

**Professor Dr Dogan Ibrahim** from the Department of Computer Engineering at the Near East University in Cyprus describes the design of a GPS data logger device with SD card storage, where the collected data can be displayed in street-level using the Google Earth mapping program

# GPS data logger with SD card storage and GOOGLE EARTH map interface

**THE GLOBAL** Positioning System (GPS) is a satellite-based navigation system developed by the US Department of Defence. The first GPS system was tested in 1960s using a constellation of five satellites. This system was implemented for military purposes and provided navigational fix data approximately every hour; it was not very accurate.

In 1993 the number of satellites increased to 24. The system became fully operational and it was also made available to the civilians with a lesser accuracy. Initially, the accuracy of the civilian GPS system was deliberately disturbed using a method called Selective Availability (SA). With the SA, the position accuracy of a typical civilian GPS receiver was about 100 metres. In the year 2000 the US Department of Defence removed the SA and, as a result, the position accuracy of a basic GPS increased to around 10 metres.

GPS has now become a widely used aid to navigation and it is commonly used in many applications such as land surveying, shipping, piloting, route guidance, map making, study of earthquakes, precise time reference and hobbies and games such as geocaching.

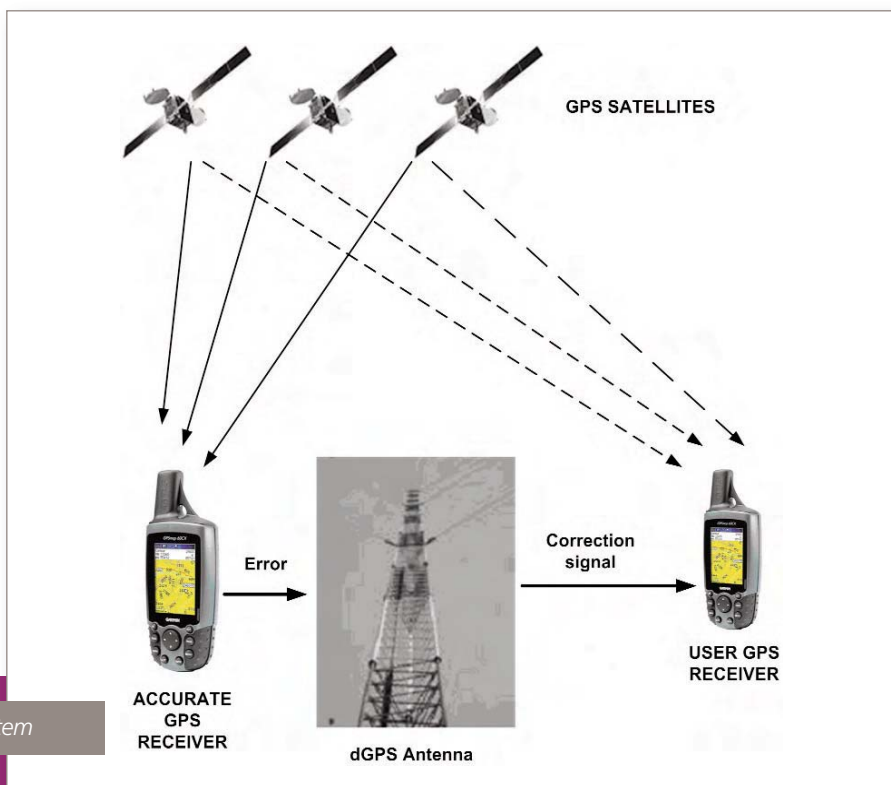
## The GPS Satellites

The GPS satellites orbit the Earth twice a day with a speed of 3.9km per second. The

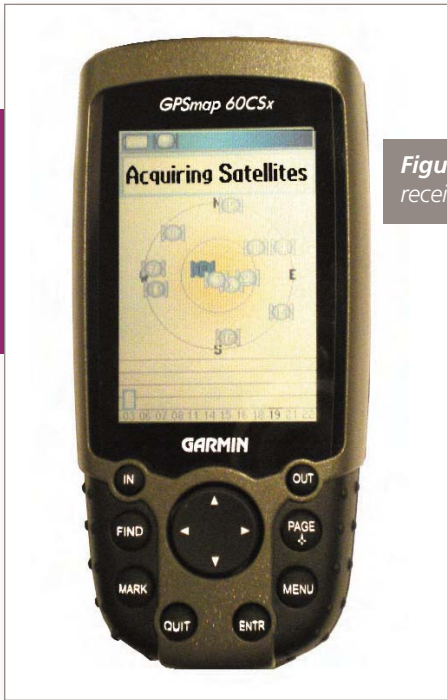
ERROR SOURCE	ERROR (METRES)
Selective Availability	30
Ionosphere	5.0
Orbit Errors	2.5
Satellite Clock	1.5
Signal Multipath	0.6
Troposphere	0.5
Receiver Noise	0.4

**Table 1:** GPS error sources

four satellites are visible at any point on earth at any time. Normally, three satellites are required to calculate the position of a point accurately on Earth's surface and four satellites are required to calculate the altitude as well. Thus, with the arrangement of the satellites it is possible to calculate both the position and the altitude of any point on Earth's surface accurately. In typical applications it is possible to get signals from at least six or seven satellites in a place with a



**Figure 1:** dGPS correction system



**Figure 2:** A typical commercial GPS receiver, such as this one from Garmin

clear view of the sky. In general, the accuracy is increased as more satellites are used in the position and altitude calculations.

### The GPS Accuracy

**Table 1** shows the typical GPS error sources ([www.kowoma.de/en/gps/errors.htm](http://www.kowoma.de/en/gps/errors.htm)). The major source of GPS error was the Selective Availability. Selective Availability was first introduced during the Gulf War in 1991 to prevent the Iraqi forces from benefiting from the accurate GPS service. What the US military chiefs did was not to prevent all non-military use of the GPS, but to degrade the accuracy of the GPS to commercial users. By the introduction of a deliberate error, the accuracy of a commercial GPS was reduced to around 100 meters and accurate receivers were only made selectively available to only the US and Allied military and to certain US Government agencies. Fortunately, the Selective Availability has been removed by the US since the 1st May 2000 and, hence, the navigational accuracy of a GPS improved significantly.

The tropospheric and ionospheric effects cause electromagnetic waves to refract. The reasons for the refraction are different concentrations of water vapour in the troposphere, caused by different weather conditions. The ionospheric errors are larger than the tropospheric errors. The errors introduced by these effects can not be eliminated, but their effects can usually be reduced in calculations.

Orbital errors are the other major sources of error, caused by the satellite geometry. If

for example a receiver sees four satellites and all are arranged in the north-west, this leads to a bad geometry and the accuracy is reduced. If on the other hand the four satellites are well distributed over the Earth, the position accuracy will be much higher. Most GPS receivers do not only indicate the number of received satellites, but also their relative positions above Earth. This enables the user to judge if a relevant satellite is obscured by an obstacle and if changing the position of the receiver might improve the accuracy.

The multipath effect is caused by reflection of satellite signals from objects on Earth's surface. The multipath error in GPS systems appear when there are large buildings near a GPS receiver and the signal reflects from these buildings. The reflected signal takes more time to reach the receiver than the direct signal and this results in accuracy errors. The multipath errors can be avoided by moving the receiver away from nearby large buildings or trees.

### Improving the GPS Accuracy

The position accuracy of a GPS signal can be improved significantly using a technique known as Differential GPS (dGPS). Using the dGPS techniques accuracies in the region of several meters can easily be achieved.

dGPS works by placing a high performance GPS receiver at a precisely known location on Earth (reference station). Since this reference receiver knows its exact location, it can determine the errors in the received GPS satellite signals. This error signal for each tracked satellite is formed into a correction message and is transmitted to ordinary GPS receivers. Users with the correct hardware can receive these correction signals and improve their accuracies. The level of accuracy obtainable with dGPS depends upon many factors, such as the quality of the reference station and user GPS receivers, and the atmospheric conditions. **Figure 1** shows the layout of a typical dGPS implementation.

Although dGPS-based systems provide high accuracies, they add extra complexity and also increase the cost of the basic navigation system.

Another technique used to improve the accuracy of the GPS system is to transmit correction signals from "correction" GPS satellites. The correction satellite is a geostationary satellite above the Equator. This technique was first developed in the US and is known as WAAS (Wide Area Augmentation System). A similar and compatible system is in operation in Europe under the name EGNOS and in the Far East under the name MSAS. Standard compatible GPS receivers can receive the WAAS/EGNOS correction signals and calculate their positions to an accuracy of around 3-5 metres.

The advantage of this technique over the dGPS is that there is no additional cost to the user since the required hardware and software are built into the GPS receiver at a very small extra cost. One disadvantage of this technique is that it may be difficult to receive correction signals at higher latitudes, away from the Equator.

### Parts of a GPS System

The GPS system consists of three major parts:

- Space segment
- Control segment
- User segment

**Space segment:** The space segment consists of the orbiting satellites. The number of satellites is increasing all the time. As of March 2008, there were 31 actively broadcasting satellites in the GPS constellation. With the increased number of satellites, the reliability and availability of the overall system has been improved.

**Control segment:** The control segment consists of the monitoring stations located on Earth. These stations constantly monitor the operational status of all the GPS satellites and also synchronize the atomic clocks on board the satellites to within a few nanoseconds of each other and adjust

```

$GPGLL,5127.3513,N,00003.2305,E,220702,A,A*40
$GPBOD,,T,,M,,*47
$GPVTG,74.7,T,77.1,M,0.0,N,0.0,K,A*26
$PGRME,14.6,M,18.5,M,23.6,M*16
$PGRMZ,72,f,*1D
$PGRMM,Ord Srvy GB*1B
$SHCHDG,245.6,...2.5,W*39
$SGPRTE,I,I,c,*37
$GPRMC,220704,A,5127.3506,N,00003.2307,E,0.0,74.7,051108,2.5,W,A*30
$GPRMB,A,,,,,,,,,V,A*1C
$GPGGA,220704,5127.3506,N,00003.2307,E,1.03,2.2,67.7,M,47.0,M,,*73
$GPGSA,A,2,,04,,,,,23,,31,,0.0,2.2,2*34
$GPGSV,3,1,12,01,27,049,00,02,09,310,18,04,51,293,39,07,18,317,00*78
$GPGSV,3,2,12,11,22,155,00,13,50,202,00,17,30,237,00,20,49,085,25*7B
$GPGSV,3,3,12,23,76,125,20,25,16,157,15,31,16,034,37,32,26,082,18*79
$GPGLL,5127.3506,N,00003.2307,E,220704,A,A*40
    
```

Figure 3: Typical NMEA sentences

the orbital model of each satellite.

**User segment:** The user segment consists of the user GPS receivers. A GPS receiver is a small battery-operated portable device similar in size to a mobile phone. As shown in **Figure 2**, the device is equipped with a large LCD display, a few buttons and an antenna (usually built-in). The device receives signals

sophisticated receivers also incorporate street level maps where the position of the user is shown dynamically in real time on the map. GPS receivers are also used in car navigation systems and can help the user to navigate to an unknown address or post-code by displaying street level turns or by audio outputs.

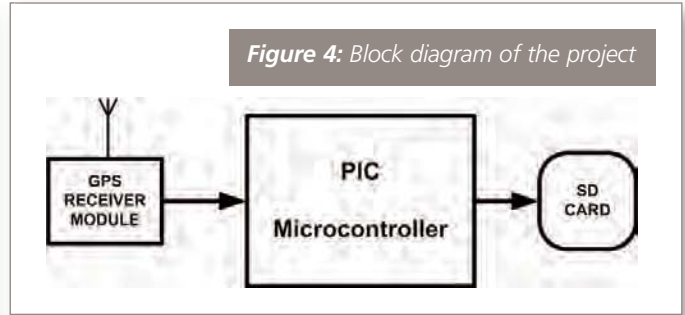


Figure 4: Block diagram of the project

from the GPS satellites and then displays user's position, altitude, speed, heading and several other navigational parameters. Some

GPS receivers may include an input for differential dGPS corrections, known as the RTCM interface. As mentioned earlier this interface improves the accuracy of the receiver considerably. In addition to the RTCM interface, some more sophisticated GPS receivers include WAAS/EGNOS correction hardware and software to improve the accuracy of the basic GPS receiver.

GPS receivers also produce RS232-compatible serial output data known as NMEA sentences, which enable them to be connected to a PC (or similar equipment) to relay the navigational data such as the latitude, longitude, altitude, speed, heading and so on. More details are given in the next section about the NMEA interface, as this is used in this project to get the time and geographical co-ordinates of the user.

**The NMEA Sentences**

Many high-end GPS receivers provide navigational output data so that the device can be connected, for example, to a PC to collect and analyse this data. This output data is usually in serial format and the communication protocol conforms to the RS232 serial standards.

The default serial communication parameters of most GPS receivers are set as follows:

- 4800 Baud
- 8 data bits
- No parity bit
- 1 stop bit

The data output from a GPS receiver is in ASCII text format and is known as the NMEA 0183, or simply the NMEA format. According to this format, navigational information are sent in the form of "sentences", where each NMEA sentence starts with a "\$" sign, the navigational parameters are separated by

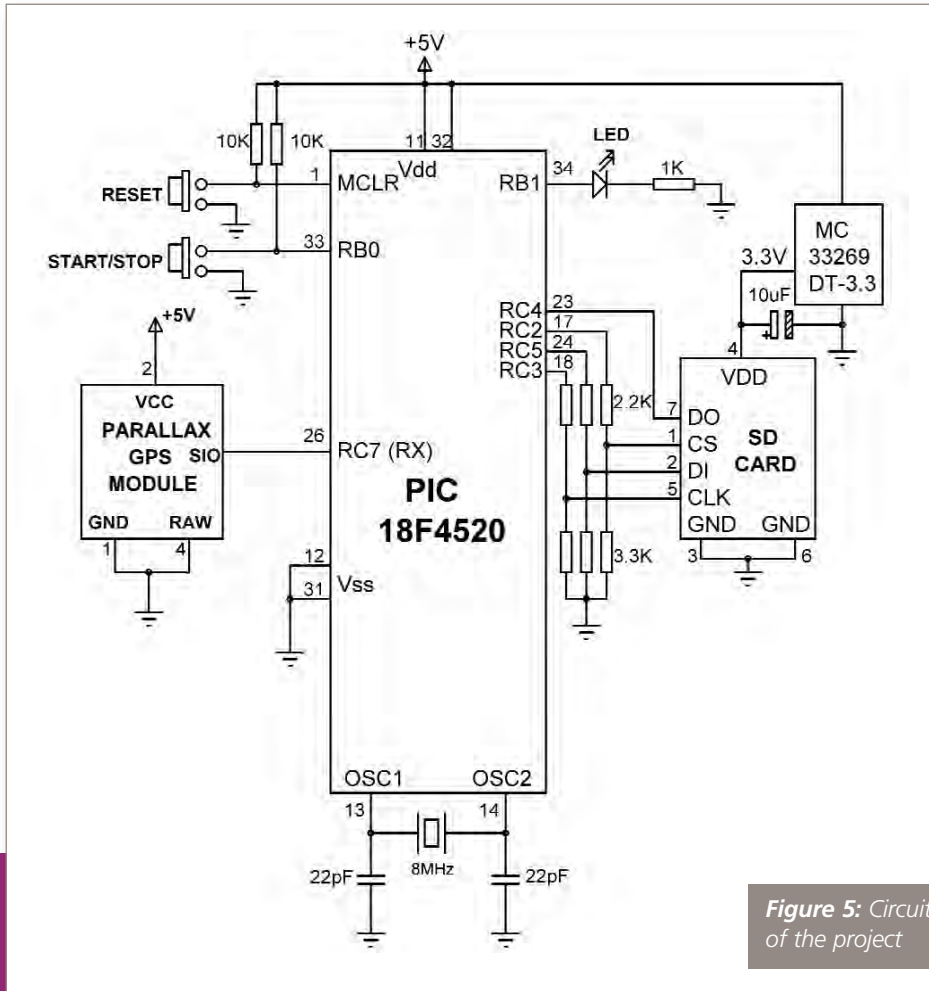


Figure 5: Circuit diagram of the project

commas and each sentence is terminated with two hexadecimal checksum characters.

The NMEA sentences are usually sent out every second. **Figure 3** shows the NMEA sentences obtained when a Garmin GPSMAP 60CSX type GPS receiver is connected to the serial port of a PC and a serial communication program, such as the HyperTerm ([www.hilgraeve.com/](http://www.hilgraeve.com/)), is activated on the PC to display the data received from the serial port.

There are many NMEA sentences for different navigational parameters. In addition to the standard NMEA 0183 protocol, some GPS manufacturers create and define NMEA sentences specific to their own products. The NMEA sentence used in this project is the \$GPRMC, having the following parameters: **\$GPRMC:** Although there are some variations in its format, this sentence basically defines the basic navigational parameters, speed, course, date of fix and the magnetic variation. An example is:

**\$GPRMC,220704,A,5127.3506,N,00003.2307,E,0.0,74.7,051108,2.5,W,A\*30**

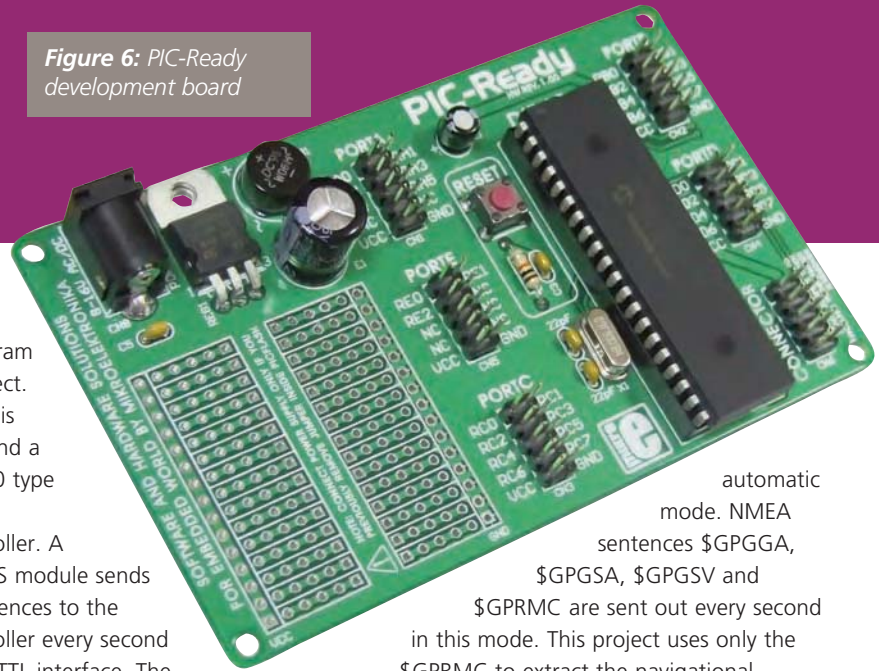
Here:

<b>220704</b>	<b>Fix taken at 22:07:04</b>
<b>A</b>	<b>Navigational data is correct</b>
<b>5127.3506,N</b>	<b>Latitude 51 deg. 27.3506 min. North</b>
<b>00003.2307,E</b>	<b>Longitude 0 deg. 3.2307 min. East</b>
<b>0.0</b>	<b>Speed over ground (knots)</b>
<b>74.7</b>	<b>Course made good</b>
<b>051108</b>	<b>Fix taken on 5 November, 2008</b>
<b>2.5</b>	<b>Magnetic variation 2.5 deg West</b>
<b>30</b>	<b>Checksum</b>

### The Project Outline

The block diagram of the project described in this article is shown in **Figure 4**. Basically, a microcontroller receives the \$GPRMC sentences from a GPS receiver module as the device moves around. The collected data is stored on an SD card continuously with time stamping. This data is then formatted and used as an input to the Google Earth (<http://earth.google.co.uk>) mapping program to display the track of the movement with or without time stamping.

**Figure 6:** PIC-Ready development board



**Figure 5** shows the circuit diagram of the project. The design is based around a PIC18F4520 type advanced microcontroller. A Parallax GPS module sends NMEA sentences to the microcontroller every second via a serial TTL interface. The microcontroller receives the NMEA sentences, extracts the \$GPRMC sentence and then extracts the time, latitude and longitude of the user co-ordinates and stores this data every two seconds in a file on the SD card. This file is then formatted to be compatible with the Google Earth mapping software, using the GPS Visualizer ([www.gpsvisualizer.com](http://www.gpsvisualizer.com)) software package. The Google Earth software can then show the user movements on a street-level map.

### The GPS Module

A Parallax GPS module is used in the design. This is a small, low-cost GPS with the following features:

- On-board passive patch antenna;
- Single wire, 4800 baud serial TTL interface;
- Provides either raw NMEA output, or specific data can be requested via a command interface;
- Operates with single +5V supply. The GPS module has four pins:
  - Pins 1 and 2 are the ground and the supply voltage respectively;
  - Pin 3 is the TTL compatible, non-inverted serial input-output pin. The data format is 4800 baud, 8 data bits, no parity.
  - Pin four is the output format selection bit, called the RAW pin. When RAW is low, the GPS module sends out NMEA sentences automatically every second. When RAW is high (or unconnected), specific GPS data (e.g. the latitude) can be requested from the device by sending commands. In this project the GPS module is operated in the

automatic mode. NMEA sentences \$GPGGA, \$GPGSA, \$GPGSV and \$GPRMC are sent out every second in this mode. This project uses only the \$GPRMC to extract the navigational parameters. Parallax GPS module outputs the \$GPRMC sentence in the following format:

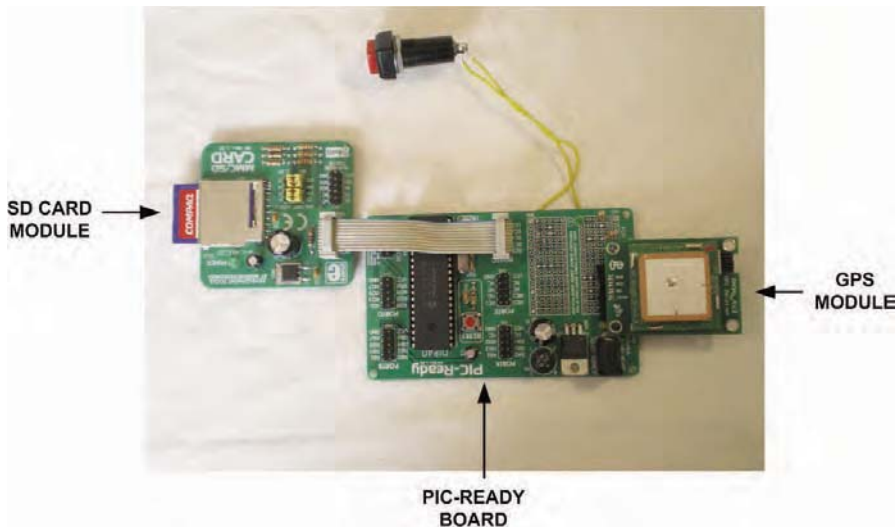
**\$GPRMC,220704,A,5127.3506,N,00003.2307,E,173.8,231.8,130608,,A\*70**

Notice that the magnetic variation and its direction are not output by the Parallax GPS module.

### The Key Elements of the Project

The SD card is connected to port pins RC2 through RC5 of the microcontroller and is operated in SPI mode. A card holder is used to physically make connections to the card pins. The voltage at output pins of the micro-controller is too high and can damage the input circuitry of the SD card. A pair of potential divider resistors (using 2.2K and 3.3K resistors) is used to lower the microcontroller output voltages to a level acceptable by the SD card inputs. The SD card is powered using a 3.3V regulated supply, obtained using a MC3269DT-3.3 ([www.volkin.com/MC3269DT-3.3.html](http://www.volkin.com/MC3269DT-3.3.html)) type regulator. In this project, a ready-made SD card module is used (see **Figure 7**), which is available from mikroElektronika ([www.mikroe.com](http://www.mikroe.com)).

A PIC18F4520 type microcontroller (MCU) is used in the design. The microcontroller is operated with an 8MHz crystal. A push-button switch is connected to port RB0 pin and a small LED is connected to port RB1 of the microcontroller. Pressing the switch



**Figure 7:** The GPS data logger device

starts and stops the data collection. During the data collection the LED flashes at a rate of about once a second. If the SD card is not inserted into its holder, the LED will flash quickly to show an error condition. The switch should be kept pressed for about five seconds to terminate the data logging and wait until the LED turns OFF before removing the SD card from its holder.

The hardware was constructed on a PIC-Ready development board (see **Figure 6**) manufactured by mikroElektronika. This is a low-cost (\$24) powerful development board with the following features:

- Socket for 40-pin PIC microcontrollers;
- 8MHz crystal;
- +5V regulator (an external 9-12V power supply is required);
- In-circuit debugger (ICD) and PIC programmer interface;
- Reset button;
- Easy access to microcontroller port pins via 10-way IDC connectors;
- Plug-in compatible with most mikroElektronika development modules;
- Small development solder area.

One of the nice things about the PIC-Ready development board is the built-in ICD and the programmer. This requires the use of a PICFlash2 ICD device, manufactured by mikroElektronika. During program development one can easily insert breakpoints or single step a program with the help of the ICD. The ICD can also program most of the PIC chips on-board, without having to

remove the chip from its socket. This feature is extremely useful during program development.

Figure 7 shows the GPS data logger device built on a PIC-Ready development board. The SD card module can be seen on the left, while the GPS module is on the right hand side of the board.

### The Software

The software was developed using the mikroC compiler developed by mikroElektronika. This is a very powerful C language compiler and supports both PIC16 and PIC18 series. The compiler provides a very rich library of routines for developing applications for SD cards, Compact Flash cards, RS232/RS485 devices, USB, CAN bus, I2C, 1-Wire bus and much more.

**Figure 8** shows operation of the software as a Program Development Language (PDL). The program is modular and consists of a number of functions and procedures for easy modification, update, or maintenance of the code. The following functions and procedures are used:

**Format\_Data:** This procedure reads the received \$GPRMC data from array DataLogger and stores in array LogIt in the following comma delimited CSV text format. This data is then written to a file on the SD card continuously every two seconds:

**T,22:05:16,5133.3627,00042.1240**

The data is stored in a format compatible

### BEGIN

```
Configure I/O ports
Wait until START is pressed
Initialise USART library
Initialise SD card library
IF SD card is not present
    Flash LED rapidly
    Wait forever
```

### ENDIF

```
Create a new file on SD card
```

### DO FOREVER

```
Turn on LED
Wait 500ms
Get GPS data
Extract $GPRMC sentence parameters
Store time, lat, lon on SD card
IF STOP pressed
    Turn OFF LED
    Wait forever
```

### ENDIF

```
Turn OFF LED
Wait 500ms
```

### ENDDO

### END

**Figure 8:** Operation of the software

with the GPS Visualizer conversion program, where:

**Character T indicates that this is a track file**  
**22:05:16 is the time the data was received**  
**5133.3627 is the latitude**  
**00042.1240 is the longitude**

As explained later, the GPS Visualizer program converts the file into a format compatible with the Google Earth mapping program.

**Get\_GPS\_Data:** This procedure reads the NMEA sentence \$GPRMC from the serial port where the GPS module is connected. The program first waits to receive the starting character "\$". Then the string "GPRMC" is matched and its parameters are read and stored in array DataLogger.

**Initialize\_SD:** This procedure initializes the SD card library routines of the mikroC compiler.

**Hex\_Byte:** This function converts a hexadecimal number into decimal. This function is used in the checksum calculation.

**Conv\_Hex:** This function converts a two digit hexadecimal number into decimal. This function is used in the checksum calculation.

**Checksum:** This function checks the checksum field of the received NMEA sentence. The checksum field in an NMEA sentence is the last two hexadecimal characters after the "\*" character. The checksum is calculated by Exclusive-OR'ing

all the characters in a sentence, except the "\$" character and the "\*" character. This function calculates the checksum and compares with the received checksum. If the two differ, a zero is returned to indicate an error in the received data and the data is discarded and read again. If the checksum is correct the function returns a one.

**Figure 9** shows how the checksum can easily be calculated by writing a "C" function. Variable `chk` is initialised to zero at the beginning of the function. A while loop is then formed to check each character of the received NMEA sentence. A "\$" or a "\*" character are ignored. Any other received character is saved in variable `chk` and then 'Exclusive-Or'ed with the older one. This way, the result is the Exclusive-OR (or the checksum) of all the received characters. Variable `chk2` is the decimal equivalent of the received checksum which is compared with the calculated checksum `chk`.

At the beginning of the program PORT B is configured to be a digital port and the program waits until the START/STOP push-button switch is pressed. After the switch is pressed the USART is initialised to 4800 Baud. The program then attempts to

initialise the SD card. If there is no card in the card holder the LED flashes rapidly to indicate an error condition. If on the other hand a card is found in the card holder then a new file is created on the card. The created filename is in the following format:

#### GPSLOGnn.TXT

Where `nn` is a number stored in the first location of the EEPROM memory. `nn` is between 01 and 99 and is incremented each time the device is started to collect new data, thus causing a new file to be created every time the data collection is started. The program then enters an endless loop where after receiving valid data from the GPS module, the time, latitude and longitude of the user are determined and stored on the SD card. The loop terminates if the START/STOP button is pressed for more than a few seconds. The data is stored on the SD card in comma delimited CSV format as shown in **Figure 10**, where the first row is a header describing what type of data is stored in each column of the file.

#### Google Earth Interface

The collected data (see **Figure 10**) should

first be converted into a format suitable to be displayed by the Google Earth mapping program. Google Earth accepts KML (Keyhole Marked Up Language) and KMZ (compressed KML file) formatted files (<http://code.google.com/apis/kml/documentaion>). There are many programs on the Internet that can be used to convert the CSV type file created in **Figure 9** into KML or KMZ format. The program used in this project is the GPS Visualizer ([www.gpsvisualizer.com](http://www.gpsvisualizer.com)). GPS Visualizer is a free, easy-to-use online utility that creates maps and profiles from GPS data, street addresses, or simple co-ordinates. The program can convert between various navigational co-ordinates, calculate the distance between two co-ordinates or two addresses, and many more.

The conversion process is very simple and is given in **Figure 11**.

- Start the GPS Visualizer;
- Select Convert a File from the top menu;
- Click on Google Earth Mapping Form;
- Click Browse under Upload your GPS data files here;
- Select the file to be converted (on the SD card);
- Select Yes, with no names in Draw as Waypoints list-box, under the Track Options;
- Choose a colour for the track (if desired) under Track Options (the default track colour is magenta);
- Click Create KML file to create a converted output file that is compatible with Google Earth.

The file will be converted and a new form will be displayed. Click on the converted filename to invoke the Google Earth program automatically and display the track (Note that the Google Earth software must be installed in order to invoke it and display the track data on the map).

```

unsigned char Checksum(void)
{
    unsigned char chk = 0;
    unsigned char i = 0;
    unsigned char chk2;

    while(DataLogger[i] != '*')
    {
        switch(DataLogger[i])
        {
            case 'S':
                break;
            case '*':
                continue;
            default:
                {
                    if(chk == 0)
                        chk = DataLogger[i];
                    else
                        chk = chk ^ DataLogger[i];
                }
                break;
        }
        i++;
    }

    // Compare calculated checksum (chk) with the Checksum read from
    // the NMEA sentence (chk2). If the Checksum is correct return 1,
    // otherwise return 0.
    //
    chk2 = Conv_Hex(DataLogger[64], DataLogger[65]);
    if(chk == chk2)
        return(1);
    else
        return(0);
}

```

**Figure 9:** Checksum calculation function

```

type,time,lat,lon
T,13:37:12,5127.3687,00003.1291
T,13:37:14,5127.3684,00003.1283
T,13:37:16,5127.3681,00003.1278
T,13:37:18,5127.3679,00003.1276
T,13:37:20,5127.3678,00003.1275
T,13:37:22,5127.3678,00003.1276
T,13:37:24,5127.3677,00003.1275
T,13:37:26,5127.3678,00003.1275
T,13:37:28,5127.3679,00003.1275
T,13:37:30,5127.3680,00003.1274
T,13:37:32,5127.3681,00003.1275
T,13:37:34,5127.3712,00003.1345
T,13:37:36,5127.3727,00003.1423

```

**Figure 10:** Example collected data

Figure 11: Example GPS Visualizer screen

Figure 12 shows an example output of displaying the track data at street-level using the Google Earth. In this example, the GPS data logger device was placed in a car and a short trip was made South East London to collect data and test the device. As can be seen from Figure 11, the collected data is very accurate and the data points are placed exactly on street co-ordinates in the Google Earth map display. In Figure 13, the same data is displayed with time stamping where the time display is enabled in the GPS Visualizer conversion program before the file is converted (i.e.

Select Yes, named with time stamps in Draw as Waypoints list-box, under the Track Options).

### Further Improvements

The device described in this article can be improved in several ways:

- Other navigational parameters can be added to the system, such as the speed, bearing and the altitude.
- The power consumption of the device can be lowered using low-power version of the microcontroller, i.e. PIC18LF4520.
- A file conversion utility can be added to

the microcontroller software so that the created file is compatible with the Google Map (i.e. in KML or KMZ format) and can be used directly without having to re-format it first.

- An LCD and a keypad can be attached to the device to make it more user friendly, e.g. to display the date, time, speed, or the geographical co-ordinates of the user as the device moves around.
- The data collection interval and the collection algorithm can be modified such that new data is stored only if the device is not stationary. ■



Figure 12: Displaying the data using Google Earth



Figure 13: Displaying the data with time stamping