Chapter 13

Data Logging and Data Processing

After constructing a radio telescope consideration needs to be given on how to log and process the information your receiver is gathering. The old way used paper chart recorders, which have been made obsolete now by the personal computer. Even an old obsolete computer can be valuable tool for the electronics bench or as a processing and control unit for a radio telescope. Old unused PCs can often be obtained free of charge and put to use in a second life. PCs can be energy hungry, so there is an intermediate step that is possible, the stand-alone data logger. A low-power device based around a microcontroller and some flash memory used to constantly record the data from the telescope. Data can be dumped once a day or so to a PC for further processing.

Here we will look at easy ways to log data. Microcontrollers are a valuable tool for instrument control and data logging. Constructing your own custom devices is certainly within your capability and is often much easier to achieve than building analog RF electronics. Space does not allow a full enough discussion on the subject, so it is recommended that you read up on the subject later. However, readymade data logging tools are available to buy. Here are some options.

The Logging Multimeter

The photograph shows a Tenma model 72-7732. Although it is not the cheapest model in its category, it has some useful features (Fig. 13.1).

For the purposes of data logging, it comes complete with a USB interface and data logging software. This relies on the personal computer to log and store data. Since a radio telescope back end usually provides a DC output signal an instrument like this acts as an analogue to digital converter, and a PC interface.

In addition to this as well as the usual DC, AC voltage and current ranges, diode tests, etc., it can measure capacitor values and has a useful frequency counter operating at up to 400 MHz. This is useful for setting up oscillators in radio circuits. The only down side to having only one of them is when you come to build the next radio telescope! To use it as a test instrument you have to stop logging data.

There are lower cost models around for as little as one fifth the price of the Tenma, which typically have an RS232 serial interface in place of the USB. This

Fig. 13.1. Tenma model 72-7732.

would suit older PCs or may work via a USB to serial interface. One model looked at was an "unbranded" unit sold via eBay. It also had a frequency counter but would only work at up to 10 MHz, much less useful. It is always worth spending a little more money, or a lot more money if you are serious, to get a better quality instrument that you can rely on for accuracy and robustness.

More Data Loggers for the PC

Pico Technology provides a wide range of data loggers and PC-based instrumentation. Shown here is the high-resolution 24-bit data logger. For more details see their website, http://www.picotech.com (Fig. 13.2).

The limitation of this model is it can only handle a range of ±2.5 V. The output of the radio telescope must therefore be scaled to operate within this range.

Fig. 13.2. High-resolution 24-bit data logger from Pico.

All Pico data loggers come with a software application called PicoLog, which handles the data capture to the PC and provides visualization in the form of graphs.

These units work out about twice the price of the Tenma meter, so maybe you should buy two Tenmas? Well, maybe not. The Pico tool will certainly provide a much better resolution for capturing fine detailed changes.

The other advantage of a tool like the ADC24 is the ability to write your own software applications and embed the functions of the unit into your program. The manual supplied gives all the information you need.

The Home Brew Method

For those of you who really want to feel you have built the system yourself, there are simple ways of constructing your own logging interface. Together with some low cost software available via the Internet you can have an eight-channel logging solution. More than one channel is useful if you have multiple instruments to log, say, by monitoring more than one radio frequency simultaneously. Since the performance of radio receivers can be influenced by ambient temperature, you may wish to build temperature sensors into your instruments and log the temperature alongside the data. Later calibration adjustments can be applied based on how the temperature varies.

A useful microchip is available from Maxim, the MAX186. Available in DIP format (dual inline package), it has 20 pins. Its data logging functions work without the need to write and store software onboard the chip, so it is an off-the-shelf solution. The chip can be configured a number of ways to talk to a computer, but the application here shows how to connect it via an old style printer port to a PC (Fig. 13.3).

Note the convention used in the circuit diagram shows a dot where lines connect. If lines cross each other without a dot being present, they are electrically

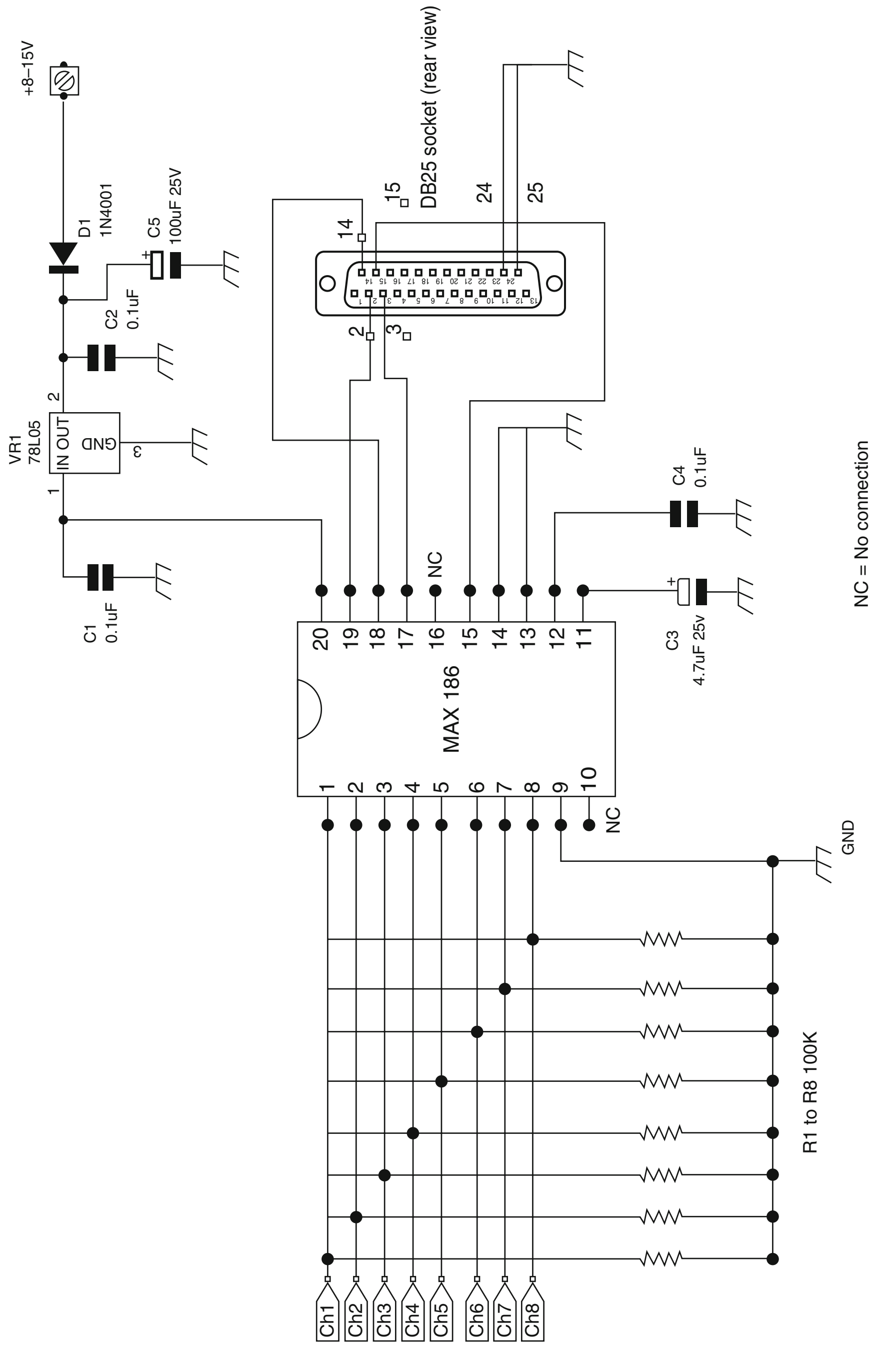

Fig. 13.3. The Maxim MAX186 chip connected to a PC.

isolated from each other. The circuit can be constructed onto strip board but the following figures show the PCB layout (Fig. 13.4).

Radio *Skypipe* Software

The previous logger design is particularly suited to the radio *Skypipe* software available for evaluation from http://www.radiosky.com. The evaluation version has a few limitations but is fully functional as a single-channel logging option. The full version is relatively inexpensive and allows the full eight channels to be logged; it also opens up the possibility of streaming data via a live website.

Skypipe is simple to use and is the modern equivalent of a chart recorder, storing its data instead to a computer drive. This is another highly recommended piece of software for the beginner.

Figure 13.5 shows a VLF receiver plot taken on June 8, 2007, showing a solar SID event.

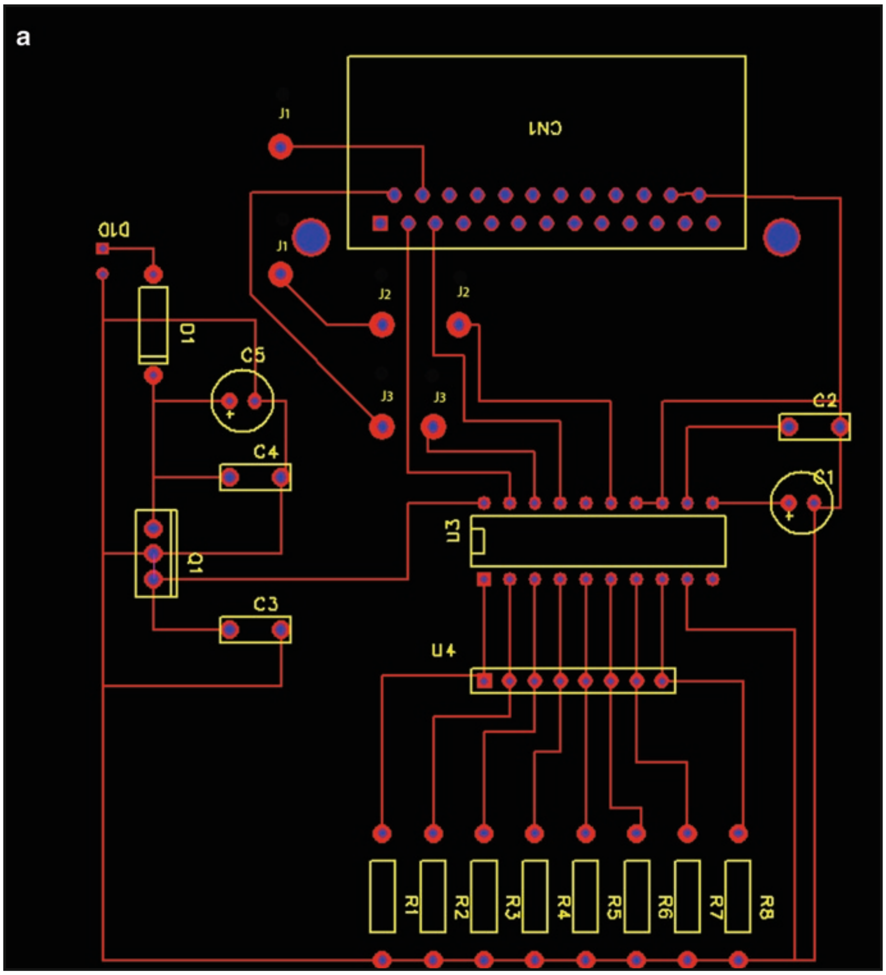

Fig. 13.4. (**a**) Top view: J1, J2, and J3 connection points require jumper cable fittings. (**b**) Copper side bottom view.

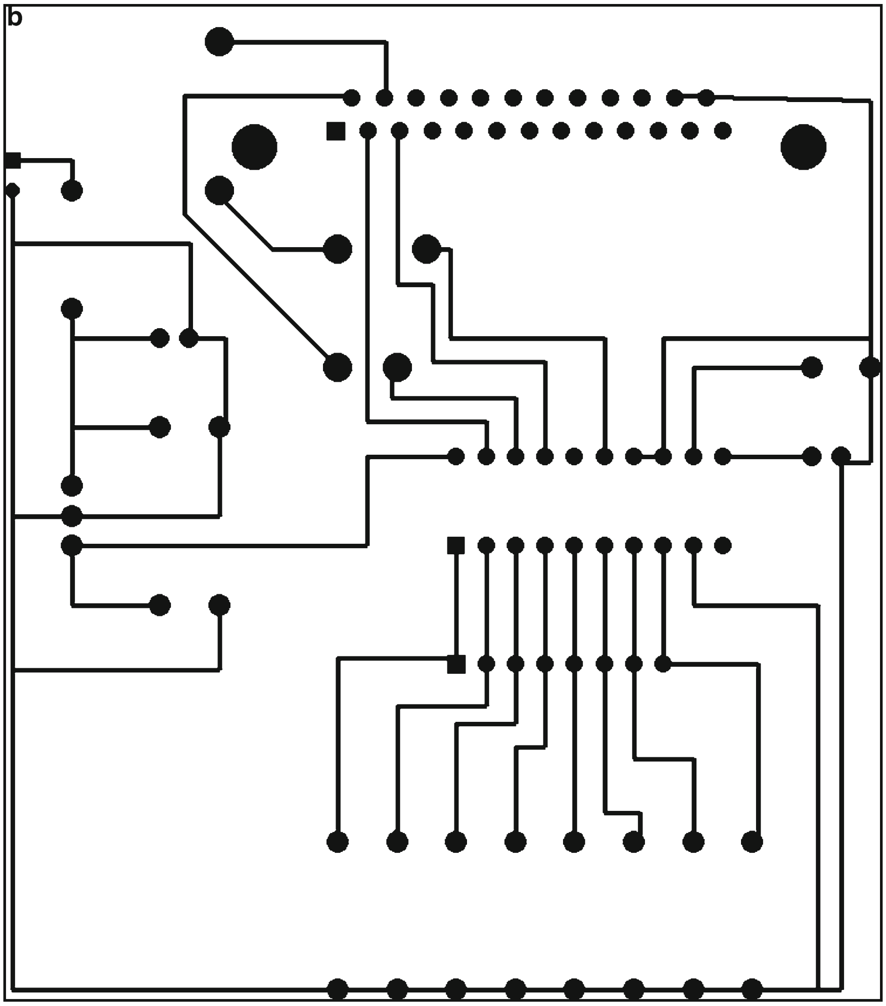

Fig. 13.4. (continued) (**b**) Copper side bottom view.

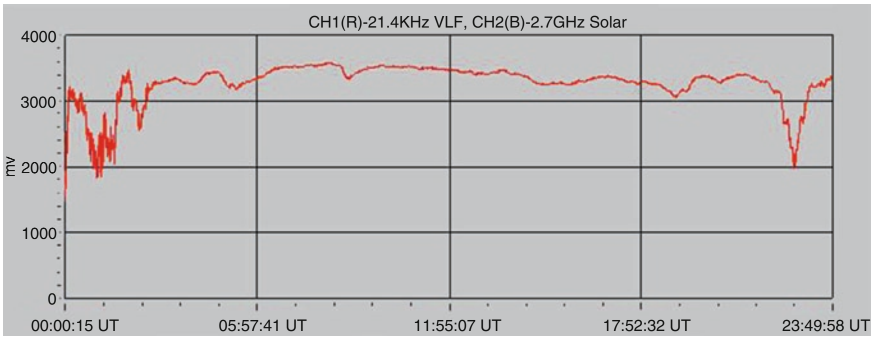

Fig. 13.5. Solar SID event logged using Radio *Skypipe* software and a VLF hardware receiver. The image is dated June 8, 2007. It shows a typical SID disturbance at around 08:00. (Image courtesy of Martyn Kinder from the British Astronomical Association Radio Astronomy Group).

The Controller Option

The controller is a more sophisticated option than we have seen so far. By controller, we mean a small single-board low-power computer based on a microcontroller. There are many available ready-built. The advantage of a controller is that it can be used to control the operation of the radio telescope, as well as handle data capture.

The controller used by the British Astronomical Association Radio Astronomy Group is the Futurlec ADuC832, available from Futurlec direct via their website, http://www.futurlec.com. This has an on-board microcontroller manufactured by Analog Devices, the ADuC832. The down side here is that you will need to create your own embedded software for it. Example software is provided with the kit (Fig. 13.6).

The features on this controller include 64K of flash program memory. (This may not sound like much, but it's surprising what can be accomplished with that amount of space.)

The software can be downloaded from a PC using a serial lead. It has eight channels of analog-to-digital conversion and two channels of digital-to-analog conversion, plus up to 1 Mb of RAM can be installed as an option. It has a real-time clock on board and can be made to communicate over very long distances using RS422 or RS485 by the addition of low-cost chips.

It would be possible to control the Philips CD1316L of the solar radio telescope project tuner via I^{2}C using this controller, as well, of course, to log the output data. With the proper software configuration the Philips tuner could easily be made to sweep quickly across a range of frequencies to turn the instrument into a spectrometer.

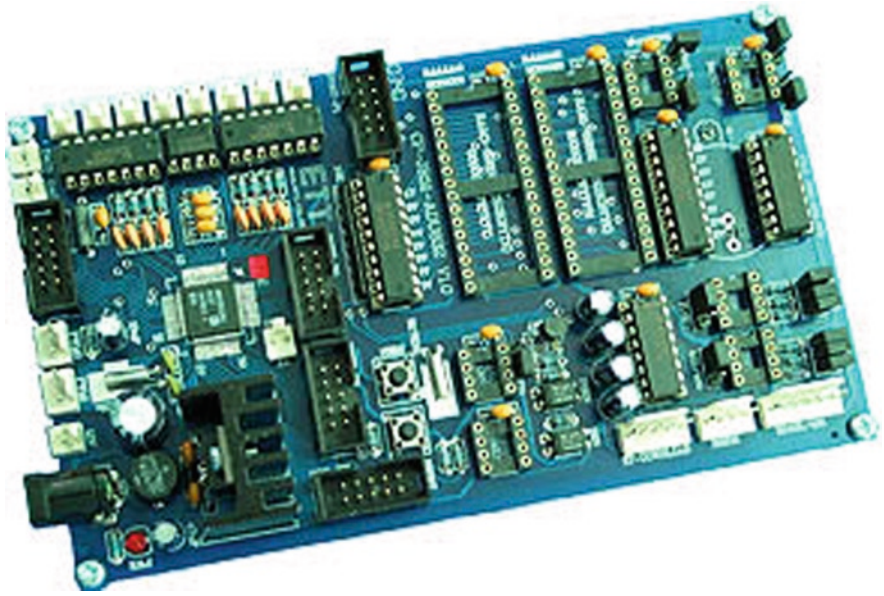

Fig. 13.6. The controller from Futurlec.

The Arduino Duemilanove

Take a look at the Arduino; it is one of the most engaging and addictive bits of technology around. It comes in a number of variations, and there are some clones such as the Seeeduino (yes, three e's). There is nothing particularly special about the hardware itself. What makes the Arduino a highly recommended platform is the community behind it. The software behind the Arduino concept is all open source and free. There are many individuals and groups using it for all kinds of projects, so there are lots of coding examples to choose from and adapt to your own projects. This is always a great start when something is new to you (Fig. 13.7).

Developing software is beyond the scope this book, but this is such a useful piece of hardware that included here is some sample software to get you going. This is truly one of the easiest microcontroller platforms to work with. There are also lots of ideas available to help interface this hardware into well-known software packages such as Flash, Visual Basic, Matlab, Java, and many more. The development environment is itself based on an open source programming language called Processing. Processing was developed to be easy to learn and use for teaching programming and as a platform for designers (rather than programmers) to be able to create visual software. Processing is an ideal language to learn if you are not already a programmer, to develop your own applications to work with the Arduino hardware.

The Duemilanove (meaning "2009" in Italian) has the following features:

- 14 digital input–output pins
- 6 of the above pins can be made to output PWM (useful for controlling motors)

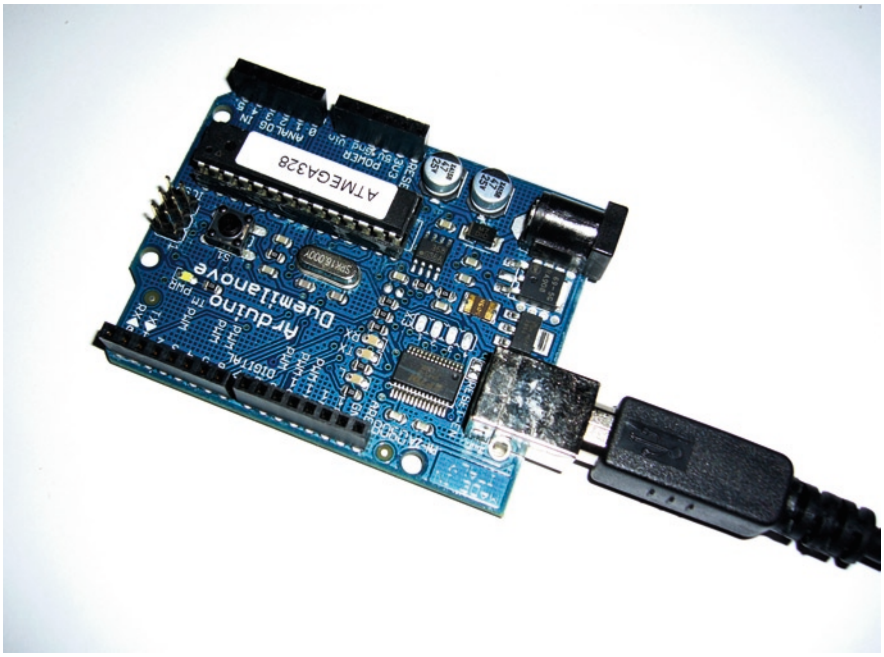

Fig. 13.7. The Arduino Duemilanove controller.

- 6 analog input ports, 10-bit resolution (giving 0–1,023 for up to 5 V inputs)
- 32K of Flash RAM for programs using the ATmega328
- 2K of SRAM
- 1K of EEPROM

When you compare the memory capacity of a microcontroller with that of a PC it seems tiny, and you wonder how this could possibly be useful for anything. Well, it goes a long way. If the primary aim is to provide an analog-to-digital interface for a PC, then an average program running on the microcontroller will probably take up about one-third of the available space or less. Most of the sophistication is handled by the PC.

Communication with the PC is by USB cable. Drivers are supplied that create a virtual COM port, making serial communications easy between the two. In fact the FTDI chip is already known to Microsoft Windows and should not pose any problems to install.

Another advantage of using the Arduino is the ability to piggyback hardware on top. Piggyback boards are known as shields. There are shields that can add SD card slots for huge storage solutions, Real-time clocks, even wireless or wired LAN boards, are available.

To get started download the Arduino programming software from the home page at http://arduino.cc and uncompress the archive to a folder on your computer. Then download the Processing software from http://processing.org and uncompress it to a folder of its own. The two environments look alike, because the Arduino software is based on Processing. It is easy to confuse the two when you are developing PC/Arduino applications!

To set up the Arduino software you need to tell it which serial port to use. The FTDI chip emulates a serial COM port on the PC. It may appear as COM3 or COM5 but may be even higher if you have used other similar devices before. In the menu "Tools/Serial Port" click over the port used for the Arduino. If you are using a modern PC or laptop it may be the only COM port available anyway. Under the menu "Tools/Board" you can set the type of Arduino you are using.

Next type in the following program to upload to the microcontroller:

```
// Code to read the analog value supplied from analog pin 0
int val;
int inputPin=0;      // Set analog input pin 0

void setup(){
 Serial.begin(9600); // Star serial communication at 9600 bps
}
void loop(){
 val = analogRead(inputPin); // Read the input pin range 0 to 1023 10 bit
 Serial.print(val/4, BYTE); // Convert to 8 bit and send via serial port
 delay(100);            // Wait 100ms
}
```

At the top of the window is a row of buttons. Click on the one square button with the right arrow in it, the upload button. This first compiles the software then programs the chip on the Arduino. You will see leads flash on the board when it is programming. Once they stop a text message appears on the PC screen. If you make errors in typing you may get warnings on the screen. Correct them, checking against the above code, and try again. When completed press the reset switch on the Arduino board, and close the PC software.

Next open the Processing software and type in this program:

```
/**
 * Simple Analog Data logging application using Arduino hardware
 * Samples are taken every second.
 * Data is read from the serial port, and displayed live as a bargraph
 * while at the same time is saved as a comma separated text file
 */

import processing.serial.*;        //import serial library

Serial myPort;                      // Create object from Serial class
int val;                           // Data received from the serial port
String d = str(day());             //Set current date
String m = str(month());
String y = str(year());
PFont font;                        // create a font object
String[] dataout = {"logger data "+d+":"+m+":"+y}; //Create String array with header

void setup()
{
 size(300, 600);                  //Define window size
 font = createFont("Arial",22);
 textAlign(CENTER);
 textFont(font);
 println(Serial.list());           //Prints a list of available COM ports
 String portName = Serial.list()[0];    //Change the [0] value for your COM port
 myPort = new Serial(this, portName, 9600); //Open serial port
}

void draw()
{
 if ( myPort.available() > 0) {        // If data is available,
  val = myPort.read();                 // read it and store it in val
  delay(1000);                         // delay 1 second
 }
 background(255);                      // Set background to white
  fill(0);                             // set fill to black
  rect(30, height, 240, -val*2);       // create bar graph rectangle
  text(val,width/2,height-val*2-20);   // Add text value above

  // create string to add to array
  String s = str(hour())+":"+str(minute())+":"+str(second())+","+str(val);
  dataout = append(dataout, s);        // append string to the array
  saveStrings("data.txt", dataout);      // save the array to a text file
}
```

Note that the // symbol is used to add comments to help you understand how the software works. When ready save the software and click the Run button at the top left of the Processing software. A window is then created, displaying a real-time bar graph of the data being streamed from the Arduino. At the same time the data is saved to a text file called "data.txt" that appears in the same folder in which you saved the program. The current date is appended to the header text, and the current (PC) time is appended to the data value. This text file could be imported into Microsoft Excel or similar for data processing and graphing, or you could develop a more sophisticated Processing application to do the same thing.

This software is intentionally simplistic, so hopefully it is easy to understand. It needs further work to be a practical tool, but it gets you started. Note that the data is captured as a 10-bit value (0–1,023) but is compressed into an 8-bit BYTE (by dividing by 4) before sending from the Arduino. The 10-bit value takes up four BYTEs, but it also makes the sensitivity very high. For example the sensitivity for a 10-bit ADC sample of a 5 V signal means 1 bit is about 5 mV. At that level even the 5 V power supply from the USB port showed extensive ripple on the order of 15–20 mV. By converting to 8-bit the output is much smoother, and 1 bit represents 20 mV. You need only code a single channel, but there are six analog input ports; so you could modify it to capture all the channels you need. It is up to you to experiment with it.

The PC Sound Card as a Data Capture Device

We have seen an example of using the PC sound card as a data capture device. The line input port of a sound card is simply another form of analog-to-digital interface, and so is the microphone port. Although the microphone port is capable of measuring very small signal levels, due to the existence of onboard preamplification within the sound card, it is the line input port that will be most useful.

The definition of line input level varies between consumer and professional equipment. Consumer equipment is usually set up to handle up to 1 V peak to peak, and professional equipment about 1.7 V peak to peak. Assuming the sound card is intended for the consumer market that means the peak amplitude the card will deal with is only 0.5 V. A circuit could be constructed to scale a 5 V signal down to 0.5 V, but it is counterproductive to do so. Where a sound card comes into its own is when a conventional radio is employed with an audio output. There are several software applications out there that can process an audio signal in order to extract useful information. The example recommended is Spectrum Lab released by Wolfgang Buescher, whose amateur radio call sign is DL4YHF. The website where you can download it is http://freenet-homepage.de/dl4yhf/spectra1.html.

One of the main uses of Spectrum Lab is a processing technique known as FFT, or Fast Fourier Transform analysis. A generic sound signal provides us with a variable waveform whose amplitude varies with time. At any given instant, the amplitude of that signal is a sum of the all the amplitudes of all the different tones or frequencies that make up a sound. Human hearing can in theory detect sounds with frequencies up to 20 kHz. In practice for most people it will be less than this.

Our hearing changes with age, and the upper limit of frequency detection falls. This is of course not important, as the PC is going to do the analysis for us. What it means in practice is that a PC sound card has a flat response up to 20 kHz, often 21 kHz, and then its response rolls off steadily and falls below ambient noise levels between 23 and 24 kHz for most cards (assuming a 48k/b sampling rate).

To picture how a sound card is useful for us, consider how the AM, or amplitude modulation, radio works for a moment. The AM radio signal is an amplitude varying RF signal, and the information it carries depends on how the amplitude varies. The amplitude "envelope" of an AM communications signal is in fact the audio signal superimposed onto the RF wave. Figure 13.8 illustrates this.

The radio translates this varying amplitude to a low frequency. The translation process of the radio removes the radio frequency "carrier" and provides a time-varying audio signal equivalent to the profile of the original wave. The FFT spectrum generated by Spectrum Lab converts this time-varying signal into the frequency domain. The plot then displays how amplitude varies with frequency. Any arbitrary waveform can be represented by a series of pure sinusoidal components of different frequencies and amplitudes. The branch of mathematics involved with converting time-varying signals into the frequency domain is Fast Fourier Transform.

Therefore a sound card can turn an ordinary radio into a narrowband spectrometer, and by tuning the radio the spectrum can be scanned across a wide range. This is not only potentially powerful but very low cost. The chances are high, if you are reading this book that you already have the PC and sound card. What's even better is that Wolfgang offers his software free for personal use.

The basic principles of using Spectrum Lab were covered in the VLF receiver chapter. But Spectrum Lab is capable of much more. It has a built-in scripting language that you can use to automatically capture spectra of plots when your test conditions are met. For example if this was used to count meteors, the script could

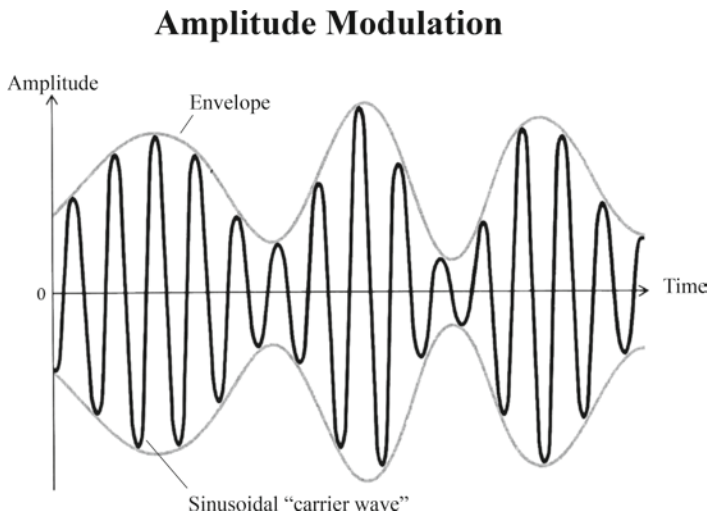

Fig. 13.8. Amplitude modulation of a sine wave. In this example the envelope of the wave is a single frequency audio tone superimposed on a higher frequency carrier. In practice the carrier is much higher frequency than shown.

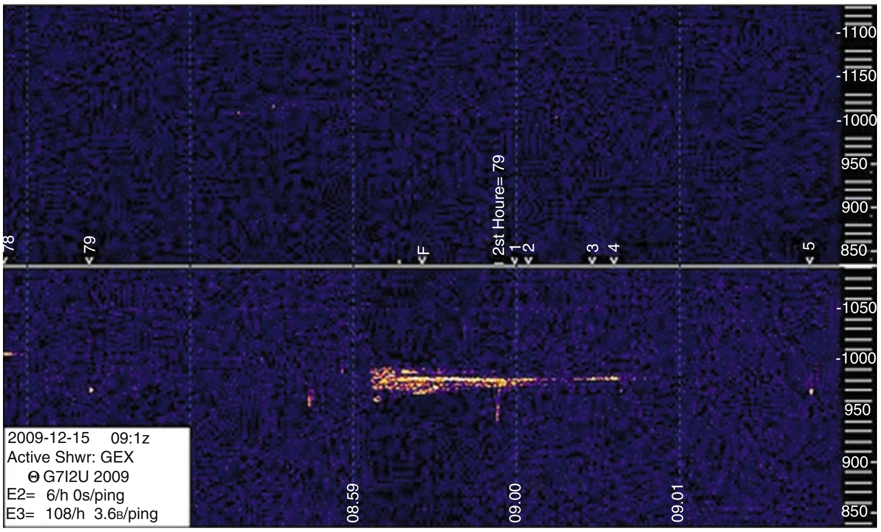

Fig. 13.9. Meteor radio spectrum of a Geminid taken on December 12, 2009, by Andy Smith G7IZU using Spectrum Lab software.

be configured to build up a count when a reflection event is detected, and if the reflection event was unusually long Spectrum Lab could be used to capture the spectrum. Long events could be due to fireball activity. This is the way Andy Smith uses Spectrum Lab to record meteor activity. An example of a spectrum he recorded is shown in Fig. 13.9.