Bluetooth® Thermometer

ELEN136 Final Project Summary
Author: Jamie Maloway (www.jamiemaloway.com)
Prepared For: Prof. David Williams
Okanagan College

Table of Contents

Section 1 - Overview

The aim of the BlueTemp project was to create a compact, lightweight, battery-powered thermometer which transmits temperature data wirelessly over a standard, 802.15.1 Bluetooth® connection. All temperature conversion and processing is taken care of on the device itself, reducing the necessary hardware and software on the host side to nothing more than a standard Bluetooth® dongle and a terminal program. As the data is sent as a simple serial data stream to a virtual COM port, simple utilities could be created to re-direct the incoming data stream for storage (data logging), conversion (Fahrenheit, Kelvin), or re-broadcast (RSS, Twitter, etc).

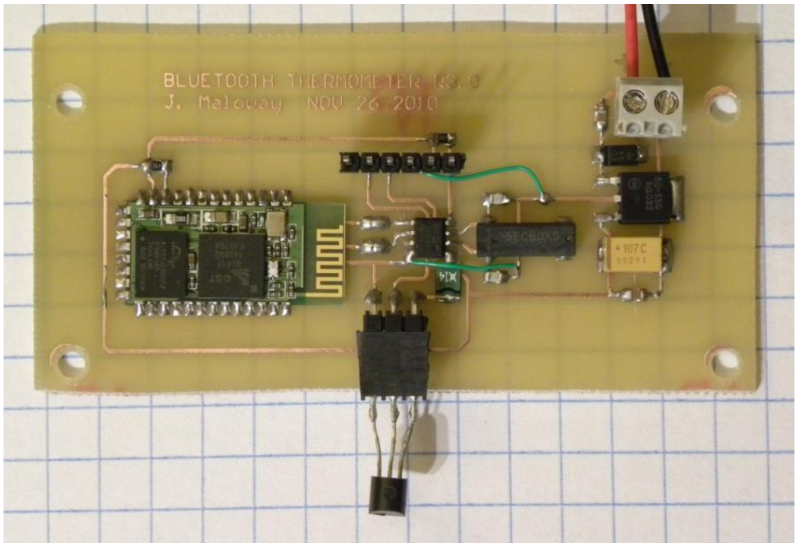


Figure 1.0: Completed Hardware

Section 2 - Hardware

The hardware is fairly simplistic, consisting of a PIC12F675 microcontroller, a Bluetooth® serial link module, a basic linear power supply circuit, and a Dallas/Maxim™ DS18B20 1-Wire® Thermometer IC. The power supply is reverse-polarity protected, and can handle a maximum input voltage of +25V DC.

Though the PIC12F675 does feature a 4MHz internal oscillator, an external clock was used due to inconsistent serial communication at lower clock speeds. Unfortunately, the Bluetooth® module used in this project is fixed at 9600 Baud, forcing the change in system clock speed rather than Bluetooth® link speed.

To insure forwards compatibility; a standard six-pin Microchip® ICSP (In-circuit Serial Programming) port is included on the board to allow for firmware upgrades and future experimentation.

The schematics and PCB layout were created in Altium Designer, with care being taken to ensure easy manufacturability. The PCB was fabricated using a positive photo etching system from MG Chemicals.

All schematics, layout data, and Gerber files are included in the ZIP archive.

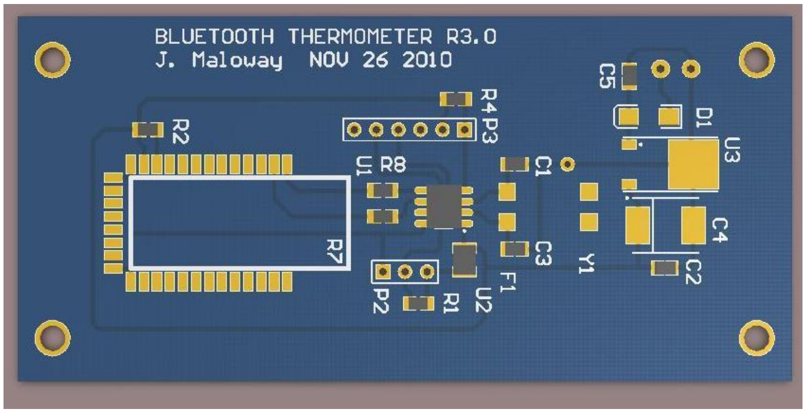


Figure 2.0 – 3D Rendering (Altium) of the PCB

Section 3 - Firmware

The firmware is based on the code used in the original thermometer design. Written in PICBasic and developed in the Proton IDE suite, the firmware makes use of the serial communication and 1-Wire® libraries built into Proton. Though the overall design does not allow for comprehensive error-checking or fault tolerance (loss of connection, power reset, etc), the firmware is able to detect if the 1-Wire® sensor is disconnected or if there is a wiring fault.

The current revision at the time of writing (R1 Build 23) drew heavily from two 1-Wire® communication examples. The Proton IDE user manual provided a number of helpful examples explaining the OREAD and OWRITE functions built into the compiler, as well as a sample temperature acquisition routine.

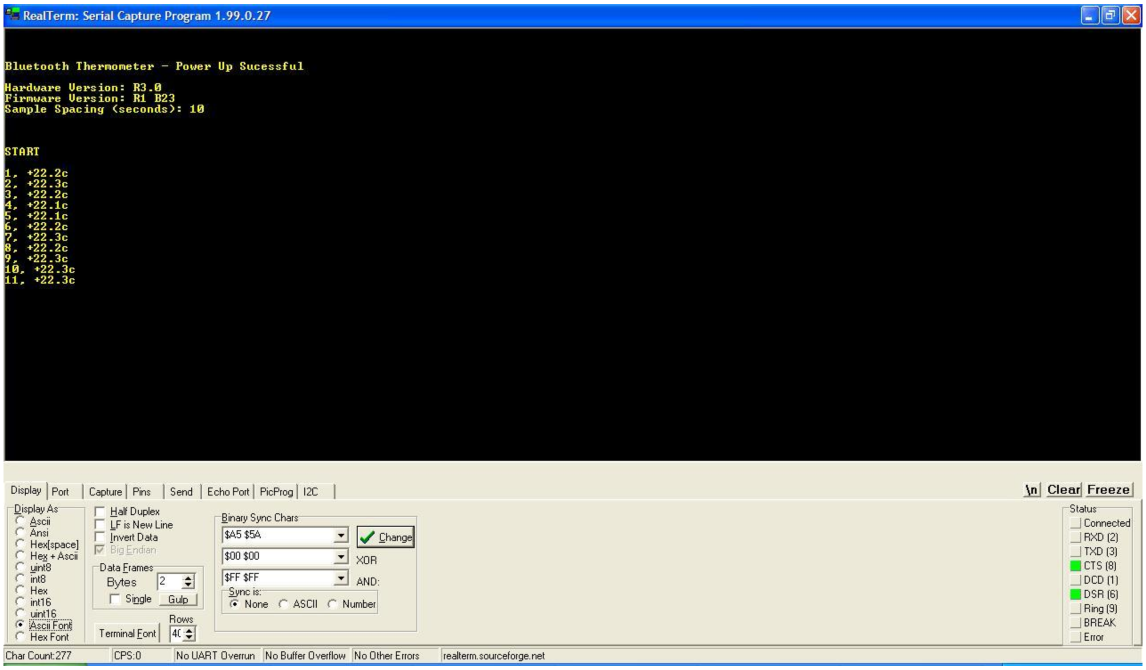
The above routines were later combined with aspects of the firmware on this website that allowed the firmware to correctly interpret and transmit negative temperature values, while at the same time simplifying the code structure.

I must also thank my instructor, David, for helping me decipher the cryptic example code I had started with, and for suggesting a workable temperature conversion scheme.

Section 4 - Software

All functional testing was performed using a PC running Microsoft Windows XP® with Toshiba® Bluetooth® hardware and drivers installed. Though the Bluetooth® protocol is meant to be cross-compatible across multiple platforms and hardware vendors, this is often not the case.

The open-source program RealTerm was used to view the incoming data stream from the embedded hardware during development. Any terminal program will work, however RealTerm was selected for this project because of its ability to “capture” the incoming data stream and save it to a text file. The data collected can then be imported into Excel as a CSV (comma-separated value) file and re-formatted as is most suitable for the task at hand.



```
Bluetooth Thermometer - Power Up Sucessful

Hardware Version: R3.0
Firmware Version: R1 B23
Sample Spacing (seconds): 10


START

1, +22.2c
2, +22.3c
3, +22.2c
4, +22.1c
5, +22.1c
6, +22.2c
7, +22.3c
8, +22.2c
9, +22.3c
10, +22.3c
11, +22.3c
```

Figure 4.0: Viewing Incoming Serial Data through RealTerm

Section 5 - Outstanding Issues

- Occasionally, an extra character appears at the start of some data transmissions.

- The sample frequency is fixed

- Though reverse-polarity protected, the power supply input could be fitted with a bridge rectifier to allow either AC or DC power supplies to be used without the need to consider polarity.

- The assembly is not RoHS compliant. The bill of materials should be revised, and any non-conforming parts should be substituted with RoHS-compliant equivalents.

- The footprint for the PTC protecting the 1-Wire VCC rail is incorrect and should be modified prior to the next PCB revision.