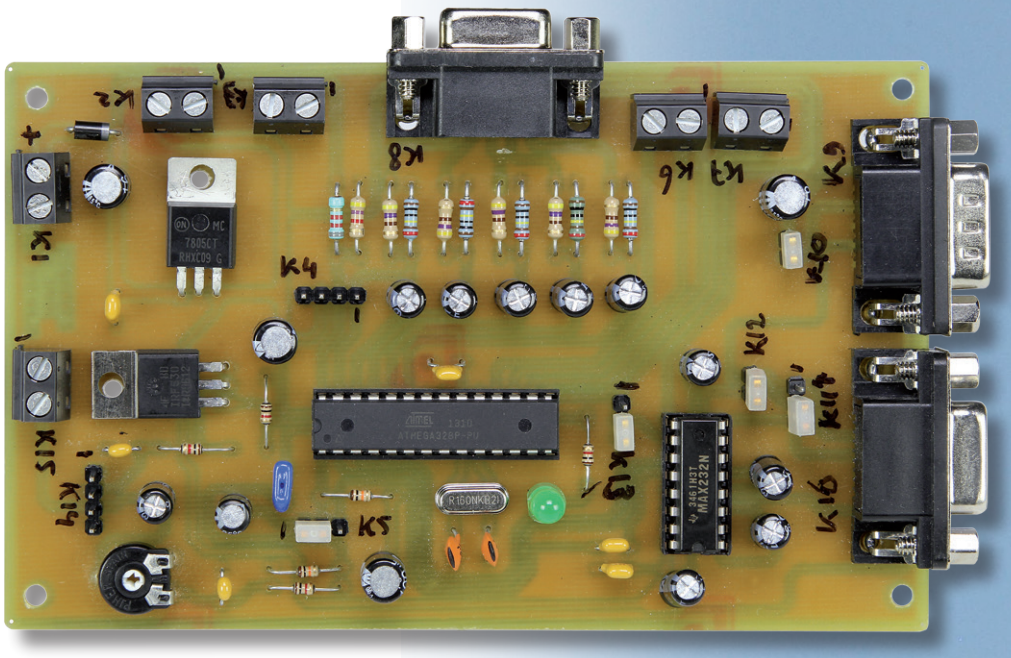
Cloud Nine and Higher

Multi-Purpose Data Acquisition Board for a Sounding Balloon

This story is about clouds of the water dropping variety. Transmitting data from a capsule dangling from a weather or sounding balloon is not easy as it often involves expensive and sophisticated equipment. Luckily, cheap alternatives exist. The board presented in this article together with a standard radio transmitter and GPS receiver can be used to beam down up to six analog signals and position data from a balloon, and all of that at very low cost. Our heads are not the clouds either—this project has been up in the air twice already!



By **Anthony Le Cren**
(France)

Features

- Arduino based software
- Five analogue inputs on 9-pin sub-d connector (the sixth input monitors the battery voltage)
- A 9-pin sub-d connector for GPS (NMEA0183A at 4800 baud)
- RS-232 port for in-circuit programming of the microcontroller (with Arduino bootloader)
- AFSK and PTT output to the transmitter
- Uses APRS/AX25 protocol (packet radio)
- Four user inputs/outputs
- 5 V regulated power supply
- Compatible with Byonics GPS4
- Compatible with Baofeng UV-3R or UV-5R two-way VHF radio

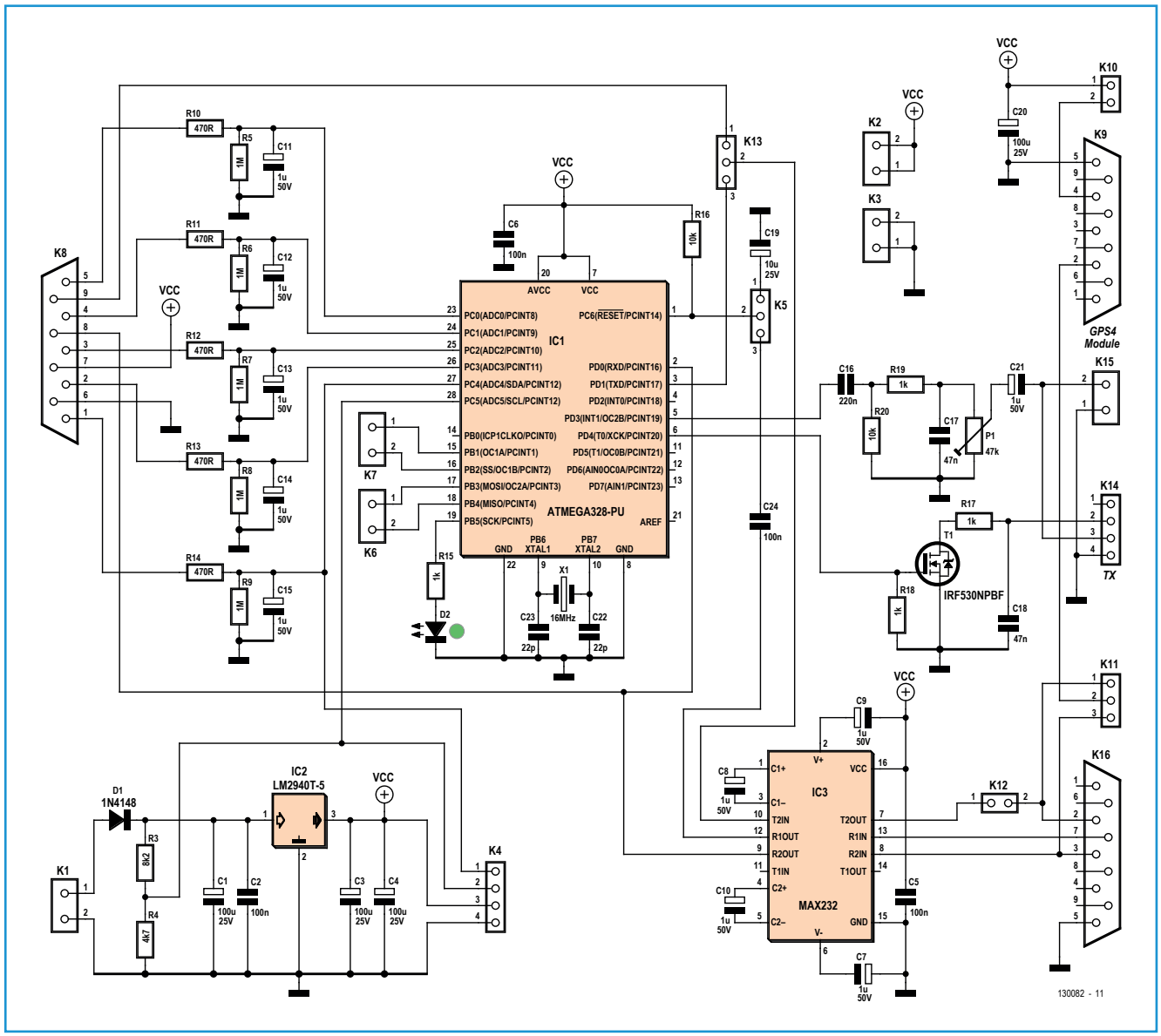


Figure 1.
Circuit diagram of the sounding balloon data acquisition board.

What's on board?

The system pictured in **Figure 1** is built around an ATmega328P microcontroller (MCU) ticking at 16 MHz. The software is based on the Trackuino open-source Arduino project [1]. Even though many sensors can be connected to the board, there is plenty of space left for extensions. First, we will go through the different parts of the design, looking at the circuit diagram in Figure 1 and the prototype of the author, shown in **Figures 2** and **3**.

Analog inputs

Five analog inputs intended for the connection of sensors (temperature, pressure, etc.) are available on K8, a 9-pin sub-D connector (Figure 2, top). After low-pass filtering, the signals are converted into digital values by the analog-to-digital converter (ADC) inside the MCU, and accessible on Port C. The sixth ADC input is used for measuring the battery voltage. Voltage divider R3/R4 reduces the 9 volts battery voltage to a safe level for MCU. A bit further on we'll provide some examples of how to read a pressure sensor and a temperature sensor by means of these inputs.

GPS

The GPS unit is connected to K9 (Figure 2, sub-D connector in top right). Its output signal

is available on pin 2 of K11. It's an RS-232 compatible signal—hence it's passed through IC3, the classic MAX232, before feeding it to the MCU. The purpose of K11 is to direct the GPS signal to the microcontroller (position 2-3), or to the TX pin of K16 (position 1-2). This allows you to view the NMEA0183A "sentences" on a conventional serial terminal. In this case, do not forget to remove jumper K12, or the TX2 output from IC3 may interfere with the GPS signal.

The GPS unit needs 5 V to work; pin 4 of K9 does the job. The supply voltage can be interrupted by removing the jumper on K10, which is useful when a PC running a simulator like SIM GPS is being used instead of a real GPS. A GPS simulator facilitates the development of your program, and allows you to check the correct operation of the system before the final launch. Connect the board and the PC with a standard null modem (crossover) cable.

VHF radio

Let's move on to the packet radio interface. The author suggests using a Baofeng UV-5R unit (Figure 3, left) or a UV-3R two-way VHF radio for this. Both have a double jack connector (**Figure 4**). Only three signals are used, connected to K14: PTT (push to talk) (pin 2), MIC+ (pin 3) and GND (pin 4).

The radio uses Frequency Shift Keying (FSK) modulation to transmit data. Consequently the audio signal alternates between two frequencies (1200 Hz and 2200Hz) in order to transmit a binary value. Because the MCU does not have a digital-to-analog converter (DAC), a pulsewidth modulated (PWM) signal is used instead. A band-pass filter consisting of a first-order high-pass and low-pass filter in series, transform this PWM signal into a quasi-sinusoidal signal. The output level can be adjusted with potentiometer P1 before feeding it to the MIC input of the VHF transmitter. The PTT virtual 'switch' effectively puts the radio in Transmit mode. An input is available on the radio so the MCU can 'push' the switch too. The PTT on port PD4 of the MCU drives an IRF530 MOS transistor, creating an open-drain output that should be compatible with the majority of VHF radios. LED D2 will light when PTT is High, showing that there's a transmission in progress.

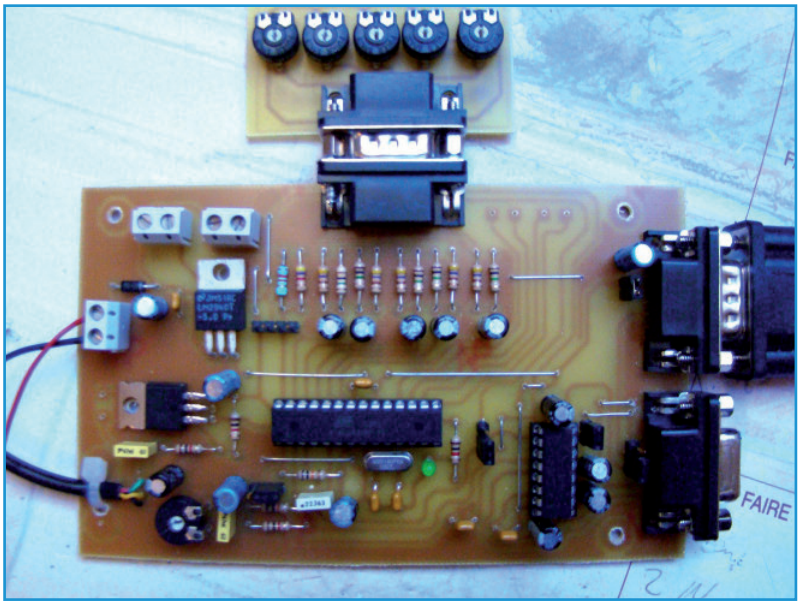


Figure 2.
The populated board as used by the author.

Power supply

The power supply is a traditional design. Note however that the use of the classic 7805 voltage regulator for IC2 is not recommended because of its huge minimum voltage drop of about 3 V. With diode D1 the battery voltage line at least 9 V would be required. Using a low-dropout (LDO) regulator like the LM2940T-5, it is possible to power the board from as low as 6 V (minimum). 9–6 = 3 volts doesn't seem like a lot, but the battery voltage drops appreciably with freezing temperatures

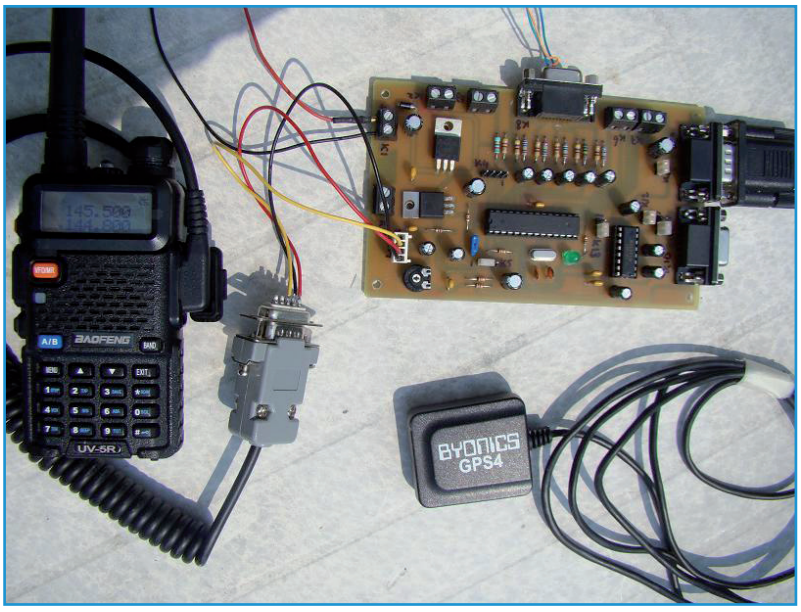


Figure 3.
Complete setup of the system, including the board, the VHF radio and the GPS.

at high altitudes. To be safe for the duration of the flight we should use an 8 to 9 V supply as a minimum. You can make one by connecting six 1.5 V batteries in series. The total current consumption of the board is below 60 mA, including the GPS unit.

Programming that ATmega

Thanks to connector K16 the Atmega microcontroller can be programmed without removing it from the board. To do so, connect the board with a one-on-one (straight trough) cable to a PC with a real RS-232 port (you still have one, do you?). IC3, a classic MAX232, adapts the RS-232 signal levels to those used by the MCU. The TX and RX signals are on pins 2 and 3 of the MCU. Pinheader K5 lets you select the source of the Reset signal for the MCU. A jumper on 2 & 3 allows the PC to force a reset via the RTS pin of K16, which will launch the MCU's bootloader (the Arduino IDE will do this for you).
The firmware is available via the project page at Elektor**.**LABS [2], together with the rest of the files. Once the MCU is programmed, fit the jumper on pins 1 & 2 to avoid accidental resets. The GPS unit that shares the same serial port must be disconnected when you program the MCU. You can do this by unplugging it, or by removing the jumper on K11. **Table 1** gives an overview of the jumper settings for every case.

Extensions

In case there are further circuits on board the capsule, connectors K2 and K3 can provide the 5 V supply voltage. Extension connectors K6 and K7 are available for switching things during the flight. Please note that it is up to you to adapt the firmware to add such functions. Connector K4 got added for those who want to use sensors that communicate over an I²C bus, connected to ports PC4 and PC5 of the microcontroller. Doing so will however cost you two analog inputs (channels 4 and 5). Finally there is also K13. With a jumper on its pins 1 & 2, you create a serial output on pin 9 of the analog extension port K8. A serial input is available on pin 8 of this connector. You can now communicate with a PC through K8. If you put the jumper on pins 2 & 3 of K13 the serial port on K8 will be connected directly to the MCU. This extra port may prove useful in case you decide to run the serial line to an external peripheral, like a second microcontroller!

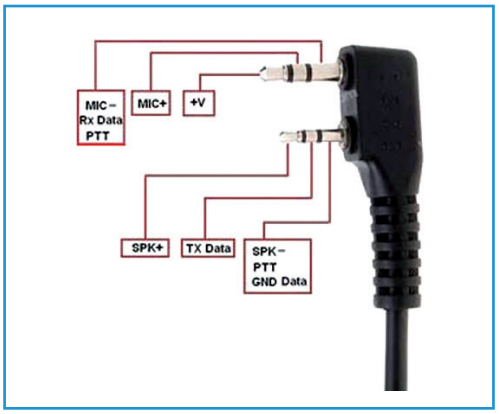


Figure 4.
Double jack connector of the Baofeng UV-5R VHF radio. Only three of the six signals are used here.

Built to fly

Assembling the board should not pose serious problems. A revised version is available at Elektor**.**LABS [2] in the form of Gerber files, PDFs and a DesignSpark PCB 5 project. As usual, solder carefully, and mount the components from smallest to largest. All are mounted at the top side of the PCB.
The board was designed for a GPS4 unit from Byonics (which may be seen in Figure 3, bottom right). If you want to use another GPS unit, you may have to adapt the connections. Make sure that the output signal of your GPS is RS-232 compatible and that the baud rate is 4800 bits/s.

Table 1. Jumper settings

	MCU programming using the Arduino IDE	Launch simulation	View GPS4 output	Launch of balloon
K5	2-3	1-2	1-2	1-2
K13	2-3	2-3	2-3	2-3
K12	Closed	Closed	Open	Closed
K11	Open	2-3	1-2	2-3
K10	Don't care	Open	Closed	Closed

The same goes for the VHF transmitter. If you want to use a model other than the Baofeng UV-5R or UV-3R, you may have to adapt the connections. It is important that your transmitter is type approved and has a PTT control input and an audio (microphone) input for the modulated signal. Typically these are jack connectors. You can test the board's radio output with a simple walkie-talkie or with a 433 MHz module that has an analog input.

Examples of analog sensors

The five inputs on connector K8 are analog and accept voltages between 0 and 5 V. The MCU's ADC is configured in 8-bit mode, producing output values from 0 to 255.

Pressure sensor

The pressure sensor is easy to connect to the board. Simply wire up the 5 V supply voltage, and connect the output of the sensor to one of the analog inputs on K8. This sensor is quite easy to use, but beyond a certain altitude, when the atmospheric pressure drops below 100 mbar, it is no longer sensitive enough. Its output remains constant where it should keep decreasing, when the balloon continues to ascend.

Temperature sensor

It is always interesting to measure temperature outside as well as inside the capsule (**Figure 5**). This can be done easily with a simple voltage divider where one of the two resistors has an NTC resistor (Negative Temperature Coefficient). These resistors are also known as thermistors. The resistor values are not critical as long as the divider produces a signal that nicely covers the range of interest. The author got good results with a 10 kΩ NTC resistor connected in series with a 36 kΩ fixed-value resistor.

Software

The software embedded in the capsule is similar to the seven-layer OSI model. Although there are many differences, we can distinguish several layers: Application → APRS → AX25 → AFSK modulation → VHF transmission.
The application layer corresponds to the analog inputs and to the acquisition and decoding of the GPS NMEA0183A data. The data obtained is put in a frame, in accordance with the Automatic Packet Reporting System (APRS) protocol [3]. Next a header corresponding to the AX25 protocol [4] is added to the APRS frame.
But that's not all. The bits of the data frame must modulate a carrier to make the data compatible with the VHF transmitter. This is the role of the Audio Frequency-Shift Keying (AFSK) modulator. It transforms the AX25 ASCII string into a series of tone bursts with frequencies of 1,200 Hz and 2,200 Hz at a speed of 1,200 bits/s. If the frequency changes between two consecutive bursts, a logic 0 is transmitted. If on the contrary the frequency remains the same, a logic 1 is transmitted. **Figure 6** provides an example of how the transmission is carried out.
The software is compatible with the current version of Arduino (1.0.5). Be sure to change the balloon's callsign in the file 'config.h':

```
#define S_CALLSIGN  "F6XXX"
```

The easiest way to get a valid callsign is to

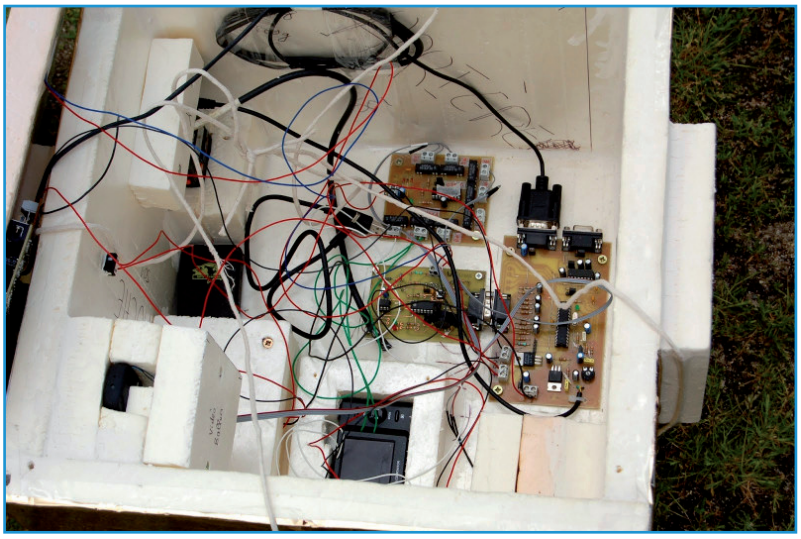


Figure 5.
A look inside the capsule: there's enough room for everything!

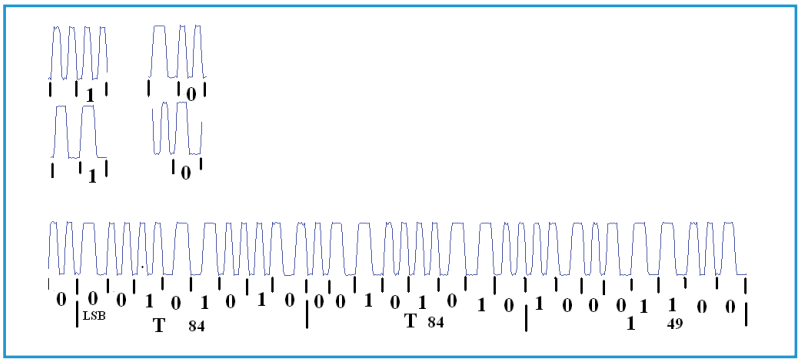


Figure 6.
Example of an AFSK transmission.

contact the radio club nearest to you [5].
The code that implements the AX25 protocol is found in the file 'aprs.cpp'. The function `ax25_send_string` allows you to prepare the AX25 frame before sending it using the function `ax25_flush_frame`. After sending the frame you can add your own code to command the devices that you may have connected to K6 and K7.

Setting up the reception chain

Receiving an AX25 frame from the balloon cane b done with a suitably tuned VHF-UHF receiver or scanner. Decoding of the frame is done by a PC by connecting the audio output from the scanner to the line input on the PC.
The following software must be installed:

- AGWPE: frame decoding;
- AGWTrackerXP: frame display and the rough location of the balloon on a map;
- Packet Engine: display the measured data.

These three may be downloaded for free at [6] too.
After installing AGWPE, restart your PC, and then run the program and configure a communication port, as shown in **Figure 7**.
Now install AGWTrackerXP. Enter the callsign as the receiving station. A TCP/IP connection should be created automatically between AGWPE and AGWTrackerXP.
The last software to install is Packet Engine. This tool saves the received data in a text file for a further analysis with a spread sheet. A screenshot is depicted in **Figure 8**. Let's have a look at one of these long strings. Here it's easy to identify the AX25 header and the APRS frame (which includes the hour, latitude, longitude, etc.). The AX25 header ends with a vertical black bar (|), and the APRS frame right after the word "*Ballon*" and two bars (||).

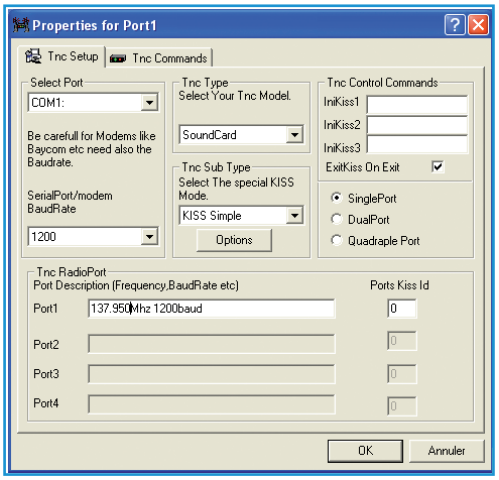


Figure 7.
Port configuration in AGWPE.

I don't wanna lose you

It should not be too difficult to track your balloon during most of the flight. However, when the balloon drops to an altitude below 2,000 m (6,000 ft.), it is common to lose the radio signal.
To ensure you can recover the capsule, it is wise to add a GSM/GPRS/GPS tracker module to the capsule. Such a device adds little weight and, equipped with a SIM card, will send text messages containing its latitude and longitude to a predefined smartphone.
Cellphone (GSM) repeater antennas are slightly oriented towards the ground meaning that the tracker will lose its network at altitudes from about 700 m (2,000 ft.). Fortunately it will wake up again as soon as it detects a network, which is when the capsule is getting close to the ground and eventually lands.

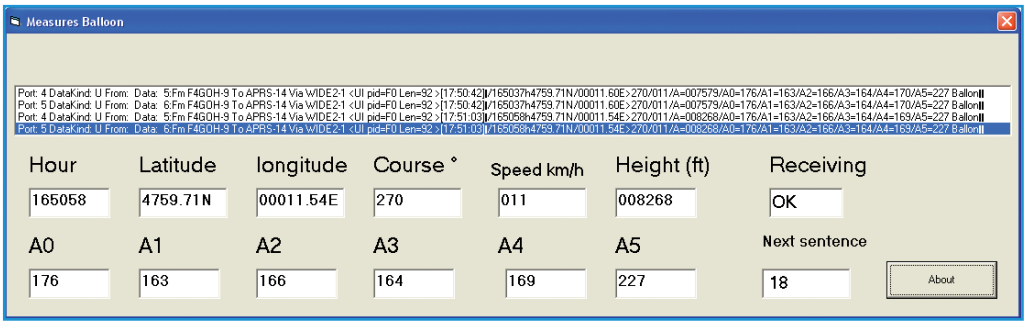


Figure 8.
Screenshot of Packet Engine, showing some received data.

Sounding off

Regulations concerning weather and sounding balloons differ per country. For France, if you have a partnership with Planète Sciences, the capsule must measure at least 30 cm (1 foot) and the transmission frequency must be 137.950 MHz.

Helium gas is quite expensive, which is why many enthusiasts build a hot air or a solar balloon. As opposed to helium-filled balloons, they do not explode and will travel hundreds of miles. For shooting photos and video, there are several solutions, selected on cost basis. The author used two types of camera—one HD, and one "keychain spy". The first provides images of very good quality, whereas the second gives good results at modest costs. Just to give an example of the measuring results you may get, the minimum outside temperature registered during the author's experiment was about –28 °C at an altitude of 15,000 m (45,000 ft.), when it was expected to be around –56 °C at 10,000 m (30,00 ft.) altitude. The sensor must have been influenced by the heat emanating from the balloon's capsule. When the balloon enters the stratosphere, the temperature rises again. The temperature inside the capsule was found to follow the same tendency. **Figure 9** shows a graph with these results.

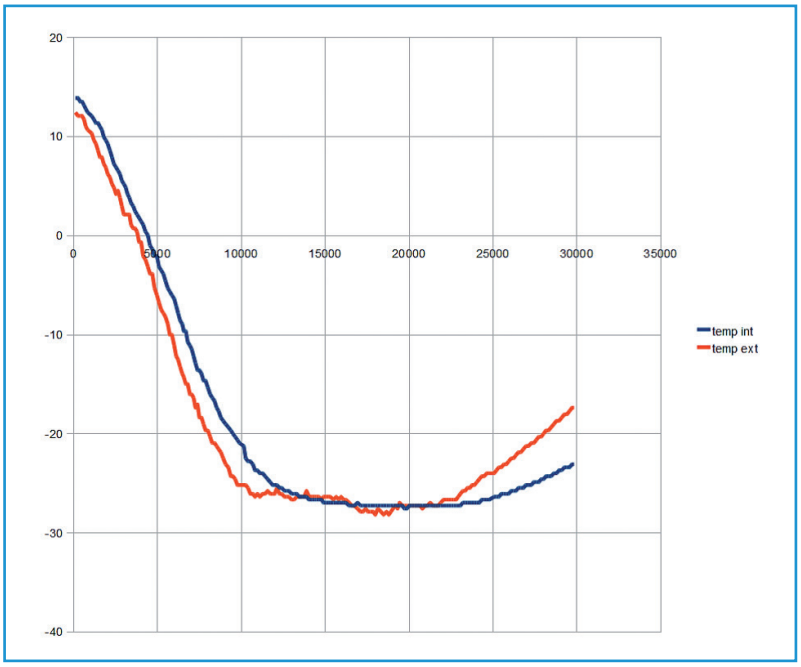


Figure 9. Temperature registered during the test flight. It must be influenced by the capsule's heat, as it was expected to be almost 30°C lower.

Checklist

Preparation

- On the PC, launch the software in this order:
  - AGWPE
  - AGWTrackerXP
  - Packet Engine
- Switch on the VHF receiver (tuned to the transmitter)
- Do not connect the VHF receiver to the PC yet

GPS simulation with SimGPS

- Check the jumpers on the board
- Connect connector K9 to the PC via a cross-wired cable with a 9-pin female sub-d connector on each end
- Launch the NMEA simulator, start sending frames
- Turn on the VHF transmitter
- Power the board with 8 to 9 V
- The LED D2 will light when a frame is sent

Reception

- Incoming frames should be heard on the receiver
- If this is the case connect the receiver to the PC
- You should see the frame appear in AGWTrackerXP
- To see the balloon move over the map, right click on its call sign, then click on 'Locate' followed by 'Show on map'.

Regarding the altitude measured by the GPS unit, from **Figure 10** you may notice the very linear first phase when the balloon rises with a speed between 4 m/s and 6 m/s depending on the volume of the helium in the balloon. The balloon burst at 30,000 m (90,000 ft), which explains the very rapid descent.
The system described in this article was developed in a college environment. The author got great pleasure from building the system with his students. Evidently, the pressure on everybody increases when the launch date is fast approaching, so a checklist must be prepared carefully and gone through several times, because there is only one balloon. Once the balloon is launched, it is great fun to observe its *voyage* in real time, and decode the data it sends down. Then, relying on weather forecasts it's time to start worrying about the landing position of the capsule. When the balloon explodes, the hunt to recover the capsule starts. Although it is easy to locate the capsule with a tracker, it may have landed high up in a tree or even in water. Once the capsule is safely recovered, it's gratifying again to look at the pictures and videos taken during the flight. An outstanding set of pictures was already presented in Elektor**.**POST #20, but in case you missed that, you may find it here [7]. The last picture represents the complete route of the balloon, departing from Le Mans (France). It's definitely worth checking!

(130082)

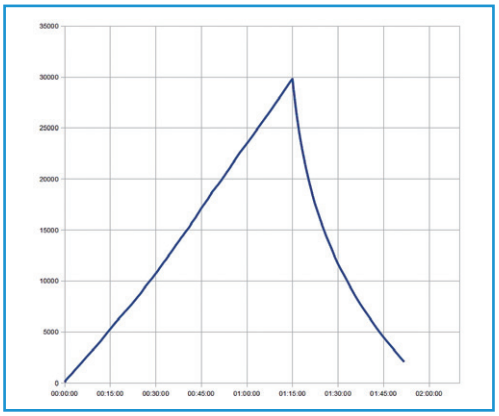


Figure 10.
Altitude registered by the GPS. The balloon burst at 30,000 m, which can be clearly seen here.

Internet Links

[1] www.trackuino.org

[2] www.elektor-labs.com/130082

[3] http://en.wikipedia.org/wiki/Automatic_Packet_Reporting_System

[4] www.tapr.org/pub_ax25.html

[5] http://en.wikipedia.org/wiki/List_of_amateur_radio_organizations

[6] www.sv2agw.com/downloads/default.htm

[7] http://bit.ly/17x8dlm