# Creating a Python Interface to the Powercast P2110-EVAL-01 Development Kit

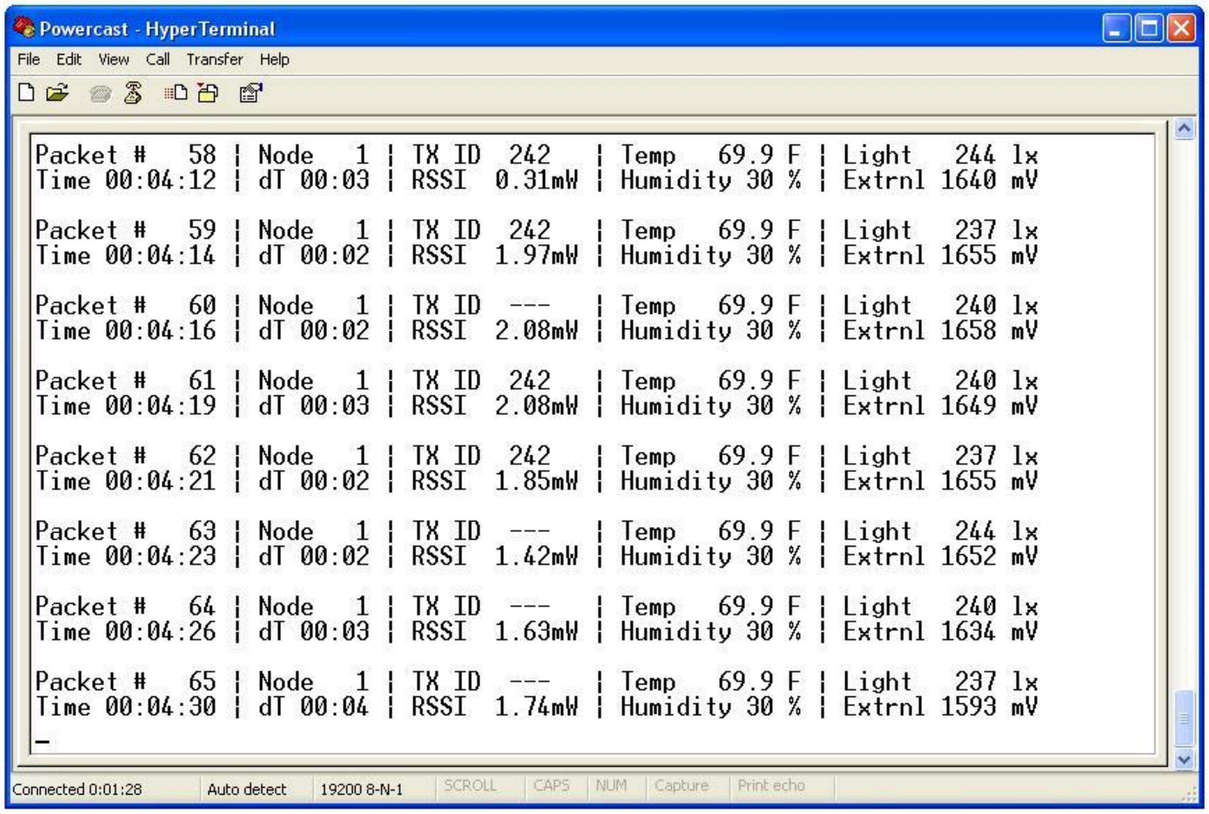
David Rogers

11/7/13

*Abstract: This application note describes an alternative way to read data from the Microchip 16-bit XLP Development Board contained in Powercast P2210-EVAL-01 sensor network development kit. This alternative way uses the flexible programming language, Python, which allows for the  rapid development of a rich user-friendly application built on top of the data being collected.*

## Introduction

A USB connection is a the typical way most electronic suppliers provide in order to directly interface with hardware. However, these suppliers only provide very simplistic manuals on how to read data from the hardware onto a computer via USB. The Powercast P2110-EVAL-01 manual[1] shows a way to connect to the Microchip 16-bit XLP Development Board using the terminal emulator application HyperTerminal.[2] This development board is a cluster head which receives all of the data from each of the sensor nodes in the network. Some sample data from the development board with network comprised of a single sensor node is shown below in Figure 1.



```
Packet #   58 | Node    1 | TX ID  242    | Temp    69.9 F | Light   244 lx
Time 00:04:12 | dT 00:03 | RSSI  0.31mW | Humidity 30 % | Extrnl 1640 mV

Packet #   59 | Node    1 | TX ID  242    | Temp    69.9 F | Light   237 lx
Time 00:04:14 | dT 00:02 | RSSI  1.97mW | Humidity 30 % | Extrnl 1655 mV

Packet #   60 | Node    1 | TX ID  ---    | Temp    69.9 F | Light   240 lx
Time 00:04:16 | dT 00:02 | RSSI  2.08mW | Humidity 30 % | Extrnl 1658 mV

Packet #   61 | Node    1 | TX ID  242    | Temp    69.9 F | Light   240 lx
Time 00:04:19 | dT 00:03 | RSSI  2.08mW | Humidity 30 % | Extrnl 1649 mV

Packet #   62 | Node    1 | TX ID  242    | Temp    69.9 F | Light   237 lx
Time 00:04:21 | dT 00:02 | RSSI  1.85mW | Humidity 30 % | Extrnl 1655 mV

Packet #   63 | Node    1 | TX ID  ---    | Temp    69.9 F | Light   244 lx
Time 00:04:23 | dT 00:02 | RSSI  1.42mW | Humidity 30 % | Extrnl 1652 mV

Packet #   64 | Node    1 | TX ID  ---    | Temp    69.9 F | Light   240 lx
Time 00:04:26 | dT 00:03 | RSSI  1.63mW | Humidity 30 % | Extrnl 1634 mV

Packet #   65 | Node    1 | TX ID  ---    | Temp    69.9 F | Light   237 lx
Time 00:04:30 | dT 00:04 | RSSI  1.74mW | Humidity 30 % | Extrnl 1593 mV
_
```

**Figure 1. Sample data collected from the Microchip 16-bit XLP Development Board via USB using HyperTerminal.**

This data stream is very hard to read and understand. As network complexity accumulates by adding additional sensor nodes it is very hard to grasp what is going on in the sensor network. Therefore there is substantial need for a robust application in order to sort and filter the data as necessary. A larger and more visual appealing view of the data can help human operators understand what is going in the network and act accordingly if there is something wrong in the environment being monitored or the network itself. A Python interface will provide the basis for the development of a feature-rich application. Python is easy to set-up and learn and has a strong development community for help and support.

[1] See http://www.powercastco.com/PDF/P2110-EVAL-01-manual.pdf for further details.
[2] See http://www.powercastco.com/zip/HyperTerminal.zip in order to download.

## Configuring Hardware and Installing Drivers

Before writing Python code a few things must be done first. The Microchip 16-bit XLP Development Board must be configured correctly and the corresponding USB-to-Serial driver files must be installed on your machine in order to communicate via USB properly. An in-depth overview of these steps is described in the Powercast P2110-EVAL-01 manual.[3] Please complete steps two and three in the manual before continuing on.

## What is Python?

From the Python programming language official website:[4]

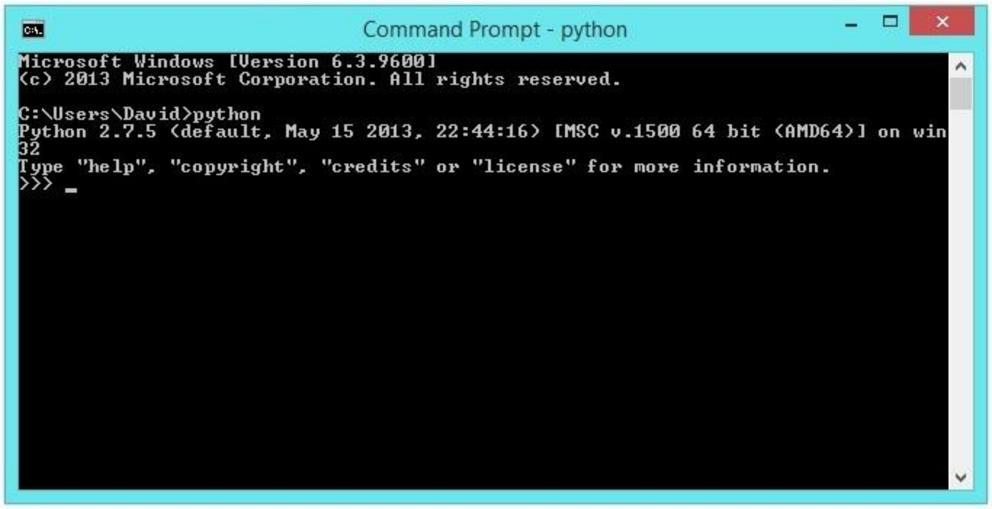
> *Python is a programming language that lets you work more quickly and integrate your systems more effectively. You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs.*

Python is available on just about any platform and is an extremely flexible interpreted language. There is a strong development community surrounding Python and this tutorial will illustrate a few modules developed by independent developers in order to interface with the development board and create a feature-rich application.

## Installing Python

This tutorial uses Python 2.7.x with the current latest stable release being 2.7.5. Please download and install the correct version for your system on the downloads page.[5] Be sure to download version 2.7.x and not the 3.3.x version of Python. Once downloaded follow the installation directions and install Python on your system.

If correctly installed on Windows one should simply be able launch a command prompt window and type python. This launches the Python interpreter. The output should be very similar to Figure 2. If an error occurs be sure to check that Python has been correctly added to your system's PATH environment variable.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\David>python
Python 2.7.5 (default, May 15 2013, 22:44:16) [MSC v.1500 64 bit (AMD64)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

**Figure 2. Showing how to access Python via command line.**

[3] See http://www.powercastco.com/PDF/P2110-EVAL-01-manual.pdf steps two and three.
[4] See http://www.python.org/.
[5] Visit http://www.python.org/download/ to download Python 2.7.x

## Installing pySerial

In order to communicate over a serial USB connection in Python an additional module is needed. This module is called pySerial.[6] Once again, this module needs to be downloaded and installed.[7] Visit the download page and be sure to download the correct version for your systems platform. PySerial only offers a 32-bit installer for Windows and therefore if your system is of 64-bit architecture you must download the source and install via the command line. If you are confused on how to install pySerial for 64-bit check out this thread[8] on Stack Overflow for help.

## Identifying the Correct USB Port

If the Microchip 16-bit XLP Development Board is not currently connected to your machine please connect it to an available USB port. Open up Device Manager on your machine and identify  the port at which the development board is connected. If you are having trouble locating the development board take a look at Figure 3 below. It should be found under the Ports (COM & LPT) heading. If the connection is not found please reinstall the drivers from the Powercast manual. Once found, please note the port at which the development board is connected.

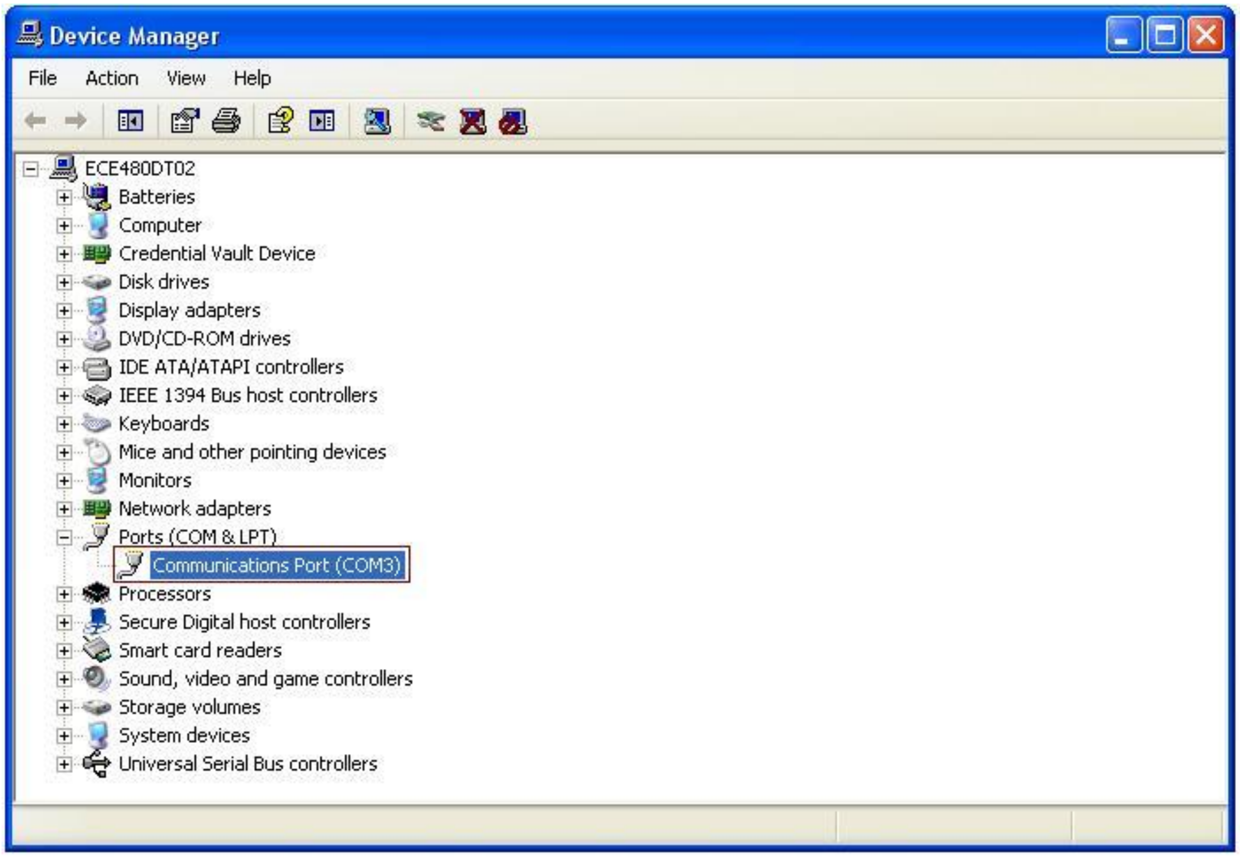


**Figure 3. A screen capture of the Device Manager interface with the development board connected to the computer. The connection is on port 'COM3'.**

[6] Documentation for pySerial can be found at: http://pyserial.sourceforge.net/
[7] Download pySerial here: https://pypi.python.org/pypi/pyserial
[8] Help on installing pySerial for 64-bit machines can be found at: http://stackoverflow.com/questions/8491111/pyserial-for-python-2-7-2

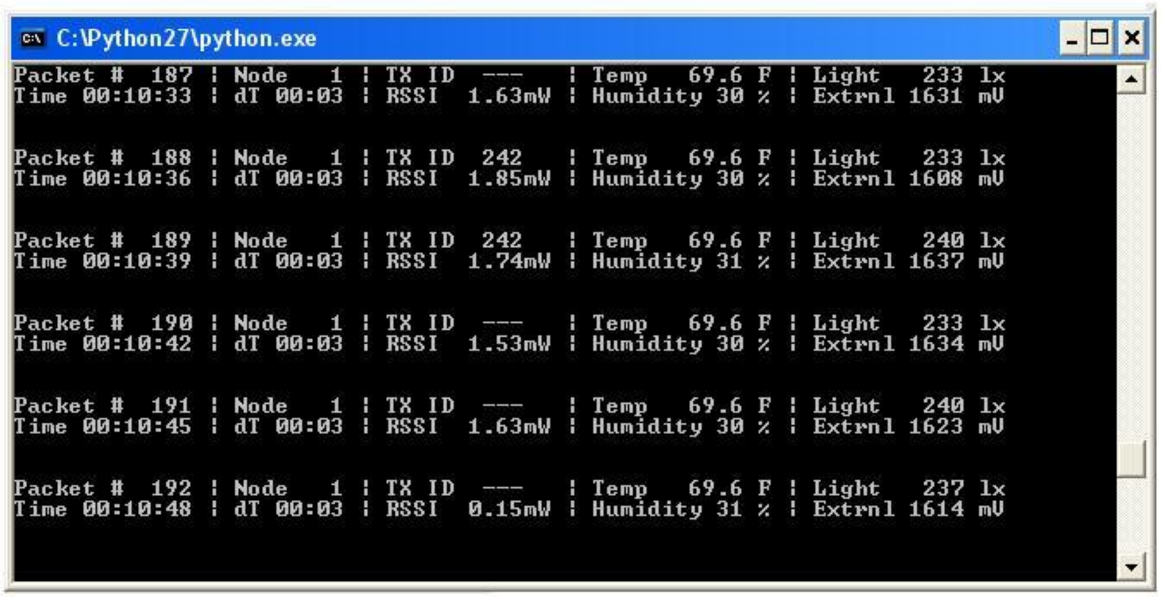
## Writing Simple Python Code

The last step is to write some simple Python code in order to interface with the development board. Figure 4 contains a few lines of sample code that will read the first 10000 bytes from the development board. In order to run this code simply copy it into a file with a '.py' extension and double click on it, but be sure to remove the line numbers and that the development board is connected properly. Looking further into the code Line 1 loads the pySerial library downloaded earlier into the namespace for use in the program. Line 3 sets-up the serial connection with the development board via USB. This is accomplished by using calling the Serial method from the pySerial library. The arguments in the expression consist of the port at which the development board is connected ('COM3' for my machine) , and the baud rate. The baud rate is hardware specific and is listed as 19200 as indicated in the Powercast manual.[9] There are additional arguments that can be ignored  because they are all set to the default values. For more information on the pySerial library check out the API on their website.[10] Line 4 simply declares a blank string called 'packet'. This will be used to capture one packet at a time and then print it out to the console. Next,  Line 6 initializes a for loop in order to count from 0 to 10000. This will allow the first 10000 bytes (in this case characters) to be read and then exit the loop. This could be changed to a while loop in order to have the program run indefinitely. In Line 7 one byte is read from the connection data stream and stored in the variable 'data'. Line 8 checks if the byte in 'data' is the capital letter 'P'. This is a simple way to check whether or not a new packet has begun. As shown in Figure 1 every packet has the same fields beginning with 'Packet #' and there is only one 'P' in the packet. Thus, it is safe, but not a best practice, to simply check for the character 'P' every time through the loop. If the condition is true then in Line 9 the packet contents are printed and the packet is reset to the letter 'P' (from the variable 'data') in Line 10. If the condition is false since 'data' does not equal 'P' then the else clause is triggered in line 11. Line 12 simply concatenates the next byte contained in 'data' to 'packet'. Finally, in Line 14 the connection to the port is closed. The outputs to this program are the same as the outputs in Figure 1 except now they are being read and outputted using Python. Figure 5 on the next page shows the output of the sample code in a command line interface. While this may not seem useful as it is the same functionality achieved using HyperTerminal, by reading in the data using Python a full-scale application can now be built on top.

```
01 import serial
02
03 connection = serial.Serial('COM3', baudrate=19200)
04 packet = ''
05
06 for i in range(0, 10000):
07     data = connection.read()
08     if (data == 'P'):
09         print packet
10         packet = data
11     else:
12         packet += data
13
14 connection.close()
```

**Figure 4. Sample Python code for printing the first 10,000 bytes received.**

[9] See http://www.powercastco.com/PDF/P2110-EVAL-01-manual.pdf step 4.4.
[10] See  http://pyserial.sourceforge.net/pyserial_api.html for more information on the pySerial library.



```
Packet #  187 ¦ Node    1 ¦ TX ID  ---    ¦ Temp    69.6 F ¦ Light   233 lx
Time 00:10:33 ¦ dT 00:03  ¦ RSSI  1.63mW  ¦ Humidity 30 %  ¦ Extrnl 1631 mV

Packet #  188 ¦ Node    1 ¦ TX ID  242    ¦ Temp    69.6 F ¦ Light   233 lx
Time 00:10:36 ¦ dT 00:03  ¦ RSSI  1.85mW  ¦ Humidity 30 %  ¦ Extrnl 1608 mV

Packet #  189 ¦ Node    1 ¦ TX ID  242    ¦ Temp    69.6 F ¦ Light   240 lx
Time 00:10:39 ¦ dT 00:03  ¦ RSSI  1.74mW  ¦ Humidity 31 %  ¦ Extrnl 1637 mV

Packet #  190 ¦ Node    1 ¦ TX ID  ---    ¦ Temp    69.6 F ¦ Light   233 lx
Time 00:10:42 ¦ dT 00:03  ¦ RSSI  1.53mW  ¦ Humidity 30 %  ¦ Extrnl 1634 mV

Packet #  191 ¦ Node    1 ¦ TX ID  ---    ¦ Temp    69.6 F ¦ Light   240 lx
Time 00:10:45 ¦ dT 00:03  ¦ RSSI  1.63mW  ¦ Humidity 30 %  ¦ Extrnl 1623 mV

Packet #  192 ¦ Node    1 ¦ TX ID  ---    ¦ Temp    69.6 F ¦ Light   237 lx
Time 00:10:48 ¦ dT 00:03  ¦ RSSI  0.15mW  ¦ Humidity 31 %  ¦ Extrnl 1614 mV
```

**Figure 5. Command Line output of the data through a Python program.**

## Extending Python Functionality

Now that data can be read into a Python program successfully, a feature-rich application can be developed. The modules matplotlib[11] and wxPython[12] are two great modules for graphing and developing a graphical user interface respectively. When used in combination an application can be built to visualize the data obtained from the development board in real-time as well as view and sort through past data. This will allow an operator properly identify problems within the network and in the environment the network is monitoring.

## Conclusion

This application note covered how to set-up and install the necessary modules to read data from the Microchip 16-bit XLP Development Board using Python 2.7. By using Python instead of HyperTerminal a full-scale application can be developed to better understand what is going on in the network. The development of this application is beyond the scope of this document but a few libraries such as matplotlib and wxPython are well developed and can be useful for further development. This document provided some sample code in order to help a user understand some Python syntax and set-up a serial connection to the development board. Overall, this note provides a great alternative to using the HyperTerminal interface provided by the Powercast manual.

[11] See http://matplotlib.org/
[12] See http://www.wxpython.org/