How To Build A Lightweight GPS Data Logger
For Model Rocket Applications

Karl Upton
February 2009

Overview

I have been dreaming of building a flight computer that will not only control the flight sequence, but also log data aboard a model rocket. But I do need to walk before I can run, so I started with a simple GPS data logger (GPSDL) that is just a "piece" of my future flight computer idea. This GPSDL will sit in a payload bay or nosecone of a rocket during flight.

My finished GPSDL weighs 62 grams with the power supply and has a 1.5"L x 3"W x 1"D footprint. The weight of the data logger can be further reduced by ¼ to ½ oz. by using a simpler GPS antennae than the one I used. Cost can run from $100 to $200 depending on how careful a shopper you are. My cost was $200 for the parts used in this article.

The design is simple consisting of three major parts: a 5.5g accelerometer switch, a BS2p microcontoller and a GPS receiver. A parts list, pictures, source code and a schematic are included in this article.

The GPSDL records the date, time, latitude, longitude, altitude, speed, heading in degrees and number of satellites that are in communication with the receiver every second for a total of 5 minutes. The source code provided will record two 5 minute flights before you have to download the data. This is completely customizable for any number of flights or a single 12 minute flight. The comments in the

source code explain not only how to make this flight time change, but also what the program is doing throughout its runtime. The source code was split into two programs to maximize the amount of data that could be stored, negating the need for a separate EEPROM. The first program parses the GPRMC and GPGGA GPS sentences for the data points and writes them to memory. Post flight, the second program is downloaded to read the data points stored in memory and prints them to your PC screen. The data points are finally copied/pasted into a spreadsheet for conversions and graphing.

Step 1

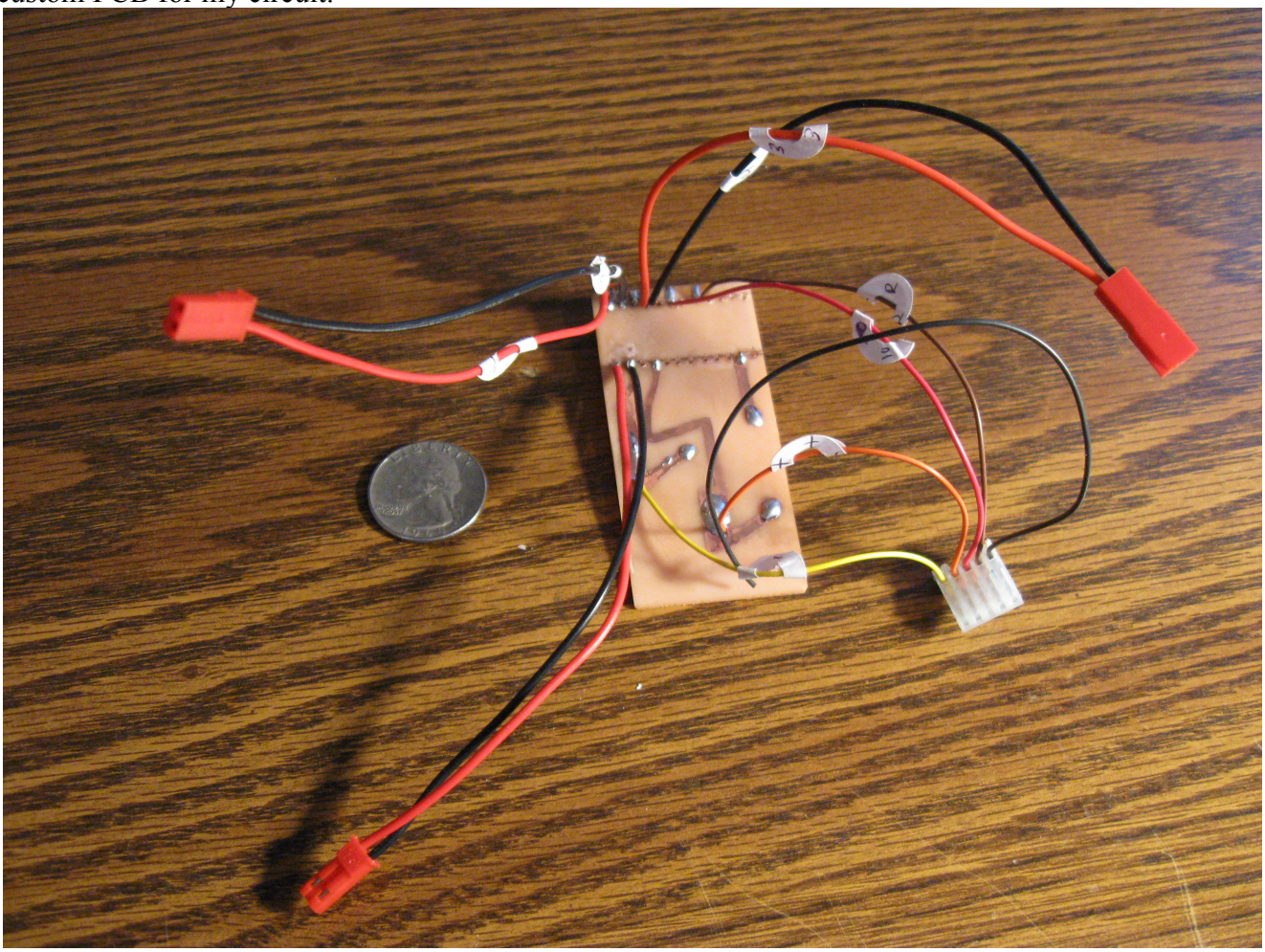
My first step was to familiarize myself with the parts chosen to work with by reading the data sheet for the GPS antennae and spending some time online learning how to use the free IDE that Parallax provides to develop and download code from my PC to the BS2p microcontroller via a serial port. This is quite simple and you can learn how to do this in an evening. If you want to just cut and paste my code onto your microcontroller that is about all you need to know code-wise to get the circuit running. If you want to customize my code or roll your own, PBASIC, which is the language that the BS2p runs on, is probably the easiest language to learn out there. There are multiple online forums catering to the Basic Stamp where help is available if you need it.

Step 2

Next, I breadboarded the circuit. Then I perched my “breadboard monstrosity” on my windowsill to get a good GPS signal and developed the code by trial and error.

Step 3

After getting the code developed and the circuit running quite nicely on a breadboard, I had to duplicate it on a PCB small and light enough for rocket work. It was not necessary, but I etched a custom PCB for my circuit.



Operation

Prior to launch, the only communication there is between you and the GPSDL is the blinking LED included on the GPS antennae PCB. A flashing LED means less than 3 satellites are acquired, a steady "on" LED signals that the antennae has acquired at least 3 of the 12 satellites available. Once you have visual that three or more satellites are acquired, the GPSDL is loaded into the rocket. At liftoff the 5.5g normally open, non-latching accelerometer switch is momentarily tripped signaling the microcontroller to log 20 bytes of GPS data every second for 5 minutes. Once 5 minutes is up it automatically resets itself to take another 5 minutes of data once the accelerometer switch is tripped again. You can record two 5 minute flights before data will need to be downloaded or you will overwrite the data already collected. The power supply is a 300 mAh 7.4V Li-Po battery. The GPSDL needs a steady diet of 5V, and it will run for approximately 3 hours on a full charge with this battery. No data is lost if power is lost. The only way data can be lost is if it is overwritten. GPS signals will travel through plastic, glass and cardboard.--The staples of model rocketry. The only things that will block the signal is concrete, metals or in the form of heavy rain or snow—H20. The antennae tested showed excellent Rx, even in a crowded, "signal noisy" urban environment where I live.

Data Recording Source Code:

```
' {$STAMP BS2p}
' {$PBASIC 2.5}
'Pins/Constants
GPSio          PIN     01
GPSraw         PIN     03
accelSwitch    PIN     09
MEMORYSIZE     CON     2020     'must be made divisible by 20 (20B per block + 6b Header/Footer)

'Variables
slotNum VAR Nib
address VAR Word
dayMonth VAR Word
year VAR Byte
hrs VAR Byte
mins VAR Byte
secs VAR Byte
latLeft VAR Word
latRight VAR Word
longLeft VAR Word
longRight VAR Word
speed1 VAR Word
speed2 VAR Nib
course1 VAR Word
course2 VAR Nib
numSats VAR Nib
alt1 VAR Word

'Initialize
INPUT accelSwitch
INPUT GPSraw
LOW GPSraw
slotNum = 0
address = 0
```

Data Recording Source Code (cont.):

```
dayMonth = 0
year = 0
hrs = 0
mins = 0
secs = 0
latLeft = 0
latRight = 0
longLeft = 0
longRight = 0
speed1 = 0
speed2 = 0
course1 = 0
course2 = 0
numSats = 0
alt1 = 0

main:
GOTO slotNumControl
'never returns to main unless a loss of power
'end main

slotNumControl:
slotNum = slotNum + 1                    'enables 14k (2047 bytes X 7) for storage in multiple slots

SELECT slotNum          'each slot holds approx. 1.5 minutes of data at 9600 baud. Slots 2-7 available
    CASE < 1            'slot1 holds source code

      END

    CASE > 6                'slot7 not used

      END

    CASE = 1

     STORE slotNum
     address = 0
     GOTO waitForAccel      'start of flight1

    CASE = 2

     STORE slotNum
     address = 0
     GOTO collectData

    CASE = 3
```

Data Recording Source Code (cont.):

```
      STORE slotNum
      address = 0
      GOTO collectData              'end dataSlot for flight1

     CASE = 4

      STORE slotNum
      address = 0
      GOTO waitForAccel             'start of flight2

    CASE = 5

      STORE slotNum
      address = 0
      GOTO collectData

     CASE = 6

      STORE slotNum
      address = 0
      GOTO collectData              'end dataSlot for flight2
ENDSELECT
'end slotNumControl

waitForAccel:                       'waits for accelerometer switch to trip
SELECT accelSwitch
     CASE = 1

       GOTO collectData

     CASE ELSE

       GOTO waitForAccel
ENDSELECT
'end waitForAccel

collectData:
        'Parse GPRMC sentence by counting commas
        '$GPRMC,hrsminssecs.sss,validitybit,latLeft.latRight,N,longLeft.LongRight,W, speed1.speed2,
                  'course1.course2,dayMonthyear,...,CRC
SERIN GPSio, 500,[WAIT("RMC,"), WAIT(",") , WAIT(","), WAIT(","), WAIT(","), WAIT(","),
                       WAIT(","), WAIT(","), WAIT(","), DEC4 dayMonth, DEC2 year]

WRITE address, dayMonth.HIGHBYTE
address = address + 1
WRITE address, dayMonth.LOWBYTE
address = address + 1
```

Data Recording Source Code (cont.):

```
WRITE address, year
address = address + 1
'end 3 byte header for slotX datablock

DO WHILE address < MEMORYSIZE                    'writes to current slot until 2026 bytes are used

        'Parse GPRMC sentence by counting bytes & commas
        '$GPRMC,hrsminssecs.sss,validitybit,latLeft.latRight,N,longLeft.LongRight,W,speed1.speed2,
                'course1.course2,dayMonthyear,...,CRC
SERIN GPSio,500,[WAIT("RMC,"),DEC2 hrs, DEC2 mins, DEC2 secs, WAIT(","), DEC latLeft,
                DEC latRight, SKIP 3, DEC longLeft, DEC longRight, SKIP 3,  DEC speed1,
                DEC speed2, DEC course1, DEC course2]

WRITE address, hrs
address = address + 1

WRITE address, mins
address = address + 1

WRITE address, secs
address = address + 1

WRITE address, latLeft.HIGHBYTE
address = address + 1
WRITE address, latLeft.LOWBYTE
address = address + 1

WRITE address, latRight.HIGHBYTE
address = address + 1
WRITE address, latRight.LOWBYTE
address = address + 1

WRITE address, longLeft.HIGHBYTE
address = address + 1
WRITE address, longLeft.LOWBYTE
address = address + 1

WRITE address, longRight.HIGHBYTE
address = address + 1
WRITE address, longRight.LOWBYTE
address = address + 1

WRITE address, speed1.HIGHBYTE
address = address + 1
WRITE address, speed1.LOWBYTE
address = address + 1
```

Data Recording Source Code (cont.):

```
WRITE address, speed2
address = address + 1

WRITE address, course1.HIGHBYTE
address = address + 1
WRITE address, course1.LOWBYTE
address = address + 1

WRITE address, course2
address = address + 1

'Parse GPGGA sentence for altitude & number of satellites (0-12) by counting commas
'$GPGGA,hrsminssecs.SSS,latLeft.latRight,N,longLeft.LongRight,W,positionIndicator,numSatellites,
        ' HDOP,MSLaltitude,...,CRC
SERIN GPSio, 500,[WAIT("GGA,"), WAIT(",") , WAIT(","), WAIT(","), WAIT(","), WAIT(","),
                  WAIT(","), DEC numSats, WAIT(","), DEC alt1]

WRITE address, numSats
address = address + 1

WRITE address, alt1.HIGHBYTE
address = address + 1
WRITE address, alt1.LOWBYTE
address = address + 1

LOOP

    'Parse GPRMC sentence by counting commas
    '$GPRMC,hrsminssecs.sss,validity bit,latLeft.latRight,N,longLeft.LongRight,W,speed1.speed2,
            'course1.course2,dayMonthyear,...,CRC
SERIN GPSio, 500,[WAIT("RMC,"), WAIT(",") , WAIT(","), WAIT(","), WAIT(","), WAIT(","),
                  WAIT(","), WAIT(","), WAIT(","), DEC4 dayMonth, DEC2 year]

WRITE address, dayMonth.HIGHBYTE                    'start 3 byte footer for slotX datablock
address = address + 1
WRITE address, dayMonth.LOWBYTE
address = address + 1

WRITE address, year
address = address + 1

GOTO slotNumControl
'endCollectData
```

Data Reader Source Code:

```
' {$STAMP BS2p}
' {$PBASIC 2.5}
'This program is used to retrieve the recorded values from all program slots and DEBUGs them to a
'monitor. The raw data is then copied/pasted into a spreadsheet for conversions and graphing.

STORE 6         'must manually change to read each slot: Flight1 = slots 1 thru 3, flight2 = slots 4 thru 6

'Pins/Constants
MEMORYSIZE      CON     2020
                                'must be made divisible by 20 + extra (20B per block + 6b Header/Footer)

'Variables
address VAR Word
dayMonth VAR Word
year VAR Byte
hrs VAR Byte
mins VAR Byte
secs VAR Byte
latLeft VAR Word
latRight VAR Word
longLeft VAR Word
longRight VAR Word
```

Data Reader Source Code (cont.):

```
speed1 VAR Word
speed2 VAR Nib
course1 VAR Word
course2 VAR Nib
numSats VAR Nib
alt1 VAR Word

'Initialize
dayMonth = 0
year = 0
hrs = 0
mins = 0
secs = 0
latLeft = 0
latRight = 0
longLeft = 0
longRight = 0
speed1 = 0
speed2 = 0
course1 = 0
course2 = 0
numSats = 0
alt1 = 0
```

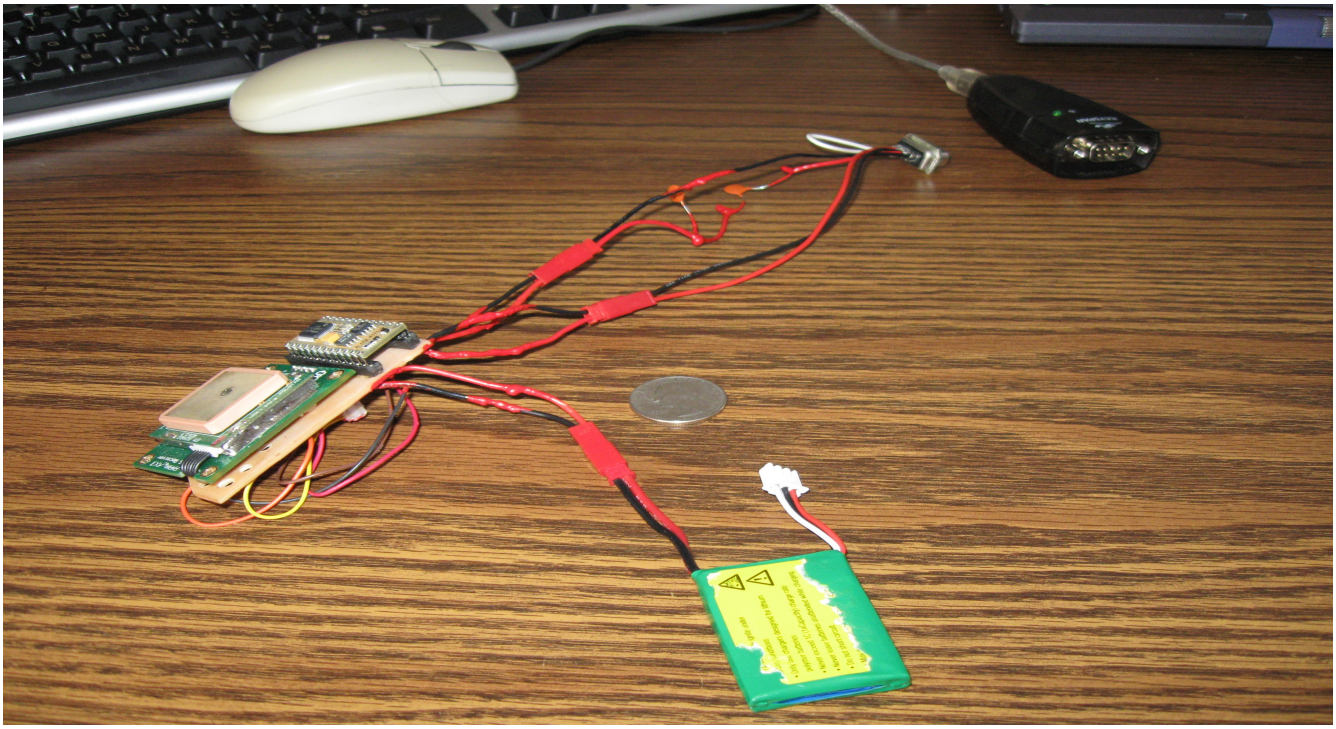
Data Reader Source Code(cont.):

```
main:
address = 0

READ address, dayMonth.HIGHBYTE
address = address + 1
READ address, dayMonth.LOWBYTE
address = address + 1
DEBUG "dayMonth: ", DEC dayMonth, CR

READ address, year
address = address + 1
DEBUG "year: ", DEC year, CR
'end 3 byte header for slotX dataBlock

DO

READ address, hrs
address = address + 1

READ address, mins
address = address + 1

READ address, secs
address = address + 1
DEBUG "HHMMSS: ", DEC hrs, ":", DEC mins, ":", DEC secs, CR

READ address, latLeft.HIGHBYTE
address = address + 1
READ address, latLeft.LOWBYTE
address = address + 1

READ address, latRight.HIGHBYTE
address = address + 1
READ address, latRight.LOWBYTE
address = address + 1
DEBUG "latitude: ", DEC latLeft, ".", DEC latRight, " N", CR

READ address, longLeft.HIGHBYTE
address = address + 1
READ address, longLeft.LOWBYTE
address = address + 1

READ address, longRight.HIGHBYTE
address = address + 1
READ address, longRight.LOWBYTE
address = address + 1
DEBUG "longitude: ", DEC longLeft, ".", DEC longRight, " W", CR
```

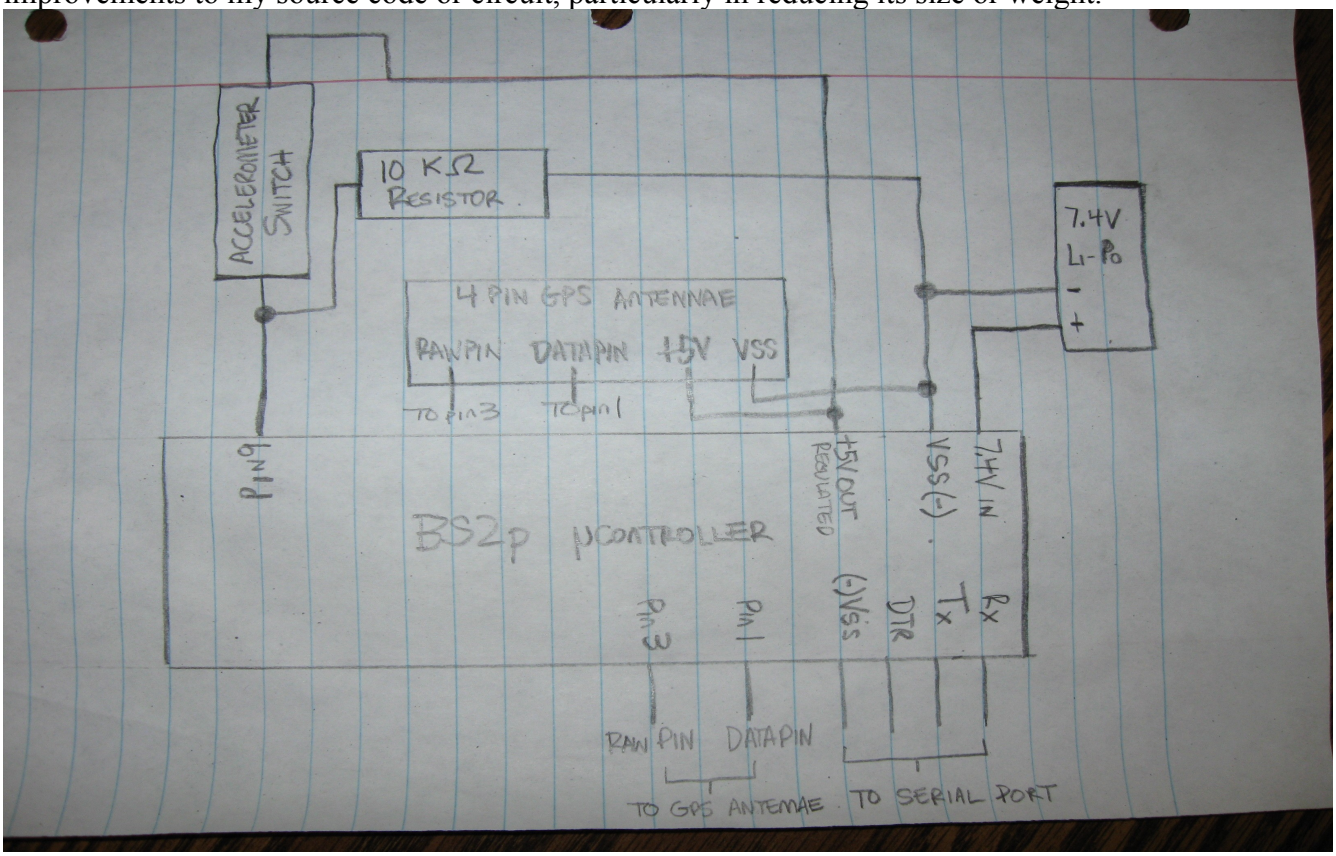
Data Reader Source Code (cont.):

```
READ address, speed1.HIGHBYTE
address = address + 1
READ address, speed1.LOWBYTE
address = address + 1

READ address, speed2
address = address + 1
DEBUG "knots: ", DEC speed1, ".", DEC speed2, CR

READ address, course1.HIGHBYTE
address = address + 1
READ address, course1.LOWBYTE
address = address + 1

READ address, course2
address = address + 1
DEBUG "heading in degrees: ", DEC course1, ".", DEC course2, CR

READ address, numSats
address = address + 1
DEBUG "number of satellites: ", DEC numSats, CR

Data Reader Source Code (cont.):
READ address, alt1.HIGHBYTE
address = address + 1

READ address, alt1.LOWBYTE
address = address + 1
DEBUG "altitude: ", DEC alt1, CR

LOOP WHILE address < MEMORYSIZE

READ address, dayMonth.HIGHBYTE                    'start 3 byte footer for slotX dataBlock
address = address + 1
READ address, dayMonth.LOWBYTE
address = address + 1
DEBUG "dayMonthFooter: ", DEC dayMonth, CR

READ address, year
address = address + 1
DEBUG "yearFooter: ", DEC year, CR
END
```

End Of Sourcecode



Unfortunately, I have not been able to test the GPSDL with an actual rocket launch. That will happen in the weeks to come. To run the GPSDL I shake it by hand to trip the accelerometer switch and take data from my dashboard as I drive. The GPSDL works flawlessly at this point. I will post not only real flight data but a video of its maiden flight to my website soon. I would appreciate any feedback on improvements to my source code or circuit, particularly in reducing its size or weight.



Parts List:

Parallax BS2p 24 pin microcontroller
Parallax GPS Receiver Module
5.5g non-latching, normally-open, accelerometer switch
Female serial port
JST battery connectors x 3 pair
10 k Ohm resistor
300 mAh 7.4 Li-Po battery

Online Resources:

www.parallax.com
www.polstargps.com
www.radioshack.com
www.aerocon.com
www.hobbyzone.com
www.grandideastudio.com
www.embeddedflightcontrol.weebly.com
embeddedflightcontrol@gmail.com