Creating a Lighttpd, PHP5, and SQLite3 Web Interface on the Raspberry Pi

Nick Falco

November 2013

Abstract
This project explains how to set up an interface between the Arduino Mega microcontroller and Raspberry Pi microcontroller running the Raspbian operating system. In the system the Arduino reads temperature values through its analog port and sends the values to the Raspberry Pi. The Raspberry Pi then displays the values on a website using a Highcharts linegraph and a table format.

Contents

1 Configuration Steps

The following section describes the necessary steps to initially configure Lighttpd, PHP5, and SQLite3.

1.1 Installing Lighttpd, PHP5, and SQLite3

Step 1: Install lighttpd

apt-get install lighttpd

Step 2: Install php5

apt-get install php5, php5-common, php5-dev, php5-cli

Step 3: Install php5-cgi

apt-get install php5-cgi

Step 4: Install sqlite3

apt-get install sqlite3

Step 5:

apt-get install php5-sqlite

Step 6: Enable the fastcgi module and the php PDO drivers

lighttpd-enable-mod fastcgi
lighttpd-enable-mod fastcgi –php

Step 7: Reload lighttpd

service lighttpd force-reload

NOTE:
At this point php PDO drivers will be enabled. and you must now verify that PHP5 is configured properly.

Step 8: navigate to the directory /var/www/

cd /var/www/

Step 9: Now create a new file index.php

nano index.php

Step 10: Add the following contents to the file:

<?php phpinfo(); ?>

Step 11: Type the following into the URL on your web browser and verify that PDO drivers are enabled.

localhost/index.php

1.2 Setup SQLite3 Database

Launch SQLite3 and create new database file SQLite3 ¡database name¿ (e.g. SQLite3 data.db) Create a table with the following command. Count is the sample number, Time is the actual time the sample was taken, Temp is the temperature value measured.

- CREATE TABLE Temperatures(Count int, Time char(20), Temp real);

Import a comma separated file tempData.csv into table named Temperatures.

- .separated ","
- .import tempData.csv Temperatures. An example tempData.csv is shown below:
  1, 3/8/2010 12:00, 22
  2, 3/9/2010 12:00, 23
  3, 3/10/2010 12:00, 21
  4, 3/11/2010 12:00, 27
  5, 3/12/2010 12:00, 26
  6, 3/13/2010 12:00, 33
  7, 3/14/2010 12:00, 32
  8, 3/15/2010 12:00, 19
  9, 3/16/2010 12:00, 21
  10, 3/17/2010 12:00, 19

Verify that the data was successfully imported into the database by running the to following query:

- SQLite3 data.db
- SELECT * FROM Temperatures;

The data you imported should appear below the SQL select statment

1.3 Give a fixed IP address to Raspberry Pi

Navigate to the following directory /etc/network/

- cd /etc/network/

Copy the original interfaces file so you have a backup

- cp interfaces interfaces.o

Edit the interfaces file using nano

- nano interfaces

In the interfaces file make the following changes:
Change the line
iface eth0 inet dhcp
to
iface eth0 inet static

Below this line enter the following using your router settings. (mine are different than yours) Here's an example:

address 192.168.1.118 (your new static ip address)
netmask 255.255.255.0 (netmask from your router)
network 192.168.1.1
broadcast 192.168.100.255
gateway 192.168.100.254 (gateway address from your router)

Now you need to set up your DNS servers. In your router find out what your DNS servers are and append the following onto the file /etc/resolv.conf:

- server <your DNS server number>
- server <your DNS server number>

2 Accessing Database Using PHP PDO Handler

At this point you should have completed the initial configuration steps. The system is now completely setup for data to be read from the database.

2.1 Using HTML and PHP PDO Handler to Print Data in a Table format

In order to access data from the the database through a *.php file you will need to make some changes to your index.php file. If you have the following line of code "<?php phpinfo(); ?>" from Section 1.1 in your index.php file - delete it. Now add the following lines of code to index.php to read data from the database and print it in an html table.

First access your index.php file using nano.

- sudo nano index.php

Now, add the following php code to your index.php file.

```
<?php
try {
$dbh = new PDO('sqlite:/var/www/realtime_data.db');
foreach(dbh → query('select * fromTemperatures') as $row)
{
result_temp[] = $row['Temp'];
result_count[] = $row['Count'];
}
}
catch(PDOException $e) {
echo  $e → getMessage();
}
? >
```

3 Arduino Mega Configuration

The following section discusses how to use the Arduino Mega to read the LM35 temperature sensor and transmit the temperature value serially to the Raspberry Pi. It is best to use an engineering approach in collecting data from the Arduino and sending data to the Raspberry Pi. This involves first verifying the temperature circuit works properly and then sending data to the Raspberry Pi.

3.1 Building the Arduino Temperature Circuit using LM35 Sensor

Tools Required

1. Arduino Mega Microcontroller
2. Raspberry Pi Microcontroller
3. 22 Gauge Electronic Wire
4. LM-35 Temperature Sensor
5. Solderless Plug-in BreadBoard

Wiring the Circuit

The process of wiring the temperature circuit is outlined below. The pinout for the LM-35 is included in Figure 1.

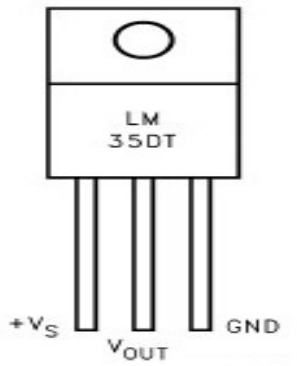

Figure 1: Pinout Diagram of LM-35 Temperature Sensor

To wire the temperature circuit to the Ardunio connect the +Vs temperature sensor pin to the 5 volt Vcc output pin on the Arduino, the Vout temperature sensor pin to the A0 analog input on the Arduino, and the temperature sensor GND pin to the GND ground pin on the Arduino. The circuit diagram is shown in Figure 2.

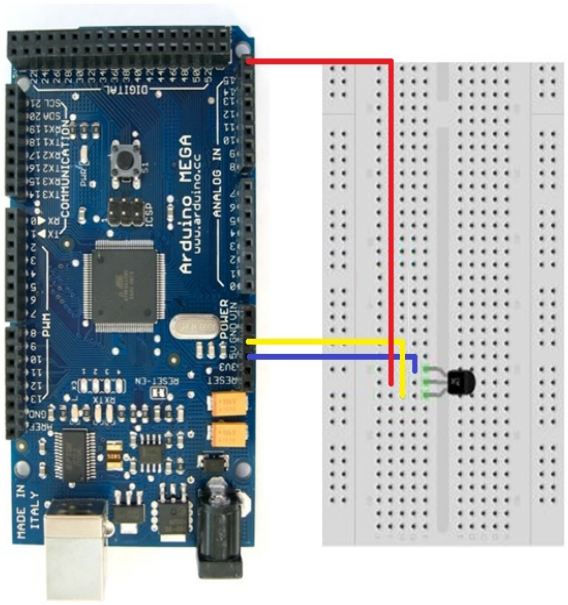

Figure 2: Temperature Sensor Circuit Diagram

3.2 Testing the Arudino Temperature Circuit

This section discusses how to verify the temperature recording circuit on the Arduino by reading in a temperature value and then sending it through the serial port to a PC.

First

Download the Arduino IDE at http://arduino.cc/en/Main/Software and upload the following code onto the Arduino Mega board.

```
double temp;
int tempPin = 0;

void setup()
{
        t Serial.begin(9600);
}

void loop()
{
         temp = analogRead(tempPin);
         temp = ( 5.0 * temp * 100.0) / 1024.0;
         Serial.print("TEMPRATURE = ");
         Serial.print(temp);
         Serial.print("*C");
Serial.println();
delay(1000);
}
```

Second

The next step is to download a serial port monitoring tool on your PC and see if you are getting accurate data from the Arduino. I choose to use the open source program "Tera Term" as my serial monitoring tool. Tera Term may be downloaded at the following address:

http://ttssh2.sourceforge.jp/index.html.en

Configure Tera Term to the COM Port that the RS232 Cable is connected. The baud rate should also be set to 9600. If all is wired and configured properly you should begin seeing temperature data on the screen being sent from the Arduino as shown in Figure 2.

In order for this program to work for the rest of your application you need to comment out the lines of code that say the following:
Serial.print("TEMPRATURE = ");
Serial.print("*C");

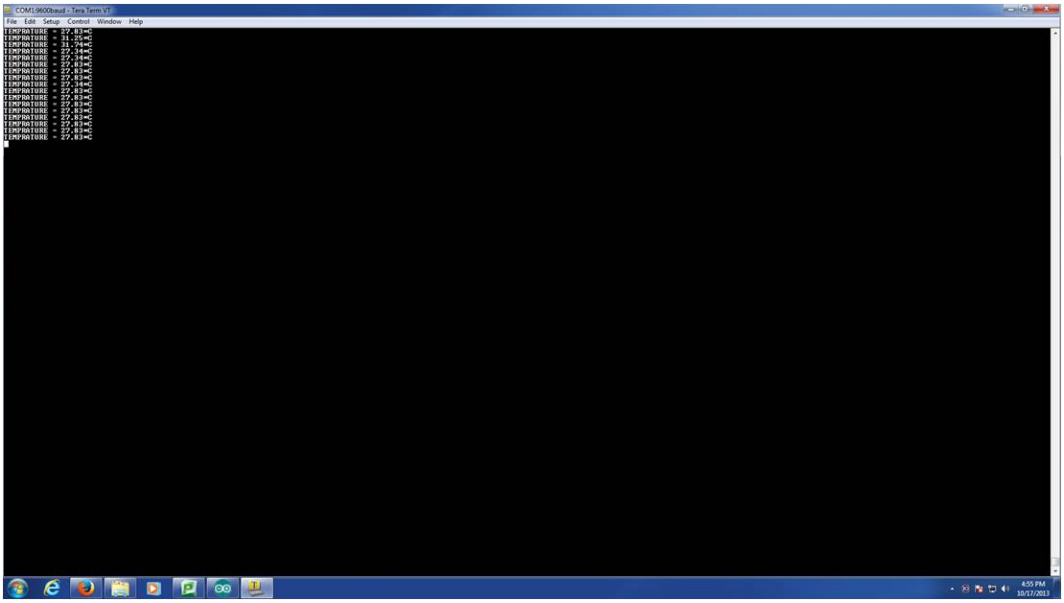

Figure 3: Tera Term After Recieving Several Temperature Values

3.3 Sending Tempeature Data to the Raspberry Pi

After verifying that the temperature circuit was working by sending data to a PC, it is easy to send data to the Raspberry Pi. To do this you must first configure the serial port on the Raspberry Pi.

3.3.1 Configuring Raspberry Pi Serial Port

There are two steps to configuring the Raspberry Pi serial port. These steps are to disable the serial port login, and to disable the Raspberry Pi bootup info.

Disable Raspberry Pi Bootup Info

To diable the serial port login you need to edit the file /boot/cmdline.txt. The contents of the file look like this:

dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

Remove all references to ttyAMA0 (which is the name of the serial port). The file will now look like this:

dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline rootwait

Disable the Serial Port Login

The Raspberry Pi serial port login is used if you want to connect to the Raspberry Pi through an RS232 connection. This setting however prevents the user from using the serial port in other applications. To disable serial port login you need to edit the file /etc/inittab.

In the file you will see a line similar to the following:

T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

Comment this line by adding a # symbol in front of it and save the file.

#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

3.4 Recieving Serial Data with Raspberry Pi

A program must be written to recieve serial data from the Raspberry Pi and upload it to the SQLite database. This program uploads a sample number "Count", the current date and time "current_time", and the temperature value "indata" to the database file /var/www/realtime_data.db. The program is shown below:

```
#!/ussr/bin/python
# -*- coding: utf-8 -*-

import serial
import datetime
import sqlite3 as lite
import sys
import time

ser = serial.Serial('/dev/ttyAMA0', 9600, timeout=1)
ser.open()

count = 0

con = lite.connect('realtime_data.db')

try:
        while 1:
          indata = ser.readline()
          current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
          count = count + 1

          print indata + current_time
          print count

        with con:
          cur = con.cursor()
          cur.execute("INSERT INTO Temperatures VALUES( ?, ?, ? )", (count, current_time, indata))
          if count >100:
            cur.execute("DELETE FROM Temperatures")
            count = 0

except KeyboardInterrupt:
        ser.close()
```

4 Building the Web Interface

This section assumes that you have already configured SQLite3 and set up the LightTPD webserver on the Raspberry Pi. It also assumes that you have already established a collecting temperature data with the Arduino and serially transmitting it to the Raspberry Pi. In this section are the instructions to constructing the web interface.

First

The first step is to create a file named "index.php" the directory /var/www/. In this file add the following code:

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="refresh" content="5">
<LINK href="styles.css" rel="stylesheet" type="text/css">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Nick's Raspberry Pi </title>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js">
<script type="text/javascript">
<?php
try{
$dbh = new PDO('sqlite:/var/www/realtim_data.db');
foreach($dbh->query('select * from Temperatures') as $row)
{
$result_temp[] = $row['Temp'];
$result_count[] = $row['Count'];
}
}
catch(PDOException $e) {
echo $e -> getMessage();
}
?>
$(function () {
$('#container').highcharts({
title: {
text: 'Sensor Temperatures',
x: -20 //center
},
subtitle: {
text: 'Last 4 Days',
x: -20
},
xAxis: {
```

```
categories: [<?php echo join($result_time, ',') ?>]
},
yAxis: {
title: {
text: 'Temperature (°C)'
},
plotLines : [{
value : 0,
width : 1,
color : '#808080'
}]
},
tooltip : {
valueSuffix : '°C'
},
legend : {
layout : 'vertical',
align : 'right',
verticalAlign : 'middle',
borderWidth : 0
},
series : [{
name : 'Sensor1',
data : [<?phpechojoin($result_temp, ',' )?>]
}]
});
});
</script>
</head>
<scriptsrc = "js/highcharts.js"></script>
<scriptsrc = "js/modules/exporting.js"></script>
<divid = "container"style = "min−width : 310px; height : 400px; margin :
0auto"></div>
<center>
<?php
print("<tableid = n"sensorn">
<th>Time</th>
<th>Temp</th>
<th>Sample#</th>");
try{
$dbh = newPDO('sqlite : /var/www/realtime_data.db');
$rc = 1;
foreach($dbh → query('select ∗ fromTemperatures')as$row)
{
if($rc%2 == 0)
{
```

```
print("<tr class = n" alt n">");
print("<td>".$row['Time']."</td>");
print("<td>".$row['Temp']."</td>");
print("<td>".$row['Count']."</td>");
print("</tr>");
}
else
{
print("<tr>"); print("<td>".$row['Time']."</td>");
print("<td>".$row['Temp']."</td>");
print("<td>".$row['Count']."</td>");
print("</tr>");
}
$rc = $rc + 1;
}
}
catch(PDOException$e){
echo$e -> getMessage();
}
print("</table>");
?>
</center>
</body>
</html>
```

Second

In order to make the website display properly copy the styles.css file and Highcharts js folder and put them in the location /var/www/.

Third

Next launch the index.php program on the web browser. You should see a bar graph displaying the sample temperature data on the top of the screen and a HTML table with the values below it. If you would like to see realtime data run the command "python serial_collect_update.py" in the /var/www/ directory to start the python script which updates the database with temperature values from the Arduino.