

**Bob Simmons, WB6EYV**1268 Veronica Springs Rd, Santa Barbara, CA 93105; [pelican2@silicon.com](mailto:pelican2@silicon.com)

# APRS Unveiled

*All the sneaky bit-level details of APRS messages...with an example packet.*

Anyone attempting to create APRS equipment “from scratch” has immediately confronted the lack of a complete, detailed summary of all the APRS message requirements. Finding these details requires considerable effort, more than a little “luck,” and a vocabulary that an APRS “novice” simply won’t have. The lack of a simple, concise (and complete) summary has probably thwarted many attempts to create APRS technology.

This article (hopefully) addresses that problem. It provides a detailed description of a typical APRS message string, byte by byte, with a complete example provided. Basically, there is enough “message protocol” information to enable the creation of a circuit that can plug into the MIC jack of a non-APRS radio, and generate APRS packets that will successfully propagate across an APRS network.

The guidelines described here have been tested and proven. I used them to develop a 2 m APRS beacon transmitter design that has

been in service for 2 years, and which has been used by several people. You can find more information about the beacon itself on my website: [www.silcom.com/~pelican2/PicoDopp/XDOPP.htm#MBCN](http://www.silcom.com/~pelican2/PicoDopp/XDOPP.htm#MBCN).

## The Bell 202 Modulation Method

APRS data is transmitted at a 1200 baud data rate, typically on the (US) national APRS channel of 144.390 MHz. The data is frequency-modulated onto the RF carrier with two audio tones (1200 / 2200 Hz) that comply with a modified version of the Bell 202 modem standard. At the moment of data bit transition, a logic “zero” data bit is signified by “flipping” between tones, (for example, 1200 to 2200, or vice versa) whereas a logic “one” data bit is signified by no “flip” (steady frequency, either 1200 or 2200 Hz)

The Bell 202 standard specifies that the tone “flip” must be “phase contiguous,” which basically means that the transition

between tones must be as smooth as possible. The phase angle of the audio waveform (at the instant of tone switching) must be preserved and used as the “starting” phase angle for the new tone waveform. This minimizes switching transients and reduces the required bandwidth of the signal, resulting in improved signal to noise ratio (SNR). An example modulation waveform is shown in Figure 1.

## Octets Versus Bytes

APRS message bytes are transmitted with no START or STOP bits, (each byte is called an octet) which is different from regular RS232 bytes. Most of the message data is encoded using ordinary ASCII characters, but there are some exceptions to this rule, described later. In all cases, octet (byte) data bits are transmitted least significant bit (LSB) first. (The bit order is B0 to B7.)

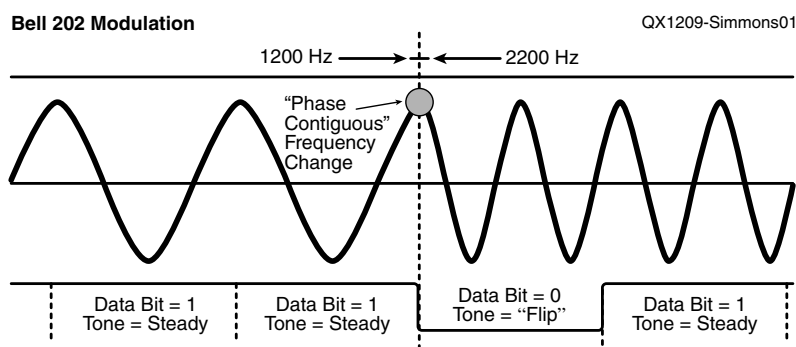


Figure 1 — This graph illustrates the Bell 202 Modem modulation specification for the transition between tone frequencies to represent a 0 data bit.

## APRS Position Reports: The Message Format

The APRS “example” message provided in this article complies with the format identified on page 33 of the APRS specification document, version 1.0.1, dated 29 August 2000.<sup>1</sup> In that document, (page 33) this message format is identified as: “Lat/Long Position Report Format with Data Extension (no Timestamp)” Byte by byte, the transmitted message has this format:

**FDDDDDDdSSSSSSsVVVVVVvVVVVVVvCPsLLLLLLHsLLLLLLHsCCCsSSSCccFF**

With each byte defined as shown below:

(begin transmission)

(NOTE: Tone modulation begins IMMEDIATELY when transmission begins... no “dead carrier” time is provided.)

**Starting Flag Bytes** (1 byte minimum, usually several identical bytes)

**F** Flag byte, always = 0x7E

**Preamble Bytes**

**DDDDDDd** Destination addr (7 bytes)(“APDF00”+ SSID = “0” in this example)

**SSSSSSs**

Source addr (7 bytes = 6 call sign bytes + 1 SSID byte)

VIA addr (0 - 8 addr = 0 - 56 bytes) (7 bytes each = 6 call sign + 1 SSID byte)

(NOTE: this example uses only 2 VIAs, shown below)

**VVVVVVv** VIA 1 address + SSID (optional, 6 + 1 bytes)

**VVVVVVv** VIA 2 address + SSID (optional, 6 + 1 bytes)

(end of preamble bytes)

**Control and Protocol Bytes**

**C** Control field (1 byte, always = 0x03)

**P** Protocol field (1 byte, always = 0xF0)

(end of control and protocol bytes)

**Information Field** (APRS Position report, no timestamp, no messaging)

**s** Symbol (1 byte)(APRS message type identifier)

(! = exclamation mark = no APRS messaging, no time stamp)

**LLLLLLH** Latitude (8 bytes, XXxx.xxH)

(XX = degrees latitude, 00 to 89)

(xx.xx = minutes + dp + decimal minutes latitude)

(H = hemisphere, N or S)

**s** Symbol (1 byte)(= primary or alternate APRS symbol table)

(This identifies the type of APRS map icon to be displayed)

**LLLLLLH** Longitude (9 bytes, XXXxx.xxH)

(XXX = degrees longitude, 000 to 179)

(xx.xx minutes + dp + decimal minutes longitude)

(H = hemisphere, E or W)

**s** Symbol (1 byte)(= map symbol displayed on APRS screens)

(data extension begins here: COURSE and SPEED)

**CCC** Course (3 bytes, xxx = 001-359, true degrees, 000 = stationary)

**s** Symbol (1 byte)(delimiter = “/”)

**SSS** Speed (3 bytes, xxx = 000-999 knots)

(end of data extension)

**C** Comment (0 - 36 bytes, 1 byte shown here)

(end of information field)

**Frame Checksum Bytes**

**cc** FCS field (2 bytes, CRC checksum, sent low byte / high byte)

(end of frame checksum bytes)

**Ending Flag Bytes**

**FF** Flag (2 bytes minimum)

(end of message.... end transmission)

NOTE: As a courtesy to the receiving decoder (to make its job easier) it is not unusual to send several bytes of 0x00 data before sending the first FLAG byte. The pattern of several successive “0” bits causes the Bell 202 tones to constantly flip between tones, which simplifies the detection of the boundary between successive data bits, at the receiving decoder. For similar reasons, it is not unusual to send several FLAG bytes at the beginning (and end) of an APRS message, even though the APRS spec states only one FLAG byte is required.

Bear in mind that the receiver “at the other end” probably is an ordinary voice radio, with ordinary squelch circuits that will require 50 to 100 milliseconds (or more) of time to detect the presence of a signal on the channel, before any speaker audio is generated. At 1200 baud, 100 milliseconds of time equates to 15 transmitted bytes of message data... so the typical “courtesy” practice of transmitting several starting flag bytes is (probably) more important than the APRS specification indicates.

<sup>1</sup>Ian Wade, G3NRW, Editor, *Automatic Position Reporting System APRS Protocol Reference*, TAPR, 2000, p 33: [www.aprs.org/doc/APRS101.PDF](http://www.aprs.org/doc/APRS101.PDF). [Yes, that should be “Automatic Packet Reporting System,” but the title of the document was not changed. — Ed.]

### Flag Bytes and Bit Stuffing

The lack of START and STOP bits in APRS messages means that some other method must be provided for an APRS decoder to “synchronize” itself with the bitstream of arriving messages.

The boundary between successive data bits can be identified by observing the 1200 / 2200 Hz tones, but the boundary between successive BYTES (end of one byte and start of next byte) must be identified by some other means. This is accomplished with special octets called FLAG bytes, consisting of a bit pattern of “01111110.” (hex 0x7e = ASCII “tilde” character: ~)

This pattern (six consecutive “one” data bits) is reserved EXCLUSIVELY for FLAG bytes in the APRS specification. Therefore, any “accidental” occurrence of the same pattern (in the transmitted data) must be detected and prevented, but the data itself must (somehow) be preserved and recovered at the receiver. To accomplish this, a method is employed called “bit stuffing.”

With “bit stuffing,” each transmitted message is examined (bit by bit) as it is transmitted, to detect any (accidental) occurrence of five consecutive “1” bits. If such an event is detected, the sixth data bit (which might be either “1” or “0”) is delayed, and a “0” bit is sent, (“stuffed” into the data stream) followed immediately by the (delayed) sixth data bit.

At the receiving end of the message, detection of 5 consecutive “1” data bits will alert the software that the following (6<sup>th</sup>) bit will determine if the data is a FLAG byte, or simply part of a regular message byte...if the 6<sup>th</sup> bit is a “1,” the byte is judged to be a FLAG byte... otherwise, the 6<sup>th</sup> bit (which is a “0”) will be ignored and discarded from the bitstream.

### The Preamble: General Description

The message preamble includes the source, destination and VIA address bytes, and their associated SSID bytes. According to the APRS specification, the number of VIAs (which are user specified) can vary from zero to eight, but in the example provided in this article, the number is limited to two VIAs.

The SOURCE address is actually the FCC call sign of the transmitter operator, (6 characters, always spelled with CAPITOL letters) with an SSID byte appended to the end (7 bytes total...more info about SSID bytes later). If the call sign is less than 6 characters long, it is left-justified and padded with trailing ASCII “blank” characters, (0x20) followed by the SSID byte.

The DESTINATION address is not actually used in APRS... it is a legacy of packet communications, but APRS is a specialized subset of packet that does not employ this data field. Instead, it is filled with a fixed string of characters that identifies the type of software used to generate the APRS message. The string provided in this article’s example was assigned to the author by Bob Bruninga, (developer of APRS) and consists of the text string “APDF00,” with an SSID character of zero. This “assignment” is a matter of social courtesy, so that any problems in the resulting APRS messages can be traced back to the software author, and corrected. Anyone creating their own software should therefore contact Bob Bruninga for a similar assignment.

The VIA call signs (and their SSID characters) are optional (two are provided in this article’s example). These are supplemental identifiers that provide information about the preferred signal path or direction for the message to take, and/or the preferred recipients for the message.

Typically these two VIAs are “WIDE1” (with SSID = 1) and “WIDE2.” (with SSID = 2) These particular VIAs are actually requests for automatic “message relays” by any digipeater station that hears the messages.

### The Preamble: SSID Bytes

SSID stands for “Secondary Station ID” (secondary station identification) SSID is encoded as a single byte that can express a

number ranging from 0 to 15. Various (somewhat complex) “rules” for selection of SSID numbers are included in the APRS specification, but their actual values do not seem to be critical to message detection / propagation through the APRS network. In this article’s example, the SSID for the DESTINATION station (= APDF00) is zero. For a WIDE1 VIA, this SSID should be one, and for a WIDE2 VIA, this SSID should be 2.

It is important to mention that the SSID values shown in ordinary computer displays (and in published articles) always include a hyphen character, so that “W6XYZ-0” indicates station W6XYZ with an SSID of zero, but in the actual transmitted message, no hyphen character is transmitted.

Furthermore, the SSID character is *not* an ASCII character; the SSID number is a 4-bit BINARY number, encoded into an 8-bit byte. The remaining bits are employed for other purposes and a description of the bits is provided below:

SSID bit 0 = extension bit (= 1 for last PREAMBLE field, = 0 otherwise)

SSID bits 1 to 4 = secondary station identification number (0 to 15, = “SSID” number)

SSID bits 5 and 6 = reserved, always = 1

SSID bit 7 = “control info,” (C-bit) always = 0

### The Preamble: Extension Bits and Byte Rotation

The APRS specification allows zero to eight VIA stations to be identified in a message, so some method must be provided to indicate how many VIAs are actually contained in any specific message. (This signals the end of the preamble block of data.) This is accomplished with the least significant bit (LSB) in ALL the preamble bytes. If the LSB (= bit 0) in a preamble SSID byte equals zero, then more preamble bytes remain in the message. If this bit equals one, no more preamble bytes remain. This bit is called the “extension bit”

This bit is often used in ordinary ASCII codes, and therefore it is not normally available for this purpose. To deal with this conflict, the ASCII codes used in the preamble (but *not* in the main message body) are limited to the 7-bit ASCII codes only (high order bit = always zero). This includes all “printable” ASCII characters, which are a subset of the entire ASCII set.

Furthermore, each ASCII byte (in the preamble only) is “rotated left” by one bit position, which is (arithmetically) equivalent to multiplying the character’s binary value by two. This can be done without loss of information because the top bit of all 7-bit ASCII codes always equals zero. As a result of this “rotation,” the LSB in each preamble byte is “liberated” for use as an APRS “extension bit.”

For example, ASCII character “3” would normally be expressed as hex number 0x33, but in an APRS preamble, (due to the byte rotation) this would be transmitted as hex number 0x66, (if the extension bit = 0) or as hex number 0x67 (if the extension bit = 1). A table of ASCII characters with their regular and “rotated” values is included in the APRS specification, in Appendix 3, Part 2.

ASCII “3” character = 0x33 = 00110011

Rotated character = 0x66 = 01100110 (if extension bit = 0)

= 0x67 = 01100111 (if extension bit = 1)

This “byte rotation” method is *only* applied to the preamble bytes — *not* to the entire contents of the APRS transmission. The first “rotated” byte is the first byte of the destination address, and the last “rotated” byte is the last byte of the last VIA address (SSID byte of the last VIA). If no VIAs are used, then the last “rotated” byte would be the SSID byte of the source address.

Summarizing, the extension bit in all PREAMBLE characters must be zero, EXCEPT for the VERY LAST character in the PREAMBLE, in which the extension bit must equal one.

### Control and Protocol Characters

The control and protocol characters consist of two octets trans-

**Table 1**  
**Sample APRS Example Message**

NAME	VALUE	HEX DATA TRANSMITTED									
(begin transmission)											
NULLS	(5X <nul>)	0x00	0x00	0x00	0x00	0x00					
FLAGS	(5X <tilde>)	0x7e	0x7e	0x7e	0x7e	0x7e					
(begin CRC calculation here)											
(begin bit stuffing here)											
(NOTE: The following bytes are left-rotated one bit position to provide bit 0 = extension bit)											
DESTINATION	APDF00	0x82	0xa0	0x88	0x8c	0x60	0x60				
DEST SSID	<SSID = 0>	0x60									
SOURCE	W6XYZ<sp>	0xae	0x6c	0xb0	0xb2	0xb4	0x40				
SRC SSID	<SSID=15>	0x7e									
VIA1		WIDE1<sp>		0xae	0x92	0x88	0x8a	0x62	0x40		
VIA1 SSID	<SSID=1>	0x62									
VIA2		WIDE2<sp>		0xae	0x92	0x88	0x8a	0x64	0x40		
VIA2 SSID	<SSID=2>	0x65									
(end left-rotation)											
CTRL CHAR	<control>	0x03									
PROTO CHAR	<protocol>	0xf0									
MSG TYPE	<msg type>	0x21									
LATITUDE	3426.22N	0x33	0x34	0x32	0x36	0x2e	0x32	0x32	0x4e		
SYMB TABLE	<primary>	0x2f									
LONGITUDE	11943.57W	0x31	0x31	0x39	0x34	0x33	0x2e	0x35	0x37	0x57	
SYMB CODE	<car>	0x3e									
COURSE	264	0x32	0x36	0x34							
DELIMITER	/	0x2f									
SPEED	000	0x30	0x30	0x30							
COMMENT	COMMENT	0x43	0x4f	0x4d	0x4d	0x35	0x4e	0x54			
(end CRC calculation)											
CRC LSB	<CRC lo byte>	0xf9									
CRC MSB	<CRC hi byte>	0x3c									
(end bit stuffing)											
FLAGS	(5X <tilde>)	0x7e	0x7e	0x7e	0x7e	0x7e					
(end of transmission)											
(total = 81 bytes = 540 ms at 1200 baud)											

mitted immediately after the preamble. The control octet is transmitted first, and always consists of 0x0f. The protocol octet is transmitted next, always consisting of 0xf0. (No explanation is offered here for their purpose.)

#### Message Body (Information Field)

The message body (called the “information field” in the APRS spec) has various forms, depending on the type of APRS message being transmitted. The format of the data contained in this field is identified by the very first character, (“symbol”) and different APRS messages use different characters for this field. (Refer to the APRS specification.)

#### APRS Map Symbol

The APRS map symbol is identified with two ASCII bytes located in the message body. One is located immediately after

the latitude data field, the other immediately after the longitude data field. These two bytes are defined in the APRS specification, in Appendix 2.

The first character identifies one of two “symbol tables” in the appendix, (PRIMARY or ALTERNATE) each containing 93 “symbols” that will be shown on a map display when the message is received. The second character identifies one of the 93 symbols in the associated table.

#### Data Extensions

Data extensions are optional 7-byte fields that (if employed) express additional information, as described in Chapter 7 of the APRS specification. In this example, a data extension is employed to express the COURSE and SPEED of the reporting station.

#### Comment Field

The COMMENT field is optional. The maximum allowed length of the COMMENT field varies depending on the type of APRS message being sent. (See the details in the APRS specification for a particular message type.)

#### Frame Checksum

The frame checksum is calculated using a CRC calculation method. CRC refers to “Cyclic Redundancy Check,” which consists of a special two byte “checksum” that allows the integrity of the message data to be tested, after it is received. The CRC checksum (= frame checksum) is generated when each message is transmitted, and evaluated at the destination, when the message is received.

The CRC checksum generation is performed by examining each byte in the transmitted message, using a special “formula”



that is applied to each bit in the message. The result of this special "formula" is a two byte number that expresses the CRC (frame) checksum.

Bits that are added to the bitstream as a result of "bit stuffing" are *not* included in the calculation of the CRC checksum. The two-byte checksum itself is also excluded from the calculation.

CRC checksum calculation begins with the first byte in the PREAMBLE block, (immediately after the last starting FLAG) and ends with the last byte in the COMMENT block. The two CRC bytes are then transmitted LSB / MSB (low byte first, then high byte).

Rather than re-explaining it here in the author's own words, I defer to the source where I learned of it myself — many thanks to Scott Miller, N1VG, for posting this simple and concise explanation of the CRC checksum calculation method on his website:

### Frame Check Sequence

One detail of the AX.25 format that deserves attention is the Frame Check Sequence (FCS) checksum. This is a two-byte checksum added to the end of every frame. It's generated using the CRC-CCITT polynomial, and is sent low-byte first.

The CRC-CCITT algorithm has plenty of published code examples, but the one I needed, and had trouble finding, was the algorithm for calculating the FCS one bit at a

time, rather than a byte at a time. That algorithm is as follows:

Start with the 16-bit FCS set to 0xffff. For each data bit sent, shift the FCS value right one bit. If the bit that was shifted off (formerly bit 1) was not equal to the bit being sent, exclusive-OR the FCS value with 0x8408. After the last data bit, take the ones complement (inverse) of the FCS value and send it low-byte first.

NOTE: this text (and more useful information) can be found at Scott Miller's website, at: <http://n1vg.net/packet/index.php>

Those who choose to double-check this information against the AX.25 protocol specification, AX.25.2.2, dated July 1998, will find in section 3.8 that the order of bit transmission for the FCS data bits is opposite to that for the rest of the packet data, that is, the FCS bits (in the spec) should be transmitted most significant bit first (bit order B15 to B0 for the two FCS bytes) whereas the bit order for all other packet bytes should be sent least significant bit first (bit order = B0 to B7).

This contradicts the author's experience, in which successful on-air tests (and iGate postings) of the beacon transmitter's APRS packets used FCS data transmitted LSB first, just like the rest of the APRS packet data. There also is no mention of this "reversed" bit order in Scott Miller's comments on the topic, so it seems that the AX.25 spec is "suspect," on this point.

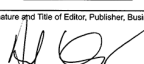
### A Message Example

The message example given in Table 1 expresses a complete APRS message generated in compliance with the guidelines described in this article. For clarity, bits added as a result of "bit stuffing" are not shown in this data. Because a few of the bytes consist of unprintable ASCII characters, the data here is expressed in hexadecimal notation.

*Bob Simmons, WB6EYV, was first licensed as a novice in 1964 at age 13, and remained licensed (more or less) constantly ever since. He also earned a commercial FCC license in 1967. He served Naval Reserve duty as a radar technician (ETR2) with about 6 months of total sea time. He spent several years of civilian work in nautical and marine electronics in Los Angeles harbor, as well as doing some land mobile radio work, followed by 5 years in flight line avionics, working on business jets. He moved to Santa Barbara, CA in 1992 and worked on vacuum deposition systems for 5 years, and held assorted odd engineering jobs at other times.*

*Presently, Bob is self employed and runs a website making and selling radio direction finding equipment and modules, with a majority of his "new" work spent creating embedded software / hardware and developing technologies to enable Internet-linked remote DF stations. His primary interest is developing and applying new technologies to old problems, and pushing the DF "art" forward.*

QEX

UNITED STATES POSTAL SERVICE® (All Periodicals Publications Except Requester Publications)			
1. Publication Title <b>QEX</b>		2. Publication Number 0 8 8 6 8 0 9 3	
3. Filing Date September 26, 2012		4. Issue Frequency Bi-monthly: Jan/Mar/May/July/Sept/Nov	
5. Number of Issues Published Annually 6		6. Annual Subscription Price \$24.00	
7. Complete Mailing Address of Known Office of Publication (Not printer) (Street, city, county, state, and ZIP+4®) 225 Main Street, Newington, Hartford County, CT 06111-1494			
8. Complete Mailing Address of Headquarters or General Business Office of Publisher (Not printer) 225 Main Street, Newington, CT 06111-1494			
9. Full Names and Complete Mailing Addresses of Publisher, Editor, and Managing Editor (Do not leave blank) Publisher (Name and complete mailing address) Harold Kramer, 225 Main Street, Newington, CT 06111-1494 Editor (Name and complete mailing address) Larry Wolfgang, 225 Main Street, Newington, CT 06111-1494 Managing Editor (Name and complete mailing address) Larry Wolfgang, 225 Main Street, Newington, CT 06111-1494			
10. Owner (Do not leave blank. If the publication is owned by a corporation, give the name and address of the corporation immediately followed by the names and addresses of all stockholders owning or holding 1 percent or more of the total amount of stock. If not owned by a corporation, give the names and addresses of the individual owners. If owned by a partnership or other unincorporated firm, give its name and address as well as those of each individual owner. If the publication is published by a nonprofit organization, give its name and address.) Full Name American Radio Relay League, Inc. Complete Mailing Address 225 Main St. Newington, CT 06111-1494			
11. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1 Percent or More of Total Amount of Bonds, Mortgages, or Other Securities. If none, check box <input checked="" type="checkbox"/> None Full Name Complete Mailing Address			
12. Tax Status (For completion by nonprofit organizations authorized to mail at nonprofit rates) (Check one) The purpose, function, and nonprofit status of this organization and the exempt status for federal income tax purposes: <input checked="" type="checkbox"/> Has Not Changed During Preceding 12 Months <input type="checkbox"/> Has Changed During Preceding 12 Months (Publisher must submit explanation of change with this statement)			
PS Form 3526, September 2007 (Page 1 of 3) (Instructions Page 3) PSN 7530-01-000-931 PRIVACY NOTICE: See our privacy policy on www.usps.com			
13. Publication Title <b>QEX</b>		14. Issue Date for Circulation Data Below Sept/Oct11-July/Aug12 Sept/Oct 12	
15. Extent and Nature of Circulation		Average No. Copies Each Issue During Preceding 12 Months	
a. Total Number of Copies (Net press run)		7767	
b. Paid Circulation (By Mail and Outside the Mail)		No. Copies of Single Issue Published Nearest to Filing Date	
(1) Mailed Outside-County Paid Subscriptions Stated on PS Form 3541 (Include paid distribution above nominal rate, advertiser's proof copies, and exchange copies)		5038	
(2) Mailed In-County Paid Subscriptions Stated on PS Form 3541 (Include paid distribution above nominal rate, advertiser's proof copies, and exchange copies)		0	
(3) Paid Distribution Outside the Mails Including Sales Through Dealers and Carriers, Street Vendors, Counter Sales, and Other Paid Distribution Outside USPS®		1458	
(4) Paid Distribution by Other Classes of Mail Through the USPS (e.g. First-Class Mail®)		535	
c. Total Paid Distribution (Sum of 15b (1), (2), (3), and (4))		7031	
d. Free or Nominal Rate Distribution (By Mail and Outside the Mail)		61	
(1) Free or Nominal Rate Outside-County Copies Included on PS Form 3541		62	
(2) Free or Nominal Rate In-County Copies Included on PS Form 3541		0	
(3) Free or Nominal Rate Copies Mailed at Other Classes Through the USPS (e.g. First-Class Mail)		72	
(4) Free or Nominal Rate Distribution Outside the Mail (Carriers or other means)		55	
e. Total Free or Nominal Rate Distribution (Sum of 15d (1), (2), (3), and (4))		188	
f. Total Distribution (Sum of 15c and 15e)		7219	
g. Copies not Distributed (See Instructions to Publishers #4 (page #3))		548	
h. Total (Sum of 15f and g)		7767	
i. Percent Paid (15c divided by 15f times 100)		97.40%	
16. Publication of Statement of Ownership <input type="checkbox"/> If the publication is a general publication, publication of this statement is required. Will be printed in the <u>Nov/Dec 12</u> issue of this publication. <input type="checkbox"/> Publication not required.		17. Signature and Title of Editor, Publisher, Business Manager, or Owner  Date September 26, 2012	
I certify that all information furnished on this form is true and complete. I understand that anyone who furnishes false or misleading information on this form or who omits material or information requested on the form may be subject to criminal sanctions (including fines and imprisonment) and/or civil sanctions (including civil penalties).			
PS Form 3526, September 2007 (Page 2 of 3)			

Pat Cain, K0PC / k0pc@arrl.net

# APRS with a Smartphone

In an earlier article I presented a method for tracking QSO party mobile operators — or “rovers” — using online maps (“Generating Your Own QSO Party Mobile Maps” by Pat Cain, K0PC, Sep/Oct 2010 *NCJ*). This tracking relies on position reporting via the VHF automatic packet reporting system (APRS) network with connections to the Internet. Typically this has been done using the combination of a GPS, a terminal node controller, and a 2 meter FM transceiver. While this method works very well in some areas, it relies heavily on fixed stations to retransmit the position reports in order for them to reach a station that can connect to the Internet.

During my QSO party travels in Minnesota, Iowa and Wisconsin I ran across large rural areas that had no VHF APRS coverage at all. As a result, my tracking map had gaps of an hour or more between position reports in many areas. These gaps make the map much less useful for the operator who is attempting to track a rover from county to county.

Modern technology comes to the rescue. For the past couple of years I've owned a smartphone running the Android™ operating system. The phone contains all the hardware needed to report your position directly to the Internet as well as a GPS and a data connection. The only missing piece is an application to provide the smarts.

I searched the Android Market (now known as Google Play) for an application that can make APRS features available on my phone and settled on an application called *APRSdroid*, written by Georg Lukas, DO1GL. *APRSdroid* is a full-featured APRS application that uses the APRS-IS network for position reporting (more on this below). *APRSdroid* provides the ability not only to report your position automatically but to track other nearby APRS-equipped stations. You can also exchange text messages with other APRS stations. I use *APRSdroid* simply to broadcast my position.

## Program Setup

*APRSdroid* is simple to set up. Start up *APRSdroid* and select the MENU button on your phone, then choose the “More” option. Select “Preferences” to access the program's setup screen. You will need to enter your call sign and the service set identifier (SSID) you plan to use. The SSID is the number after your call sign. For

example, I use an SSID of 9, so my APRS identification is K0PC-9. Most of the other preferences can be left at their default values to begin with.

## APRS-IS Server

*APRSdroid* uses the APRS-IS server system to send and receive location data. The APRS-IS server is really a coordinated group of servers that share data. In order to connect to an APRS-IS server, you must first have a passcode issued by the program author. The first thing to do when you start up *APRSdroid* is to request a passcode for your call sign. If you have an APRS-IS passcode from another application, you can use it in *APRSdroid*. The Preferences screen contains an entry called “Request Passcode,” which will bring up a Web form for submitting your information. The author may ask for verification that you hold an Amateur Radio license. It seems that using an e-mail of the form <call sign>@arrl.net makes the process easier and can speed the response. Once you receive your passcode, enter it in the Preferences screen.

The Preferences screen also has an

entry for “Connection Preferences,” where you can enter a server name. I use **rotate.aprs.net**, a round-robin DNS for all core APRS-IS servers. It will automatically direct you to one of the available servers.

## Reporting Your Position

*APRSdroid* has several methods of reporting your position. You can generate a single-shot position report and leave it at that, but that's not useful when you're in motion. One automatic method involves sending a position report at a fixed interval that you define. This works but is not always terribly efficient. The best method is called SmartBeaconing™. This method is dynamic, based on your speed and direction. As you go faster the reporting interval is reduced, and if you stop moving the interval shifts to its maximum setting. In addition, SmartBeaconing also sends a position report when your heading changes. This makes sure you are tracked around corners.

## Data Usage

This application uses your phone's data



Figure 1 — The *APRSdroid* “Hub” screen identifies stations and locations.

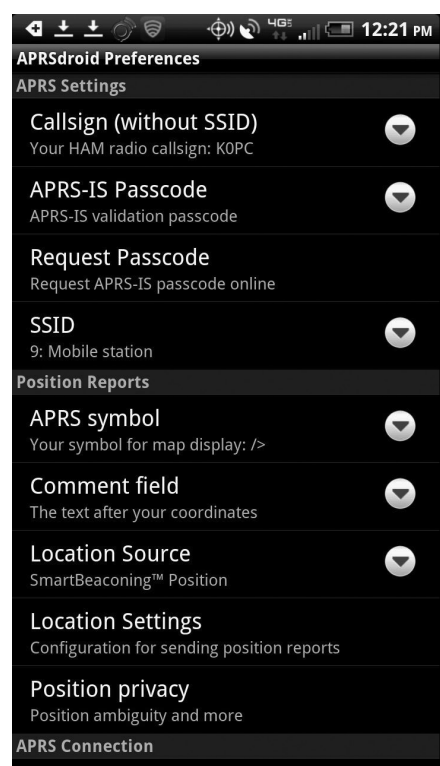


Figure 2 — A list of some available *APRSdroid* s “Preferences”

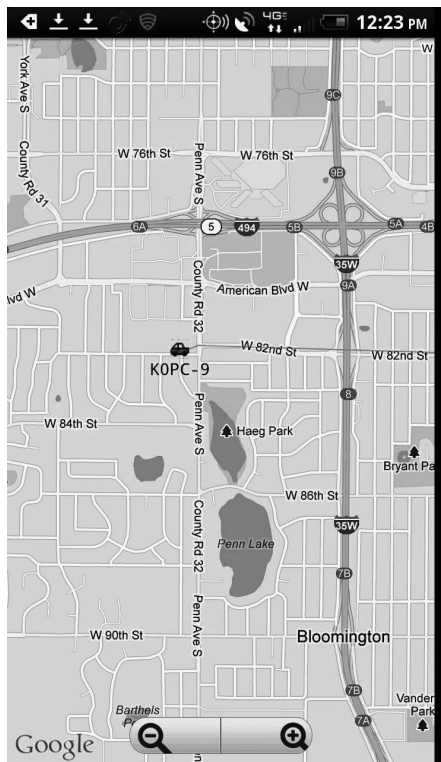


Figure 3 — A map showing the location of K0PC-9 as a tiny automobile graphic

network, so it is prudent to consider the data usage requirements. It would be a shame to have a great application, only to have it blow up your data plan. *APRSdroid* is very frugal on data usage. I have found it uses about 50 to 75 kB of data per hour. Over the course of a 12 hour QSO party this adds up to less than 1 MB of data.

One way to minimize the data the application uses is to reduce the “Neighbor Radius” setting on the “Connection Preferences menu.” This setting determines the radius for nearby stations that the application is tracking.

### Conclusion

*APRSdroid* greatly improved the usefulness of my QSO party tracking maps. Accurate location information lets the fixed operator know when the mobile station will cross into the next county. This program has worked very well for me. Georg is still enhancing it, and it is well worth more than its price, which is less than \$5.

NCJ

## UM Unified Microsystems



### New! USB Interfaces

Free your PC serial ports. USB to CW/PTT & USB to VK-64 versions.

### VK-64 Combo Voice/CW Keyer

Voice keyer and full feature CW memory keyer in a single package. Front panel operation or control through your laptop or PC.

### BCD-10 Band Decoder

Use band port signals from selected Yaesu® rigs or PC printer port for automatic antenna switching as you change bands.

### XT-4 CW Memory Keyer

Battery powered and small size for VHF rover, FD, DXpeditions and vacations. 4 memories.

### XT-4BEACON - CW Beacon IDer

Easy to program IDer for VHF beacons. Low power. Selectable speeds 5-25 WPM.

Visit our web site for new products.

### Unified Microsystems

PO Box 133

Slinger, WI 53086 262-644-9036

[www.unifiedmicro.com](http://www.unifiedmicro.com)

**SuperBertha.com**

**Ultimate Antenna Systems**

Scott Johns W3TX • 814-881-9258 • [Sales@SuperBertha.com](mailto:Sales@SuperBertha.com)

## Get Your PDF Catalog NOW at SuperBertha.com

### SuperBertha: 75ft high to 200ft+ high

### LittleBertha: 35ft high...45ft high...55ft high...65ft high

- Ultimate Antenna Tower • Individually Engineered to TIA-G For Your Exact Location and Your Custom, Stacked Antenna Load • No Guy Wires
- Entire Pole Rotates • Ground Level Rotor • Small Yard Footprint
- Stack and Rotate All Your Antennas at Optimum Heights on One Pole!

### “OWA” Optimized Wideband Array Yagis...

### The Ultimate Monoband Yagi

- Monoband OWA Yagis: 6M, 10M, 12M, 15M, 17M, 20M, 30M, 40M, 80M,
- Custom Commercial OWA Yagis for Business-Government-EmComm
- Sustained Optimal Gain, F/B, and Clean Pattern Across Entire Monoband
- SWR Better Than 1.4 to 1 Across the ENTIRE SSB, CW, RTTY, and DIGITAL Passbands.

### Rotating Guyed Tower Hardware and Ring Rotors

- Rotating Base Systems, Rotating Guy Rings, and Ring Rotors for 45G, 55G, LittleBertha, SuperBertha or Custom.

### Estate Sales Division

- All great things must come to an end. And when your station must be de-commissioned, rely upon SuperBertha's pre-arranged Estate Sales Division to keep life simple for your survivors.
- Pre-arranging relieves you & your family of the worry surrounding the eventual decommissioning, take-down, and disposition of your amateur radio station and towers.
- Professional and Fully Insured...Contact Scott W3TX for a confidential discussion of your needs today.



Specifications subject to change without notice.



### Custom CW Ring Tones for your Cell Phone

Visit Website for details.

- iPhone
- BlackBerry
- Android



# ARRL Education and Technology Program Space/Sea Buoy

## Mark Spencer, WA8SME

*"As hurricane Irene approaches, the sustained winds are 75 MPH as reported by a buoy located in Long Island Sound."*

*"The west coast is on high alert after an earthquake hit Japan a few hours ago."*

*"The tsunami early warning buoy system is being monitored for the first signs of potentially dangerous waves that are generated by seismic events of this magnitude."*

*"The debate about global climate change is again heating up, discussions on both sides are being supported from environmental buoys reporting weather data from the North Pole."*

All of these fictitious scenarios have one thing in common — remote sensing from seaborne or airborne buoys. But how does that work?

## A Hands-On Teachable Experience

The latest addition to the ARRL's Education and Technology Program (ETP) portfolio of offerings is a Teachers Institute-2 based on just that question — the technology behind environmental buoys. The current cost of a single seaborne environmental buoy with simple temperature and pressure sensors is approximately \$16,000, excluding the cost of deploying the buoy and the satellite data link time to report out the data collected. Obviously the cost is far outside the reach of most schools, yet our lives can depend on this technology.

The goal of the ETP is to allow schools to access, study and participate in the use of this kind (and other kinds) of technology in an affordable, hands-on way. It also provides the teachers with the fundamental understanding of the technology so that they can in turn develop lessons to teach the technology to their students. Yes, the ETP encourages schools to include Amateur Radio in their curriculum, but not necessarily as a stand-alone subject. Amateur Radio more importantly provides a conduit, avenue or tool that schools and teachers can use to support the study of other curricular areas. In other words, Amateur Radio provides a gateway to discovery. The ETP Sea/Space Buoy project is an example.

## The ETP Sea/Space Buoy Project

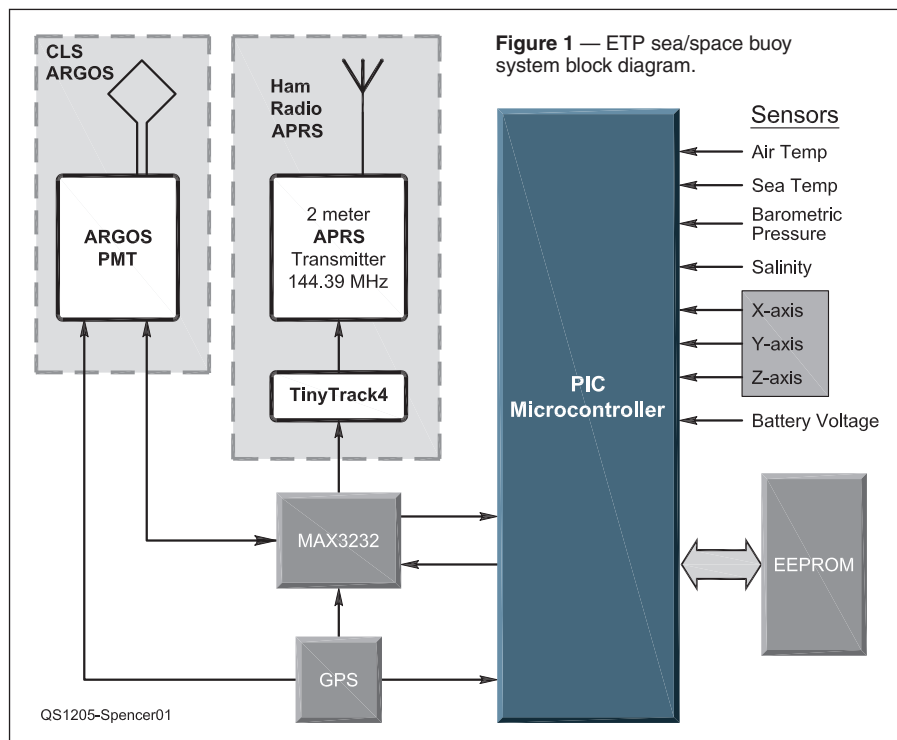
This introductory article summarizes the sea/space buoy system that has been developed. This development is a work in progress; modifications and improvements will be made as the system develops. This is an introductory article because the scope of the technology cannot be covered adequately in one printed article. Consequently, the details of the individual components of the system will be provided online through links on the ARRL website as they are developed. The buoy is called a sea/space buoy because the system can be deployed either in a surface buoy or attached to a weather balloon payload and deployed into near space.

The purpose of a buoy is to access remote areas for long periods of time that are either too dangerous, too expensive or too remote to visit and to make direct observations for study and monitoring. Setting up the technology to make the observations is one thing; getting the information out of the buoy in a timely manner is another matter altogether.

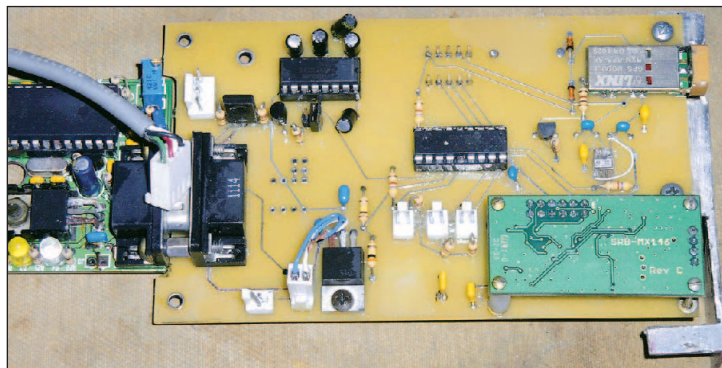
**Remote measurement technology can be the beginning of many rewarding classroom experiences.**

The weather services around the globe deploy weather balloons many times a day to take climatological readings in the upper atmosphere.

These readings are data linked down by a transmitter, and the balloons are tracked by ground stations to make wind measurements. At a predetermined altitude, the balloons burst and the sensor packages return to Earth,







**Figure 2** — Sensor package circuit board mounted on buoy body top hatch below the 2 meter data link vertical dipole antenna.

in all likelihood never to be seen again.<sup>1</sup> Government and education institutions are doing climate change studies by deploying sea and ice borne buoy systems around the globe and in the regions of the Earth's poles.

### Measuring Data Remotely

The battery operated sensor packages take temperature, pressure, salinity and other environmental readings and transmit the information to passing NOAA weather satellites using a commercial package carried on the satellites called Advanced Research and Global Observation Satellite (ARGOS). Once the data is in the ARGOS system, it is disseminated to interested users via the Internet. The use of the ARGOS system is not limited to environmental research. Users also track wildlife in remote areas.

Position information can be transmitted from on-buoy (or attached collar when tracking animals) GPS systems or the ARGOS system can use the Doppler shift of the transmitted signal to determine the transmitter's location with remarkable accuracy. The ARGOS system is a subscription service and is not free. The ETP sea/space buoy system is intended to simulate, in the typical classroom environment, the actual buoy systems used. Amateur Radio and the Automatic Position Reporting System (APRS) is a perfect, alternative, affordable fit for this project for the data link.

### What's in the Package?

The ETP sea/space buoy system is depicted in the block diagram of Figure 1, and the prototype board outside the buoy body is shown in Figure 2. The centerpiece of the system is the microcontroller that is used to manage the attached sensor packages. The system is flexible enough to adapt the sensors to a particular application. In the case of the ETP sea buoy, the sensors include air and sea temperature sensors, an air pressure

sensor, a salinity probe, a three axis accelerometer to measure wave activity and a GPS module to report position information. Alternatively, if the system were to be deployed as a balloon payload, the sensor package might include just the air pressure sensor, air temperature and accelerometers.

Because the space buoy package will have a short mission duration, and the data is to be collected at a shorter time interval (seconds or minutes versus the hour duration between data points in the sea borne application), power limitations (battery life) are not a factor but data link stability is. Consequently, the space buoy has an onboard recording capability to back up the data link so that the collected data can be recovered when the payload is retrieved.

### Internal and External Data Links

Besides the microcontroller and sensors, the other integral part of the buoy system is the data link. The ETP system is flexible enough to interface to the ARGOS transceiver for a professional application, or use the ham radio APRS system. The majority of the schools that will use the ETP Space/Sea Buoy system as an educational resource will use the APRS as a data link. The ETP system interfaces to the Bionic TinyTrak 4 modem and the SRB Electronics SRB-MX-146 APRS 2 meter transmitter module.

The sensor/microcontroller interface requires the extensive use of basic electronics from Ohm's law, to voltage dividers, voltage regulation, current limiting, filtering and beyond. Basic algebra skills are employed

## ARRL's Education and Technology Program

This curriculum was developed as part of ARRL's Education and Technology Program (ETP). The ETP is funded by donations from individuals and clubs in the amateur radio community. The ETP is an outreach program to US schools to introduce teachers to Amateur Radio as an instructional resource and to provide an educationally sound curriculum focused on wireless communications. The goal is to offer the resources to build a foundation of wireless technology literacy among American teachers and students. Find more information about the ETP at [www.arrl.org/etp](http://www.arrl.org/etp). To make a donation, visit [www.arrl.org/education-and-technology-fund](http://www.arrl.org/education-and-technology-fund).

to calibrate the sensors and to translate the raw data collected (basically proportional voltages and currents) into useable data (temperature in degrees C and pressure in millibars, for example).

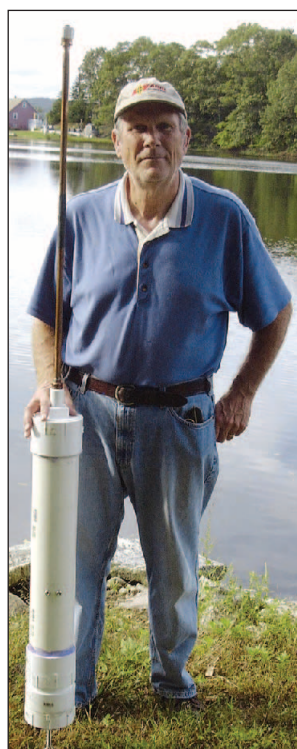
The intrasystem communications between the microcontroller and the sensor systems use a number of the common communications strategies including universal synchronous asynchronous receiver transmitter

(USART serial communications) and inter-integrated circuit bus (I2C) protocols. For voltage and current measurements, the microcontroller's analog to digital converters (ADC) are used.

All of these topics and some basic C language programming techniques for the microcontroller will be introduced to the teachers during the Teachers Institute and will be covered in greater detail in the supporting web-based articles on this project.

### The Physical Plant

The sea borne buoy body is constructed out of common household building supplies that can be found at home improvement stores. Figure 3 illustrates the construction of the "spur" ETP buoy. The sensor package, battery pack and ballast are located inside the buoy body. The sea temp and salinity



**Figure 3** — Buoy body constructed out of common PVC plumbing fixtures available in home improvement retail stores.

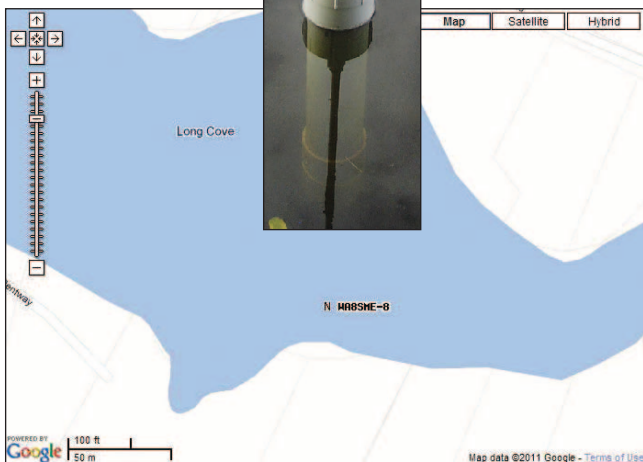
<sup>1</sup>The US National Weather Service attaches a return mailing sticker so the finder can simply drop the sensor package in the mail to return it to the NWS.

sensors are mounted on the outside of the buoy below the water line. The air temperature sensor is mounted in the tip of the 2 meter vertical dipole antenna used for the APRS data link. The buoy is ballasted to sink to a predetermined level to keep the GPS antenna above the water surface while having the center of gravity and buoyancy well below the water line for stability.

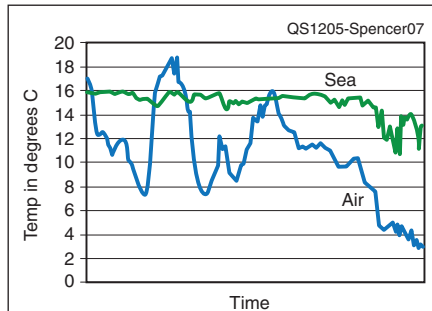
### Setting it Free

The prototype sea borne buoy was deployed for “wet” bench testing and shakedown in a brackish water cove located near the mouth of the Thames River in Connecticut where it spills into Long Island Sound (see Figure 4). In operation, the data is collected at 30 minute intervals and packaged as UNPROTO text that is sent via the APRS system. Once captured by the local APRS node, the data is disseminated through the Internet to interested users. Using *FindU* and *call sign* in a Google search, users can access the raw data. Figure 5 is a map display of the

**Figure 4** — The buoy is ballasted to keep the GPS antenna above the water line.



**Figure 5** — Map plot of buoy location generated by *findU.com*.



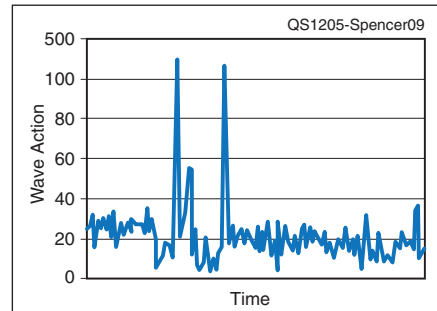
**Figure 7** — Air and sea temperature data.

buoy location from a *FindU* server. The raw data as captured and relayed by APRS is illustrated in Figure 6.

After the identification, time, location and routing fields, the sensor data is shown in comma-delimited fields that can be imported into an *Excel* spreadsheet for interpretation. The system owners would have to publish the algorithms needed to translate the reported data into meaningful information — and that is where the rubber meets the road for this project. Figures 7 and 8 show representative graphs of environmental data generated during the wet bench testing of the system and represent the end product that would be generated, interpreted and used in the classroom. There is not sufficient space to go into the interpretation of this data set here, but suffice it to say that the teachable moments you are looking at are *huge*!

Alternatively, the buoy sensor package can be modified and adapted for a balloon payload and launched with a weather balloon. This is a very popular school activity. There are, however, a few notes of caution about both the sea and air buoy systems. If you are going to deploy the systems, you have to assume that they will be lost.

There are also numerous liability issues that need to be addressed,



**Figure 8** — Wave action data.

particularly with balloon launched systems. I hope the liability concerns that some schools might have about ballooning activity and retrieving sensor packages that land in less than ideal locations is obvious. But frankly, even the discussion of potential liabilities is a teachable moment.

### The Next Steps

I hope this introduction to the ETP sea/space buoy project has piqued your interest. More details will be published and posted on the ARRL web pages as they are finalized. If you can't wait, feel free to contact me for the specifics that are immediately available. For those teachers who are interested in this project and would like to participate in the Teachers Institute-2 this summer, fill out and submit an application at [www.arrl.org/teachers-institute-on-wireless-technology](http://www.arrl.org/teachers-institute-on-wireless-technology).

There are two prerequisites for this TI2. First, the applicants must have attended a basic TI, and second, because the system depends on ham radio for the data link, the applicant must have an Amateur Radio operator's license. See the sidebar for more on the Education and Technology Program.

Photos courtesy of the author, unless otherwise noted.

Mark Spencer, WA8SME, is the ARRL Education and Technology Program Director. You can reach him at [mspencer@arrl.org](mailto:mspencer@arrl.org).



```
WA8SME-8>APTT4,WA8SME,WATECT,WIDE2*,qAR,N1MIE-5:/190914z4124.66N\07205.10WN/1005,0432,0262,0314,0562,0244,0516,0470,0904
WA8SME-8>APTT4,WA8SME,SALECT,WIDE2*,qAS,KB1BVF:/193913z4124.65N\07205.10WN/1005,0433,0260,0323,0569,0236,0535,0461,0904
WA8SME-8>APTT4,WA8SME,WATECT,WIDE2*,qAR,N1MIE-5:/193913z4124.65N\07205.10WN/1005,0433,0260,0323,0569,0236,0535,0461,0904
WA8SME-8>APTT4,WA8SME,WATECT,WIDE2*,qAR,N1MIE-5:/200920z4124.66N\07205.10WN/1007,0434,0259,0328,0556,0247,0536,0452,0901
WA8SME-8>APTT4,EKONCT*,WIDE2,qAR,W2DAN-15:/200920z4124.66N\07205.10WN/1007,0434,0259,0328,0556,0247,0536,0452,0901
```

**Figure 6** — Data reported through the APRS system and posted on *findU.com*.

frequencies passed.

5. Transmit (downlink) frequency varies with temperature. Due to the wide range of temperatures we are seeing in the eclipse cycle, the transmitter can be anywhere from around 500 Hz low at 10°C to near 2 kHz low at 40°C.

6. Receive frequency has been generally agreed to be about 435.170 MHz, although the AFC makes that hard to pin down and also helps with the uplinks that are off frequency.

We must remember that science is the reason behind these satellites. Not only does science help with the launch cost, it provides a great amount of educational value both from the science payload and in amateur radio itself. The data-under-voice (DUV) telemetry is an excellent way to provide the science without sacrificing the use of the satellite for communications, which would be the case if higher speed downlinks were needed. DUV provides constant science as long as the repeater is in use, which in turn provides more downlink data for the science – a mutually beneficial combination.

Fox-1A is AMSAT-NA's first in a series of Fox-1 CubeSats. Many new techniques are incorporated and lessons will be learned, as with any new product. A total of five will be built and flown. Launches are scheduled for the next three, and a new NASA CubeSat Launch Initiative proposal will be submitted for the fifth. We will incorporate changes from what we learn in each subsequent Fox-1 launch, to the extent possible.

Of the four NASA sponsored CubeSats on the ELaNa XII launch October 8, I am sad to report that ARCI was never heard from and BisonSat was lost after a few weeks of operation. I extend my deepest sympathy to the people who worked so hard on these projects. To our members, I want to say that we on the Fox Team are very proud and pleased that our first CubeSat is very successful and hopefully will be for some time. 🌐



## Fox-1 Satellite Telemetry – Part 1: On The Satellite

**Burns Fisher • W2BFJ**  
**Fox-1 Flight Software**

**F**ox-1A launched on October 8, 2015, and is now orbiting and operating as new amateur radio satellite AO-85. The first four Fox-1 1-Unit CubeSat amateur radio satellites, Fox-1A through Fox-1D<sup>1</sup>, are FM repeaters with 200 bps data-under-voice (DUV), frequency-shift keying (FSK) telemetry and a special high-speed 9600 bps FSK telemetry mode. This article describes the telemetry generation and modulation, as well as the software audio path, in these satellites. In this article, “Fox-1” refers only to these FM/FSK/DUV satellites.

### FSK Modulation

In order to describe Fox-1's telemetry, we need to understand FSK modulation and its relationship to FM transmitters. We typically think of FSK modulation as a carrier switched between two frequencies, one frequency representing a digital 1 (mark) and one frequency representing a digital 0 (space). A simplified diagram of this concept is shown in Figure 1 while Figure 2 shows the data versus RF waveform.

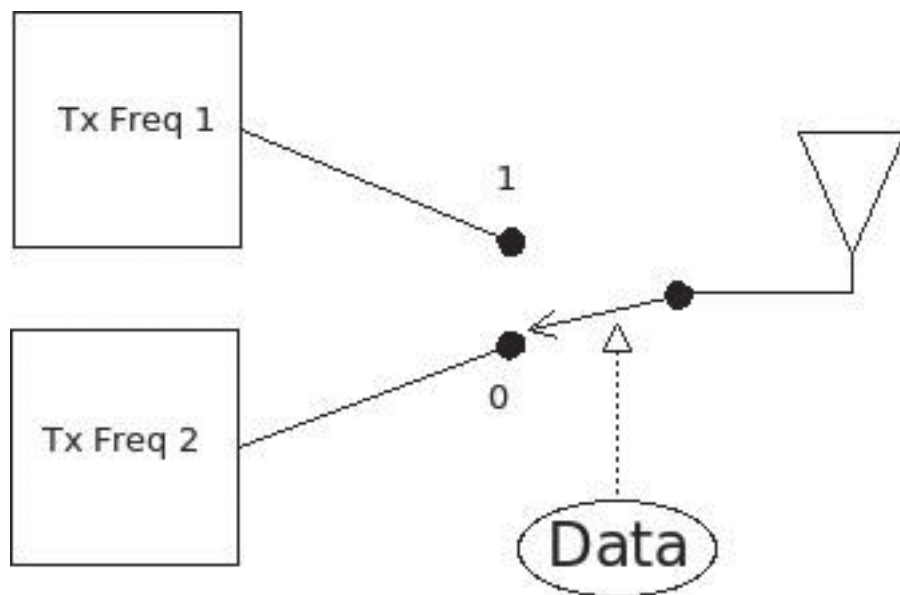


Figure 1 - Conceptual drawing of ideal FSK transmitter

But how does this relate to an FM transmitter? Remember that an FM transmitter uses a central carrier frequency that is increased and decreased depending on the amplitude of the modulating signal. If the modulating signal is a square wave of fixed amplitude, then the FM transmitter shifts between two frequencies. We can say that the “high” voltage of the square wave represents “mark,” and the “low” voltage represents “space,” and we have exactly what we want, the output shown in Figure 2 with an FM transmitter. An FM receiver, which is designed to reproduce the waveform input to the transmitter, would output the same square wave.

This describes an ideal case, assuming infinite frequency response. In the real world, however, signals with fast changes (like the square corner on a square wave) and non-sinusoidal shapes (like the flat top on a square wave) have high frequency content. A typical band-limited FM transmitter/receiver pair, if given a perfect square wave to transfer, would output a signal with rounded corners, “ringing,” and wavy tops like Figure 3.





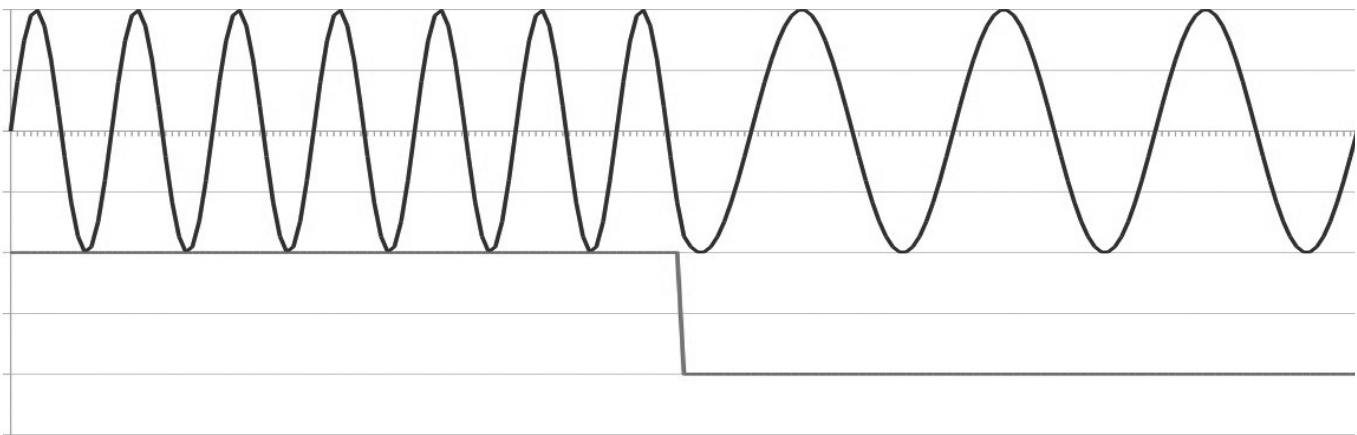


Figure 2 - Data into (bottom) and RF out (top) of an ideal FSK transmitter (not to scale)

## Audio Path

Knowing how audio is processed on Fox-I allows us to see how telemetry is incorporated into it. The Fox-I designers chose to reduce the number of components on the crowded RF boards by passing the audio path, by default, through the Internal Housekeeping Unit (IHU, i.e. the onboard computer) so that the 67 Hz tone detection, as well as some filtering and the telemetry mixing, could be done in software. A block diagram of the path is shown in Figure 4.

The uplinked audio from the UHF receiver is routed to an Analog-to-Digital Converter (ADC) in the IHU. The voltage of the audio signal is sampled and converted into numbers from 0 to 4095 -- the higher the voltage, the higher the number. The processor then manipulates that data and sends it on to the VHF downlink transmitter by way of a Digital-to-Analog converter (DAC), which turns the samples into voltages.

Since we plan for the Fox-I satellites to be operating for a long time, we also designed a fallback in case the computer fails. In that case, there is a direct analog audio path between receiver and transmitter called COR (Carrier Operated Relay). Under COR, the transmitter is keyed when the receiver detects an uplink carrier. The receiver audio is passed directly to the transmitter, with no tone detection and no telemetry.

More specifically, the processor samples the input signal at 48 KHz; 48 12-bit samples per millisecond are delivered to an input buffer via direct memory access direct memory access (DMA). There are two input buffers so that while one is being filled the other can be processed. Every millisecond, the processor performs a number of digital signal processing steps on the samples that arrived the previous millisecond, as follows:

- Reduce the number of samples from 48 to 8 (i.e., a sample rate of 8 KHz) by a process known as decimation. Decimation also acts as a low-pass filter on the signal so frequencies above 4 KHz will not cause aliasing.<sup>2</sup>
- Enter each sample into a signal processing method known as a "Goertzel" algorithm and look for a 67 Hz tone.
- Pass the samples through a high-pass filter to remove any frequencies below around 300 Hz.
- If the transponder is running, the resulting samples are used. If we are sending a voice ID, use instead on-board voice ID samples (pre-filtered to leave a gap below 300 Hz).
- Inject telemetry into the "gap" below 300 Hz.
- Raise the sampling rate back to 48 KHz through interpolation.

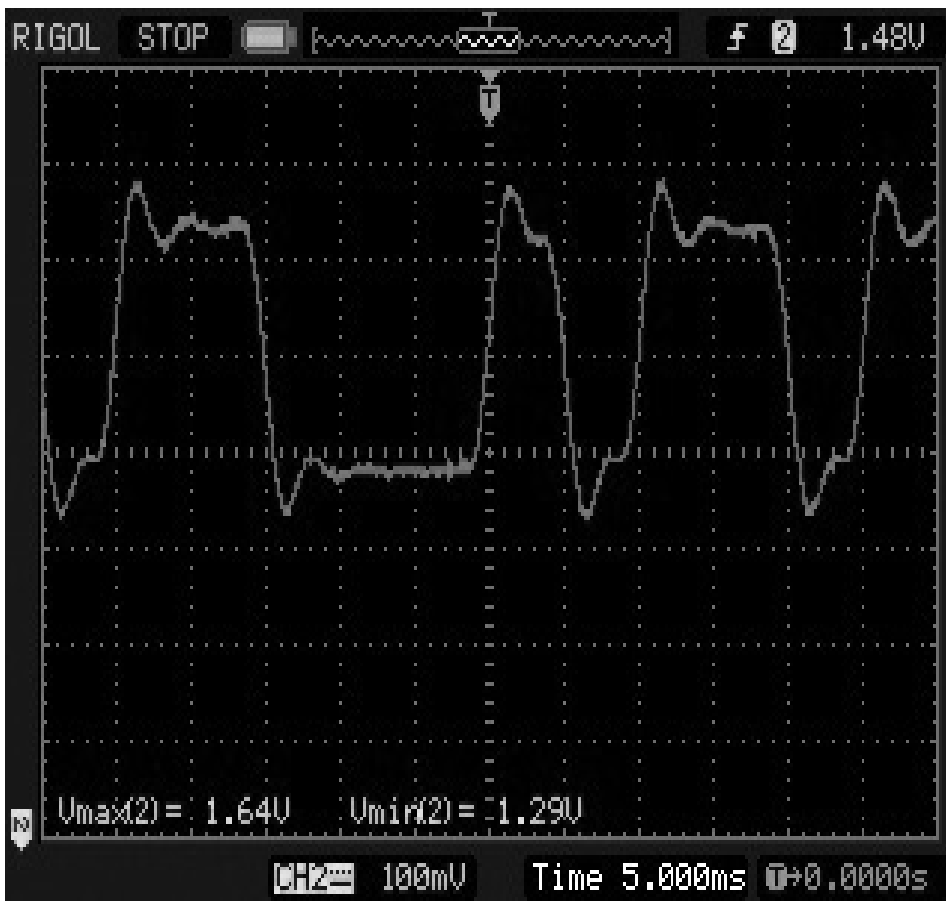


Figure 3 - Frequency limited square waves



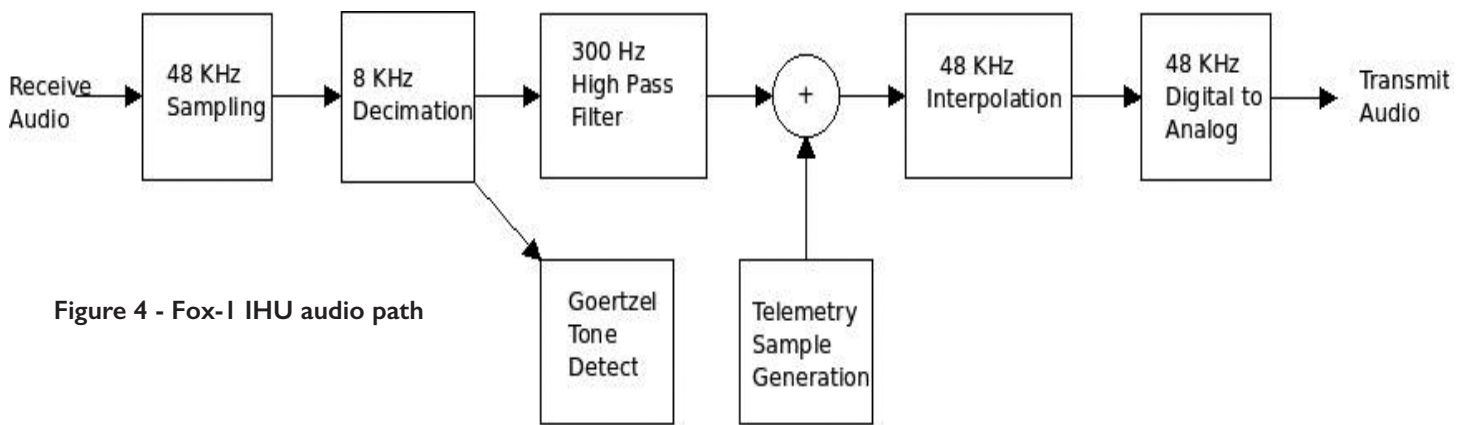


Figure 4 - Fox-I IHU audio path

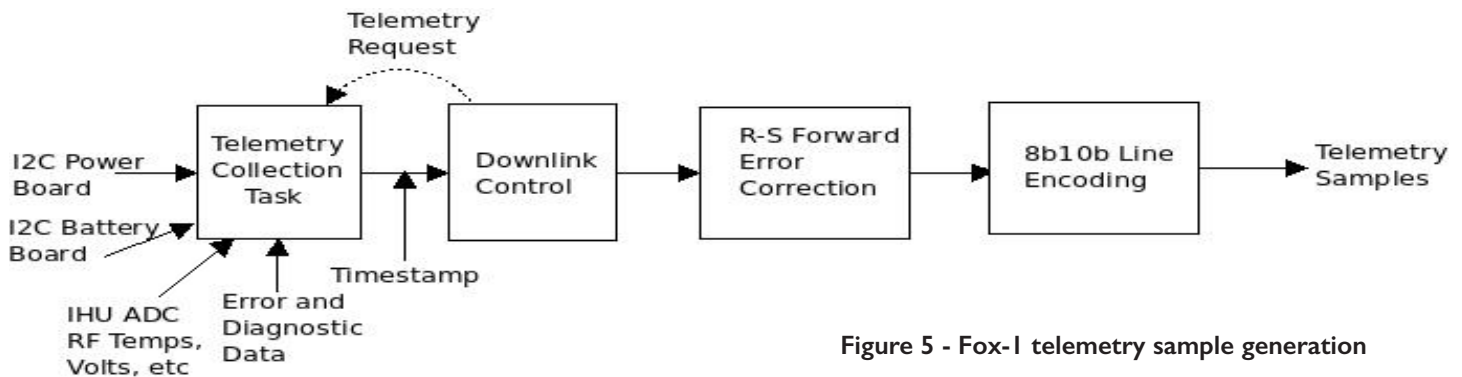


Figure 5 - Fox-I telemetry sample generation

During the next millisecond, the samples are output via DMA through the DAC to the transmitter. The length of the audio delay in Fox-I satellites' audio path is 2 ms, broken down as follows:

- 0 to 1ms, input audio is collected
- 1 to 2ms, it is processed, and
- 2 to 3ms, it is transmitted.

The 2 ms time interval is comparable to the amount of time required for an uplink signal to reach an LEO satellite and is unnoticeable to the ear.

Why start at 48 KHz, reduce to 8, and then increase back to 48 again? The reason for 8 KHz in the middle is to reduce the work required by the low-power and fairly slow IHU processor when it performs tone detection, filtering, and telemetry injection. There are several reasons for running the ADC and DAC at 48 KHz. We will see later that high-speed data mode must use the DAC at 48 KHz. The ADC and DAC are synchronized together so they have the same clock speed, and it is simplest to run the ADC and DAC all the time and not switch frequencies. And finally, a 48 KHz sample rate simplifies the analog filtering required on input and output to avoid aliasing issues.

## Data Under Voice

To understand how DUV works in Fox-I, first assume that Fox-I is receiving an unmodulated carrier, so the receiver is experiencing full quieting and is sending no audio to the ADC. In that case, we simply put numerically high samples into the output buffer for a mark and numerically low samples for a space. This makes a square wave and produces FSK modulation as described above.

In terms of the number of required samples, 200 bits of telemetry per second means 5 ms per bit. At 8 samples per millisecond, that is  $8 \times 5$  or 40 samples per bit.

We said earlier that square waves contain high frequencies. However, we only have allocated a bandwidth of 300 Hz for the telemetry. The solution is to use a low-pass filter on the square wave that represents the data, reducing it to about 200 Hz. (The 100 Hz cushion allows us to use a fairly simple and not very sharp filter.) This produces a signal more like Figure 3, with a wavy top, rounded corners and some ringing or oscillation around changes between mark and space. Notice, though, that the signal is relatively stable in the middle of the bit. The decoder knows the spacing of the bits, so it can look in the middle to identify a mark or space.

With DUV, suppose that the uplink is carrying voice modulation. The buffer where we want to put the telemetry already has samples representing the uplink voice. We must decide what percentage of the downlink modulation will be voice and what will be data. If we want 90% voice and 10% data, we scale the uplink samples by 90% and ensure that the data samples we generate do not exceed 10% of the maximum value. Then we simply numerically add the value of each filtered data sample to the corresponding filtered data sample.

What does this mean for the output carrier frequency? Remember that the carrier frequency varies depending on the voltage of the input modulation. By adding the data to the uplink sample, we have increased or decreased the corresponding output signal voltage depending on whether we are adding a mark or a space. This increased or decreased signal voltage increases or decreases what the frequency of the carrier would be without the data. Imagine a graph of the entire carrier frequency versus time shifting higher or lower with the data. You can still consider this Frequency Shift Keying, but it is just not quite traditional.

To decode the data, we use an FM receiver and work with the resulting audio. We use a sharp low-pass filter around 200 Hz (we have more processing power on the ground). Now we have reproduced approximately the data samples that we injected into the audio stream in the satellite. Imagine talking over an audio link with a hum or buzz in the background and filtering out the hum with a high-pass filter or the voice with a low-pass filter. This scheme is completely analogous.

We can now see why you cannot just plug FoxTelem, the telemetry decoder, into the speaker port of your FM receiver. CTCSS tones, which you typically use to trigger a repeater, are in the range of 67 Hz to 250 Hz. Thus, most ham FM receivers filter out these low frequencies so you don't hear them. However, that is just the frequency range where Fox-I data is located. If it gets filtered out, there is no data.

Many ham FM receivers have special high-speed TNC output ports. Some of these will work fine for telemetry, but you must look at the low frequency specifications for them. High-speed TNC output has reduced or eliminated the low-pass filter to allow high frequencies to pass through, but it still may have a high-pass filter that blocks the low frequencies we care about.

One important point is that, while the data is in the bottom 200 Hz of the audio, when transmitted on FM and mixed with the voice uplink, data is spread all across the FM channel's bandwidth.

## High Speed Data

Data mode allows us to send data at 9600 bits per second. The main purpose of this mode is to send data from on-board experiments for which 200 bps is insufficient. For example, Fox-I Cliff and Fox-I Delta will have a camera experiment supplied by Virginia Tech. While AO-85 does not strictly require data mode, we included it to validate the concept. In addition, it gets more satellite health data to us more quickly, which is an advantage during commissioning.

High-speed data is implemented in the satellite as follows:

- First, save some processing power by discarding the uplink samples. We

don't spend processor time decimating or filtering them, or looking for the 67 Hz tone.

- The extra processing time is used to collect and encode telemetry and experiment data faster.
- A speed of 9600 bps requires much more than the 4 KHz bandwidth that can be accommodated by an 8 KHz sample rate, so we put the data directly into the 48 KHz buffer. (Note: 9600 bps means .104 ms per bit multiplied by 48 samples per millisecond, which is five 48 K samples per bit.)
- We do not need to filter the data samples because there is hardware anti-aliasing to match the 48 KHz rate.
- With no voice and much less low-pass filtering, Fox-I data mode is just traditional FSK.

## On-board Telemetry Collection

The telemetry collection path in the satellite is shown in Figure 5. Note that telemetry, with few exceptions, gets collected asynchronously from its transmission to the ground. Telemetry is collected every 4 seconds on the satellite, and each value (where it makes sense) is compared with the previous minimum and maximum for that value. If necessary, a new minimum or maximum is stored along with the time (in seconds since last reset and number of resets) that a min or max changed.

The telemetry modulator uses a double buffer scheme so that, when data from one buffer is being transmitted, the other buffer can be filled. When one telemetry buffer is empty (about every 5 seconds while the transponder is on), the modulator uses the next full buffer, while the downlink manager requests the telemetry collection task to return a new packet of the data that is already collected. It is at this point that the downlink time is filled in.

The entire packet of data is now passed through the forward error-correction routines to generate Reed-Solomon check blocks as well as through an 8b10b line coding algorithm (see "Coding and Modulation Design for AMSAT Fox-I", Phil Karn, KA9Q in *AMSAT Symposium 2013 Proceedings*). The buffer is finally marked "ready" for the telemetry modulator to take when it is needed. When the telemetry modulator

freed the buffer that it was working on, the cycle begins again.

## Telemetry Values


While some telemetry values are true and false (for example "Antenna deployed"), and some telemetry values are encoded states or counts (for example, "Hard error type"), many values are analog measurements that originated from one of several 12-bit ADCs. The data is either collected directly by the ADC on the IHU board or sent by an I<sup>2</sup>C bus from other boards. All data are sent to the ground unprocessed. In other words, they are sent as a number between 0 and 4095, leaving it to the ground software to convert them to degrees, volts, or amps.

When low-speed data is being sent, four types of telemetry frames alternate according to a set pattern. Type 1 contains the real time values. These values are current readings on the satellite within the last few seconds. Type 2 contains the minimum values along with a few extra bits that are not required frequently. These are the minimum values that have been seen since the satellite min/max memory was reset by ground command. Type 3 is similar to Type 2 but uses the maximum values. Type 4 contains low speed experiment data. Each high-speed frame contains all of the above types as well as experiment data.

A future article will discuss FoxTelem, our ground-based telemetry decoder, in more detail.

## Notes

1. Fox-I C has been renamed Fox-I Cliff in honor of Cliff Buttschardt, K7RR (SK).

2. For those unfamiliar with digital signal processing, you must sample at least twice the rate of the highest frequency component in the input signal to reproduce the signal faithfully. Too slow sampling leads to an error called "aliasing." Our initial 48 kHz rate can reproduce a signal containing up to about 24 kHz frequencies. However, our decimated 8 kHz sample rates can only work with frequencies up to about 4 kHz. 



# Fox-I Satellite Telemetry Part 2: FoxTelem

Chris Thompson • AC2CZ

**F**oxTelem is the AMSAT ground station software for the Fox-I series of satellites that supports decoding of the Data Under Voice (DUV) and high speed telemetry. This article will discuss some of the advanced features of FoxTelem and will preview some of the interesting things that will be available as future satellites are launched.

## How FoxTelem Works

At its simplest, FoxTelem reads audio from a source, samples that audio into binary digits, then decodes the binary format. You can see this flow in Figure 1. The output is a frame of telemetry data. The frame is then unpacked into telemetry values. We call them “raw” values because they are the original source readings from the instruments on the spacecraft. They are shown when you click Display Raw Values on one of the FoxTelem screens. We apply conversion routines to give us human-readable data when we display them on the screens and graphs.

Let's follow the flow in Figure 1 where FoxTelem is reading audio from the soundcard and decoding 200 bps DUV. Let's also assume the sample rate is 48000 samples per second. One bit will be stored in  $48000/200 = 240$  samples. FoxTelem then reads 70 bits worth of audio from the soundcard, or 16800 samples.

The audio chunk contains the telemetry and any voice transmissions. FoxTelem runs the 70 bits through a digital filter to remove all of the audio above 200 Hz. The user can configure the filter, but in most cases the standard 200 Hz Raised Cosine Filter with 512 coefficients works well. We have broken the audio into chunks, which upsets the Digital Filter, giving us pops and crackles as the audio cycles through. We use the overlap add method to eliminate this. In summary, we take the excess audio that is created by the digital filter at the end of a chunk and add it to the start of the next chunk.

FoxTelem knows how wide one bit is by counting the samples -- 240 samples for 200 bps when the sample rate is 48000. The samples can be analyzed quickly and tagged as a one or a zero. But that rapidly fails because the first sample is very unlikely to be the start of the first bit. It is probably somewhere in the middle of a

bit. Step 1 in decoding the bits, therefore, is recovering the clock.

## Clock Recovery

Consider the situation in Figure 2 where we examine the first five bits. Each bit has 200 samples of audio. We find that the middle of the bit has values of something like 0.5, 0.5, 0, 0.5, 1. The clock is clearly misaligned. FoxTelem calculates when each bit starts to transition, on average, and works out how much the audio stream needs to be shifted. It reads some additional bits from the audio channel, effectively pulling the audio stream forward slightly. This will be a fraction of a bit, and we end up with the situation in Figure 3. If the clock stays aligned, then we won't need to adjust the clock again, but that is never the case. The clock on the spacecraft, in the hostile environment of space, and the clock in your computer, safe in your shack, do not stay aligned.

With the clock recovered, FoxTelem samples each bit in the middle of the sample period. At 9600 bps we only have 5 bits for each sample, so we average the middle 3 bits to try to compensate for noise. For 200 bps, we use a more sophisticated algorithm that measures the distance from the last bit. This better compensates for sloping bits that have

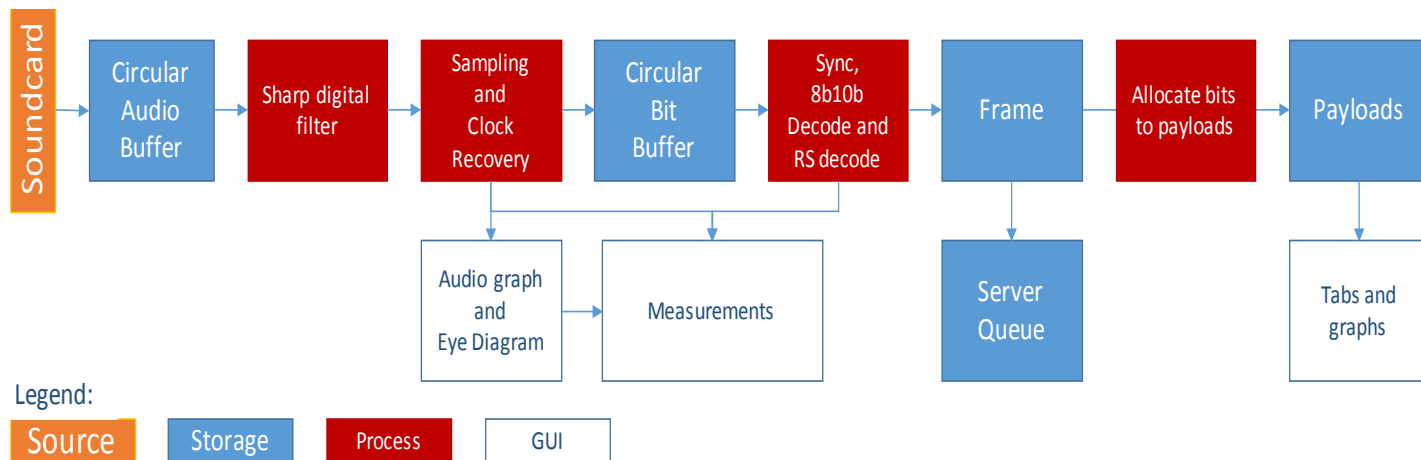


Figure 1 - FoxTelem decoding from a soundcard



Figure 2 - Mismatched sample periods

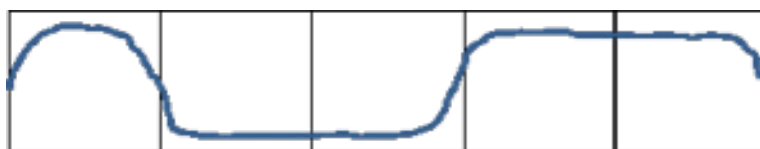


Figure 3 - After we have read an extra third of a bit



been distorted by high pass filtering, such as those shown in Figure 4. You can see that bits start at their full value then steeply slope. If we have several bits in a row representing 1, then the value can drop below the centerline, and we can incorrectly sample the last bit as a zero. The “distance to last bit” algorithm compensates for this and allows radios such as the Yaesu FT-817 to be used as the audio source.

Once the bits are sampled, they are written into a large circular bit buffer, shown in the middle of the flow in Figure 1. After each chunk of audio is sampled, we check to see if the SYNC word has been received, which is 1100000101 or its compliment 0011111010. If we find a SYNC word, then FoxTelem notes its position in the circular bit buffer. A DUV frame has 96 bytes (64 bytes of data and 32 check bytes). Each byte is sent from the spacecraft as a 10-bit word. So, if we have two SYNC words that are 960 bits apart, then we might have a valid frame. We find lots of SYNC words in random noise, so we ignore SYNC words, waiting for a valid frame length before trying to decode further.

With two sync words the right distance apart we sample at the next level of detail. It's like unwrapping a parcel with many layers of paper. Taking 10 bits at a time and decoding them into an 8-bit byte using a lookup table decodes this next layer. The bytes have been 8b10b encoded (see [en.wikipedia.org/wiki/8b/10b\\_encoding](http://en.wikipedia.org/wiki/8b/10b_encoding)) to minimize the number of 1's and 0's that we get in a row. This minimizes the slope of the bits and helps with decoding when high pass filtering is present.

Sometimes the 10-bit word is not in the lookup table, so FoxTelem notes the byte as corrupt. The 96 bytes are then passed to the Reed Solomon Decoder (see [www.ka9q.net/code/fec/](http://www.ka9q.net/code/fec/)), together with the list of corrupted bytes, called “erasures.” The RS Decoder uses the 32 check bytes to fix any errors in the received data. If it succeeds, then it reports the number of errors corrected. If it fails, then the frame is corrupt and was either too damaged or was not a real frame in the first place.

FoxTelem displays the number of errors and erasures on the eye diagram (Figure 5). This is the number from the last

successfully decoded frame. You can also plot the Errors and Erasures on a graph from the Measurements Tab. This gives you a feel for how corrupt the data stream is.

## Unpacking the Data

We now have 64 bytes of valid data -- a DUV frame -- from the spacecraft. The raw data values are packed into the 64 bytes as tightly as possible. FoxTelem uses a set of files in the spacecraft directory to unpack the data. First, we decode the header to work out what type of payload we have inside the frame. The header is laid out as follows:

```
| 1 0 1 1 0 } [0 0 1] <- [FoxID] (3 bits)
= 001
| 0 0 0 0 0 0 0 1 <- {Reset count}
(16 bits = 000 00000001 10110 = 54)
| 0 1 0 1 1 } {0 0 0
| 0 0 0 0 0 0 0 1 <- [Uptime] (25 bits)
= 0000 00000000 00000001 01011 = 43
| 0 0 0 0 0 0 0 0
| {0 0 0 1} {0 0 0 0 <- {Frame type} (4
bits = 0001)
```

We might have hoped for a simpler layout, but it turns out that this is rocket science, and apparently, even when you try to make it simple, somehow it becomes

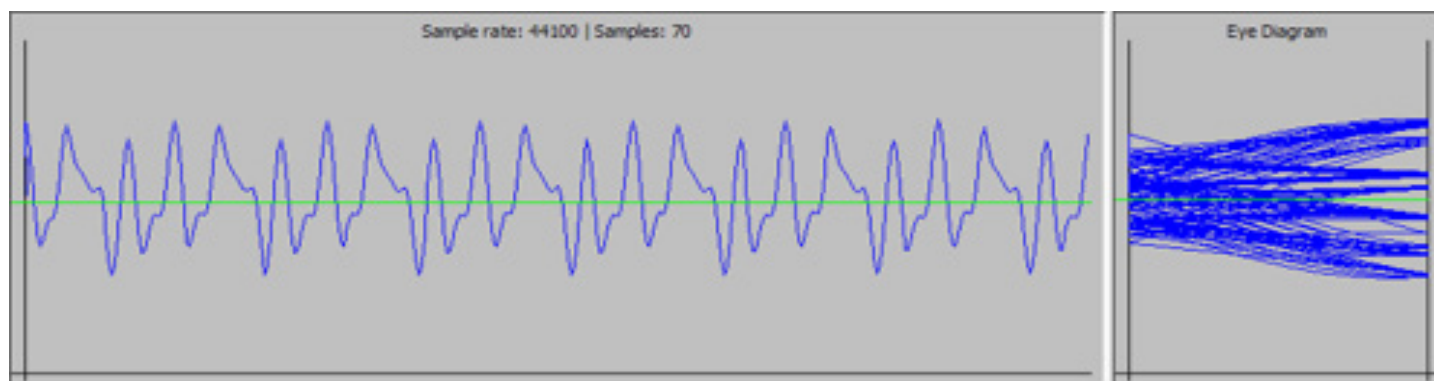


Figure 4 - Bits distorted by high pass filtering

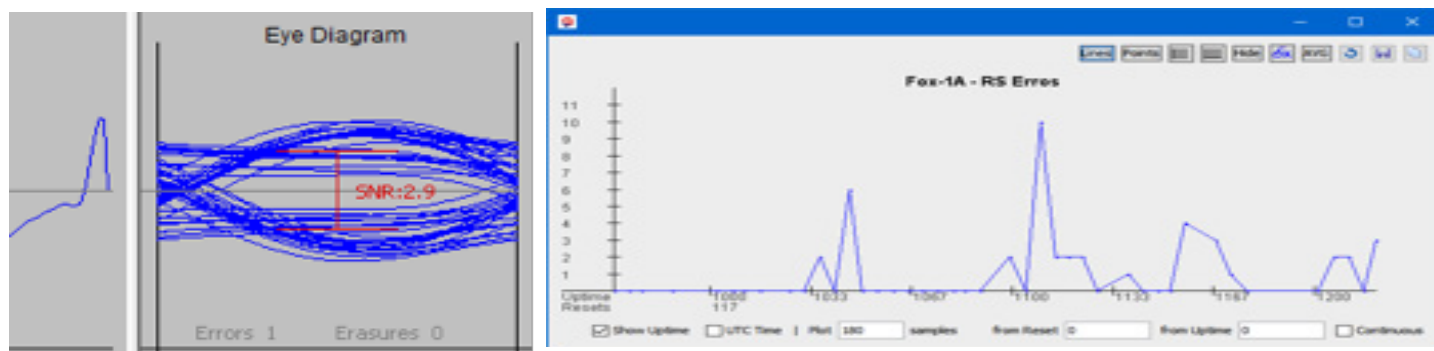


Figure 5 - Errors and erasures



complex. The complexity results from the spacecraft sending the most significant bit first (as we can see with the FoxID), but Fox's computer is a little-endian system; therefore, we get the least significant byte first. Decoding the data is full of fun little challenges like this.


Having sorted out the bit and byte order, we next decode the header and recover the Fox ID, Reset Count, Uptime and Payload Type. The FoxID and Payload Type tell us which layout we are going to decode. Types 1, 2 and 3 are telemetry for the spacecraft. Type 4 is experiment data.

We use the same bit and byte order logic as the layouts in the spacecraft directory to unpack the remaining 60 bytes into raw values. The raw values are then written to disk. If the user has selected Display Raw Values, then these are shown on the screen. Otherwise, each value is converted to a human-readable form using a conversion routine specified in the layout file. The FoxTelem manual specifies these conversions.

## High Speed

High speed data is decoded in the same way as DUV data. Each Frame is 52720 bits long. We search for SYNC words in exactly the same way and attempt to decode a frame when we find SYNC words are the right distance apart.

Instead of running the Reed Solomon Decoder for the whole frame, we break the frame into 21 Reed Solomon words and run the RS Decoder for each one. The data is sent from the spacecraft with the bytes allocated round robin to the RS words. The check bytes are all sent at the end of the frame in a block. This pattern allows us to better cope with fading and interference.

The high speed frame also contains a header, so FoxTelem first decodes it and identifies the spacecraft. Each high speed frame then contains real time, max and min telemetry payloads. The layout of the rest of the high speed frame is then determined by the experiments onboard the spacecraft: Fox-1A contains Type 4 experiment payloads; Fox-1Cliff will contain Type 5 camera lines; and Fox-ID will contain either University of Iowa HERCI data or Type 5 camera payloads. We will talk more about these payloads in a future article. 

## UPDATE: A Full-Duplex VHF-UHF Satellite System Using Flex SDR

**Ronald G. Parsons • W5RKN**  
**w5rkn@w5rkn.com**

The article, "A Full-Duplex VHF-UHF Satellite System Using SDR," that appeared in the July/August 2013 issue of *The AMSAT Journal* – [www.flexradio.com/downloads/w5rkn\\_julyaugust2013withcover-pdf](http://www.flexradio.com/downloads/w5rkn_julyaugust2013withcover-pdf) -- described a fixed ground station for working satellites using two FlexRadio Systems® radios. This update describes a number of modifications and improvements to the original station configuration. The two transmit and receive radios were replaced with a single Flex-6500, a tuning knob was added, and a software program, FlexSATPC, written by David Beumer, W0DHB, was incorporated to tie all the functions together. [See "Using SatPC32 with FlexRadio Systems Radios," p. 13.]

## Implementation

Three major changes were made to the system. The first was the addition of a software program, FlexSATPC, which acts as the system central control point. FlexSATPC provides an interface between SatPC32, the tracking and Doppler control software, and SmartSDR, the FlexRadio control and display software because SatPC32 does not support SmartSDR. FlexSATPC also includes a logging interface that enables all required logging by just entering the callsign. All other fields required for logging are automatically entered. FlexSATPC also facilitates uplink frequency calibration to match the satellite's transponder.

The second change was to improve the full-duplex operation, made possible with version 1.5.0 of FlexRadio's SmartSDR control program. The third was adding a FlexControl tuning and control knob. The knob controls the frequency of the active slice and other programmable functions. If the operator is using the FlexControl to tune the radio and a separate logging operator is entering data in other programs, the FlexControl only controls SmartSDR, leaving other programs under mouse control. I made good use of

this feature when operating as W1AW/5 and W7O.

The original design used a FLEX-1500 for transmit and a FLEX-3000 for receive, but the required switching between various bands and antennas was very complex. The single antenna port on the FLEX-3000 made it difficult to operate HF or satellites which use HF such as AO-7 Mode A.

When FlexRadio introduced its newest Signature Series model in May 2014, the FLEX-6300, the SmartSDR software that controls the new radio did not support use of transverters. In August 2014, FlexRadio added transverter support, so I replaced the two radios in the original system with one FLEX-6300. Effecting the change was simple, requiring adjusting the CAT COM port and moving the coax from the receive transverter coax switch for the FLEX-3000 to the FLEX-6300 ANT2 port and moving the coax from the transmit transverter coax switch on the FLEX-1500 to the FLEX-6300 XVTR port. That left the FLEX-6300 ANT1 output free for 6-meter use.

FLEX-6300 users must be very careful with implementing and operating a system that allows the physical hardware to transmit into the RX IF port of the transverters. I tried to minimize this risk by automating the signal routing and other software settings. FlexSATPC also contains checks to help avoid this potential problem. Still, the operator must be aware of this risk and operate accordingly.

To completely eliminate this risk, I traded in my FLEX-6300 for a FLEX-6500, which has a separate receive-only port RX A that is connected to the RX IF ports of the transverters. The XVTR port is connected to the TX IF ports of the transverters for transmitting.

## Major Features

### Full Duplex

The addition of full-duplex capability in FlexRadio's Signature Series radios – FLEX-6300, 6500, and 6700 – enables the audio from the radio's receive slice to pass through to the speakers/headphones even when another slice is transmitting.

"Slice" is the term used by FlexRadio for a segment of the entire spectrum sampled by



John E. Post, KA5GSQ

Embry-Riddle Aeronautical University, 3700 Willow Creek Road, Prescott, AZ, 86301; john.post@erau.edu

# A Simple Sensor Package for High Altitude Ballooning

*High-altitude ballooning provides a unique opportunity to motivate high school students to pursue careers in science and engineering, as well as introduce them to exciting Amateur Radio applications.*

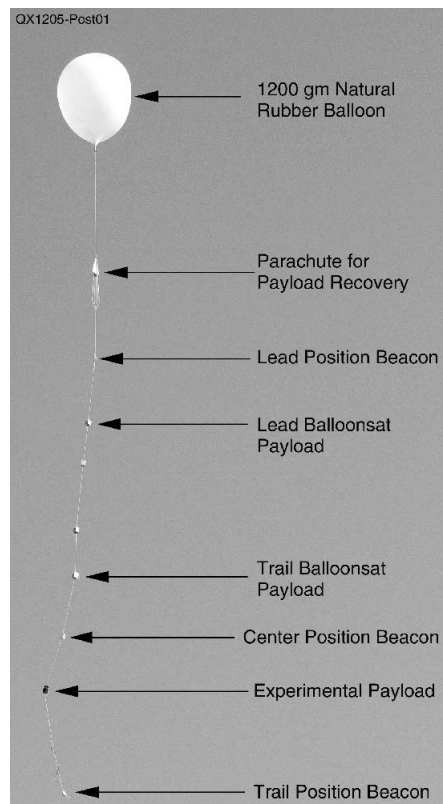
This paper describes a simple sensor system designed for a balloonsat that was assembled and flown by high school students during an intensive four-day summer camp experience. The camp was held on

the campus of Embry-Riddle Aeronautical University in Prescott, Arizona. The purpose of the camp was to motivate students to consider pursuing careers in science and engineering, specifically in electrical and computer engineering. At the end of the camp, the students were given a written survey to complete. Student responses indicated that for the most part the objectives of the camp were achieved. Several students indicated that the camp increased their interest in science and engineering topics, while one

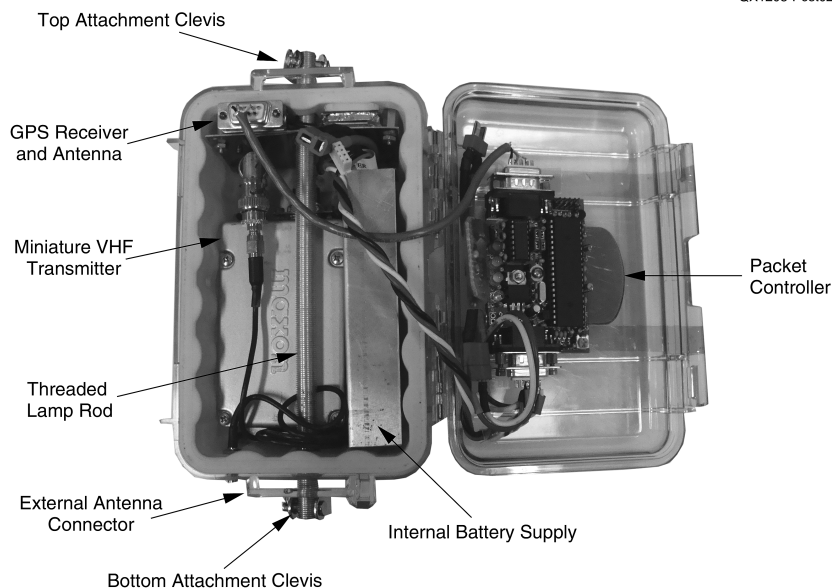
student stated a direct interest in studying electrical engineering upon graduation from high school.

Figure 1 shows the high-altitude balloon system shortly after launch. The system consists of a balloon, a parachute, and a string of payloads commonly referred to as the “balloon train.”<sup>1</sup> The balloon is typically a 1200 to 1500 gram natural rubber meteorological

<sup>1</sup>Notes appear on page 19.



**Figure 1** — This photo shows the high-altitude balloon system shortly after launch. You can see the trail of payloads that are known as the “balloon train.”



**Figure 2** — This photo shows a typical GPS/APRS position beacon with the case open and major components labeled.

or “weather” balloon that inflates to a diameter of 18 to 22 feet, and bursts at a diameter of approximately 22 to 27 feet.<sup>2</sup> The balloon train consists of individual payloads or “balloonsats,” which are attached end-to-end using 10 to 15 foot lengths of high-strength nylon cord. The first and last payloads in the balloon train are usually position beacons containing a GPS receiver that determines position from the GPS satellite constella-

tion and a packet transmitter that broadcasts that position using the Automatic Packet Reporting System (APRS). The balloon train includes multiple position beacons in order to improve the likelihood of recovering the payloads if an unanticipated structural or electrical failure occurs during flight. Figure 2 is a close up view of a position beacon constructed for operation in the 70 cm Amateur Radio band.

The purpose of the sensor system is to record sensor data taken during the ascent and descent of a balloonsat payload during a high-altitude balloon flight. The data recorded during the flight includes the temperature inside the payload, the external ambient temperature, the ambient atmospheric pressure, as well as the voltage of the internal Lithium-ion Polymer (LiPo) battery supply. The balloonsat payloads also contained still and video cameras that obtained breath-taking images from the edge of space. After recovery of the balloonsat payloads, the students were able to download and analyze the data in order to better understand the characteristics of the near-space environment.

This article details the design and operation of the sensor package developed for the balloonsat payloads. Additionally, the article will discuss application of the APRS digital communications protocol in order to track the payload during flight, and facilitate payload recovery after landing. Finally, the article will review a selection of the sensor data and still images obtained during the flight of ERAU-06, which occurred the morning of June 29, 2011.

### Sensor System Overview

Figure 3 presents an overview of the sensor and data recording system. As shown in the figure, four sensors measure atmospheric pressure, internal and external temperature, and battery voltage during the flight. The output of each sensor drives an op amp subtractor to remove the dc offset voltage and rescale the sensor readings so that they are within the 0 to 3.3 V data logger input range. The data logger digitizes the four analog voltages at a sampling rate of 1 Hz and writes the data in table format onto a micro-SD card installed on the data logger board.

### Sensor System Sub-Circuit Details

**Data logger:** The data logger used for the sensor system is a Logomatic v2, obtained from Sparkfun Electronics.<sup>3</sup> See Figure 4. The Logomatic v2 digitizes analog voltages between 0 and 3.3 V with a resolution of 10 bits, which is equivalent to  $2^{10} = 1024$  possible integer values (0 to 1023). This results in a resolution of  $1/1023 \text{ bits} \times 3.3 \text{ V} = 3.22 \text{ mV per bit}$ . The Logomatic writes data to a microSD card installed on the circuit board. Additionally, the Logomatic includes a type B mini-USB connector, so the microSD card mounts on the desktop of a PC just like a standard flash drive. Once the drive mounts on the desktop it is straightforward to edit, copy or delete the data files from the PC desktop. Table 1 lists the configuration file used to initialize the data logger for this

QX1205-Post03

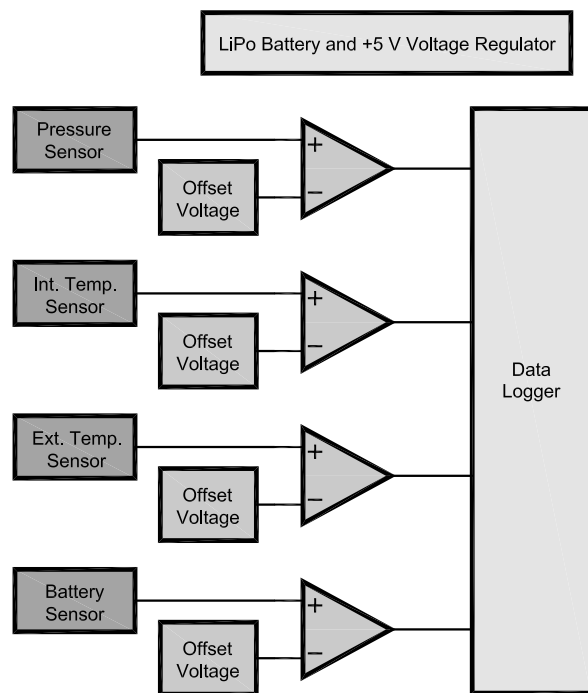


Figure 3 — Here is the block diagram of the sensor system, depicting the analog signal processing that occurs prior to digitization and recording of sensor data by the data logger.

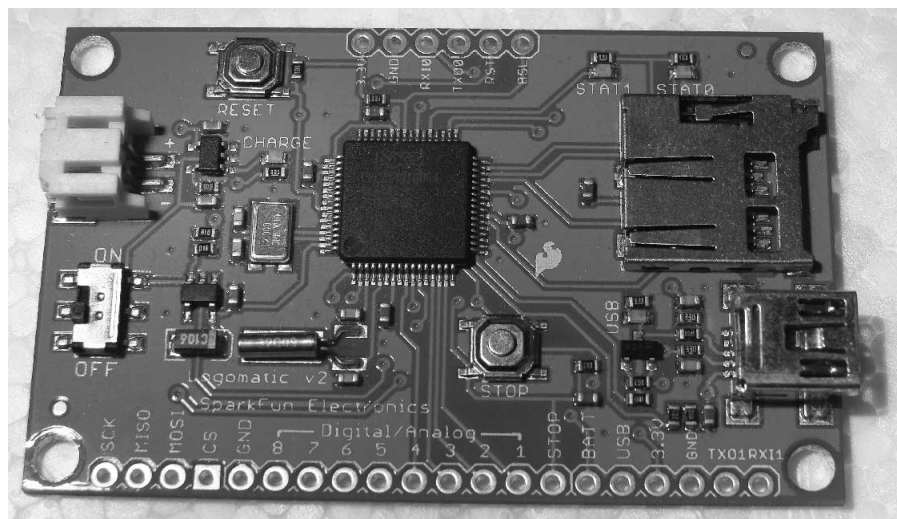


Figure 4 — This photo shows the Logomatic v2 purchased from [www.sparkfun.com](http://www.sparkfun.com).

**Table 1****Datalogger Configuration File.**

This file configures the datalogger for analog-to-digital conversion (MODE=2), in text format (ASCII=Y), at 1 sample/sec (Frequency=01), with four channels active (AD0.3=Y, etc.) and four channels disabled (AD1.3=N, etc.). The other commands do not apply to ADC logging.

```

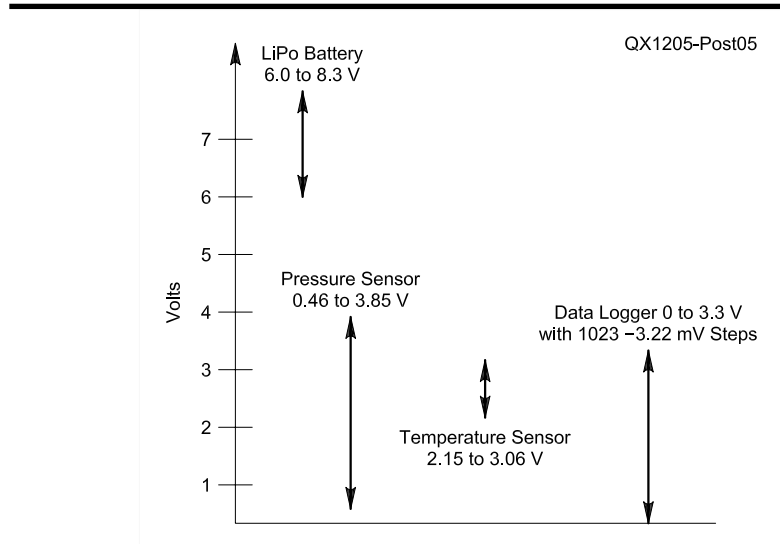
MODE = 2
ASCII = Y
Baud = 4
Frequency = 01
Trigger Character = $
Text Frame = 100
AD1.3 = N
AD0.3 = Y
AD0.2 = Y
AD0.1 = Y
AD1.2 = N
AD0.4 = Y
AD1.7 = N
AD1.6 = N
Safety On = Y

```

application, along with a description of the more important control commands.

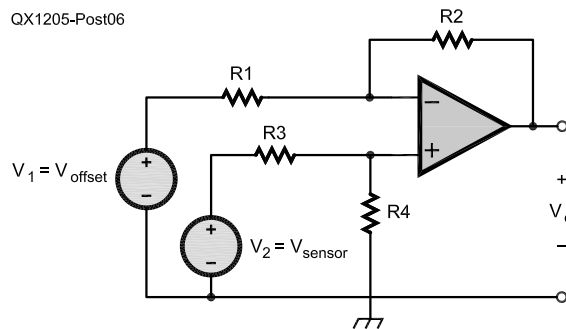
**Analog Signal Processing:** Analog signal processing of the output of each sensor is necessary in order to maximize the resolution of the recorded data. This design uses op amp difference amplifiers to level shift and scale the output of each sensor so that the output remains inside of the 0 to 3.3 V range of the data logger. See Figure 5. Since the input voltage to the data logger must remain positive, a single-supply quad op amp such as the LM324 is satisfactory.<sup>4</sup> The LM224 was selected for this application because it is capable of operating at temperatures as low as  $-40^{\circ}\text{C}$ . The four independent op amps on the LM224 provide the ability to level-shift and scale the outputs from four different sensors.

**Pressure Sensor:** Atmospheric pressure is determined with a Honeywell ASDXACX015PAAA5 0 to 15 PSI absolute pressure sensor.<sup>5</sup> This sensor has a maximum sensor output of 4.58 V at 15 PSI, with a sensitivity of 0.267 V/PSI. The ambient pressure at the launch location (5000 feet) is approximately 12.23 PSI. This reduces the maximum expected sensor output voltage to  $4.58 - 0.267 \times (15 - 12.23) = 3.85$  V. The minimum zero pressure offset voltage for the sensor is 0.42 V. Adding this value to the sensor voltage at the anticipated maximum altitude of 100,000 feet (0.162 PSI) gives the minimum sensor voltage of  $0.42 + 0.162 \times 0.267 = 0.463$  V. Thus, in order to maximize the resolution of the recorded data, the output



**Figure 5 — This graph shows the relationship between sensor output voltages and data logger input voltages. This illustrates the need to level shift and scale each sensor voltage in order to maximize the resolution of the recorded measurements.**

QX1205-Post06



**Figure 6 — This is a schematic diagram of an op amp difference amplifier. This circuit amplifies the difference between the two input signals  $v_{\text{sensor}}$  and  $v_{\text{offset}}$  in order to level-shift and scale the sensor output voltage**

of the pressure sensor must be level shifted by 0.463 V and amplified by  $3.3 / (3.85 - 0.463) = 0.974$  by the analog signal processing circuit prior to digitization.

**Temperature Sensor:** Internal and external temperatures are determined using the LM135 and LM335 precision temperature sensors.<sup>6</sup> The LM335 operates from  $-40^{\circ}\text{C}$  to  $100^{\circ}\text{C}$ , so it is suitable for obtaining internal temperatures, whereas the LM135 operates from  $-55^{\circ}\text{C}$  to  $150^{\circ}\text{C}$  so it is suitable for obtaining external temperatures. Each device has a sensitivity of  $+10$  mV/ $^{\circ}\text{C}$  with maximum and minimum output voltages of 3.04 V and 2.92 V at  $25^{\circ}\text{C}$ . Assuming a maximum launch temperature of  $80^{\circ}\text{F}$ , or  $27^{\circ}\text{C}$ , the maximum output voltage is  $3.03 + (27 - 25) \times 0.01 = 3.05$  V. Assuming a minimum temperature at altitude of  $-50^{\circ}\text{C}$  gives a minimum output voltage of  $2.92 - (25 - (-50)) \times 0.01 = 2.17$  V. Thus, the output of

the temperature sensors must be level shifted by 2.17 V and amplified by  $3.3 / (3.05 - 2.17) = 3.75$  by the analog signal processing circuit prior to digitization.

**Power Supply and Voltage Regulator:** A lightweight, rechargeable 7.4 V LiPo battery provides sufficient capacity to power the sensor package for approximately five hours.<sup>7</sup> Note that LiPo batteries require a recharging system specifically designed for this type of battery. (Do not recharge them with chargers designed for NiCd or NiMH batteries.) Additionally, an MC78L00A 5 V regulator establishes a constant voltage supply for the op amp difference amplifiers as well as providing a stable reference from which to establish the necessary offset voltages.<sup>8</sup>

**Battery Monitor:** The data logger records the voltage of the LiPo battery during the flight in order to monitor the reduc-



tion in battery voltage at colder temperatures. The battery voltage varies from a maximum of about 8.3 V when the battery is fully charged to approximately 6 V, at which the MC78L00A 5 V voltage regulator drops out. Level shifting by 5 V gives a maximum voltage of  $8.3 - 5 = 3.3$  V and a minimum voltage of  $6 - 5 = 1$  V. In this case, no additional scaling was necessary, since this is a satisfactory resolution for this application.

### Op Amp Difference Amplifier Design

Figure 6 gives the schematic diagram of a general difference amplifier.<sup>9</sup> This circuit amplifies the difference between two input voltages, so it is useful for performing the analog signal processing necessary to interface the output of each sensor to one input

channel of the data logger. The mathematical relationship between the output voltage,  $v_o$ , as a function of the input voltages  $v_1$  and  $v_2$  and the resistors  $R_1 - R_4$  is given by Equation 1.

$$v_o = \frac{R_2 \left( 1 + \frac{R_1}{R_2} \right)}{R_1 \left( 1 + \frac{R_3}{R_4} \right)} \times v_2 - \frac{R_2}{R_1} \times v_1 \quad [\text{Eq 1}]$$

Under the constraint that  $\frac{R_1}{R_2} = \frac{R_3}{R_4}$ , then Equation 1 becomes:

$$v_o = \frac{R_2}{R_1} (v_2 - v_1) \quad [\text{Eq 2}]$$

If the offset voltage is applied to  $v_1$ , the sensor output voltage is applied to  $v_2$ , and the ratio  $R_2 / R_1$  selected properly, the difference amplifier is able to accomplish the level shifting and scaling necessary to interface each sensor to the data logger. Inverting Equation 2 yields Equation 3, which allows recovery of the original sensor voltage  $v_2$  from the digitized voltage  $v_o$ .

$$v_2 = \frac{R_1}{R_2} v_o + v_1 \quad [\text{Eq 3}]$$

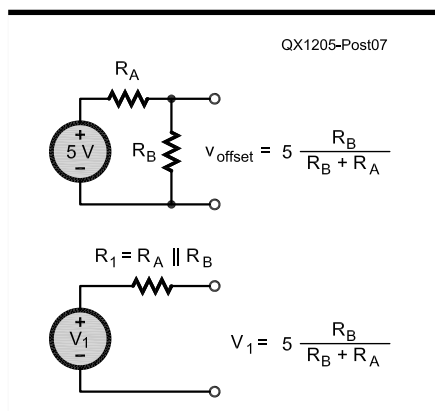
As discussed in the next section,  $R_1$  is actually a Thevenin equivalent resistance that may not be readily available as a standard resistor value for  $R_2$ . Thus, a more general result requires inverting Equation 1 to solve for the sensor voltage  $v_2$ , given with Equation 4.

$$v_2 = \left( \frac{R_1}{R_2} v_o + v_1 \right) \left( \frac{1 + \frac{R_3}{R_4}}{1 + \frac{R_1}{R_2}} \right) \quad [\text{Eq 4}]$$

### Design of Offset Voltage Sources

Voltage sources are required to provide the proper offset voltage ( $v_1$  in Figure 6) for each sensor, as discussed earlier. For this design a simple voltage divider was applied to reduce the +5 V provided by the voltage regulator. See the top circuit in Figure 7. Comparing the Thevenin equivalent circuit of the voltage divider (bottom circuit of Figure 7) with the series combination of the voltage source  $v_1$  and the resistance  $R_1$  in Figure 6 demonstrates that

$$v_1 = 5 \frac{R_B}{R_B + R_A} \quad [\text{Eq 5}]$$

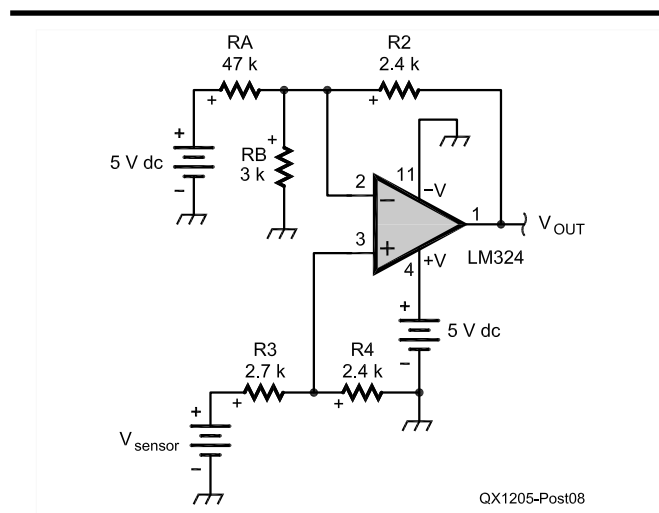


**Figure 7** — At the top is the schematic diagram of the offset voltage source, with its Thevenin equivalent circuit on the bottom. Comparing the Thevenin equivalent circuit with  $v_1$  and  $R_1$  in Figure 6 shows the effects of Equations 5 and 6, which are also shown on the right side of this diagram.

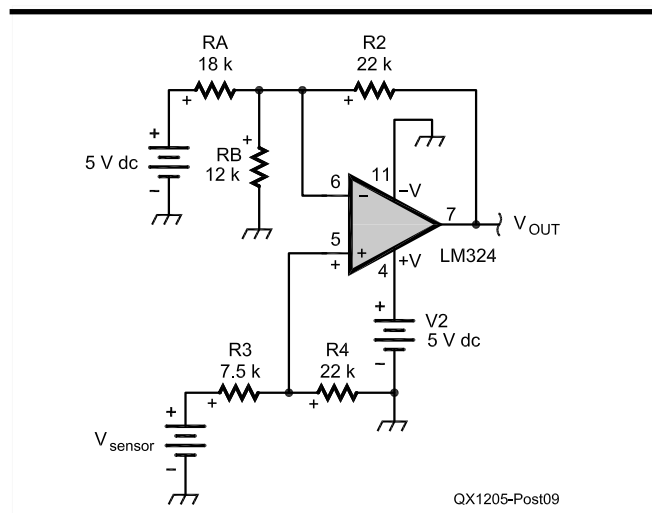
**Table 2**

List of standard resistor values  $R_A$ ,  $R_B$ ,  $R_2$ ,  $R_3$ , and  $R_4$  for design of op amp analog signal processors along with the resulting voltage gain,  $A_v$ , and offset voltage,  $v_1$ , produced for each choice.

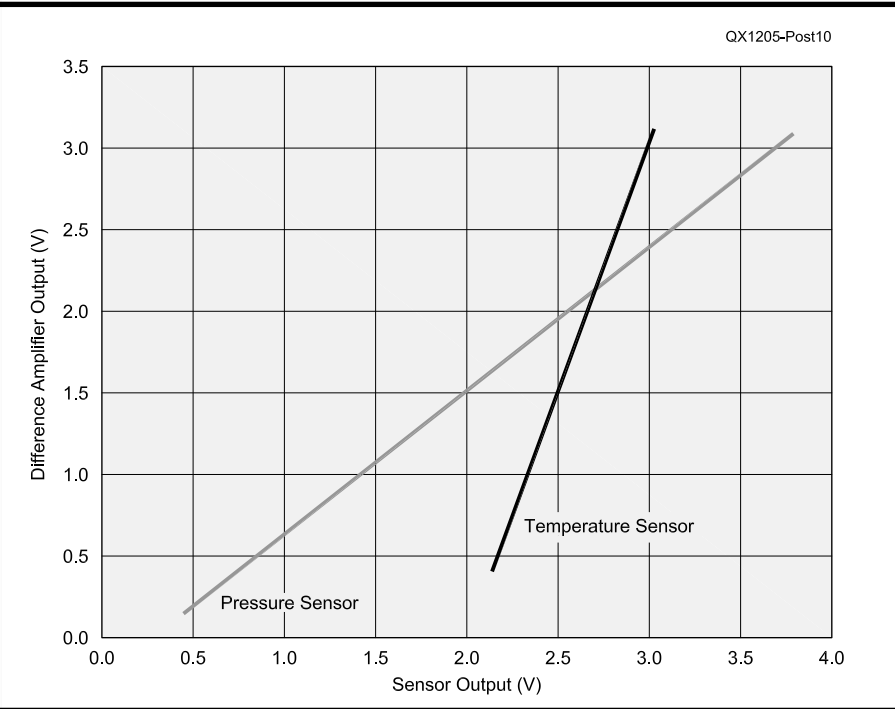
Case	$R_A$ (k $\Omega$ )	$R_B$ (k $\Omega$ )	$R_A    R_B = R_1$ (k $\Omega$ )	$R_3$ (k $\Omega$ )	$R_2 = R_4$ (k $\Omega$ )	$A_v = R_4/R_3$ (V/V)	$v_1 = v_{offset}$ (V)
Pressure	47	3	2.82	2.7	2.4	0.889	0.3
Temperature	18	12	7.2	7.5	22	2.93	2.0
Voltage	10	10	5	10	10	1	5



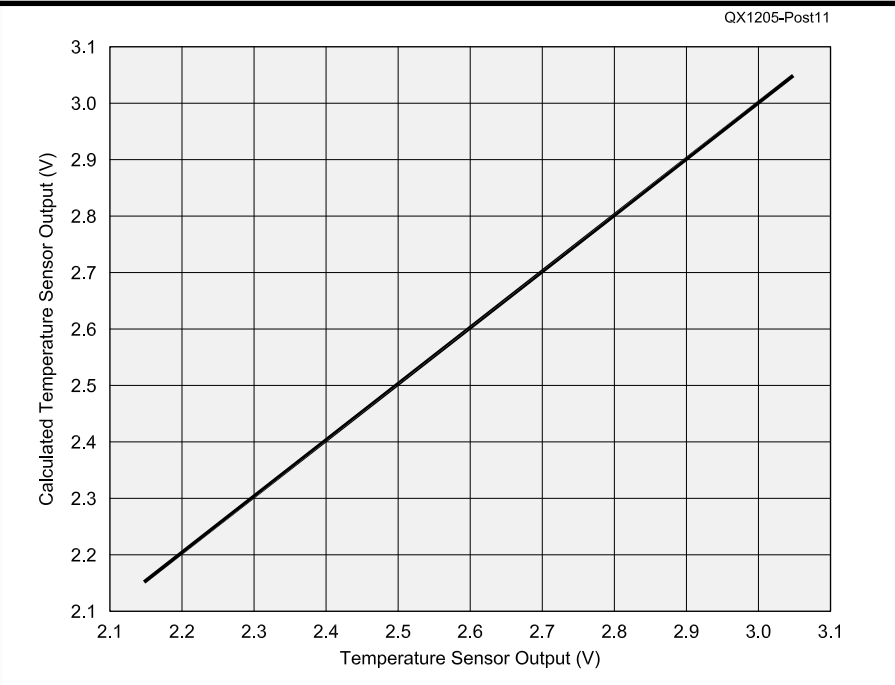
**Figure 8** — This schematic shows the final design of the difference amplifier that processes voltage from the pressure sensor.



**Figure 9** — Here is the final design of the difference amplifier that processes voltage from the temperature sensor.



**Figure 10 —** This graph is a plot of the output voltages from the simulation of the difference amplifier circuits shown in Figures 6 and 7. This plot shows that the output of the difference amplifier remains between 0 to 3.30 V as the output of the pressure sensor ranges from 0.463 V to 3.85 V and the output of the temperature sensor ranges from 2.15 V to 3.06 V.



**Figure 11 —** Here is the output of the temperature sensor versus the calculated temperature sensor output obtained by applying Equation 4 to the output voltage produced by the simulation of the difference amplifier circuit. The figure demonstrates the exact correspondence of the calculated result to the original sensor output voltage.

and

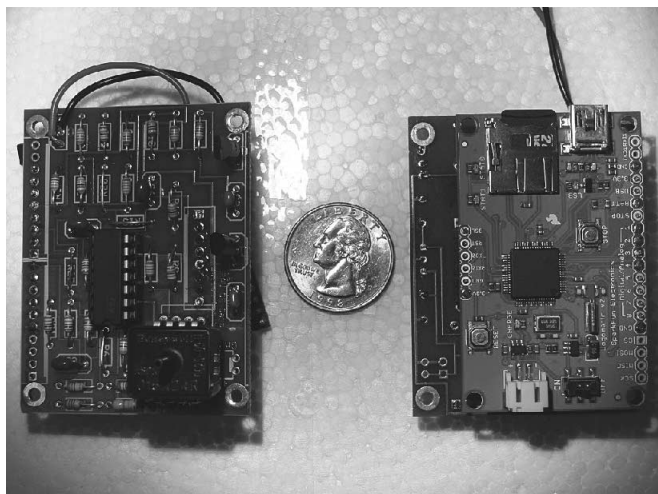
$$R_1 = R_A \parallel R_B \quad [\text{Eq 6}]$$

Since only standard resistor values were available, it is not possible to obtain the exact offset voltages and gains for each sensor derived previously. This situation is reflected in Table 2, which shows the offset voltage,  $v_1$ , and scaling factor for the temperature, pressure, and voltage sensor as functions of the resistors  $R_A$ ,  $R_B$ ,  $R_2$ ,  $R_3$  and  $R_4$ . Referring to this table, columns 2 and 3 give the values for resistors  $R_A$  and  $R_B$ , column 4 shows the equivalent value of  $R_A \parallel R_B$ , while column 5 shows the nearest standard resistor value to the adjacent value shown in column 4. Continuing to refer to the table, column 6 gives the standard value for resistor  $R_2$ , which equals resistor  $R_4$ , while columns 7 and 8 give the scaling factor and offset voltage that result from these design choices. It is important to note that even though the constraint  $R_1/R_2 = R_3/R_4$  no longer applies, the sensor voltage is still recoverable uniquely through application of Equation 4. A slight loss of resolution does occur because the amplitude scaling factors of 0.889 and 2.93 V/V shown in Table 2 are somewhat smaller than the ideal scaling factors of 0.974 and 3.75 derived in the previous section.

**Difference Amplifier Simulation**

Figure 8 shows the difference amplifier circuit of Figure 6, using resistor values from the first row in Table 2 for processing the output of the pressure sensor. Figure 9 shows the difference amplifier circuit, Figure 6, using resistor values from the second row in Table 2 for processing the output of the temperature sensor. Figure 10 shows the difference amplifier output voltage obtained by *PSpice* circuit simulations of Figs. 8 and 9 when the appropriate sensor output voltage range is applied to each circuit.<sup>10</sup> Maximum resolution is obtained when the amplifier output ranges between 0 and 3.3 V as the sensor output voltage ranges between its minimum and maximum values. Review of Figure 10 demonstrates that the goal of performing analog signal processing on the sensor output voltages so that they remain within the 0–3.3 V input range of the data logger was achieved.

As a final check of the design of the analog signal processing circuit, Equation 4 was applied to recover the temperature sensor voltage from the simulated difference amplifier output, in order to verify that Equation 4 is correct. Figure 11 gives this result and shows an exact correspondence between the sensor voltage with the voltage recovered after applying Equation 4. An identical



**Figure 12** — Here is a photo of the front (left side) and back (right side) of the sensor system circuit board. The photo of the back of the board shows how the Logomatic v2 board was mounted as a daughterboard using the sensor system circuit board as the motherboard.



**Figure 13** — Students assembled the sensor system circuit boards under the watchful eye of a veteran instructor.



**Figure 14** — Balloonsat payloads were constructed by folding  $\frac{3}{8}$  inch thick foam-core board into a cube shape approximately 6 inches on a side. The cubes were covered with adhesive-backed aluminum tape and the interior filled with automotive-type foam to provide insulation from both thermal and mechanical shocks.

procedure was followed to verify that the pressure sensor voltage was also properly recovered.

### Circuit and Balloonsat Construction

A custom printed circuit board was designed and manufactured for the sensor system.<sup>11</sup> The left side of Figure 12 shows the analog signal processing circuitry installed on the front side of the printed circuit board. The right side of Figure 12 shows that the back side of the printed circuit board served as a motherboard to which the Logomatic v2 was installed as a daughterboard using low-profile connection sockets and terminal pins. This approach allowed for separate verification of the sensor system and data logger prior to mating them together and testing the entire assembly. This was important because most of the students who assembled the circuits had little or no prior experience in electronic circuit construction. Thus, it was important to verify circuit functionality frequently during the construction phase. Figure 13 is a photograph taken during the construction phase of the project.

Figure 14 is a photo that was taken looking down at the top of a balloonsat payload. The balloonsats were constructed by cutting a cross-shaped pattern out of  $\frac{3}{8}$  inch thick foam-core board, which was then folded up to form a cube approximately 6 inches on each side. The joints of the cube were hot glued together and the entire cube covered in adhesive-backed aluminum tape in order to provide a rigid structure. The interior of the cube was filled with pieces of  $\frac{1}{2}$  inch thick automotive-type sound insulating foam to provide both for thermal insulation as well as to cushion the components against mechanical shock. Voids were left in the foam packaging to provide space for the sensor system, batteries, a small electric heater, and either a video or still camera.

### Preflight Environmental Testing

In order to simulate the extreme conditions encountered by the balloonsat in the near-space environment, as well as verify the functionality of the sensor system, thermal and vacuum tests were conducted prior to the flight. Figure 15 shows the balloonsats stacked in the vacuum chamber ready for testing. Once the door is closed and the chamber is sealed a vacuum pump is started, which evacuates the air from the chamber simulating a “flight” up to an altitude of 100,000 feet. Next the vacuum pump is switched off and a small valve opened to allow the air pressure to return to the ambient state in order to open the door and retrieve the balloonsats.

Next, the balloonsats were placed in a



**Figure 15** — Balloonsat payloads were placed in a vacuum chamber from which the air was evacuated to simulate a flight to 100,000 feet.



**Figure 16** — Balloonsat payloads were placed in a thermal chamber, which was cooled to approximately  $-40^{\circ}\text{C}$  to simulate the temperatures encountered during flight through the troposphere.

thermal chamber in order to simulate the sub-zero temperatures encountered during flight in the troposphere. See Figure 16. Thermal tests were important to verify that the small onboard electric heater had sufficient capacity to maintain the balloonsat's internal temperature such that the LiPo battery voltage did not drop below the minimum voltage necessary to power the sensor system.

During the approximately 1 hour long environmental tests, the sensor system and data logger were enabled in order to record the temperatures and pressures that the balloonsat encountered, as well as to monitor

the voltage of the onboard LiPo battery during testing.

### **Balloon Tracking and Recovery using Position Beacons**

The balloon train includes one or more APRS position beacons, in order to facilitate tracking of the balloon during flight and recovery of the balloonsat payloads after landing. Figure 2 shows one of the trackers. The position beacons are enabled prior to launch to verify their functionality using portable or mobile receivers tuned to the appropriate APRS frequency. Next, the bea-



**Figure 17** — Four students act as “handlers” to stabilize the balloon during inflation. The balloon is tethered to 15 pounds of weights to give an approximate indication of the amount of lift the balloon provides.

cons are inserted into the balloon train along with the other balloonsat payloads, attached to a properly inflated meteorological balloon, and then released for flight. Figure 1 shows the whole system ascending.

In the APRS digital communications protocol digital repeaters (digipeater) receive and rebroadcast APRS packets, while Internet gateway stations (iGates) transfer the packets to the Internet. Once the balloon has ascended to an altitude so that position beacon broadcasts are received by an iGate, the balloon's position and altitude are available to anyone by entering the callsign of the appropriate beacon at <http://aprs.fi>. Additionally, APRS packets are digipeated as long as the altitude of the balloon allows for digipeater reception.

Prior to launch the approximate landing location for the balloonsat payload is predicted based on forecast winds aloft and an

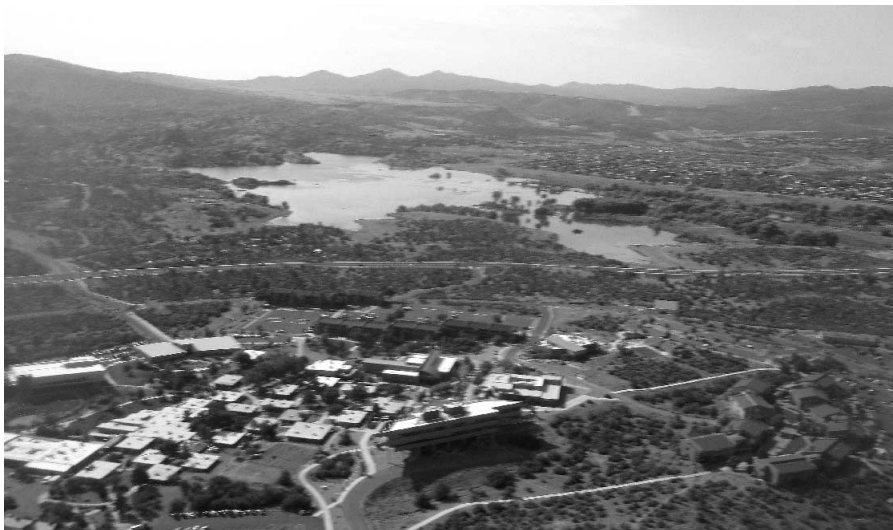
estimate of the balloon burst altitude. Based on this position estimate, one or more mobile recovery teams equipped with GPS enabled APRS receivers are stationed on high terrain in the recovery area as the balloonsat payloads return to the surface under parachute. This is necessary because the final position beacon broadcasts prior to landing are normally terrain masked from the digipeater and are not retransmitted. The goal of the recovery team is to position themselves so that they receive the final position beacon transmissions before landing. This will minimize the search area if the position beacons are disabled by a hard landing or rough terrain or inclement weather prohibits immediate recovery of the payloads before beacon power is exhausted.

Once the payload has landed the recovery teams follow distance and direction information provided by a GPS enabled APRS portable receiver to the final location transmitted by the position beacon. The best-case scenario is that one or more position beacons are still in operation at the landing location and the recovery team is within range to receive these packets containing the actual location of the payload. Payload recovery is easy in this case since the recovery team can literally walk or drive to within a few meters of the actual location of the payloads.

The situation quickly becomes more complicated and recovery of the payload is problematic if one or more of the following occurs: Failure to capture the final position beacon packets by the recovery team, failure of the beacons during flight or after landing, landing in rough terrain or inclement weather, inaccurate forecast of surface winds or winds aloft, separation of one or more of the balloonsats from the balloon train during flight, and many other difficulties that space constraints preclude listing.

### Sample Flight Images and Data

Wednesday morning, June 29, 2011 a high altitude balloon designated ERAU-06 (Embry-Riddle Aeronautical University-06) was launched from the Embry-Riddle campus in Prescott, AZ, shortly after 8:30 AM local time. The balloon payload consisted of the four student-constructed balloonsats, along with three position beacons that periodically transmitted position information using APRS. Figure 17 shows the balloon during inflation while Figure 18 shows an image taken at an altitude of approximately 1000 feet by an onboard camera carried by one of the balloonsats shortly after launch. Figure 19 shows an image of the cloud layer recorded by a camera as the balloon ascends through the stratosphere. Figure 20 shows an image taken by a camera immediately following the “post burst chaos” that ensues



**Figure 18 — This is one of the first images captured after launch. It shows a panoramic view of the Embry-Riddle Prescott campus as well as Willow Lake just east of campus.**



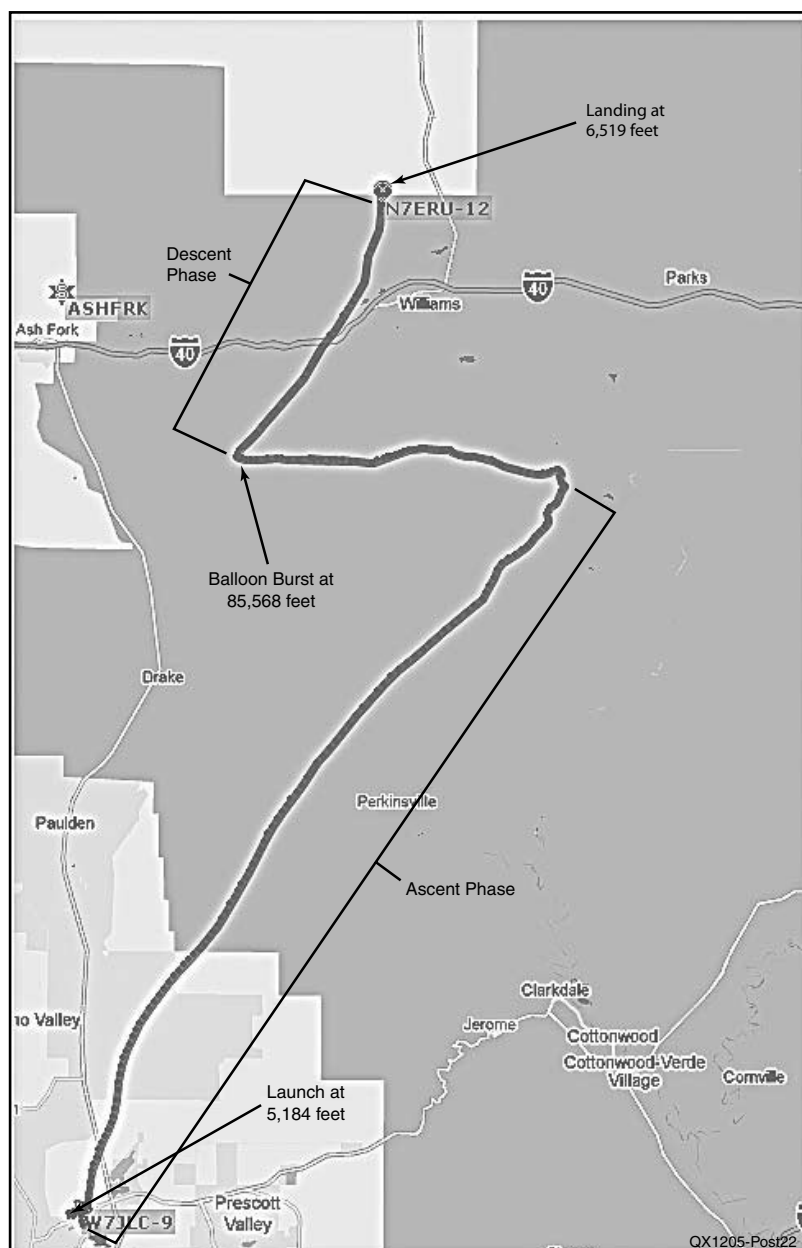
**Figure 19 — Here is a photo taken from high in the stratosphere of clouds over northern Arizona.**



**Figure 20 — This near-space image was captured shortly after the balloon burst at an altitude of approximately 86,000 feet. The curvature of the earth as well as the blackness of space are clearly visible in this image.**



**Figure 21 — During landing the balloon train became entangled in a dead tree. None of the balloonsat payloads sustained much noticeable damage, however.**



**Figure 22 — This two-dimensional depiction of the flight of ERAU-06 on June 29 2011 was generated by plotting GPS data on a Google Earth map. You may view this flight by searching for "N7ERU-12" on 29 June 2011 at <http://aprs.fi>. (See Note 12.) This map shows the launch point near Prescott, Arizona, the burst elevation near 85,568 feet, as well as the landing location approximately five miles north of Williams, Arizona.**

after the balloon bursts.

The balloonsat payloads and position beacons were recovered intact near the final position location, as shown in Figure 21. The sensor systems and cameras were powered down after recording data and images for approximately 4½ hours. All sensor systems and cameras appeared to operate normally and over 1 MB of data, hundreds of still images, and several hours of video images were recovered from the flight.

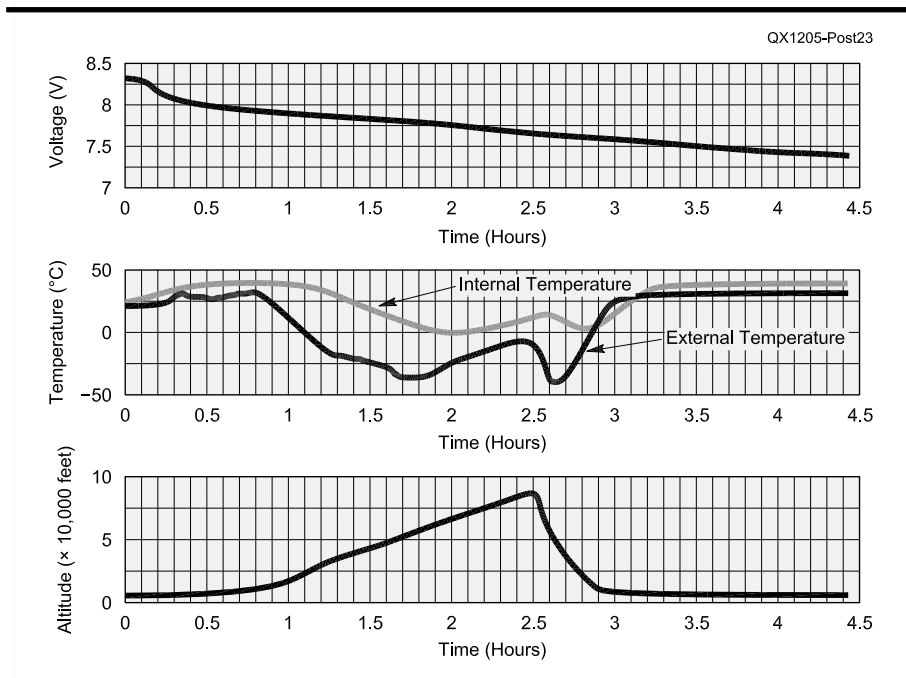
Figure 22 shows a two-dimensional depiction of the flight created by plotting GPS flight data on a Google map.<sup>12</sup> The GPS flight data shows that the balloon ascended through the troposphere in a generally northeasterly direction until easterly winds were encountered in the stratosphere, which directed the balloon almost due west. Once the balloon burst it again drifted in a generally northeasterly direction while parachuting through the troposphere until landing on national forest land approximately 5 miles north of Williams, Arizona.

Figure 23 is an example of the data generated by one balloonsat's sensor systems before, during, and after the flight. The top panel depicts the battery voltage versus time. The flat spot at the beginning of the curve is due to the voltage of a freshly-charged LiPo battery exceeding the maximum of 8.3 V the system was designed to measure. Once the battery voltage falls below 8.3 V, the curve shows a gradual reduction in voltage as the battery discharges.

The middle panel in Figure 23 depicts the internal and external temperatures recorded before, during, and after the flight. Upon power up the two temperatures are almost identical, but the internal temperature quickly shows a steady increase in temperature due to the small electric heater inside the balloonsat container. About 45 minutes after power up, launch occurs and the external temperature decreases steadily to a minimum of about -40°C as the balloonsat ascends through the troposphere. The thermal mass of the balloonsat package, along with the heat produced by the internal heater, delays the decline of the internal temperature and limits its minimum value to approximately 0°C. Once the balloonsat encounters the stratosphere, ambient temperatures begin to increase with increasing altitude until they reach a maximum temperature of approximately 0°C.

Approximately 2½ hours after sensor system power up the balloon bursts and the process is reversed. As the balloonsat parachutes down through the stratosphere temperatures plunge from 0°C to -40°C at the top of the troposphere. After this, temperatures rise steadily while the balloonsat parachutes through the troposphere to the





**Figure 23** — These three graphs represent data recovered from one of the balloonsats during the flight of ERAU-06 on 29 June 2011. The top panel depicts battery voltage versus flight time. The middle panel shows internal and external temperature versus flight time. The bottom panel shows altitude computed from air pressure readings versus flight time. Results from the other three balloonsats were similar to those presented in this figure.

surface of the Earth. After landing, the internal heater, along with the afternoon Arizona sun, overcomes the thermal mass of the balloonsat payload and increases the internal temperature well above the maximum value of approximately 38°C the system was designed to record. This limitation produces the flat spot on the internal temperature curve that is clearly visible after landing.

The bottom panel in Figure 23 shows the altitudes recorded before, during, and after the flight. The launch site elevation was approximately 5000 feet, so this value was used to calibrate the altitudes computed from the measurements that followed. The altitude curve clearly depicts balloon launch at approximately 48 minutes after sensor system power up. After that the balloon ascends at a relatively constant rate of approximately 900 ft/min until burst at a measured altitude of approximately 90,000 feet, about 2½ hours after power-up (compared with a maximum altitude of 85,568 feet reported by the position beacon). The curve then depicts the decrease in altitude as the balloonsat returns by parachute at an average descent rate of 2000 ft/min. The balloonsat payloads landed on the Earth's surface at a measured altitude of approximately 6300 feet (compared with an altitude of 6519 feet reported

by the position beacon). In this case, the altitude measurements taken at burst and landing were within 5% or better of the GPS positions, which seems satisfactory for such a simple system. The results from the other three balloonsats were similar to those presented here.

### Conclusion and Future Opportunities

The data returned from the flight of ERAU-06 validated the sensor system design methodology. Given the modest size and weight requirements for the balloonsat payloads, a low-power, light-weight sensor system with a very small form factor was required. The approach described in this article successfully met these constraints.

Finally, many opportunities exist for further expansion of the sensor system. One of the most interesting possibilities is to incorporate a low-power packet-radio transmitter that would broadcast sensor system measurements periodically during flight. This would provide students with real-time snapshots of the atmospheric conditions and system status during the flight. Successfully introducing a more complex system will require rethinking how to go about the construction and testing phases in order to keep the project within the capability of high school students, the

patience of the instructors, as well as within the time constraints of the program.

### Acknowledgement

The author gratefully acknowledges many helpful discussions with Mr. Jack Crabtree, W7JLC, concerning the topics covered in this article. Jack is the founder of two high-altitude balloon groups, Edge of Space Sciences (EOSS) in Denver, Colorado and Arizona Near Space Research (ANSR).

*John E. Post, KA5GSQ, is an assistant professor of electrical and computer engineering with Embry-Riddle Aeronautical University in Prescott, AZ. He holds an Amateur Extra class license and has BS, MS, and PhD degrees in electrical engineering.*

### Notes

<sup>1</sup>For information about the parachute recovery system used, see: [www.the-rocketman.com/recovery.html](http://www.the-rocketman.com/recovery.html).

<sup>2</sup>Scientific Sales is one source for weather balloons. See [www.scientificsales.com/8244-Weather-Balloon-1200-Grams-Natural-p/8244.htm](http://www.scientificsales.com/8244-Weather-Balloon-1200-Grams-Natural-p/8244.htm).

<sup>3</sup>Sparkfun Electronics has many electronics experimentation projects. For the datalogger board, see: [www.sparkfun.com/products/10216](http://www.sparkfun.com/products/10216).

<sup>4</sup>The data sheet for the LM324 series of op amps is available on the National Semiconductor website. [www.national.com/mpf/LM/LM324.html#Overview](http://www.national.com/mpf/LM/LM324.html#Overview).

<sup>5</sup>Information about the Honeywell pressure sensor is available at: [http://sensing.honeywell.com/index.php/ci\\_id/45330/la\\_id/1/document/1/re\\_id/0](http://sensing.honeywell.com/index.php/ci_id/45330/la_id/1/document/1/re_id/0).

<sup>6</sup>You can find information about the Texas Instruments temperature sensors at: [www.ti.com/lit/ds/symlink/lm135.pdf](http://www.ti.com/lit/ds/symlink/lm135.pdf).

<sup>7</sup>Lithium ion polymer batteries provide a very lightweight power source for the balloonsat electronics. For one source of battery packs, see [www.electrify.com/batteries/batteries-lipo.html](http://www.electrify.com/batteries/batteries-lipo.html).

<sup>8</sup>You can find more information about the voltage regulator used with the sensor electronics package at: [www.onsemi.com/pub\\_link/Collateral/MC78L00A-D.PDF](http://www.onsemi.com/pub_link/Collateral/MC78L00A-D.PDF).

<sup>9</sup>Charles K. Alexander and Matthew N.O. Sadiku, *Fundamentals of Electric Circuits*, 4th Edition, McGraw Hill, Boston, Massachusetts, 2009.

<sup>10</sup>You can download a student version of PSpice at [www.electronics-lab.com/downloads/schematic/013/](http://www.electronics-lab.com/downloads/schematic/013/).

<sup>11</sup>ExpressPCB is a convenient way for electronics experimenters to design and purchase circuit boards. See [www.express-pcb.com](http://www.express-pcb.com).

<sup>12</sup>See <http://aprs.fi/?call=N7ERU-12>. There are tracks available on the map for N7ERU-11, N7ERU-12 and N7ERU-13, which are all APRS beacons used on balloon flights. Along the right side of the screen, select the 2011 tab, then select the 2011/6 date and that will bring you to the track for the June 29th balloon flight.

# Touching Near Space on a Budget

By Paul Verhage, KD4STH (all photo and graphic credits are Paul Verhage unless noted otherwise)

## Introduction

Science, technology, engineering, and mathematics (STEM) education is more important than ever as our current STEM workforce nears retirement and our nation competes with a STEM-educated workforce in other countries. Amateur Radio operators have a history of bringing STEM to the public through Field Days, licensing classes, and radio club activities. Here's another way hams can bring STEM education to the public; helping school clubs and groups like the Scouts launch science experiments into near space. To make near space flights possible, it's necessary to track and recover student experiments and this is the perfect job for an APRS tracker. Below are some directions on how to build an APRS tracker suitable for the harsh conditions found in near space.

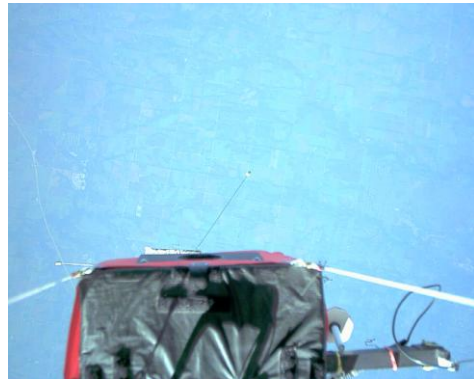


*This is what it looks like at an altitude of 86,000 feet. The distance to the horizon is 360 miles and the sky has the inky blackness of outer space. The horizon becomes noticeably curved at this altitude. The snow-capped volcano is Mt. Bachelor in Oregon and the moon is visible at the upper right.]*

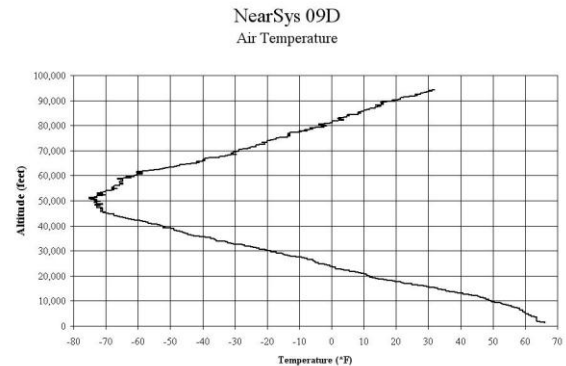
## Near Space

Near space begins at 60,000 feet (the top of controlled air space) and extends to 328,000 feet (the International Aeronautical Federation's boundary for space). Within this region of Earth's atmosphere the sky above turns black, the Earth below turns blue, and the horizon becomes visibly curved. By most measures, the conditions found in near space are much closer to outer space than to Earth's surface.

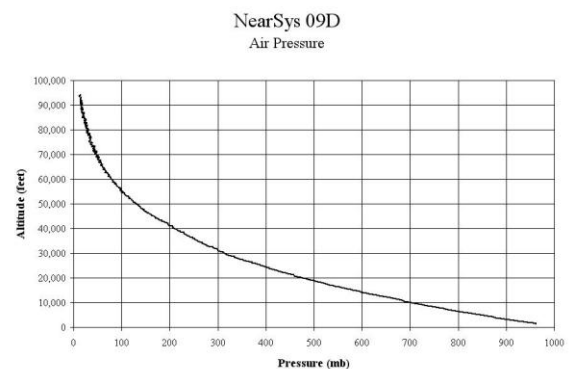
Because of its similarity to outer space, many people feel an excitement launching experiments into near space.



*In near space the blue sky is below your feet and not above your head. Notice that the atmosphere subdues some of the ground color. The scene in this picture is about 13 miles across.]*



*It gets exceedingly cold in near space. Notice however that the air temperature increases with altitude once the weather balloon enters the stratosphere. This is caused by the blockage of dangerous solar ultraviolet by ozone in the stratosphere.*



*Air pressure decreases by 50% for every 18,000-foot increase in altitude. At 100,000 feet, an altitude that a near space balloon mission can attain, the air pressure is around 10 millibars, or 1% of the average air pressure at mean sea level.*



## Near Space Tracker: An Overview

The near space tracker described below makes use of commonly available materials where ever possible. That makes it more affordable for amateur radio clubs to bring student experiments to near space. This article focuses specifically on the airframe, antenna, and batteries. (Additional information is available in other articles on the *ARRL Handbook's* supplemental CD.)

### Items to Purchase

- A reusable lunch cooler (the insulated, soft-sided, zippered type – see photo)
- A sheet of 2" thick urethane foam rubber (available at fabric and craft stores)
- A sheet of 1" thick urethane foam rubber
- A small sheet of Coroplast (corrugated polyethylene plastic available at sign shops)
- 1" wide nylon strap material (available at fabric and craft stores)
- #8-32 hardware (11 screws 1" long, 11 nylon locknuts, and 22 fender washers)
- Eight 1-¼ inch diameter metal split rings



*When not carrying a lunch, this cooler can carry student experiments into near space.]*

## Assembly: Airframe Attachment

### Straps

The attachment straps will connect the near space tracker to the recovery parachute, to a second tracker (the backup unit), and to student experiments.

The straps attach to the lunch cooler with screws, which is much easier than trying to sew them to the cooler.

1. Cut the nylon straps into four pieces, each 12" long (longer if your lunch cooler is tall)
2. Fold each strap with a 1" overlap as shown below



*This nylon strap is shown clamped with this butterfly clip to illustrate the folds and overlap in the strap.]*

3. Melt two holes into each strap with an old soldering iron; do not breathe the fumes or smoke. The holes need to be large enough for #8 screws.
4. Place each strap next to a different corner of the lunch cooler and melt holes into the lunch cooler that correspond to the holes in the straps.
5. Attach the straps to the lunch cooler using #8 hardware (place fender washers inside and outside the cooler and the locknuts on the outside).



*Two screws, fender washers, and nylon locknuts are more than enough to attach the straps to this lunch cooler.*

## Assembly: Antenna Boom

The Antenna Boom is a stiff and lightweight platform for holding the antenna to the cooler. And it's easy to replace in case it's ever damaged at landing.

1. Cut a Coroplast strip 18" by 2.5" aligned lengthwise with the cells of the Coroplast for strength
2. Drill three holes in the Coroplast
3. Place the Coroplast against the back of the lunch cooler and melt three holes into the lunch cooler matching the holes in the Coroplast
4. Attach the Coroplast to the lunch cooler in the same way that the straps were attached to the cooler.



*The antenna boom is very lightweight, so three small screws are sufficient to hold it into place (the five screws in this image are overkill).*

## Assembly: Bottom Cushion

A sheet of foam rubber in the bottom of the lunch cooler cushions the tracking electronics at landing. It also creates a layer of insulation to protect them from the cold.

1. Cut a piece of the 2" thick foam rubber so its fits snugly into the bottom of the lunch cooler.
2. Place the foam rubber into the bottom of the lunch cooler.
3. Place the flight battery on the foam rubber and draw and then cut out a pocket in the foam rubber to hold the battery snugly in the foam rubber.

## Assembly: Avionics Deck

The electronic modules must be held together rigidly or they're liable to break apart, especially when the balloon bursts. The Avionics Deck creates a

lightweight method for keeping the electronics connected together, to the battery, and to the antenna.



*This near space tracker is one of my kits. You can also use a TNC and HT.*

1. Cut a piece of Coroplast so it fits tightly into the lunch cooler and on top of the Bottom Cushion.
2. Mount the flight computer or TNC/radio to the Coroplast avionics deck (use screws, washers, and locknuts – in a pinch you can use nylon wire ties)

## Assembly: Top Cushion

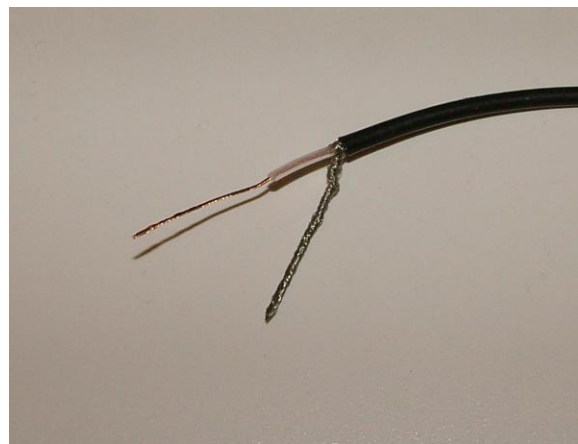
The GPS can't be allowed to flip upside down during the mission or you'll lose track of the near spacecraft. To hold it steady, it is sandwiched between a sheet of foam rubber and the top of the lunch cooler. Sandwiching it like this also gives the receiver a little more insulation from the cold.

1. Cut a piece of the 1" thick foam rubber so its fits snugly into the lunch cooler (use a thicker sheet if your lunch cooler is tall).
2. Place this foam rubber on top of the Avionics Deck and trim it to fit around the items mounted to the avionics deck (if necessary)
3. Place the GPS receiver on top of the Top Cushion and then close and zip the lid of the lunch cooler.

## Assembly: Antenna

The antenna for the near space tracker is a simple dipole antenna. The balloon rotates along its vertical axis during its flight and will constantly change the orientation of the antenna with respect to chase crews on the ground. A dipole is fairly immune to this and except for the null directly beneath, allows for continuous radio reception. The null is not a significant issue since chase crews aren't directly below the balloon for long and there's almost always a digipeater some distance away that's unaffected by the antenna null. The dipole is very lightweight, as well. You'll need the following materials to make the antenna.

- Two foot long coax cable, like RG-174
  - #12 AWG solid copper wire
  - Perforated board
  - Three 1/8" nylon wire clamps
  - Seven pairs of nylon screws and nuts (#10-24 3/4" or 1" long, for example)
1. Cut off one of the SMA connectors from the cable (this is the end of the coax that will attach to the dipole elements).
  2. Slide a 1" piece of heat shrink tubing over the coax
  3. Strip about 1" of the outer jacket off the coax to expose its braid.
  4. Remove approximately half of the exposed inner insulation to reveal the inner conductor (leave some insulation remaining between the inner conductor and outer braid).
  5. Push the inner conductor and its insulation through the braid.
  6. Twist the braid tightly.



*The stripped end of the coax.*

7. Cut a 2" square piece of perforated board.
8. Solder both coax conductors (the twisted braid and inner conductor) to two different pads on the perforated board (you'll need to expand the diameter of one of the holes in the perforated board for the twisted braid).
9. Slide the heat shrink tubing to where a nylon wire clamp will wrap around the coax.
10. Clamp the coax to the perforated board with a nylon wire clamp and bolt (the clamp provide a strain relief for the coax).
11. Cut two pieces of #12 AWG solid wire to a length of 20 inches. Each of these makes up one-half of the antenna.

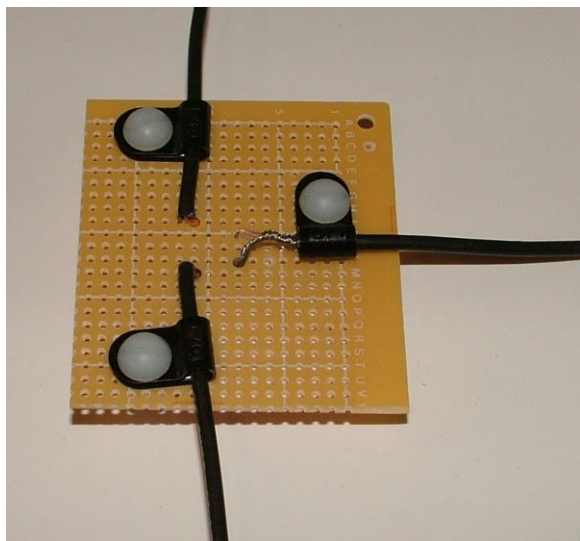
Note: these directions assume you're building a 2 meter tracker. The proper length of the wires depends on the frequency of the transmitter and can be calculated with the equation,  $L \text{ (feet)} = 485/\text{frequency (MHz)}$ . You'll need to divide the result by two for the correct length of each antenna half.

12. Strip off ¼" of insulation off of one end of each #12 AWG wire
13. Bend the short bare end in a right angle.



*Very little of the #12 AWG wire needs to be stripped and bent. This is where it will be soldered to the perforated board.*

14. Solder the bare ends of the #12 AWG wire to the perforated board so that each half of the dipole electrically connects to one lead of the coax (you'll need to expand the diameter of the holes in the perforated board).
15. Slide a 1" piece of heat shrink tubing onto the antenna halves.
16. Slide the heat shrink tubing to where a nylon wire clamp will wrap around the antenna halves.
17. Clamp each half of the antenna to the perforated board with a nylon wire clamp and bolt (provides strain relief for the element).



*The finished antenna.*

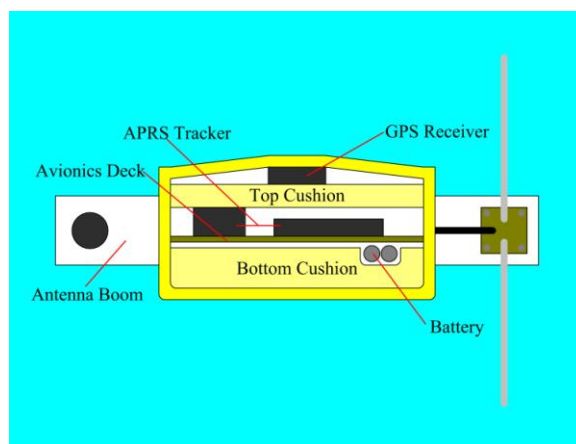
18. Drill or enlarge a hole in each corner of the perforated board.

19. Place the perforated board against the Antenna Boom and drill four holes into the Correplast that match the holes in the perforated board.
20. Bolt the perforated board to the Antenna Boom using nylon locknuts, washers, and bolts.



*The antenna after it's been attached to the Antenna Boom. This is a circuit board of my design that I use for making near space dipole antenna.*

21. Use an old soldering iron to melt a hole through the side of the lunch cooler where the antenna's coax will enter the lunch cooler and connect to the radio on the Avionics Deck.
22. Use another nylon wire clamp to secure the antenna coax to the Antenna Boom.



*An x-ray view of an idealized Near Space Tracker. It's essentially a layer cake design.*



## Batteries

Since it gets bitterly cold in near space, lithium-based batteries are strongly recommended as the power source as they tend to handle the cold temperatures without dropping their voltage. A practical source of lithium batteries is an RC hobby shop. Racing car batteries come in several voltages and capacities. Since many APRS trackers are designed to operate on five volts, select a two- or three-cell battery pack. If your tracker electronics has a low dropout voltage regulator, then your tracker can operate with a two-cell battery pack.

Not only do you have a choice of battery voltages with RC racing car batteries, you also have a choice of capacities. So measure the current draw of your APRS tracker and multiply that value by 24 hours. That will give a rough idea of a good battery capacity for your tracker. Even though missions only last three hours from power up to shut down, it's nice to have the extra reserve to make sure the tracker produces position reports overnight should it get lost during descent.



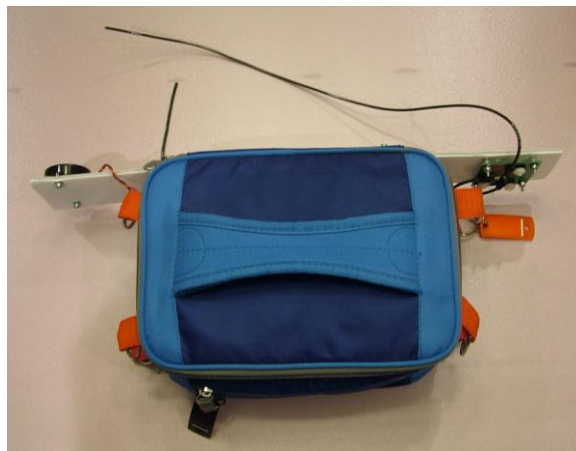
*The battery on the left is a 2-cell, 25C, 2200 mAh rechargeable lithium battery. The battery's original connector was replaced with an Anderson PowerPole connector. The 9 V battery is for an audio beacon mounted to the Antenna Boom.*

## Ready to Launch!

This completes the near space tracker. You'll need to find a parachute before you can launch it, but that's relatively easy to do. Afterwards, you'll need to learn how to fill a weather balloon and how to predict a near space flight. Fortunately, these steps are very easy to learn. You will need a balloon filler — directions for assembling one can be found in my online book, *Near Space with the BASIC Stamp* ([nearsys.com/pubs/book/index.htm](http://nearsys.com/pubs/book/index.htm)).

You should practice with a couple of near space flights before offering to carry student experiments. The student experiments are often built into modules called BalloonSats. You'll be able to attach six or more BalloonSats (depending on their weight) to your tracker, parachute, and balloon.

Since you're providing the launch and tracking services, students won't need to add tracking electronics to their BalloonSat or need to earn an Amateur Radio license just to fly an experiment. Your help will let students concentrate on their science experiment and the analysis of its data.



*The completed near space tracker as seen from the top. Inside this reusable lunch cooler are an APRS tracker and GPS receiver. The left side of the Antenna Boom is carrying an 80 dB piezo alarm that will make locating the tracker much easier.*