How to use the Vernier Go Direct Sensor Library, GDXLib

Installation

Vernier produces a family of Go Direct, or GDX sensors, that can communicate via Bluetooth Low Energy (BLE) radio. This library works with the Arduino BLE library to read those sensors. There are more than 45 different GDX sensors.

Arduino libraries are packaged sections of code that allow you to add functionality to a sketch. You can download the GDXLib library from the Arduino section of the Vernier website. It will be a ZIP file. To install it, start the Arduino IDE, and choose Include Library from the Sketch menu. Then choose Add .ZIP Library and select GDXLib. You should get a message that the library has been added.

Using the GDXLib Library

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

```
#include “GDXLib.h”
GDXLib GDX
```

GDX.open() - Three different variations:

There are three variations of this statement to connect to a GDX device by Bluetooth.

If you use just:

```
GDX.open:
```

the program will connect to the nearest GDX device and use its default channel.

If you use a statement like this:

```
GDX.Open("GDX-ACC 0H1019K1", 1, 500);
```

you are picking a GDX device, specifying the channel to be read, and the sample period in milliseconds.

If you use a statement like this:

```
GDX.Open("GDX*ACC XXXXXXXX", 1, 500);//Note the ‘*’ in the middle of
the device type
```

You are specifying the type of GDX device, but not the exact serial number, and also specifying the channel to be read and the sample period in milliseconds.

GDX.start();

Starts the periodic data collection by the GDX sensor.

`GDX.stop();`

Stops the periodic data collection by the GDX sensor.

`GDX.close();`

Closes the Bluetooth radio connection.

`GDX.readSensor();`

This function reads the latest reading from the specified channel of the Vernier GDX sensor.

Parameters that can be read from the GDX sensor:

Once a Bluetooth connection has been made to a Vernier GDX sensor, you can retrieve any of the following values from the sensor, by using these statements:

`GDX.readSensor()` The latest reading from the specified channel of the Vernier GDX sensor.

`GDX.deviceName()` The name and serial number of a GDX device. Use it to specify the device to which you want to connect.

`GDX.orderCode()` The name of the device connected. Use this to determine the device you actually connected with.

`GDX.serialNumber()` The 8-digit serial number of the device device you actually connected with.

`GDX.channelName()` The channel of the device being read

`GDX.channelUnits()` The units of measurement

`GDX.channelNumber()` The channel number selected on the device

`GDX.batteryPercent()` The battery charge level (0 to 100%)

`GDX.chargeState()` Status of battery charging (0=idle, 1=charging, 2=charging complete, 3=error)

`GDX.RSSI()` The "Received Signal Strength Indicator", which is a measure of the Bluetooth radio signal level from the sensor. The range is from about -90 to -30. The larger the number the stronger the signal (that is -40 is stronger than -50). We have the minimum acceptable level as -44 in the code.

`GDX.samplePeriodInMilliseconds``()` The time between readings in milliseconds.

Sample Sketches

There are several sample sketches included in the Examples folder.

`GDXLibDemoSimple` This is the simplest sketch. It shows the main parts of a program to read a GDX sensor.

```
//GDXLib DemoSimple (v. 20201123, using the 0.90 library code)
#include "ArduinoBLE.h"
#include "GDXLib.h"
GDXLib GDX;

void setup(){
  Serial.begin(9600);
  delay(500);
  GDX.open();  //use this line for proximity pairing
      //or
  //GDX.open("GDX-ACC 0H1019K1",1, 1000);//or specify specific device,
channel and period here
     //or
  //GDX.open("GDX*ACC XXXXXXXX",1, 1000);//or specify device type,
channel and period here
  delay(100);
  Serial.print("Found: ");
  Serial.print("GDX.orderCode() ");
  Serial.print(" ");
  Serial.println (GDX.serialNumber());
  Serial.print("GDX.channelName() ");
  Serial.println (GDX.channelName());
  GDX.start();

  for(int row=1;row<11;row++){
     Serial.print(row);
     Serial.print(" ");
     float channelReading =GDX.readSensor();
     Serial.print(channelReading);
     Serial.print(" ");
     Serial.println(GDX.channelUnits());
  }//end of for
  GDX.stop();
}//end of setup
```

```
void loop(){
}
```

GDXLibDemoWithSerialDisplay:

This is the similar to as GDXLibDemoSimple, except it has a function added to display parameters on a 16-character, 2-line display connected using three serial lines.

GDXLibDemoWithI2CDisplay:

This is the similar to as GDXLibDemoSimple, except it has a function added to display parameters on a 16-character, 2-line display connected using four I2C lines.

Known limitations of this version:

It reads only one device at a time. Later versions can read more than one.

It reads only one channel of the GDX device at a time. Later versions can read more than one.

It is limited to 5 readings/second.

For the reason above, it will not work with sensors like EKG, respiration Monitor Belt, the SND sensor doing sound waves, or any channel that requires dozens or hundreds, or thousands of readings per seconds. Later versions should support that.