ZX Distance and Gesture Sensor Hookup Guide

CONTRIBUTORS: *SHAWNHYMEL*

FAVORITE 5

Introduction

The ZX Distance and Gesture Sensor is a collaboration product with XYZ Interactive. The very smart people at XYZ Interactive have created a unique technology that allows for simple infrared 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 right). Additionally, the sensor can also recognize the distance of an object away from the sensor up to about 10 inches (25 cm), which we will call the "Z" axis, and the location of the object from side to side across the sensor in about a 6 inch (15 cm) span, which we will call the "X" axis.

Covered in This Tutorial

We can use I^2C or UART to communicate with the ZX Sensor. In this tutorial, we will show you how to connect the sensor to an Arduino as well as a computer. The tutorial is divided into the following sections:

- Board Overview – We go over the various pins and jumper settings on the ZX Sensor.
- Hardware Hookup – How to connect the sensor to an Arduino.
- Arduino Library Installation – Install the ZX Sensor library if you plan to use it with an Arduino
- Arduino: ZX Example – How to read Z- and X- axis data from the sensor on an Arduino.
- Arduino: Gesture Example – How to read gestures with an Arduino.
- PC: ZX Example – How to read Z- and X- axis data on a Windows computer.
- Resources and Going Further – This section gives you some additional resources for getting more use out of the ZX Sensor.

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-12780 The ZX Distance and Gesture Sensor is a touchless sensor that is capable of looking for simple gestures. Developed in conjunction with [XYZ Interactiv…
	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 connector that usually comes with USB Hubs, Cameras, MP3 players, etc. You can us…
	SparkFun RedBoard - Programmed with Arduino DEV-12757 At SparkFun we use many Arduinos and we're always looking for the simplest, most stable one. Each board is a bit different and no one board has everyt…
	Breadboard - Mini Modular (White) PRT-12043 This white Mini Breadboard is a great way to prototype your small projects! With 170 tie points there's just enough room to build and test simple circ…
	Jumper Wires Premium 6" M/M Pack of 10 PRT-08431 This is a SparkFun exclusive! These are 155mm long jumpers with male connectors on both ends. Use these to jumper from any female header on any board,…
	Break Away Headers - Straight PRT-00116 A row of headers - break to fit. 40 pins that can be cut to any size. Used with custom PCBs or general custom headers.**Features: *** Pin Style: Squar…

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
$14.95

★★★★☆ 81

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:

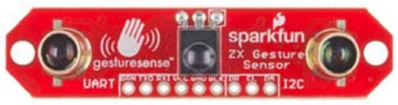
- What is an Arduino? – Two of the examples use an Arduino to control the ZX Sensor
- I^{2}C – I^{2}C is the one of the protocols used by the ZX Sensor
- Serial Communication – We use serial communications to program the Arduino, view debugging information, and transmit data from the ZX Sensor
- How 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 I^{2}C or UART connection.

Pin Descriptions

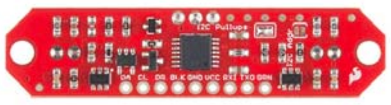
The ZX Sensor gives us 2 ports to connect to: I^{2}C and UART.



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 I^{2}C
CL	I^{2}C clock
DA	I^{2}C 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 I^{2}C lines. Remove the solder on this jumper to disconnect the pull-ups.

I2C Addr - By default, this jumper is open. Close it to change the I^{2}C address of the sensor.

Jumper	I^{2}C Address
Open	0x10
Closed	0x11

The Infrared Shields

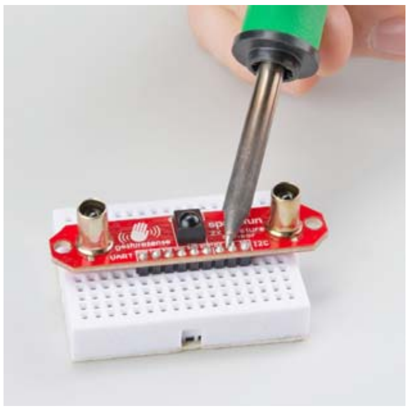
See those little brass tubes on the LEDs? They are needed to block any IR light going directly from the LEDs (inside the brass tubes) to the receiver (the black domed component in the center of the sensor). We want the light to bounce off an object first.

They are held on with Super Glue. You can remove them, but it might require a bit of acetone. If you plan to mount the ZX Sensors in your own housing, make sure IR light can't travel directly from the sides of the LEDs to the receiver (i.e. you will want to make your own IR shields).

Hardware Hookup

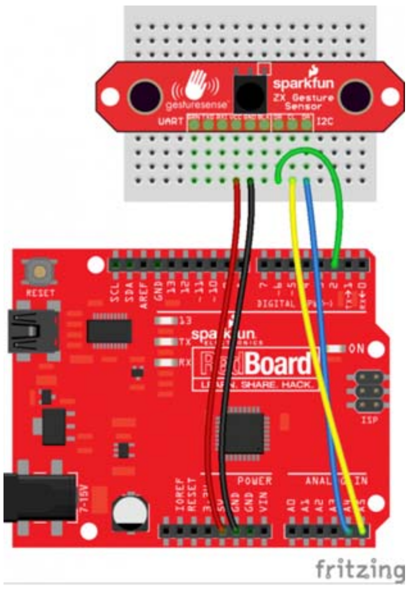
Add Headers

Solder a row of break away male headers to the 9 headers holes on the board.



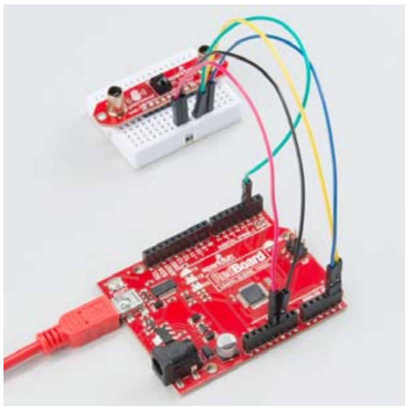
Connect the Breakout Board

For the Arduino examples, we will be using I^{2}C. 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.

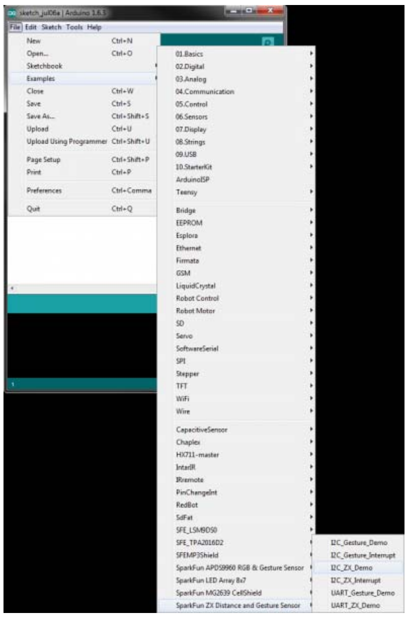
DOWNLOAD THE ZX SENSOR ARDUINO LIBRARY!

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

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.

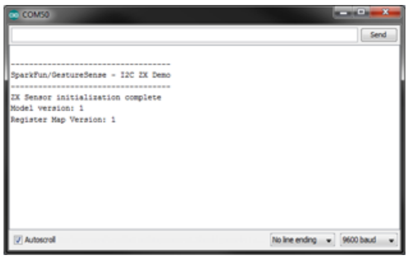
For reference, here is the `I2C_ZX_Demo.ino` sketch.

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

Tests the ZX sensor's ability to read ZX data over I2C. This demo
configures the ZX sensor and periodically polls for Z-axis 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 SparkFun
employee) at the local, and you've found our code helpful, 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 ID)
  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 number");
  } 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 it
  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

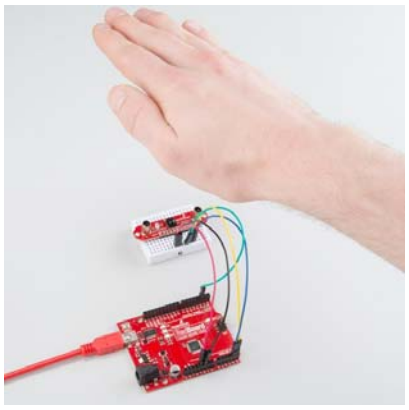
Make sure you have the correct serial port selected under Tools → Serial Port and “Arduino Uno” selected under Tools → Board. If you have never used the Arduino IDE before, this turoial should get you started.

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.”

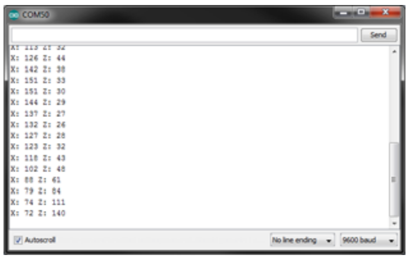


```
--------------------------------
SparkFun/GestureSense - I2C ZX Demo
================================
ZX Sensor initialization complete
Model version: 1
Register Map Version: 1
```

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.



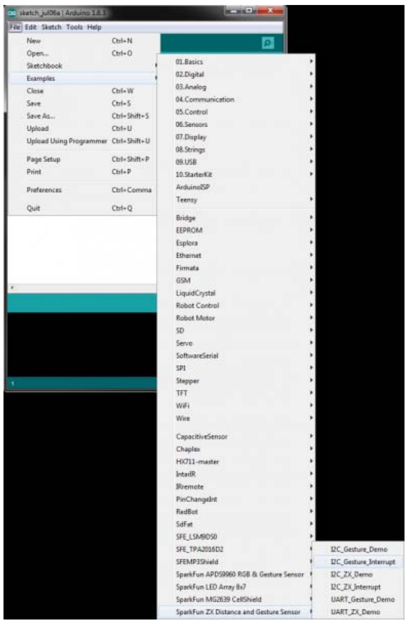
```
X: 113 Z: 32
X: 126 Z: 44
X: 142 Z: 38
X: 151 Z: 33
X: 151 Z: 30
X: 144 Z: 29
X: 137 Z: 27
X: 132 Z: 26
X: 127 Z: 28
X: 123 Z: 32
X: 118 Z: 43
X: 102 Z: 48
X: 88 Z: 61
X: 79 Z: 84
X: 74 Z: 111
X: 72 Z: 140
```

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.

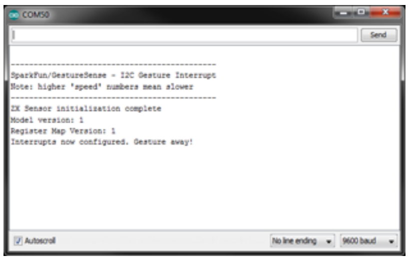


Here is the `I2C_Gesture_Interrupt.ino` sketch for reference.

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

Tests the ZX sensor's ability to read gesture data over I2C using
an interrupt pin. This program configures I2C and sets up an
interrupt to occur whenever the ZX Sensor throws its DR pin high.
The gesture is displayed along with its "speed" (how long it takes
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, 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
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 Interrupt");
  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 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 number");
  } 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;
}
```

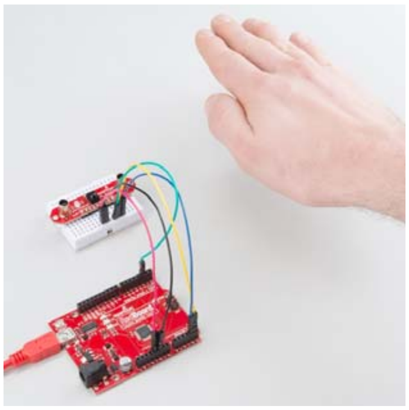
Run

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

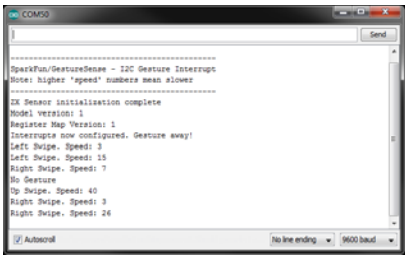


```
------------------------------------------------
SparkFun/GestureSense - I2C Gesture Interrupt
Note: higher 'speed' numbers mean slower
------------------------------------------------
ZX Sensor initialization complete
Model version: 1
Register Map Version: 1
Interrupts now configured. Gesture away!
```

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.



```
============================================
SparkFun/GestureSense - I2C Gesture Interrupt
Note: higher 'speed' numbers mean slower
============================================
ZX Sensor initialization complete
Model version: 1
Register Map Version: 1
Interrupts now configured. Gesture away!
Left Swipe. Speed: 3
Left Swipe. Speed: 15
Right Swipe. Speed: 7
No Gesture
Up Swipe. Speed: 40
Right Swipe. Speed: 3
Right Swipe. Speed: 26
```

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

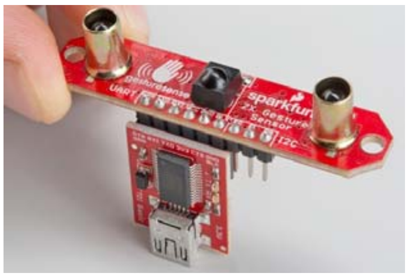
The ZX Sensor, in addition to responding to I^{2}C commands, continually transmits ZX data over its UART port. We can connect an FTDI Breakout directly to the ZX Sensor and read the output. You can use serial applications or the screen command (Linux or Mac) to view the output.

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.



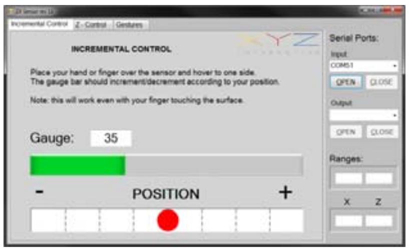
Download the ZX Demo application, and unzip it.

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.



INCREMENTAL CONTROL

Place your hand or finger over the sensor and hover to one side.
The gauge bar should increment/decrement according to your position.

Note: this will work even with your finger touching the surface.

Gauge: 35

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 I^{2}C 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 I^{2}C 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.