1-Wire APRS Weather Station

William Beals NØXGA
15014 East Idaho Place
Aurora, CO 80012
will@beals5.com

Russell Chadwick KBØTVJ
4371 North 63rd Street
Boulder, CO 80301
russ@wxqa.com

Abstract

This project and resulting TAPR kit implement the necessary hardware and software to sense and display data from various low-cost 1-Wire weather sensors. With the addition of a TNC and radio, it forms a complete, stand-alone APRS-compatible weather station. Weather information is also available on a 4-line by 20-character LCD. The project is designed to be easily extensible and upgradeable—even for applications that are not weather related.

Key Words

One-Wire, 1-Wire, APRS, Weather, Peet, Dallas Semiconductor, Motorola, LCD, temperature, wind direction, wind speed, humidity.

Introduction

After much research, the authors came to the conclusion that there wasn't quite the "perfect" weather station intended for stand-alone APRS use. Our definition of "perfect" had the following requirements. 1) Affordable 2) Reproducible 3) Expandable 4) Accurate 5) No computer required 6) Low power and single supply, and 7) APRS compatible. Multiple consumer-grade weather stations exist, but in order to get on the APRS airwaves required either a dedicated computer or special "logging software" to transmit a non-APRS format. We had no convenient "buy it" option available. We considered several reputable projects where sensors were made out of household parts, but they relied on surplus parts that failed our second criterion, reproducibility. About that time, Dallas Semiconductor introduced a line of sensors called 1-Wire devices that required only two wires (marketing folk don't count ground I guess). To promote these parts they introduced a wind/temperature sensor based on these chips. With a decent and affordable sensor in hand, a micro chosen, and some welcome TAPR support, the project was born.

Project History

The project started on a 6805 microcontroller. Based on personal preference, all of the source code was written from the ground up, starting with and referring to little else beyond the Dallas chip specifications. The only times I "cheated" and looked at other code was referencing some C

code for the CRC checkers. This clearly had drawbacks in that I had to learn several basic lessons the hard way, however by starting from scratch, I had significant flexibility in the overall code architecture and chose one best suited for this project.

During the early project evolution, several major changes took place. The micro was upgraded to a Motorola 6808, the external serial EEPROM functionality was incorporated into the 6808 internal flash, and we even changed revisions of the 6808, from a 20Kbyte part to a 32Kbyte part. At that point the printed circuit board was created, so only software changes were allowed.

Even after the hardware platform became fixed, several significant features were added and upgraded. Most notable were adding a software downloader, overhauling the code for purely interrupt-driven data gathering, and the adding of several weather sensors as well as a time reference. All these upgrades have been available free to anyone who purchased the weather station or constructed it from scratch. Starting fairly early, the full source code has also been available for download as well.

The Hardware

The 6808 requires very little in the way of external support components for normal operation. These include a 5V regulator, crystal oscillator (not really required), the one-wire interface, and the UART buffers. Additional circuitry is included for program debug, this includes an extra crystal oscillator, additional UART interface, and a slightly unusual reset circuit.

The need to put the debug circuitry on every "production" board was heavily debated during the board design phase. This is a very general purpose board and we expect projects other than the weather station to be possible with this hardware platform. To support this, it made sense to enable as many people as possible to be developers.

Weather Data

During the initial stages of the project, the official APRS data formats were in the final stages of getting defined by the APRS working group. The final version I used was version 1.01. The only area were the station is in a gray zone is in the definition of the station type. The station identifier field calls out a one-letter software type followed by a multi-character weather unit type. For the software type, the specification defined six categories of equipment based mostly on the operating system: Windows, DOS, Mac, and Linux. A unique letter identifies each of these categories. Unfortunately, the weather station fit none of these categories. I arbitrarily chose a seventh identifier letter "e" for Embedded or Experimental as it was an unused letter. The weather unit type characters do not need to follow any format, so I chose "1w" for one-wire. Hence the station identifier is "e1w".

Both positioned and positionless weather data formats are supported. If the user specifies their coordinates in a setup screen the positioned format is chosen.

Data transmitted by the weather station clearly depends on the sensors installed. Sensors currently supported are wind speed, wind direction, temperature, rain, and humidity. In addition to the basic weather data from those sensors, calculated data such as dewpoint is displayed but

not transmitted. For rain, the intervals reported on APRS are rain last hour andsince midnight. In addition, rain for month-to-date, and "since-user-reset" are displayed on the LCD. Wind speed is measured every 5 seconds. Average wind speed over 5 minutes is reported for each APRS packet along with the peak gust measured over the 5-second sampling time.

Daily stats are also displayed for the current as well as previous day. This includes high temperature, low temperature, peak gust, and rainfall.

This is only the current sampling of weather data gathered and reported. This project by its nature is easily expandable. The most noticeably absent sensor is barometric pressure. The authors are waiting for a suitable mass producible sensor to be available before adding the support for it. There is a decent chance such a sensor will be available before this article gets published.

Peet Mode

In addition to APRS-formatted data, a mode exists to send data out of the unit using Peet format. There was a specific user request to support some existing equipment that understood only Peet formatted data. When in this mode, only raw (unaveraged) data is sent every five seconds.

Reporting accurate weather data

For a project such as this, being able to get, process and present the data is only half the battle. The other half of the battle is to do so as accurately as possible. Russ brought his expertise in this area, providing the most accurate algorithms possible and practical for this project. Weather data is inherently very "noisy" data. To sample this raw data every five minutes and broadcast that one sample does not give a good representation of the current weather conditions. Additionally, simple averaging is not practical, especially for information such as wind direction.

Download Monitor

Early in the project the main processor was changed from a UV-erasable EPROM-based 6805 micro to a Flash EEPROM-based 6808 micro. Once the main micro became flash-based, that brought the possibility of allowing users to upgrade their code without having to remove the micro from the main board. It would further allow this upgrade without needing the full debugger toolset. This quickly became a critical feature of the design and has proven to be a great asset during the projects lifetime, allowing users to easily upgrade to newer versions of the software without requiring anything more than a PC and a simple DOS-based program.

The Flash memory inside the micro is divided into three regions. The first is the boot area. This is never altered during normal operations. The second is the user-data area. This area stores user data and preferences. It is preserved during code updates. The final area is the application area. This is the area that is erased and update during a code update.

The boot area resides in the boot vector space of the flash and assumes control after reset. It occupies 2K of the flash space. After reset, this code checks to see if the up and down buttons are being pressed simultaneously. If not, the boot controller passes control to the application area. If both buttons are pressed, the boot loader initializes the display and presents a message

saying it is ready for a download. It then sends out a query character, waiting for the PC to start a download. As soon as this character is recognized and acknowledged, the boot loader erases the application area and requests new application code from the PC until the new application is loaded.

The user data area is where sensor IDs are stored as well as any other permanent information such as position string, UTC offset, etc. It is also approximately 2K in size. A significant advantage of the user-download feature is that this data is NOT erased when new application is loaded, hence preserving any previous settings. The development tools can only erase the entire flash including the user settings. A default of 0xff is required for any unused data, allowing a known value to be present for any new variables that become defined with a new software download.

The application area is the remainder of Flash, approximately 28K in size. This area is not divided any further, if there is a code update, this entire area is erased and reprogrammed. Since inception, there have been nearly 30 releases to both introduce significant new features as well as fix numerous coding bugs.

An Upgradeable Project

As evidence of the usefulness of the upgrades, the initial release of the weather station only included basic monitoring of wind direction, speed, and temperature. During the nearly thirty releases, the following features were added either as pure software enhancements, or additional sensor support for which the sensor just needed to be added to the 1-wire network.

Computer (non-averaged) data mode
Updated calibration constants for wind speed
Added error counters screen
Made all data gathering routines interrupt-driven
Wind direction optionally, N, NW, vs. just degrees
Added today/yesterday stats screen
Added Peet mode support
Added rain gauge support
Added support for better DS18S20 temp sensor
Added support for TNCs with LTP command
Added positioned weather format
Added metric display screens
Added support for DS1994 real-time clock
Added full baud rate support in all modes
Added bus voltage monitoring (great for debug!)
Added support for humidity sensor
Added support for new AAG wind/temp sensor
Added ability to "remove" a sensor
Added external radio power control (external HW required)
Added battery voltage monitoring (external HW required)
Added dewpoint option vs. humidity only

This list only includes new features, not the numerous bugs that were discovered while adding these features. Also, in many cases the features were requests from users who purchased the weather sensor and placed requests for additional features after using the unit for a while.

Single-supply design

(A small digression here) A significant drawback with PC-based weather stations in this authors opinion is power consumption. Assuming an average PC draws approximately 200 watts (even more if the monitor is on), power consumption alone adds up to a tangible impact to the electricity bill. If you are curious, multiply 146.4 (the number of kilowatt hours in a month of 24/7 usage at 200 watts) times your local utilities cost per kilowatt hour. For our local utility, this adds up to about 11 dollars a month! Naturally, your numbers may vary—especially if your 24/7 computer is a laptop or some other watt-miserly machine. This ongoing cost was a major reason for wanting to expend the effort for an embedded weather station instead of just using one of the many excellent PC-based solutions that were available at the time.

Fully configured with a standard setup and some margin, the weather station draws a maximum of four watts (12V at 300ma or 22 cents per month). Almost half this power is for the LCD backlight, so if you leave it off, power is even further reduced. This is sufficient for any domestic use. The choice to use a 12V supply even though only 5V is needed is that this is already a very common power voltage in almost any ham shack. Also, for a remote situation 12V is also a very common DC supply.

During the design we debated over the type of regulator to use to get the 5V supply from the 12 rail. The quick and dirty linear regulator was certainly the easiest and cheapest. For a remote location that might be solar powered, a switching regulator would clearly have made more sense. We ended up choosing the cheaper solution that wasted some power in recognition that most of the station would be AC-powered. For anyone running from solar power, they could make the choice to use the existing linear or install a switching regulator in the patch area.

Future Work

The only major sensor still missing is barometric pressure. This sensor has proven to be somewhat problematic to make cheaply and accurate at the same time. The author also really wanted to support a sensor that was produced by either Dallas Semi or AAG, but neither company has offered one as of this writing. A design from the amateur (not just radio) community is gathering momentum and support for this sensor may exist about the time of this papers publication.

Other future work includes using more of the DS1994 features besides the real-time clock. The part also has a 4Kbit non-volatile memory. It would be nice to log daily statistics to that memory in a format that would be readable by any PC running Dallas Semiconductors TMEX file system. This would allow logging up to approximately one month of data and then be able to transfer these logs to a PC in a very convenient manner.

There is also nothing required other than programming time to be able to support parsing of incoming APRS data and displaying it on the screen. The numerous formats, exceptions, and

variations of incoming data would be a formidable chore and so far continues to be avoided by the author.

Finally, one of the original intents of the hardware design was to be a general-purpose board for projects besides weather stations. So far no other projects have materialized. Hopefully some day this will happen.

Conclusion

It is a rare opportunity in the hobbyist community to not only want to work on a project but also find a group of people and an association that is willing to help you in that process. This weather station is such a project. It would be dangerous to try to mention the numerous folk that helped with this project either directly in material ways or the hundreds of emails on the weather reflector. TAPR clearly deserves thanks as it provided the forum for all of the helpful people to work through.

References

Ian Wade, Editor. (2000) APRS Protocol Reference. The APRS Working Group

Dallas Semiconductor. DS18S20 High Precision 1-Wire Digital Thermometer
Dallas Semiconductor. DS1994 4Kbit Plus Time Memory iButton
Dallas Semiconductor. DS2401 Silicon Serial Number
Dallas Semiconductor. DS2407 Dual Addressable Switch Plus 1K–Bit Memory
Dallas Semiconductor. DS2422/DS2423 1K/4K–Bit 1–Wire RAM with Counters
Dallas Semiconductor. DS2438 Smart Battery Monitor
Dallas Semiconductor. DS2450 1-WireTM Quad A/D Converter