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. This library works with the Arduino BLE library to read these 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.Begin() or GDX.Begin(deviceName, channel selected, samplePeriodInMilliseconds)

There are two ways to connect to a GDX device by Bluetooth. If you use just:

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

If you use a statement like GDX.Begin("GDX-ACC 0H1019K1",1, 500);

You are picking a device, speciying the dhannel to be read and the sample period in milliseconds.

This function will check for an analog (BTA) sensor connected to the Analog 1 connector of the Vernier Interface Shield. It is only needed if you plan to read data from a Vernier analog (BTA) sensor. If a sensor is found, it will read information about it, including:

deviceName, shortName, sensorUnits, sensorNumber

To use the information about the sensor in your sketch, you use a statement like:

```
Serial.print(GDX.sensorName);
```

This will print the name of the sensor on the Serial Monitor.

or:

```
Serial.print(GDX.sensorUnits);
```

This will print the slope used in the sensor's calibration units.

GDX.readSensor();

This function reads the specified channel of the Vernier GDX sensor. Just add a line of code like:

```
sensorReading = GDX.readSensor();
```

Here is a very simple sample sketch using the GDXLib library. This sketch is in our examples folder and is named GDXLibDemoSimple.

```
//GDXLib DemoSimple (v. 20200819, using the 0.85 GDXLib )
#include "ArduinoBLE.h"
#include "GDXLib.h"
GDXLib GDX;

void setup(){
  Serial.begin(9600);
  delay(500);
  GDX.Begin();  //use this line for proximity pairing
     //or
  //GDX.Begin("GDX-ACC 0H1019K1",1, 1000);//or specify device, channel and period here

  Serial.print("Found: ");
  Serial.println (GDX.deviceName());
  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(){
}
```

Parameters that can be read from the GDX sensor:

GDX.deviceName()

GDX.channelName()

GDX.channelUnits()

GDX.channelNumber()

GDX.batteryPercent()

GDX.chargeState() (0=idle, 1=charging, 2=charging complete, 3=error)

GDX.RSSI()

GDX.samplePeriodInMilliseconds()

Sample Sketches

There are XX sample sketches included in the Examples folder:

GDXLibDemoSimple:

The very simple sketch listed above.

GDXLibDemo:

Similart to GDXDemoSimple, but it prints out all the paramters that can be read from the GDX device on the Serial Monitor. It has code for reading the senor in the loop available, if you want to use it.

GDXLibDemoWithSerialDisplay:

This is the same as GDXLibDemo, except it has a function added to display all the parameters associated with the sensor, like deviceName, sensorName, units, sample Period, battery condition, and RSSI on a 16-character, 2-line serial-interfaced display.

GDXLibDemoWithI2CDisplay:

This is the same as GDXLibDemoWithSerialDisplay, except it assumes that the display is connected, by 4 I2C wires, 16-character, 2-line serial interfaced display.

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/s.

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