Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

GPS Receiver for Arduino (Model A) (SKU:TEL0083-A)

Introduction

GPS Receiver for Arduino is an unit embedding GPS module and antena in a small foot-print enclosure. By using TinyGPS library, Arduino can retrieve geographic coordinates (latitude and longitutde, altitude), speed, heading and GMT time.

The update rate is an important performance index of a GPS receiver. Most GPS in mobile phones provide an update rate of 1Hz, which means, only one set of data can be retrieved in one second. For GPS receivers with 5Hz, the data interval is much reduced and thus can be used for more demanding applications (e.g. on fast-moving vehicles)

Specifications

- U-BLOX G6010 solution
- 5Hz output
- 38400bps TTL serial interface
- 3.3V-5V input voltage
- 50-channel receiver
- Extremely high sensitivity -161dBm
- Accuracy 2.5m (Autonomous) / <2m[SBAS]
- Operating temperature: -40°C to 85°C
- Power consumption: 3.3V @ 41mA
- Hot Start : 1s
- Warm Start : 32s
- Cold Start : 32s
- Ceramic Antena 25*25*2mm
- Module Size 25*28*7.5mm
- Enclosure Size 40x30x10mm
- Wire length: about 23cm

Line Definition

- Black: RxD
- White: TxD
- Red: VCC
- Black (thick): GND

Tutorial

To use, simply create an instance of an object like this:

```
#include "TinyGPS.h"
TinyGPS gps;
```

Feed the object serial NMEA data one character at a time using the encode() method. (TinyGPS does not handle retrieving serial data from a GPS unit.) When encode() returns “true”, a valid sentence has just changed the TinyGPS object’s internal state. For example:

```
void loop()
{
  while (Serial.available())
  {
    int c = Serial.read();
```

```
    if (gps.encode(c))
    {
      // process new gps info here
    }
  }
}
```

You can then query the object to get various tidbits of data. To test whether the data returned is stale, examine the (optional) parameter “fix_age” which returns the number of milliseconds since the data was encoded.

```
long lat, lon;
unsigned long fix_age, time, date, speed, course;
unsigned long chars;
unsigned short sentences, failed_checksum;

// retrieves +/- lat/long in 100000ths of a degree
gps.get_position(&lat, &lon, &fix_age);

// time in hhmmsscc, date in ddmmyy
gps.get_datetime(&date, &time, &fix_age);

// returns speed in 100ths of a knot
speed = gps.speed();

// course in 100ths of a degree
course = gps.course();
```

GPS with LCD Sample Code

GPS receiver Arduino
VCC ———————— VCC
GND ———————— GND
RX ———————— TX
TX ———————— RX

Library：TinyGPS V1.3 https://github.com/mikalhart/TinyGPS/archive/v13.zip

```
#include <TinyGPS.h>
#include <LiquidCrystal.h>

TinyGPS gps;
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);              //LCD driver pins
int led = 13;

long lat, lon;
unsigned long fix_age, time, date, speed, course;
unsigned long chars;
unsigned short sentences, failed_checksum;
//int year;
//byte month, day, hour, minute, second, hundredths;

int DEG;
int MIN1;
int MIN2;
```

```
void LAT(){                          //Latitude state
  DEG=lat/1000000;
  MIN1=(lat/10000)%100;
  MIN2=lat%10000;

  lcd.setCursor(0,0);              // set the LCD cursor   position
  lcd.print("LAT:");
  lcd.print(DEG);
  lcd.write(0xDF);
  lcd.print(MIN1);
  lcd.print(".");
  lcd.print(MIN2);
  lcd.print("'   ");
}
void LON(){                           //Longitude state
  DEG=lon/1000000;
  MIN1=(lon/10000)%100;
  MIN2=lon%10000;

  lcd.setCursor(0,1);              // set the LCD cursor   position
  lcd.print("LON:");
  lcd.print(DEG);
  lcd.write(0xDF);
  lcd.print(MIN1);
  lcd.print(".");
  lcd.print(MIN2);
  lcd.print("'   ");
}




void setup()
{
  Serial.begin(38400);            //Set the GPS baud rate.
```

```
  pinMode(led, OUTPUT);

  lcd.begin(16, 2);               // start the library
  lcd.setCursor(0,0);             // set the LCD cursor   position
  lcd.print("GPS test");          // print a simple message on the LCD
  delay(2000);
}

void loop()
{
  while (Serial.available())
  {
    digitalWrite(led, HIGH);
    int c = Serial.read();                    // Read the GPS data
    if (gps.encode(c))                         // Check the GPS data
    {
      // process new gps info here
    }

  }
  digitalWrite(led, LOW);
  gps.get_position(&lat, &lon, &fix_age);     // retrieves +/- lat/long in 10
0000ths of a degree
  gps.get_datetime(&date, &time, &fix_age);   // time in hhmmsscc, date in dd
mmyy

  //gps.crack_datetime(&year, &month, &day,    //Date/time cracking
  //&hour, &minute, &second, &hundredths, &fix_age);

LAT();
LON();

}
```