A Simple JeeNode Barometer

David Taylor, GM8ARV

I had been looking for a way to record a few weather parameters on a casual basis for some time and had already found indoor and outdoor temperature probes available on the Internet; I later discovered an indoor temperature and humidity probe. However, pressure seemed to elude the Chinese makers of these probes so, late last year, I bought a wireless weather station—which would have been ideal were the software up to the same standard as the hardware. Unfortunately though, while the indoor display panel worked most of the time, it was so unreliable when interrogated by software that I sent the unit back for a refund. During my Internet explorations I discovered a low-cost pressure sensor, which was an add-on hardware to an *Arduino* compatible computer. The discovery was the *JeeNode* from JeeLabs.

http://jeelabs.com/products/jeenode

Seeing Guy Martin's recent article made me realise that this alternative approach might be of interest to GEO readers, as only a very few non-surface-mount components are required, and the programming model is the widely supported *Arduino*.

http://www.arduino.cc/

The JeeNode

JeeLabs offer a variety of formats of hardware, all designed for 'physical computing' (i.e. real-world measurements) and the hardware is designed to be very easy to connect together. The main computing part is designed around *Arduino* compatibility, so there's plenty of software available, and there is an extensive library of software to support the hardware boards you can add to a *JeeNode*. I wanted to drive this from a PC, so I chose a *JeeNode* with a USB port, but there is quite an interest in battery-powered *JeeNode*s which can be interrogated remotely with an 868 MHz RF signal.

The *JeeNode* can be programmed with the standard *Arduino* development environment, which is well described and well supported. There is forum here where you can get help.

http://arduino.cc/en/Guide/Environment

The development environment communicates with the *JeeNode* via a USB lead with a serial COM port emulation. Once your program (in *Arduino* nomenclature, your sketch) is running on the *JeeNode*, the same COM port is available for the PC to communicate with the *JeeNode*.

Hardware

For this project, I chose a *JeeNode* with a USB interface and a *Pressure Plug*, a small board for theis atmospheric pressure sensor, which is based on a BMP085 chip.

http://jeelabs.net/projects/hardware/wiki/JeeNode_USB

http://jeelabs.net/projects/hardware/wiki/Pressure_Plug

You can get various multi-pin connectors to solder to the board to adapt the configuration to best suit your own needs—I used just a single 6-pin header so that I could

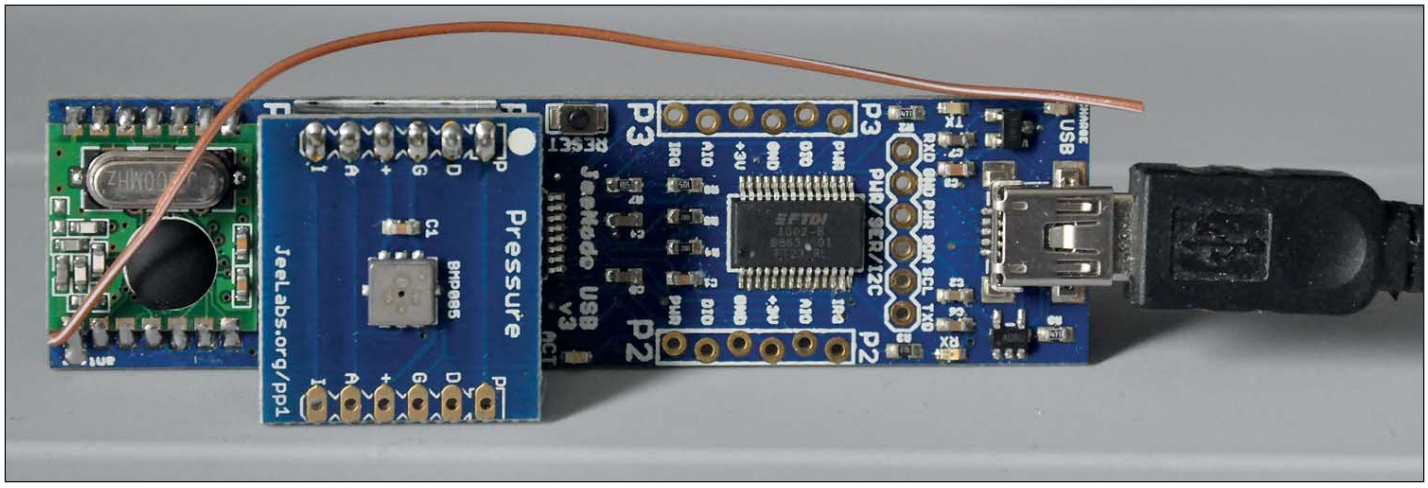

Figure 1 - General view of the JeeNode Barometer.

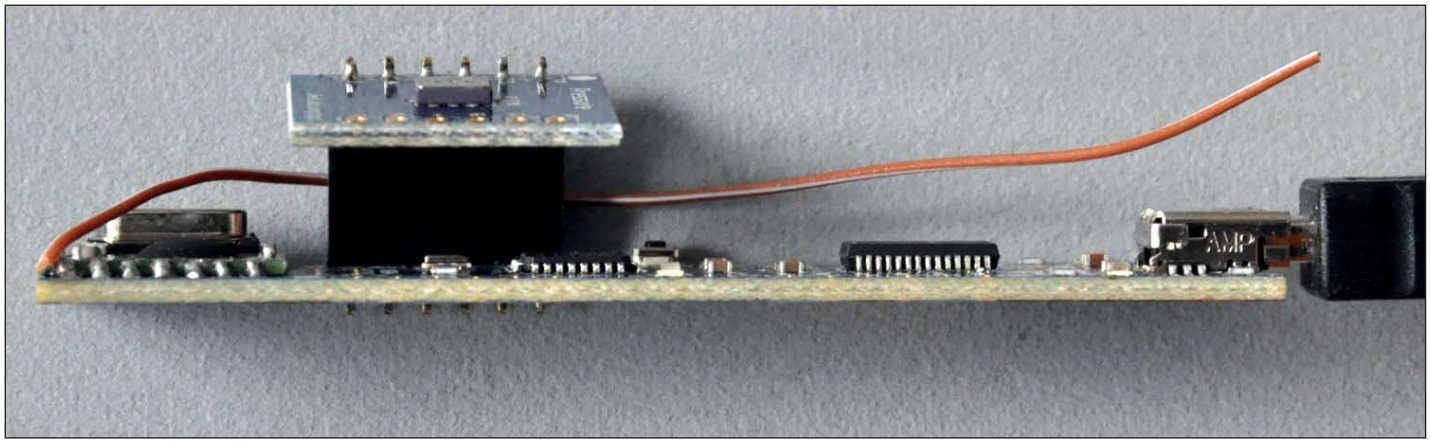

Figure 2 - An elevation view, showing how the Pressure Plus is mounted above the main JeeNode USB board.

unplug the pressure sensor if required. You can buy these boards ready-made (as I did), and there are some kit options if you have surface-mount facilities (which I do not).

Software

Getting data from the JeeNode

There is sample code provided to drive the BMP085 in a *JeeNode* configuration, and all I had to do was to provide serial access to that data. Most of the listing, in Appendix 1, is simply copied from the sample provided. As this is both my first *Arduino* and first *JeeNode* program, I could probably have done things in a much better way. The function I wanted was to be able to send a serial character to the board and get back the pressure data. Line 32 in the execution loop waits until a character is received on the serial line. Any character will do to elicit a response. Lines 36-45 get the current pressure (and temperature) reading, and calibrate it into the expected values. Lines 48-56 send the values back to the serial port, including a prefix of the characters 'BMP' and a carriage return suffix so that you could read the data a line at a time.

Using the Data

As regular readers may recall, I am a fan of MRTG for simple graphing of data over both short and long intervals, so I wrote a small program which interrogates the *JeeNode* and returns data in a suitable form for MRTG. Please contact me if you wish a copy of this program.

You could easily write a program in *Visual Basic*, or your favourite language, to interrogate the data by sending one character to the COM port, and reading the response.

An alternative, should you wish a stand-alone unit such as Guy Martin described in GEO Quarterly 34, would be to add a compatible graphics board from the JeeLabs shop and program it appropriately.

http://jeelabs.com/products/graphics-board

The Photos

Figure 1 shows a general view of the *JeeNode* Baraometer, looking directly down on the board. The USB connector is on the right and the (unused) RF interface on the left. The pink wire is the antenna for the RF interface. The Pressure Plug is the square board left of centre, and you can just see a few of the pins from the surface-mount processor to the right of it. The chip right of centre is the FTDI serial-to-USB chip. There are red and green LEDs which flash when serial data is sent or received.

Figure 2 is an elevation view of the board, and illustrates how the *Pressure Plug* is mounted above the main *JeeNode* USB board. The black item is a multi-way socket allowing the *Pressure Plug* to be removed and different hardware substituted.

Appendix 1 - Code Listing

```
(0 ) // Ports demo, reads out a BMP085 sensor connected via I2C
(1 ) // 2009-02-17 <jc@wippler.nl> http://opensource.org/licenses/mit-license.php
(2 )
(3 ) // 2010-05-22: added support for all resolution modes
(4 ) // 2010-05-25: extended to also broadcast all readings over wireless
(5 ) // 2010-06-17: add power saving logic, should reduce consumption by over 90%
(6 ) // 2010-06-24: improved power savings, several "hot spots" optimized
(7 )
(8 ) // see http://news.jeelabs.org/2010/06/20/battery-savings-for-the-pressure-plug/
(9 ) // see http://news.jeelabs.org/2010/06/30/going-for-gold-with-the-bmp085/
(10)
(11) #include <JeeLib.h>
(12) #include <PortsBMP085.h>
(13)
(14) PortI2C two (4);
(15) BMP085 psensor (two, 3); // ultra high resolution
(16)
(17) // This power-saving code was shamelessly stolen from the rooms.pde sketch,
(18) // see http://code.jeelabs.org/viewvc/svn/jeelabs/trunk/jeemon/sketches/rooms/
(19)
(20) void setup() {
(21)   Serial.begin(57600);
(22)   Serial.print("\n[bmp085demo]");
(23)
(24)   psensor.getCalibData();
(25) }
(26)
(27) void loop() {
(28)   // sensor readout takes some time, so go into power down while waiting
(29)   // int32_t traw = psensor.measure(BMP085::TEMP);
(30)   // int32_t praw = psensor.measure(BMP085::PRES);
(31)
(32)  if (Serial.available())
(33)
(34)  {
(35)   char ch = Serial.read();
(36)   psensor.startMeas(BMP085::TEMP);
(37)   delay(16);
(38)   int32_t traw = psensor.getResult(BMP085::TEMP);
(39)
(40)   psensor.startMeas(BMP085::PRES);
(41)   delay(32);
(42)   int32_t praw = psensor.getResult(BMP085::PRES);
(43)
(44)   struct { int16_t temp; int32_t pres; } payload;
(45)   psensor.calculate(payload.temp, payload.pres);
(46)
(47)   // this code is not needed for use as remote node, keep it for debugging
(48)   Serial.print("BMP ");
(49)   Serial.print(traw);
(50)   Serial.print(' ');
(51)   Serial.print(praw);
(52)   Serial.print(' ');
(53)   Serial.print(payload.temp);
(54)   Serial.print(' ');
(55)   Serial.print(payload.pres);
(56)   Serial.print("\n");
(57)
(58)  }
(59)  delay (300);
(60) }
```