial.print("Register Map Version: ");
      Serial.println(ver);
    }
    if ( ver != ZX_REG_MAP_VER ) {
      Serial.print("Register map version needs to be ");
      Serial.print(ZX_REG_MAP_VER);
      Serial.print(" to work with this library. Stopping.");
      while(1);
    }
  }

  void loop() {

    // If there is position data available, read and print i
t
    if ( zx_sensor.positionAvailable() ) {
      x_pos = zx_sensor.readX();
      if ( x_pos != ZX_ERROR ) {
        Serial.print("X: ");
        Serial.print(x_pos);
      }
      z_pos = zx_sensor.readZ();
      if ( z_pos != ZX_ERROR ) {
        Serial.print(" Z: ");
        Serial.println(z_pos);
      }
    }
  }
```

Run

Make sure you have the correct serial port selected under Tools → Serial
Port and "Arduino Uno" selected under Tools → Board. If you have never
used the Arduino IDE before, this turoial should get you started.

Click the Upload button and wait for the program to finish uploading to the Arduino. Select Tools → Serial Monitor to open up the serial terminal. More info on the Serial Terminal can be found here. Note that the Serial Monitor settings are the default settings (9600, 8, n, 1). You should see a couple of messages noting that "ZX Sensor initialization complete."

Hover your hand 4 to 10 inches (10 to 25 cm) above the sensor.
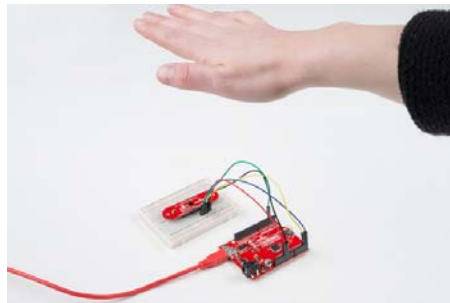
Move your hand around above the sensor, and you should see Z (height above the sensor) and X (position side to side) appear in the serial terminal.
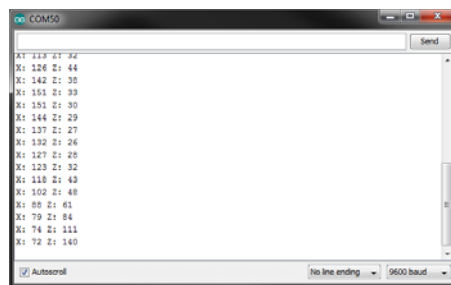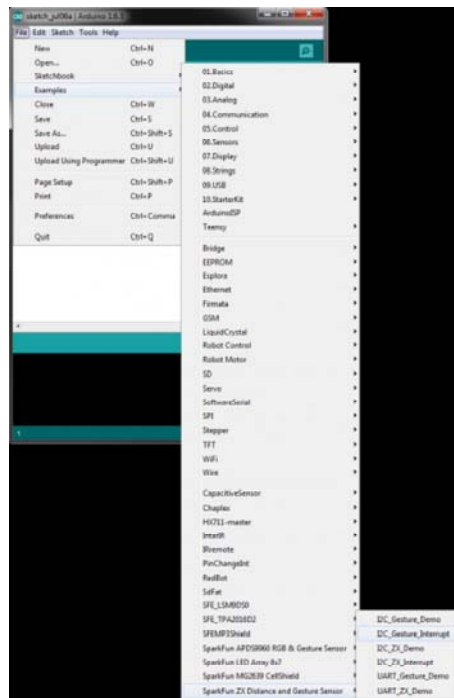

NOTE: Z- and X- data is given as an unsigned integer between 0 and 240 (inclusive).

Arduino: Gesture Example

Load the Gesture Interrupt Demo

In addition to providing Z- and X- axis data about an object, the ZX Sensor is also capable of detecting simple gestures. To see an example of this, open File → Examples → SparkFun_ZX_Distance_and_Gesture_Sensor → I2C_Gesture_Interrupt.

Here is the I2C_Gesture_Interrupt.ino sketch for reference.

```
/**************************************************************
***
I2C_Gesture_Interrupt.ino
XYZ Interactive ZX Sensor
Shawn Hymel @ SparkFun Electronics
May 6, 2015
https://github.com/sparkfun/SparkFun_ZX_Distance_and_Gesture_S
ensor_Arduino_Library

Tests the ZX sensor's ability to read gesture data over I2C us
ing
an interrupt pin. This program configures I2C and sets up an
interrupt to occur whenever the ZX Sensor throws its DR pin hi
gh.
The gesture is displayed along with its "speed" (how long it t
akes
to complete the gesture). Note that higher numbers of "speed"
indicate a slower speed.

Hardware Connections:

 Arduino Pin  ZX Sensor Board  Function
 ---------------------------------------
 5V           VCC              Power
 GND          GND              Ground
 A4           DA               I2C Data
 A5           CL               I2C Clock
 2            DR               Data Ready

Resources:
Include Wire.h and ZX_Sensor.h

Development environment specifics:
Written in Arduino 1.6.3
Tested with a SparkFun RedBoard

This code is beerware; if you see me (or any other SparkFun
employee) at the local, and you've found our code helpful, ple
ase
buy us a round!

Distributed as-is; no warranty is given.
**************************************************************
**/

#include <Wire.h>
#include <ZX_Sensor.h>

// Constants
const int ZX_ADDR = 0x10;     // ZX Sensor I2C address
const int INTERRUPT_NUM = 0; // Pin 2 on the UNO

// Global Variables
ZX_Sensor zx_sensor = ZX_Sensor(ZX_ADDR);
volatile GestureType gesture;
volatile bool interrupt_flag;
uint8_t gesture_speed;

void setup() {

  uint8_t ver;

  // Initialize gesture to no gesture
  gesture = NO_GESTURE;
```

```
  // Initialize Serial port
  Serial.begin(9600);
  Serial.println();
  Serial.println("-------------------------------------------
-");
  Serial.println("SparkFun/GestureSense - I2C Gesture Interrup
t");
  Serial.println("Note: higher 'speed' numbers mean slower");
  Serial.println("-------------------------------------------
-");

  // Initialize ZX Sensor (configure I2C and read model ID)
  if ( zx_sensor.init(GESTURE_INTERRUPTS) ) {
    Serial.println("ZX Sensor initialization complete");
  } else {
    Serial.println("Something went wrong during ZX Sensor ini
t!");
  }

  // Read the model version number and ensure the library wil
l work
  ver = zx_sensor.getModelVersion();
  if ( ver == ZX_ERROR ) {
    Serial.println("Error reading model version number");
  } else {
    Serial.print("Model version: ");
    Serial.println(ver);
  }
  if ( ver != ZX_MODEL_VER ) {
    Serial.print("Model version needs to be ");
    Serial.print(ZX_MODEL_VER);
    Serial.print(" to work with this library. Stopping.");
    while(1);
  }

  // Read the register map version and ensure the library wil
l work
  ver = zx_sensor.getRegMapVersion();
  if ( ver == ZX_ERROR ) {
    Serial.println("Error reading register map version numbe
r");
  } else {
    Serial.print("Register Map Version: ");
    Serial.println(ver);
  }
  if ( ver != ZX_REG_MAP_VER ) {
    Serial.print("Register map version needs to be ");
    Serial.print(ZX_REG_MAP_VER);
    Serial.print(" to work with this library. Stopping.");
    while(1);
  }

  // Initialize interrupt service routine
  interrupt_flag = false;
  zx_sensor.clearInterrupt();
  attachInterrupt(INTERRUPT_NUM, interruptRoutine, RISING);
  Serial.println("Interrupts now configured. Gesture away!");
}

void loop() {

  // If we have an interrupt, read and print the gesture
  if ( interrupt_flag ) {
```

```
      // Clear the interrupt flag
      interrupt_flag = false;

      // You MUST read the STATUS register to clear interrupt!
      zx_sensor.clearInterrupt();

      // Read last gesture
      gesture = zx_sensor.readGesture();
      gesture_speed = zx_sensor.readGestureSpeed();
      switch ( gesture ) {
        case NO_GESTURE:
          Serial.println("No Gesture");
          break;
        case RIGHT_SWIPE:
          Serial.print("Right Swipe. Speed: ");
          Serial.println(gesture_speed, DEC);
          break;
        case LEFT_SWIPE:
          Serial.print("Left Swipe. Speed: ");
          Serial.println(gesture_speed, DEC);
          break;
        case UP_SWIPE:
          Serial.print("Up Swipe. Speed: ");
          Serial.println(gesture_speed, DEC);
          break;
        default:
          break;
      }
    }
}

void interruptRoutine() {
  interrupt_flag = true;
}
```
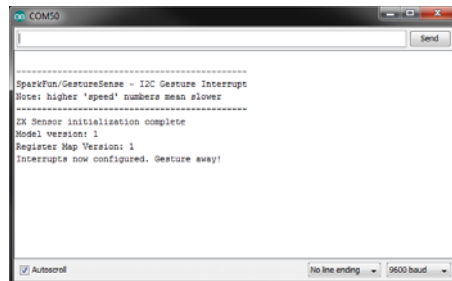
Run

Upload the sketch, and open the Serial Monitor. You should see a message stating that initialization is complete.

Start with your hand off to one side (a "side" being the one of the infrared LEDs with the brass covers) about 4 to 10 inches (10 to 25 cm) above the sensor. Swipe your hand horizontally across the sensor so that your hand passes over the one infrared LED and then the next infrared LED.
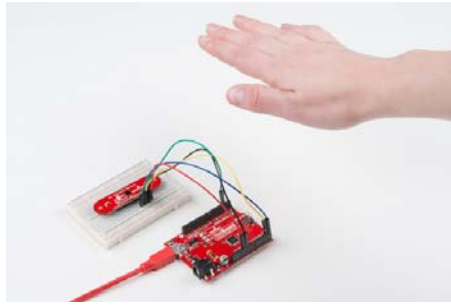
If you performed the gesture correctly, you should see a message appear in the Serial Monitor.

> **NOTE:** The "Speed" of the gesture is a measure of how fast the gesture occurred. Note that the *lower* the number, the *faster* the gesture occurred (e.g. 3 being very fast and 25 being very slow).

Supported Gestures

Here is a list of the currently supported gestures. Make sure each gesture begins outside of the range of the sensor, moves into the range of the sensor, and ends outside the range of the sensor.

Gesture	Description
Right Swipe	A swipe from the left side of the board to the right and out of range of the sensor. Make sure that your wrist/arm is not in the sensor's range at the end of the swipe!
Left Swipe	A swipe from the right side of the board to the left and out of range of the sensor.
Up Swipe	Object starts near the sensor, hovers for at least 1 second, and then moves up above and out of range of the sensor.
No Gesture	The sensor could not correctly determine the gesture being performed.

PC: ZX Example

The ZX Sensor, in addition to responding to I²C commands, continually transmits ZX data over its UART port. We can connect an FTDI Breakout directly to the ZX Sensor and read the output. You can use serial applications or the screen command (Linux or Mac) to view the output.

> **NOTE:** You can use either 3.3 V or 5 V FTDI. 5 V gives you a bit better range with the sensor.

If you are on a Windows computer, you can use the demo application (linked below) provided by XYZ Interactive to test the ZX Sensor.

Setup

Connect the FTDI Breakout board to the ZX Sensor. Ensure the pins on the FTDI Brekaout line up with the pins on the ZX Sensor (e.g. GRN connects to GRN and BLK connects to BLK). Connect the FTDI Breakout to your computer with a USB cable.

Download the ZX Demo application, and unzip it.

DOWNLOAD THE ZX DEMO APPLICATION

Run

Double-click to run the ZX Demo application. Under "Input:" on the right side, drop down the list and select the COM port that corresponds to your FTDI Breakout (if you need a refresher on find the right COM port, check out this section of the Terminal Basics tutorial). You do not need to choose an "Output:" port.

Click **Open** to connect to the FTDI Breakout.

Move your hand around above the sensor, and you should see the red ball move.

Try out the other tabs in the application! The Z-Control tab lets your try moving your hand toward and away from the sensor, and the Gestures tab computes a few different gestures based on the Z- and X- data.

Resources and Going Further

After trying the basic ZX and gesture demos, you can try the other examples in the Arduino library. A description of each of the examples is given below:

- **I2C_Gesture_Demo** – Poll the sensor over I²C to see if any gestures have occurred.
- **I2C_Gesture_Interrupt** – The DR pin will go from low to high when a gesture is detected. This example reads the gesture over I2C and tells the sensor to clear DR.
- **I2C_ZX_Demo** – Poll the sensor periodically over I²C for Z- and X-axis data.
- **I2C_ZX_Interrupt** – The ZX Sensor will throw DR high whenever valid ZX data is ready.
- **UART_Gesture_Demo** – **NOTE:** Gestures over UART are not supported at this time. This demo is a placeholder for the time being.
- **UART_ZX_Demo** – Read Z- and X- axis data from a software serial port and display them on the Serial Monitor.
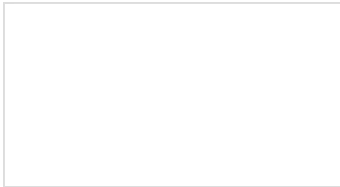
Resources

Here are some additional resources to help you with the ZX Sensor:

- ZX Sensor Datasheet
- Using the ZX Sensor with Arduino
- ZX Sensor Schematic
- ZX Sensor GitHub Repository
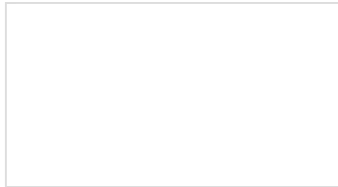- ZX Sensor I2C Register Map

Other Tutorials

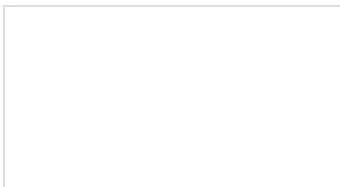What will you make with the ZX Sensor? If you need some inspiration, check out these related tutorials:

Connecting Arduino to Processing
Send serial data from Arduino to Processing and back - even at the same time!
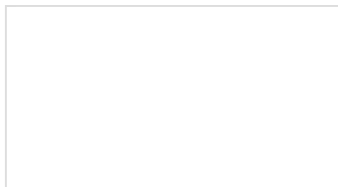
Serial Graphic LCD Hookup
Learn how to use the Serial Graphic LCD.

RGB Panel Hookup Guide
Make bright, colorful displays using the 32x32 and 32x16 RGB LED panels. This hookup guide shows how to hook up these panels and control them with an Arduino.

APDS-9960 RGB and Gesture Sensor Hookup Guide
Getting started guide for the Avago APDS-9960 color, proximity, and gesture sensor.