Multi-purpose GPS

For those who are keen to know everything!

Christian Tavernier (France)

Although satnav systems have been in our cars for some years now, it's only recently that stand-alone GPSs have appeared aimed at the amateur market. Despite their small size and often very affordable prices, they have nothing to envy their predecessors, and if we team them up with suitable electronics, they can perform our positioning or navigation tasks just as accurately as the manufactured products.

Although these receivers are relatively simple to use, Internet research shows that many amateur electronics or robotics enthusiasts don't know how to set about exploiting these little technological jewels. So in this article we're suggesting three different projects where you can learn how to use these receivers.

Totally standardized dialoguing – or almost!

Historically, the first GPS information was used by mariners, and Naval ones at that. No surprise then that the GPS should have adopted the NMEA (National Marine Electronics Association) communication protocol, originally designed to facilitate data exchange between the various instruments on board, long before GPS even existed.

If you only want to build the GPS receivers described here, the subtleties of this protocol aren't of any great importance to you. However, if you want to modify these receivers or perhaps build a GPS receiver into your robot, we invite you to read the box about it.

Then you'll be able to see that it's still relatively simple... well, almost!
In fact, although all current GPS receivers do indeed respect the hardware part of the protocol, along with the basic character format, during our experimenting we noticed that certain receivers do take a few liberties with the frame content. The format of certain data – for example, altitude or ground speed – can actually vary from one device to another, making decoding the data a bit more complicated (pun intended). We'll come back to this later.

The receiver chosen for these projects

For these projects, we've chosen the EM-406A receiver, made in South-east Asia (where else, these days?) by Globalsat Technology Corporation [1] and available from Lextronic [2], amongst others.

This receiver is very compact at only 30 × 30 × 10.5 mm yet offers characteristics that will be the envy of certain car model manufactures. Fitted with an SiRF Star III, it achieves a remarkable sensitivity of –159 dBm – to the point that the author was able to acquire 8 satellites even with it shut in a metal desk drawer located inside a reinforced concrete building.
Powered from a voltage between 4.5 and 6.5 V, the GPS receiver draws just under 45 mA. The only slight niggles in this fanfare of praise are the technical documentation – which, although it gives the appearance of being comprehensive, is in fact inaccurate, particularly in terms of the NMEA frames format; its output level – described as 'TTL', it isn't really, which may pose problems of interfacing with certain microcontrollers, as we'll be seeing further on; and finally, the format of certain data in its NMEA frames, which is a little… 'creative'!

Despite this, the EM-406A is still an excellent choice, as you're about to see. However, as all GPS receivers output the NMEA frames used by these projects, there's nothing to stop you choosing another model of receiver, or even using your car satnav, provided its data output is accessible – and that's usually the case with the better models.

Receiver

A PC/GPS interface

The first project on offer is extremely simple, and makes it possible to interface the GPS receiver with the RS-232 serial port of any computer equipment, and hence, a PC. It will enable you to use your PC for navigation using different pieces of software – and not neces-

GPS NMEA frames

The NMEA standard was created to facilitate the exchange of data between electronic navigation equipment. So it's only to be expected that the GPSs use it, since the first users of these devices were mariners.

The standard used by GPS receivers is NMEA version 0183, which states that data transmission occurs in asynchronous serial form with 8 data bits, 1 stop bit, and no parity, at a speed of 4,800 b/s. This standard operates by sending messages or frames, all organized in the following way:

A frame always starts with the character $ (dollar) followed by two letters that identify the type of sender (GP in the case of a GPS, but you might also find EP for a position beacon, HC for a compass, etc.). Then come three other letters identifying the type of frame for the sender concerned. In the case of GPSs, the RMC frame is the most common (for the simplest devices, the only one even), but you do also find GGL, GGA, GSA, etc.

Then come the various data fields, each separated by commas. The order and size of these fields depend on the frame involved and so there is no common rule for all the frames. The * (asterisk) character immediately follows the last data (i.e. with no separating comma) and precedes a 2-character checksum, calculated by performing an exclusive OR on all the message characters from $ to * inclusive. So here, for example, is what an RMC frame provided by a GPS looks like:

$GPRMC,114630.325,A,4315.0426,N,00603.2734,W,0.20,307.53,310508,,E*54

If one data element is missing from a field, the place for it must be present all the same, so you may find two successive commas, as shown at the end of the example above. The various frames arrive in an order not defined in the standard, since they are in any case clearly identifiable by their header (the five letters following the $ character).

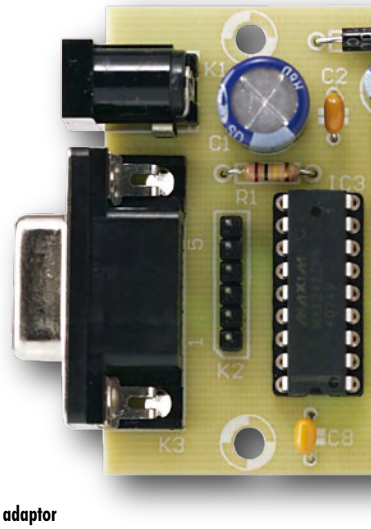
The standard defines more than 30 frames just for GPSs alone. So it's quite out of the question for us to reproduce them all here, especially since the majority of GPS receivers provide only a few of them. So we're offering you just the two most useful frames in the two tables here. You can find out all about the others in the instructions for your GPS, if it provides them.

As far as the EM-406A receiver used for these projects is concerned, it provides the GGA, GLL, GSA, GSV, RMC, and VTG frames, the formats of which are (more or less) described in its data sheet, downloadable from the manufacturer's website (see address list).

Before investigating these formats, do note that the RMC frame is the minimum frame needed for navigation, and is sometimes the only one used by the simplest devices.

Contents of a typical GGA frame	
Format	**Signification**
$GPGGA	Indicates the start of the frame, the sending device, and the type of frame
hhmmss.dcm	Time in UTC, to the nearest thousandth of a second
ddmm.mmmm	Latitude in degrees, minutes, and fractions of minutes
N or S	North or South
dddmm.mmmm	Longitude in degrees, minutes, and fractions of minutes
W or E	West or East
X	'fix' indicator : 0 = fix invalid, 1 = fix valid, 2 = fix valid, differential mode
NN	Number of satellites being used
XX	HDOP horizontal dilution of precision
AAAA	Altitude in metres above sea level
M	Altitude units (M for metres)
DDDD	Difference in altitude between the WGS84 ellipsoid and sea level
M	Difference units (M for metres)
SSSS	Age of the differential data in seconds
NNNN	Differential station ID
*XX	End of frame indicator and checksum

Contents of a typical RMC frame	
Format	**Meaning**
$GPRMC	Indicates the start of the frame, the sending device, and the type of frame
hhmmss.dcm	Time in UTC, to the nearest thousandth of a second
A ou V	Valid (A) or invalid (V) data
ddmm.mmmm	Latitude in degrees, minutes, and fractions of minutes
N or S	North or South
dddmm.mmmm	Longitude in degrees, minutes, and fractions of minutes
W or E	West or East
VVV	Ground speed in knots
CCC.CC	True heading in degrees and fractions of degrees
JJMMAA	Date
XXX	Variation
E or W	Sense of the correction (East or West)
*XX	End of frame indicator and checksum



GPS/PC adaptor

sarily of the payware type, as we'll be telling you below about some excellent freeware.

This interface will also let you display, on the PC, the NMEA frames actually transmitted by the GPS receiver. If you plan to build this receiver into a project of your own, or modify the software for the stand-alone GPS receiver we're proposing, this will prove a very handy investigative tool.

Figure 1 gives the circuit diagram of this interface, which shows a certain originality, as it can be powered either from a PC's USB port or by an external PSU (depending on the position of jumper JP1: external towards the outside, USB towards the inside of the PCB). This board can be connected to the PC in two ways: via either a COM port or a USB port.

In the case of the serial approach, the MAX242 converts the receiver's (almost) TTL output signal levels into RS-232 levels, and vice-versa. In this case, use of an external power supply will be obligatory.
As both the MAX and the GPS receiver need +5 V, a common or garden 3-terminal regulator will allow us to obtain this from any voltage between +8 and +12 V applied to power socket K1.
For the USB link, we could use the Elektor/FTDI TTL-232-R cable (SHOP item 080213-71) which will connect header K2 to the PC's USB port.
In addition, activating the EN signal via the USB cable allows the MAX242 to disable the RS-232 port transceivers, thereby protecting the USB cable from overloads if the COM port is still connected (dual connection is possible!). We recommend you not to use both power supplies at the same time.

If you are only ever going to be powering via the USB, the supply components (K1 to C3) may be omitted, and JP1 fitted towards the inside, or even replaced with a wire link.

Even though most of the communications will be going from the GPS to the PC, it should be noted that this interface is bi-directional, as the EM-406A receiver is capable of understand-

ing a certain number of elementary commands.

Construction

Construction of this project is very easy, thanks to the PCB we've produced, the overlay for which is shown

EM-406A GPS receiver module.

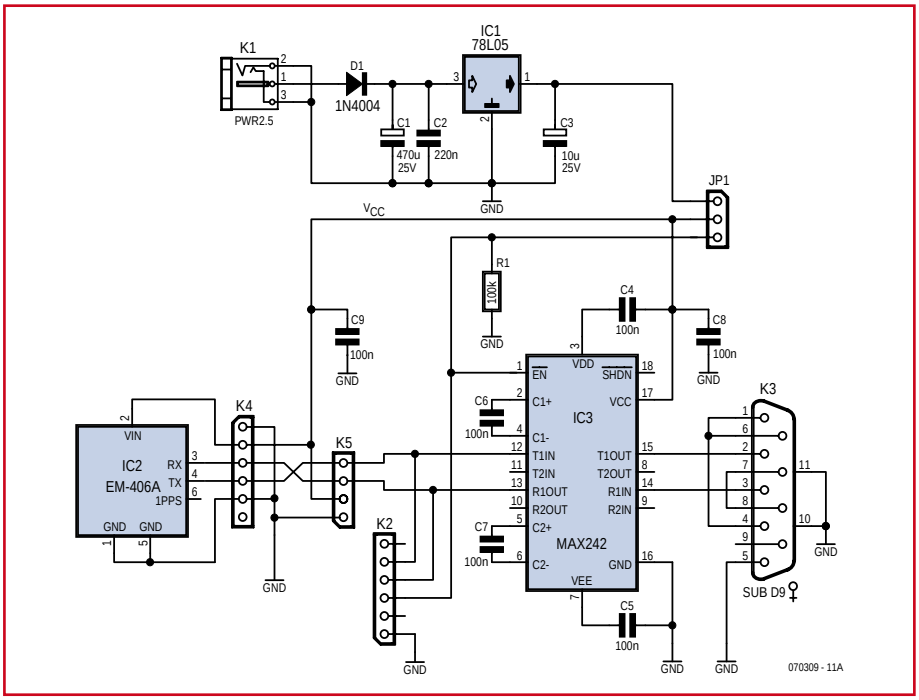
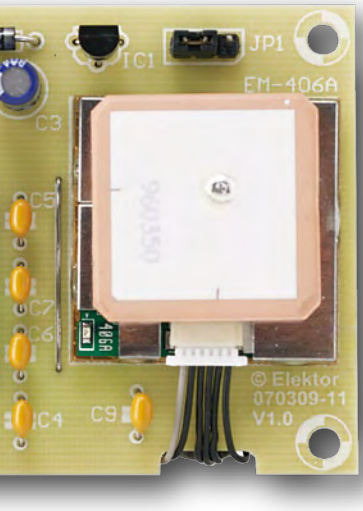


Figure 1. Circuit diagram of GPS/PC adaptor.



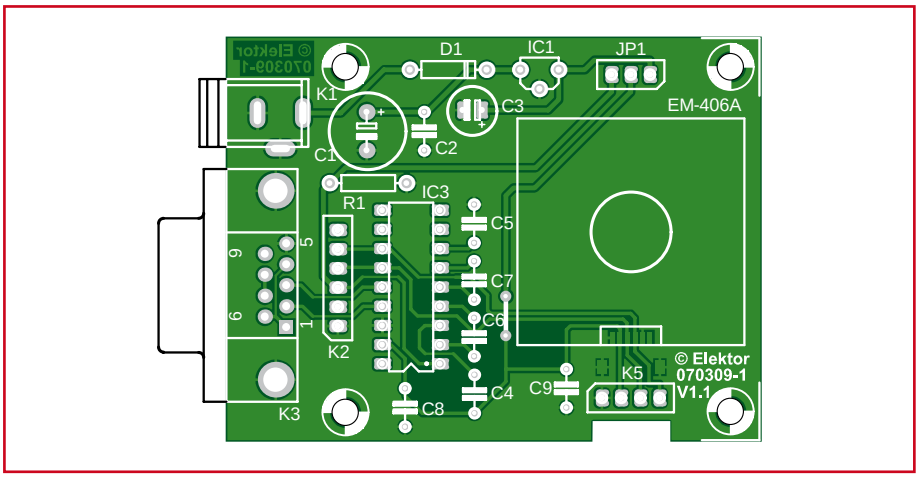


Figure 2. PCB component overlay for GPS/PC adaptor.

in **Figure 2**. To minimize wiring, it carries all the components in **Figure 1**. The connector on the GPS link cable is really tiny and its female socket (K4) very difficult to source (see component list). Also, it is tricky to solder on the track side of the PCB. To avoid this problem, we've also made provision for four pins on the board on a 2.54-mm (0.1-inch) pitch (K5) which can be used to make the link with the receiver. In this case, it will be necessary to cut the plug off the receiver's little cable to free off the conductors so they can be fitted with a 2.54 mm-pitch SIL female connector, cut down to a length of four contacts, which will mate with header K5 (note that header K5 hasn't yet been fitted to the prototype in the photo).

GPS software for PC

The unit is very simple to use. All you have to do is connect it to the PC using a cable fitted with 9-pin sub-D connectors. Take care! Given the pinout of J1, you need to use a straight-through cable (the commonest type), not one with crossed wires.

The supply should be connected between the pins marked +8 to +12 V and Ground, if you are using a plug-top PSU, which must be capable of providing 100 mA or so and in that case doesn't need to be regulated. In this case, link S1 must be fitted.
If the +5 V supply is tapped off the keyboard or a USB port, S1 is removed, freeing up the +5 V connection pad intended for this very purpose.
As soon as power is applied, the red LED on the GPS receiver will light up – steadily at first while it acquires satellites, then flashing when it has 'seen' enough to be able to provide valid data.

Note that the receiver includes a super-capacitor, enabling it to 'hot re-

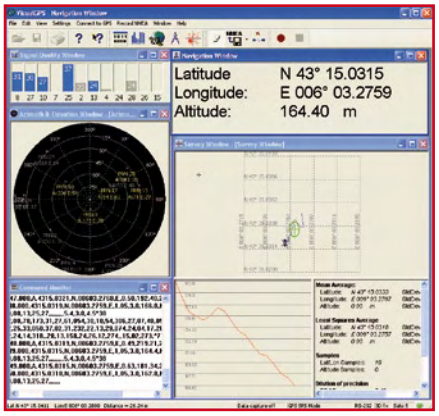


Figure 3. VisualGPS offers a very comprehensive display.

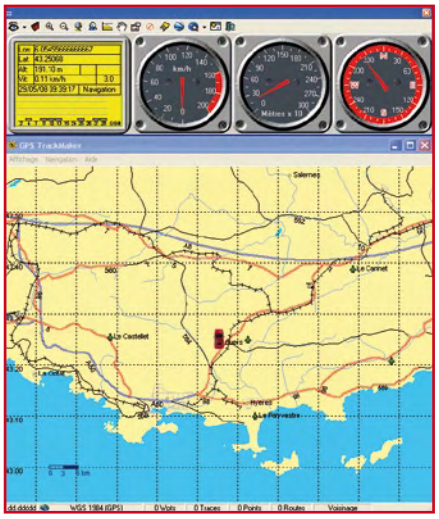


Figure 4. GPS TrackMaker has a cartography system.

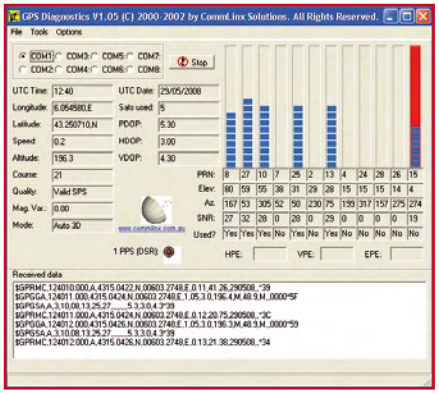


Figure 5. GPS Diagnostics lets you display and decode the NMEA frames.

start', i.e. acquire the satellites very quickly if it has not been moved since it was powered-down, as long as it hasn't been left without power for too long.
To exploit the data provided by the GPS, you can use one of the countless free or paid up applications available on the Internet. A simple Google search will offer you an overwhelming choice, but as far as we're concerned, we recommend you try, for example:

– **VisualGPS** [3], one of the most comprehensive, offering (**Figure 3**) both a decoded display of the GPS data and the contents of its output frames, with the added possibility of saving them to a file. Take care, though, not to get muddled up: VisualGPS is free, while VisualGPSXP is paid for.

– **GPS Track Maker** [4], just as free, also allows the GPS output data to be displayed, but also offers navigation facilities (**Figure 4**) by means of a large number of maps that can be downloaded from the Internet.

– **GPS Diagnostics** [5] has a slightly less rich interface (**Figure 5**), but does still display all the data output by the GPS receiver, along with the raw content of the frames it provides.

So whether you opt for GPS Diagnostics or VisualGPS, you'll have an excellent tool for saving the frames provided by the GPS receiver and then if necessary analyse them at leisure. Let's not forget these frames are just a series of ASCII-coded characters (as shown in the box), and so can be manipulated using any text editor.

A very simple stand-alone GPS receiver

This first receiver is very simple to construct – so much so that we haven't even designed a PCB for it. However, *let's render unto Cæsar that which is… Lextronic's.* It actually comes straight out of Application Note 36 relating to the Cubloc circuits from Comfile Technology, a well known advertiser in the French edition of Elektor.

As **Figure 6** shows, the circuit is incredibly simple: apart from the GPS receiver itself, it requires only an LCD display – admittedly a bit special, we'll be coming back to that later – and a Cubloc CB220. Let's not forget that this circuit, which we've already discussed in our Summer 2007 special robotics double issue, is in fact a hybrid microcontroller, similar in concept to the famous Basic Stamps from Parallax, with which it is also pin compatible. Though still based on an Atmel ATMega128 processor, the Cubloc CB220 is more powerful than the Basic Stamp and has a much fuller and better-performing Basic interpreter.

The GPS receiver serial output is connected to the input of the UART built in to the CB220, which in turn drives the display – an LCD alphanumeric type with 4 lines of 20 characters – via just the two port lines P8 and P9. An I²C bus is available, and the CLCD display from Comfile Technology [6] is also fitted with this type of interface, referred to by Comfile as 'Cunet'. Only pull-up resistors for the I²C bus SDA and SCL lines need to be added.
Switch S1 lets you choose two different display screens: the first indicates the first eight satellites received with their respective levels in the form of bargraphs; the second displays time, latitude, longitude, number of satellites being received, and indicates if the details displayed are valid.

The 9-pin sub-D connector marked J1 is not used in normal operation — only during the programming phase of the CB220, which is carried out from the serial port of any PC running the free Cubloc Studio development software.
Given the simplicity of the circuit, we built the project on prototyping board, also on offer from Comfile under part no. 'CB220 proto', as the price of about € 10 (approx. £ 7.50) is less than the cost of the items you'd need to build it.
Board wiring amounts to fitting resistors R1 and R2, the connectors for the display and the connection to the GPS receiver. For the latter, we proceeded in the same way as for the RS-232 interface above, but with only 3 pins, as this project operates only in the direction GPS → microcontroller.

Take care! The CB220 can be powered using an unregulated voltage between +5.5 V and +12 V via its pin 24 (VIN), and it will then output a stabilized voltage of +5 V from its pin 21 (VDD). However, this option must not be used in this particular project, as the CB220's built-in regulator can't supply the combined current of the GPS receiver and display together. So the whole circuit needs to be powered via pin 21 (VDD) of the CB220, which in this case becomes its power input.
If you still want to use the jack wired onto the CB220 prototype board, two minor modifications need to be made. The first is to slightly bend pin 24 of the CB220 so that it no longer enters the

24-pin socket and hence stays 'floating'. The second is to make a wire link – underneath the IC, for example – between the cathode of diode D1 and the +5 V pads in the experimental wiring area so that the centre contact of the jack connects the +5 V to the CB220's VDD input on pin 22.

The CB220 is programmed via the Cubloc Studio software, downloadable free from Lextronic's website. Then all you need do is connect one of the PC's serial ports to the 9-pin sub-D connector on the CB220 prototype board, configure Cubloc Studio accordingly, and run the software programming corresponding to Application Note 36. This software can also be downloaded from the Lextronic website in the form of a .zip file containing all the resources for all the application notes relating to the CB220.

This listing is provided in the form of source code, and in BASIC moreover, and is well covered in the aforementioned application note, so we won't repeat these comments here. But we do still recommend you to only study

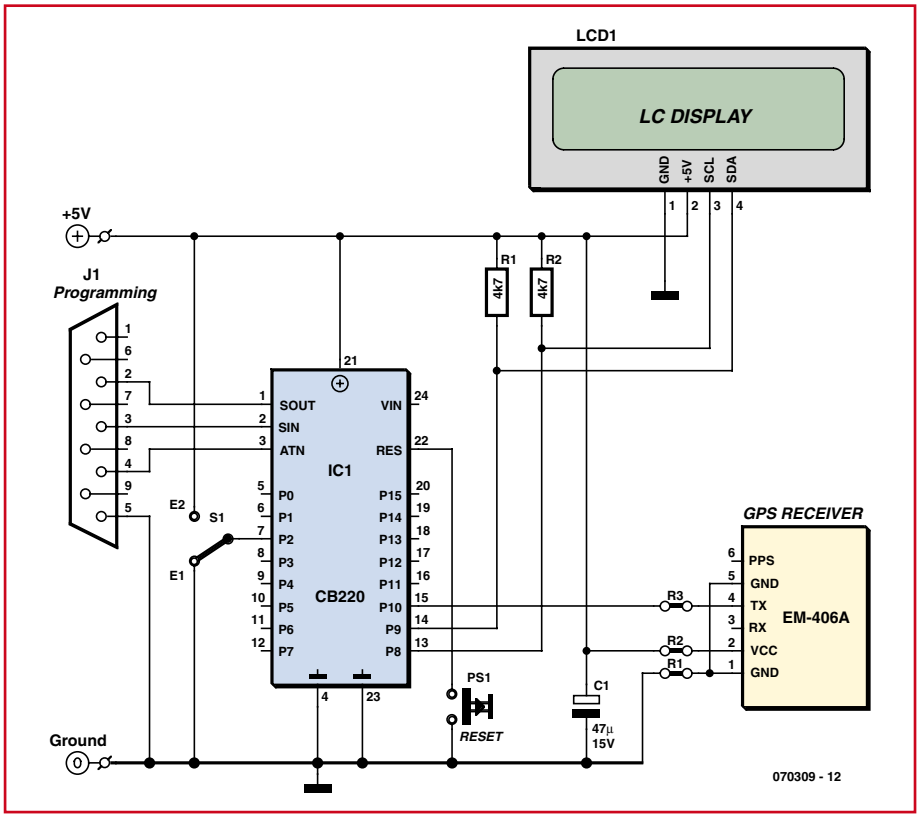


Figure 6. Circuit diagram of the very simple stand-alone GPS receiver.

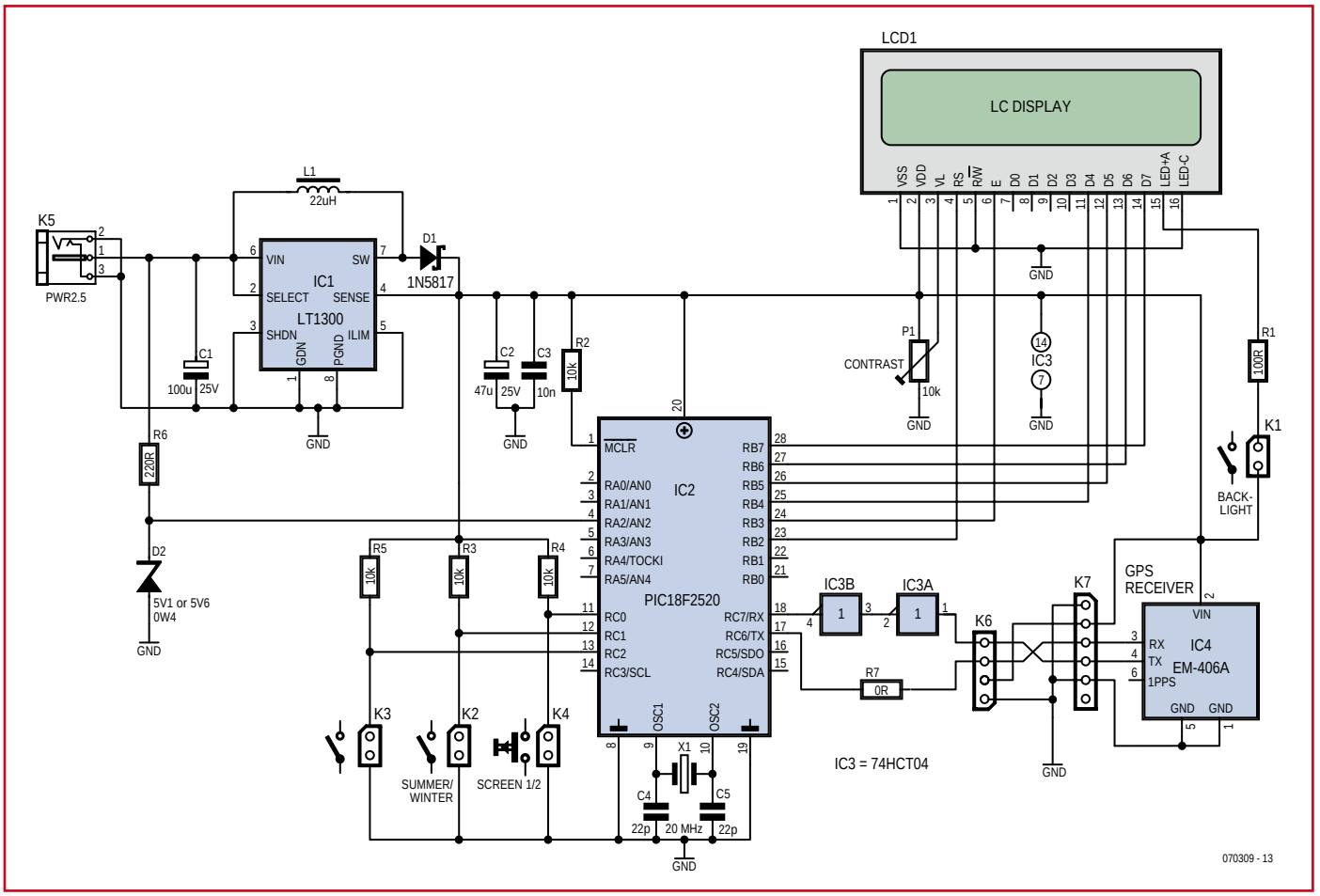


Figure 7. Circuit diagram of the comprehensive autonomous GPS receiver.

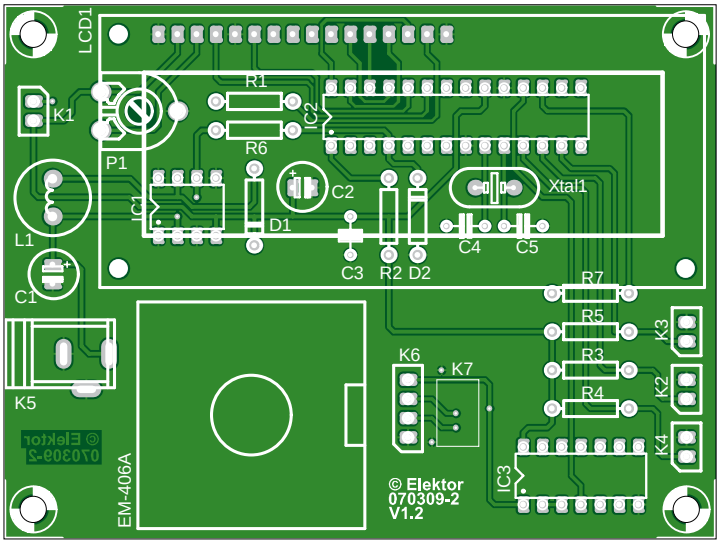
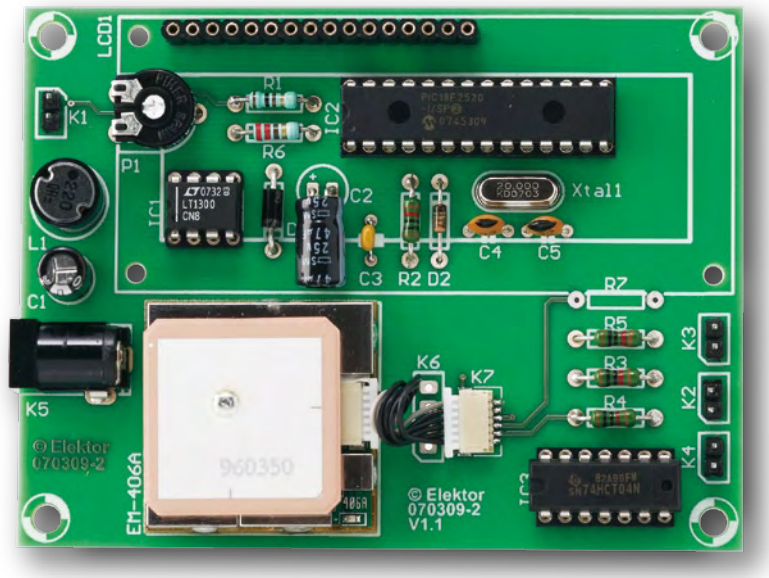


Figure 8. Component overlay for the comprehensive self-contained GPS receiver.



The wiring for the very simple stand-alone GPS receiver on the CB220 prototype board amounts to very little.

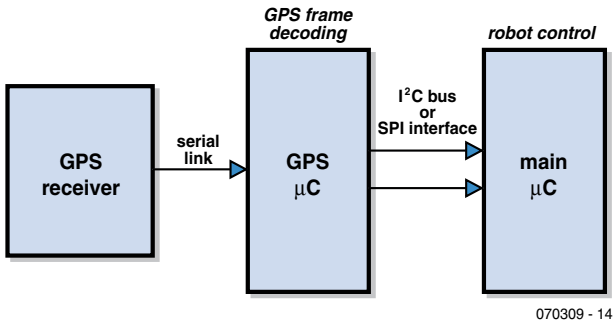


Figure 9. Block diagram for fitting a GPS receiver into a robot.

it after you've read the **inset** about the contents of NMEA frames. Note that this software decodes frames using detection of commas (the field delimiters).

Once the program has been downloaded into the CB220, the link with the PC can be disconnected for good, and the circuit then becomes stand-alone. Moreover, since you have both the source listing and the Cubloc Studio development tool, you can modify this software at your leisure if the displayed information doesn't suit your needs. In this way you might, for example, decode the end of the GGA frame to add a display of altitude, or the end of the RMC frame to display speed.

The circuit operates as soon as the program has finished loading, but if the display only shows its blinking cursor instead of the GPS information, check its I²C address is correct. This is set to 1 by the program line: `Set Display 2,0,1,50`, while the CLCD displays seem to be supplied with an I²C address set to 0. In that event, modify accordingly either the position of the display's DIP switches (to make its address 1), or the program line (to use a display at the I²C zero address).

A comprehensive self-contained GPS receiver

The above receiver is certainly perfectly functional and very quick to implement, but it still costs quite a bit, mainly because of the cost of the CB220 and the associated CLCD display. So now we're suggesting you really 'get your hands dirty' by building a receiver from scratch.

Just like the previous one, this has an LCD display with 4 lines of 20 characters – but a type without serial or I²C interface, and so much cheaper to buy. In addition, more and different information is provided. What's more, our receiver is totally self-contained, as it can be powered from two 1.5 V dry cells, or better still, two 1.2 V rechargeables. It is relatively compact, as it fits onto a PCB that mounts onto the back of the display board. The source listing of the software used is provided for you to download free, allowing readers to adapt it to their needs if the choices made by the author do not suit you.
The receiver circuit shown in **Figure 7** is easy to follow. The heart of the circuit is IC2, a PIC18F2520 microcontrol-

ler. Using an IC from the PIC18 family allows us to have an extended instruction set and larger program and data memories than with the PIC16 family, for almost the same price. It controls a standard LCD display with parallel interface in 4-bit mode, which explains why DB0–DB3 are not connected. The presence of switch K1 means you are free to choose a backlit display – at the price of increased power consumption, of course, but more legible in low-light conditions.

The signal from the GPS receiver goes to the input of the PIC's built-in UART, but first has to be brought up to level by IC3. This is one of the criticisms we made above of the EM-406A module: its logic High output level of 2.85 V is lower than the 4.00 V minimum High threshold of input RC7 on the PIC. So without IC3, the UART would never receive any data whatsoever.

The pushbutton connected to header K4 lets you toggle between the display screens. One shows latitude, longitude, altitude, and speed; the other, time (real, summer/winter depending on the position of switch S2), date, number of satellites being used, and battery voltage.

Switch S2 (K2) lets you choose between BST and GMT so as to display the 'real' time and not the raw UTC (Universal Time) provided by the satellite. Switch S3 (K3) is not used in the version given here, but it's still provided for anyone who might want to add functions to this project.

To obtain a high-efficiency power supply, we've dispensed with a conventional linear regulator and used an LT1300 from Linear Technology (IC1). From an input voltage applied to power jack header K5 between +2 V and +5 V, this remarkable circuit provides a stabilized voltage of +5 V at a current of up to 400 mA in the configuration used here. So two 1.5 V dry cells or two 1.2 V rechargeables are perfect for powering this project, giving it around ten hours of operation.

The battery voltage is also applied, via resistor R6 and protective zener D2, to input AN2 of the PIC's analogue-to-digital converter (ADC), enabling us to display it and hence be able to judge how much operating time is left. Resistor R7 (0 Ω = it's a wire link!) is only fitted if you want to establish communication from the PIC to the GPS module and thereby be able to experiment with sending commands using the PIC.

Construction

Construction is particularly easy as there are no SMD components and the PCB is a single-sided type with a track pattern that's easy to copy, even with limited resources, as you can see from **Figure 8**. The display is mounted onto the board.

The choice of components presents no difficulty, but if you don't use the recommended type for L1, be careful to choose a choke capable of handling a current of 800 mA without saturating. If you don't, the LT1300 will work badly, or not at all.

We have provided two options for connecting the GPS receiver. If you're lucky enough to be able to get hold of a male header for K7, you'll be able to solder it in place on the top of the board. If not, as in the other two circuits, you can use 2.54 mm-pitch male pin headers and modify the receiver cable as explained above. The connections to the various switches are likewise brought out on the same type of pin headers.

As far as the display is concerned, we fitted 2.54 mm-pitch female strip contacts to the back of it and male contacts at the same pitch on the copper side of the PCB. This is an easy way to construct an assembly that's easy to dismantle in the event of problems.

In order to simplify the PCB layout and keep it single-sided without needing too many wire links, we've not fitted an 'in-circuit' programming connector for the PIC. So it will need to be programmed using an external programmer before it's fitted into its socket. Elektor has described numerous projects of this type, and you'll also find one you can build yourself on the

COMPONENTS LIST

GPS/PC Adaptor

Capacitors
C1 = 470µF 25 V radial
C2 = 220nF MKT
C3 = 10µF 25 V radial
C4-C9 = 100nF ceramic

Semiconductors
D1 = 1N4004
IC1 = 78L05
IC2 = GPS receiver type EM-406A (see text)
IC3 = MAX242 (Maxim IC)

Miscellaneous
K1 = PCB mount DC adaptor socket
K2 = 6-way SIL pinheader
K3 = 9-way sub-D socket (female), PCB mount
K4 = connector type SM06B-SRSS-RB (Digikey)
K5 = 4-way SIL pinheader
JP1 = 3-way SIL pinheader with jumper
PCB copper track layout, free download # 070309-1 from www.elektor.com

Comprehensive Autonomous GPS Receiver

Resistors (0.25W / 5%)
R1 = 100Ω
R2-R5 = 10kΩ
R6 = 220Ω
R7 = 0Ω*

Capacitors
C1 = 100µF 25V radial
C2 = 47µF 25V radial
C3 = 10nF ceramic
C4,C5 = 22pF ceramic

Semiconductors
LCD1 = alphanumeric LCD, 4 lines of 20 characters
IC1 = LT1300
IC2 = PIC18F2520, programmed, Elektor SHOP # **080309-41**
IC3 = 74LS04
D1 = 1N5817 (must be Schottky)
D2 = 5.1V or 5.6V 400 mW zener diode

Miscellaneous
L1 = 22 µH, Panasonic type ELC08D (e.g. RS Components)
Qz1 = 20MHz quartz crystal
P1 = 10kΩ preset H
K1-K4 = 2-way SIL pinheader
K5 = DC adaptor socket
K6 = 4-way SIL pinheader
K7 = connector type SM06B-SRSS-TB(LF)(SN) (Digikey)
Type EM-406A GPS receiver (see text)
PCB copper track layout, free download # 070309-2 from www.elektor.com

author's website. The IC is also available ready-programmed via the Elektor website (see component list).
The software to be programmed into the PIC is available for download from the Elektor website, as well the author's own website, in two forms: an object file (with a .hex extension), ready to be programmed into the PIC, and the source file in Mikroelektronika Basic [7], since this is the compiler we used for this project.

Before fitting the PIC into its socket and connecting up the display and GPS receiver, connect the batteries to the circuit and check you have got +5 V on the LT1300 output. If this is OK, you can turn off the power, fit these last elements, and apply power again. The display should first indicate "Acquiring Sat." then "Invalid Data»while the receiver is not ready. However, as soon as the receiver's LED starts to blink, the information it provides should appear on the screen.

Software and incorporating into a robot

The software used for this receiver is inspired by that suggested by Marcel Durieux in the Mikroelektronika forum, with a certain number of adaptations aimed at making it work with the EM-406A receiver, not used in the original version.

While the previous receiver decoded the frames by detecting and then counting the commas (field delimiters), this one receives a complete frame and the range into an indexed variable. The advantage of this method is that it is simpler to implement. The reception of the characters sent by the EM-406A module is taken care of by the following very short function:

```
sub function ReadChr as byte
  do
  loop until USART_Data_Ready = 1
  result=USART_Read
end sub
```

All that remains is to call it up in a loop to store the characters sent by the GPS receiver into an indexed variable. End-of-frame (EOF) detection can be achieved either by receiving the asterisk that precedes the checksum, or when the number of characters received is clearly greater than the maximum allowed by the standard. Here's an example:

```
do
  GPSstr[i]=ReadChr
  if GPSstr[i]="*" then
   i=0
  else
   i=i+1
  end if
 loop until (i=0) or (i=75)
```

Frame decoding is then performed, according to the position of the data in the indexed variable `GPSstr[i]`. According to the NMEA standard, this position is fixed for a given piece of data. Unfortunately, the EM-406A receiver is a little 'creative' in terms of the length of the fields at the end of the RMC frame, requiring some adjustments in order to display the date, as you can see from a look at the BASIC listing.

We won't go right through the listing here, but have highlighted two extracts from it to help those who might want to incorporate such a receiver into a mobile robot, for example. This is actually very simple with the information in this article, but we do need to draw your attention to a couple of points.

The first concerns the accuracy of a GPS receiver, which is not unlimited, and depends heavily on the number of satellites being correctly received. In the best cases, we can hope for around 2 to 3 m with 7 or 8 satellites being received. When this number is reduced, the accuracy drops off rapidly, approaching 10 m with only 4 satellites. This is enough for route-finding in a car, but not for your robot to be able to pass between a chair and a table in your lounge, for example – unless of course you live in a castle!

The second concerns the decoding of the GPS frames, which is a time-consuming task, especially if you want 'good' accuracy, which implies constant position updating. So it must not be tasked to the robot's main processor, which would have to spend most of its time just doing this. So we recommend using the architecture given in **Figure 9**, where a microcontroller – and why not the PIC18F2520 of our previous receiver? – receives and decodes the frames. All it then has to do is provide the robot with just the required information – most usually, latitude and longitude. This can take place via its SPI port for example, or over the I²C. In this way, the robot's main processor is free to get on with its most important primary responsibilities, and only come and collect the ready-to-use information when it actually needs it.

Conclusion

Apart from suggesting three projects we hope you'll have found interesting, we also hope this article has been able to answer some of the questions frequently found on the Internet about how to interface a GPS receiver with a microcontroller.

(070309-I)

Internet Links

[1] Globalsat Technology Corporation:
www.globalsat.com.tw
GPS receiver manufacturer. Product datasheet downloading.

[2] Lextronic:
www.lextronic.fr
Distributor for the GPS receiver and Comfile products. Downloading the datasheets and instructions for the CB220, the LCD display, Cubloc Studio and the very simple GPS receiver software.

[3] VisualGPS LLC:
www.visualgps.net
Downloading the VisualGPS software.

[4] GPS Track Maker:
www.gpstm.com
Downloading the GPS Track Maker software.

[5] CommLinx Solutions:
www.commlinx.com.au
Downloading the GPS Diagnostics software.

[6] Comfile Technology:
www.comfile.co.kr
Manufacturer of the Cublocs. Downloading the data sheets and instructions for the CB220, the LCD display, and Cubloc Studio.

[7] Mikroelektronika:
www.mikroe.com
Publisher of the MikroBasic BASIC compiler used for the full GPS receiver.

[8] Author's website:
www.tavernier-c.com
Downloading of the software for both GPS receivers.

[9] Elektor website:
www.elektor.com
Downloading the software for both GPS receivers.