The Arduino Uno is capable of many functions. Having fourteen digital input-output pins and six analog input pins it is able to read data and users are able to use that data in many ways. This application note shows how to implement a web server that hosts and displays the data that is read by the Arduino. This is achieved through a combination of the Arduino Ethernet shield and programming using software that comes with the Arduino.

**Keywords**

Arduino, Web, Server, Ethernet Shield, Data, Monitoring, Hosting
Introduction

The Arduino Uno is a microcontroller board that uses the ATmega328 microcontroller. The board itself comes with everything needed to support the microcontroller, so all that is needed to use it is to connect it to a computer via USB, power it via an AC-to-DC adapter or a battery. The Arduino is programmed using software provided. The java-based programming language is similar to C++ with some modifications and modifications. It is a simple language that most programmers will not have trouble learning and using. An important feature of the Arduino is its ability to connect different add-on modules called shields. This is thanks the way the Arduino’s connector are positioned. The Arduino Ethernet shield is one of these modules that can be easily added onto the Arduino. The Arduino Ethernet shield allows the Arduino board to connect to the internet. It uses the Wiznet W5100 Ethernet chip which provides a network stack for the Arduino. The pin layout is the same as the Arduino Uno which allows other shields to be stacked on top.

With the combination of Arduino Uno and the Arduino Ethernet shield, users are able to take data that is read by the analog inputs of the Arduino and display the data on a web browser. This allows for user to monitor the data from anything that can access the internet and without being near the Arduino itself. This application note will go through implementing this use starting from how to setup the Arduino Uno and the Arduino Ethernet shield to read analog data and then how to take that data and post it on a web browser to be viewed over the internet. The ports used for reading inputs will be shown on the Arduino and the base program will be given. This application note will also go over rewriting the base program to fit the user.

Setup

Hardware:

- Arduino Uno
- Arduino Ethernet Shield
- Router connected to network
- Personal Computer connected to router

Software:

- Arduino IDE
Setting up the Arduino Uno and Ethernet Shield

The Arduino Uno is designed to have the capability to add various shields onto it with relative ease. Figure 1 shows the top of the Arduino Uno. The red circles on Figure 1 show the pins of the Arduino. The pins are carefully labeled on the Arduino. For now we will not worry about the functions of each pin, but only note their location. If you wish look into the functions of each pin, please refer to [1] found in the references. There you can find pin names and their corresponding functions.

Figure 2 shows the Arduino Ethernet shield. Note the red circle in Figure 2. These show the long wire-wrap headers of the board. These headers allow the Ethernet shield to be stacked on top of the Arduino Uno by placing the headers into its pins. After placing the shield on top of the Arduino, it should look like Figure 3. Thanks to this design, the layout of the pins is the same as if the Arduino had no shield. Also, this allows for additional shields to be stacked on top of each other. With this setup, we are ready to program the Arduino.

Programming the Arduino

The people of Arduino give a wonderful example that gives a base for programming our web server. Open up the Arduino software and select File > Examples > Ethernet > Webserver. This will load the code that is referenced in [2]. The code should look like Figure 4. We will be altering this program to fit our needs.
A simple web server that shows the value of the analog input pins using an Arduino Wiznet Ethernet shield.

Circuit:
* Ethernet shield attached to pins 10, 11, 12, 13
* Analog inputs attached to pins A0 through A5 (optional)

created 18 Dec 2009
by David A. Mellis
modified 20 Mar 2012
by Tom Igoe

#include <SPI.h>
#include <Ethernet.h>

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = {
    0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192,168,1, 177);

// Initialize the Ethernet server library
// with the IP address and port you want to use
// (port 80 is default for HTTP):
EthernetServer server(80);

void setup() {
    Serial.begin(9600);
    // start the Ethernet connection and the server:
    Ethernet.begin(mac, ip);
    server.begin();
    Serial.print("server is at ");
    Serial.println(Ethernet.localIP());
}

void loop() {
    // listen for incoming clients
    EthernetClient client = server.available();
    if (client) {
        Serial.println("new client");
        // an http request ends with a blank line
        boolean currentLineIsBlank = true;
        while (client.connected()) {
            char c = client.read();
            // if you've gotten to the end of the line (received a newline character) and the line is blank, the http request has ended,
            // so you can send a reply
            if (c == '\n' && currentLineIsBlank) {
                // send a standard http response header
                client.println("HTTP/1.1 200 OK");
                client.println("Content-Type: text/html");
                client.println("Connection: close");
                client.println();
                client.println("<!DOCTYPE HTML>");
                client.println("<html>");
                // add a meta refresh tag, so the browser pulls again every 5 seconds:
                client.println("<meta http-equiv="refresh" content="5"></html>");
                // output the value of each analog input pin
                for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
                    int sensorReading = analogRead(analogChannel);
                    client.print(analogChannel);
                    client.print(" is ");
                    client.print(sensorReading);
                    client.println("<br />");
                }
                client.println("</html>");
                break;
            }
            if (c == '\n') {
                // you're starting a new line
                currentLineIsBlank = true;
            } else if (c != '\r') {
                // you've gotten a character on the current line
                currentLineIsBlank = false;
            }
        }
        // give the web browser time to receive the data
        delay(1);
        // close the connection:
        client.stop();
        Serial.println("client disconnected");
    }
}

Figure 4 – Arduino Example Web Server Code
Now that we have the base code for the web server, let’s look closer at the code to see what it does.

```cpp
for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
    int sensorReading = analogRead(analogChannel);
    client.print("analog input ");
    client.print(analogChannel);
    client.print(" is ");
    client.print(sensorReading);
    client.println("<br />");
}
```

The code above takes readings from the analog inputs of the Arduino and prints them out. You can change the print various print strings to print out what kinds of values are being read. Specifically, the client.print() command is used to send text or data to the web page we will be using. We can see in the code that some of the print statements use HTML code such as \texttt{br \/>.}

Now that we know where in the code our values are read and printed to our webpage, we will go into setting up the webpage.

There are two options when creating the webpage. One route is to specify an IP address of the Ethernet send and then sending the data to a local web browser based on that IP address. The code

```cpp
IPAddress ip(192,168,1, 177);
```

allows you to specify the IP address of the Ethernet shield. You should alter it so that it matches your network setup. The data then can be accessed by entering the IP address of the Ethernet shield into the URL bar of a web browser. This is seen in Figure 5. The data then can be updated by reloading the webpage. Being a local web browser you cannot access it from outside your own network. This makes the applications of this very limited. In order to take our data and put it into a web browser that can be accessed anywhere we need to get a static IP. A static IP is an IP address that your internet service provider assigns to your connection. This is the second option when creating a web page.

To find the static IP address, you must log into your router’s administration page. This is done by entering your router’s default IP address into a web browser’s address bar. This can be
found in your router’s manual or online. Figure 6 gives a table of default IP address based on the router manufacturer.

<table>
<thead>
<tr>
<th>Router Manufacturers</th>
<th>Default IP Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Com</td>
<td>192.168.1.1</td>
</tr>
<tr>
<td>Apple</td>
<td>10.0.1.1</td>
</tr>
<tr>
<td>Asus</td>
<td>192.168.1.1, 192.168.1.220</td>
</tr>
<tr>
<td>Belkin</td>
<td>192.168.2.1, 10.1.1.1</td>
</tr>
<tr>
<td>Buffalo</td>
<td>192.168.11.1</td>
</tr>
<tr>
<td>Dell</td>
<td>192.168.1.1</td>
</tr>
<tr>
<td>D-Link</td>
<td>192.168.0.1, 0.30, 0.50, 1.1, 10.1.1.1</td>
</tr>
<tr>
<td>Linksys</td>
<td>192.168.0.1, 1.1</td>
</tr>
<tr>
<td>Microsoft</td>
<td>192.168.2.1</td>
</tr>
<tr>
<td>Motorola</td>
<td>192.168.10.1, 20.1, 30.1, 62.1, 100.1, 102.1, 1.254</td>
</tr>
<tr>
<td>MSI</td>
<td>192.168.1.254</td>
</tr>
<tr>
<td>Netgear</td>
<td>192.168.0.1, 0.227</td>
</tr>
<tr>
<td>Senao</td>
<td>192.168.0.1</td>
</tr>
<tr>
<td>SpeedTouch</td>
<td>10.0.0.138, 192.168.1.254</td>
</tr>
<tr>
<td>Trendnet</td>
<td>192.168.0.1, 1.1, 2.1, 10.1,</td>
</tr>
<tr>
<td>U.S. Robotics</td>
<td>192.168.1.1, 2.1, 123.254</td>
</tr>
<tr>
<td>Zyxel</td>
<td>192.168.1.1, 2.1, 4.1, 10.1, 1.254, 10.0.0.2, 0.138</td>
</tr>
</tbody>
</table>

**Figure 6 – Default Router IP Addresses**

Once logged into the router’s administration page, look for the router’s IP address (this can also be labeled as WAN IP). Once found, record that address because we will be using it after we setup up port-forwarding on the router. Port-forwarding tells the router where to redirect incoming requests from outside its own network. We want the router to redirect specific requests to the Ethernet shield. This is done by assigning the Ethernet shield a port number. The default number is seen in the code `EthernetServer server(80);` where the port number is 80. In the router’s port-forwarding page, enter the IP address of the Ethernet shield into the corresponding field. Then enter the 80 into the port field. In some cases they will give you a
range for your port. In this case enter 80 in both fields. Figure 7 is an example of a port-forwarding page. For more information about port-forwarding visit [3] where you can find specific information on various routers and how to port forward them. Once that is done, all that is left is to enter the static IP address into a web browser’s address bar with the addition of the port number. The number being entered should be in the form of [Static IP Address]:[Port#]. A more specific example would be 213.123.456.128:80 where 213.123.456.128 is the static IP address and 80 is the port number. What this will do is that it will send a request to your router. The router sees the port 80 and redirects the address to the Ethernet shield. This gives you the web page with the data retrieved from the Arduino. With the ability to retrieve information from a system without actually being there, the applications of this web server are various.

References