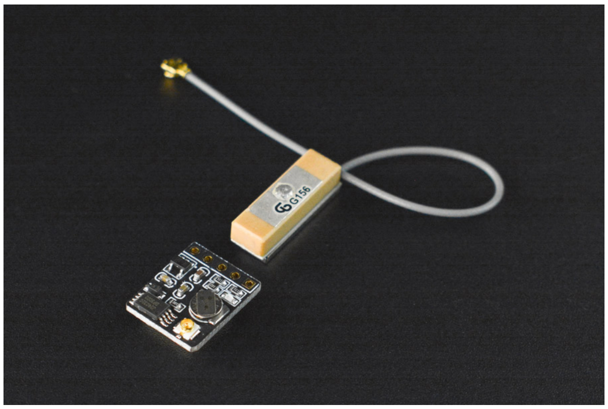
SKU:TEL0132 (https://www.dfrobot.com/product-2080.html)

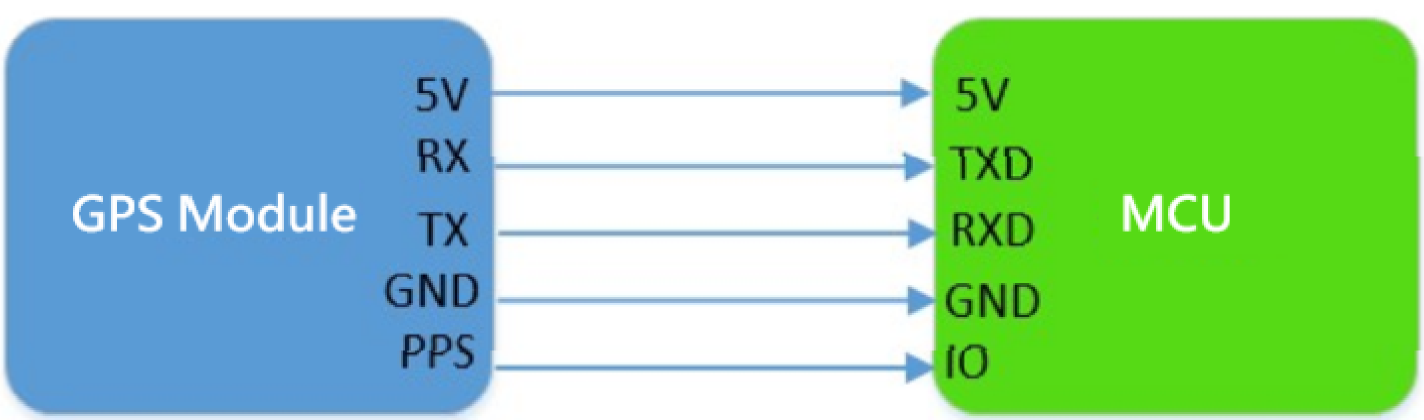


(https://www.dfrobot.com/product-2080.html)

Introduction

This is a small-size positioning and navigation module based on the AT6558 (a real sixes in one multi-mode satellite navigation and positioning chip). It can provide real-time location information, and support a variety of satellite navigation systems, including 32 tracking channels. Besides, the module can receive GNSS signal from 6 satellite navigation systems simultaneously, which include China BDS (Beidou navigation satellite system), the American GPS, the Russian GLONASS, European GALILEO, Japan QZSS and SBAS satellite augmentation system (WAAS, EGNOS, GAGAN MSAS), and realize the joint positioning, navigation and timing.

In serial output mode, the module is compatible with various mainboards equipped with serial output: Arduino, Raspberry Pi, STM32 and so on. The positioning accuracy error is measured at about 3m, basically the same as smartphones. The module's power consumption is as low as 0.1W, and it can work continuously for a long time with a small power supply.

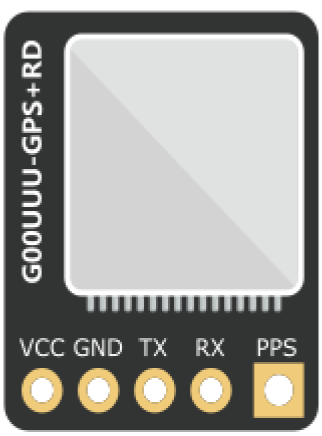


The small size of the module allows it to be embedded in various types of drones and smart cars. Furthermore, this module can be very suitable for vehicle navigation, handheld positioning and wearable devices or be used as a replacement of Ublox MAX series module.

Specification

- Operating Voltage: 3.3-5V
- SMA and IPEX Antenna Ports
- Onboard E2PROM for storing configuration settings
- Onboard XH414 rechargeable battery, speed up hot-start and search
- Support A-GNSS
- Cold- start Acquisition Sensitivity: -148dBm
- Tracking Sensitivity: -162dBm
- Positioning Accuracy: 2.5m(CEP50, open field)
- Time to First Fix: 32 seconds (or few minutes, depends on the environment)
- Low Power: <25mA (work continuously)
- Built-in antenna detection and antenna short circuit protection function
- Dimension: 13.1 x 15.7mm/0.52 x 0.62″

Board Overview

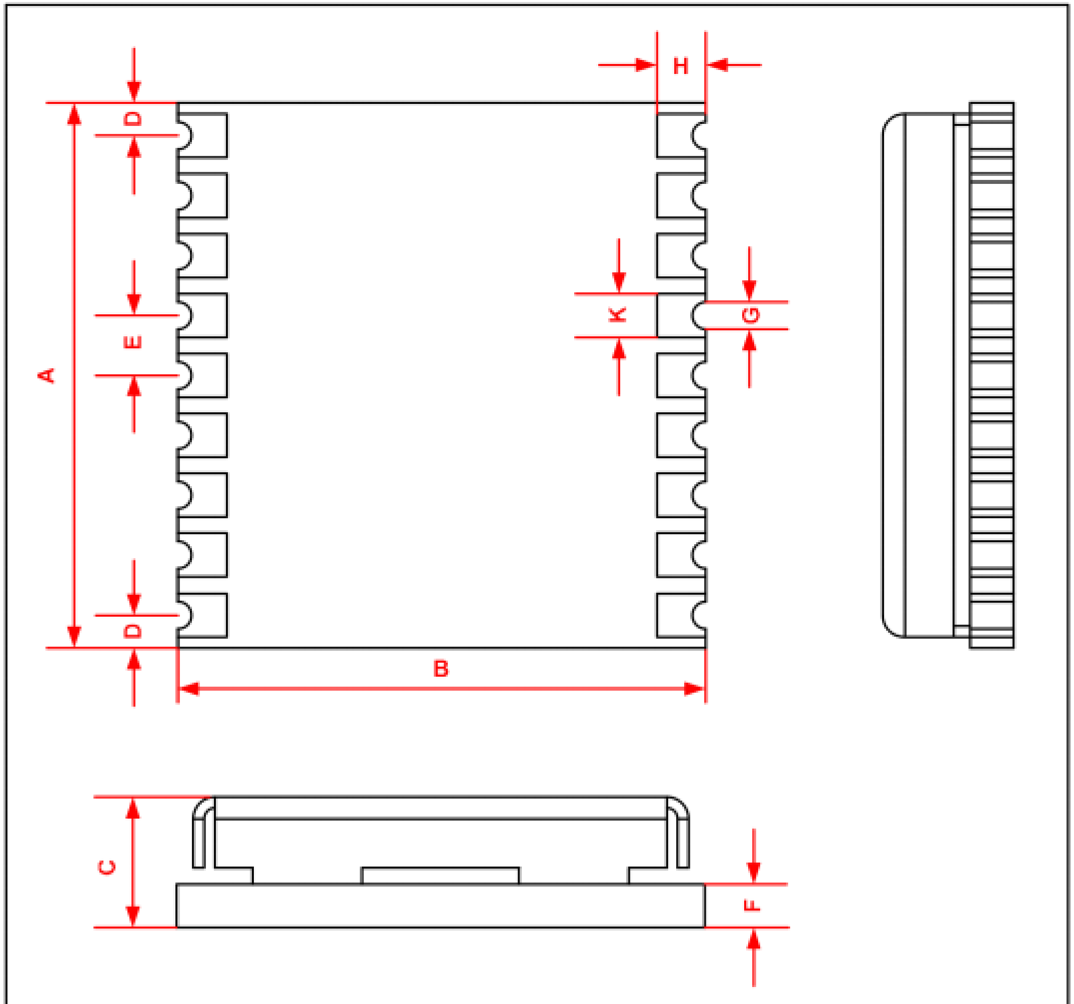


Name	Function
VCC	Power Input(5V)
GND	Ground
TX	Transmit
RX	Receive

Name	Function
1PPS	Pulse Output Per Second

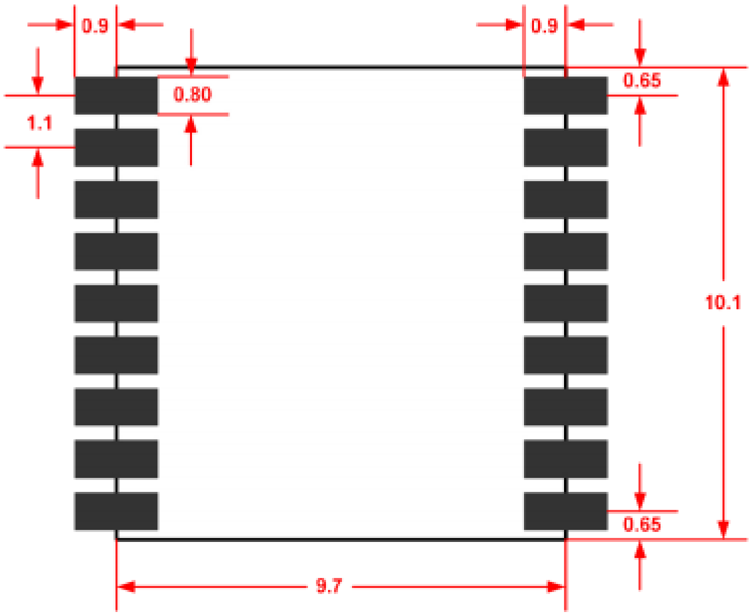
Module Introduction

- Outline Dimension (Unit: mm)

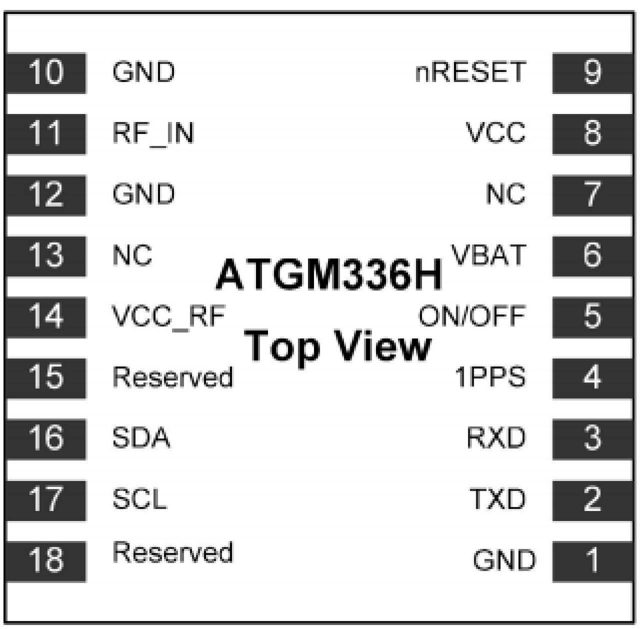


Symbol	Min.(mm)	Typ.(mm)	Max.(mm)
A	10.0	10.1	10.7
B	9.6	9.7	9.8
C	2.2	2.4	2.6
D	0.55	0.65	0.95
E	1.0	1.1	1.2
F		0.8	
G	0.4	0.5	0.6
H	0.8	0.9	1.0
K	0.7	0.8	0.9

- PCB layout (Unit: mm)



- PinOut



- Pin Definition

Number	Name	I/O	Description	Electrical
1	GND	I	Ground	
2	TXD	O	Navigation Data Output	NMEA0183 Protocol

Number	Name	I/O	Description	Electrical
3	RXD	I	Interactive Command Input	Configuration Command Input
4	1PPS	O	Second Pulse Output	

5	ON/OFF	I	Module Shutoff control, active Low	
6	VBAT	I	RTC and SRAM backup power supply	Power with 1.5~3.6V to insure a warm start for the module
7	NC	I		
8	VCC	I	Module Power Input	DC 3.3V±10%, 100mA
9	nRESET	I	Module Reset Input, active Low	Left floating when not used
10	GND	I	Ground	
11	RF_IN	I	Antenna Signal Input	
12	GND	I	Ground	
13	NC			
14	VCC_RF	O	Power Output	+3.3V, can be used as power for antenna
15	Reserve			Left floating
16	SDA	I/O	I2C data interface	Left floating
17	SCL	O	I2C clock interface	Left floating
18	Reserve			Left floating

- Electrical Parameters

1. Limit Parameter

Parameters	Symbol	Maximum	Minimum	Unit
Power Supply Voltage (VCC)	Vcc	-0.3	3.6	V
Backup Battery Voltage (VBAT)	Vbat	-0.3	3.6	V
Digital Input Pin Voltage	Vin	-0.3	Vcc+0.2	V

Digital Input Pin Voltage	Vin	-0.3	Vcc+0.2	V
Parameters	**Symbol**	**Maximum**	**Minimum**	**Unit**
Maximum acceptable ESD level	VESD (HBM)		2000	V

2. Operating Conditions

Parameter	Symbol	Maximum	Typical	Minimum	Unit
Power Supply Voltage	VCC	2.7	3.3	3.6	V
Vcc Peak Current(Antenna not included)	Ipeak			100	mA
Backup Power	Vbat	1.5	3.0	3.6	V
Backup Power(Vbat Current)	Ibat		10		uA
Input Pin	Vil			0.2*Vcc	V
Input Pin	Vih	0.7*Vcc			V
Output Pin	Vol Io=-12mA			0.4	V
Output Pin	Voh Io=12mA	Vcc-0.5			V
Active Antenna Output Voltage	VCC_RF		3.3		V
Antenna short circuit protection current VCC_RF (=3.3V)	Lant Short			50	mA
Antenna open circuit current Power from VCC_RF (=3.3V)	Lant Open		3		mA
Antenna Gain	Gant	15		30	dB

- Technical Specification

	Technical Paramter
Signal Receive	BDS/GPS/GLONASS/GALILEO/QZSS/SBAS
Number of RF channels	Three-channel RF, support the full sonstellation BDS, GPS and GLONASS receiving at the same time
Cold Start TTFF	≤35s
Hot Start TTFF	≤1s

Recapture TTFF	**Technical Paramter** ≤1s

Cold Start capture sensitivity	-148dBm
Hot Start capture sensitivity	-156dBm
Recapture sensitivity	-160dBm
Tracking sensitivity	-162dBm
Positioning precision	<2m（1σ）
Speed measurement precision	＜0.1m/s（1σ）
Timing precision	<30ns（1σ）
Positioning update rate	1Hz(default), Maximum 10Hz
Serial port characteristics	Baud rate range：4800 bps~115200 bps, default
9600bps, 8 data bits, No check, 1 stop bit	
Protocol	NMEA0183
Max Height	18000m
Max Speed	515m/s
Max Acceleration	4g
Backup battery	1.5V ~ 3.6V
Supply Voltage	2.7V ~ 3.6V
GPS&BD Typical power consumption	<25mA @3.3V
Working Temperature	-40 to +85°C
Storage Temperature	-45 to +125°C

	Technical Paramter
Size	10.1mm×9.7mm×2.4mm
Weight	0.6g

Data Analysis

It is recommended to test this module in an outdoor open place since if the antenna is placed on a balcony, its signal may be influenced by the buildings around. The positioning can be done within one minute in open spaces. When the onboard LED keeps flashing at a regular frequency, the positioning is completed. The default baudrate is 9600. Now let us check the data in the serial monitor.

$GNGGA,084852.000,2236.9453,N,11408.4790,E,1,05,3.1,89.7,M,0.0,M,,*48

$GNGLL,2236.9453,N,11408.4790,E,084852.000,A,A*4C

$GPGSA,A,3,10,18,31,,,,,,,,,,6.3,3.1,5.4*3E

$BDGSA,A,3,06,07,,,,,,,,,,,6.3,3.1,5.4*24

$GPGSV,3,1,09,10,78,325,24,12,36,064,,14,26,307,,18,67,146,27*71

$GPGSV,3,2,09,21,15,188,,24,13,043,,25,55,119,,31,36,247,30*7F

$GPGSV,3,3,09,32,42,334,*43

$BDGSV,1,1,02,06,68,055,27,07,82,211,31*6A

**$GNRMC,084852.000,A,2236.9453,N,11408.4790,E,0.53,292.44,1412 16,,,A*7 5
$GNVTG,292.44,T,,M,0.53,N,0.98,K,A*2D**

$GNZDA,084852.000,14,12,2016,00,00*48

$GPTXT,01,01,01,ANTENNA OK*35

There are three data types in the data: GN, GP and BD, which respectively represent the dual-mode mode, GPS mode, and Beidou mode.

NMEA0318 protocol frame format content can refer to the following forms:

(1) $GPGGA (GPS location information)

Num	Name	Example	Unit	Description
	Message ID	$GPGGA		GGA protocol header
<1>	UTC Position	161229.487		hhmmss.sss

Num	Name	Example	Unit	Description
<1>	UTC Position	161229.487		hhmmss.sss
<2>	Latitude	3723.2475		ddmm.mmmm
<3>	Latitude Direction	N		N=north or S=south
<4>	Longitude	12158.3416		dddmm.mmmm
<5>	Longitude	W		E=east or W=west
<6>	Position Fix Indicator	1		0: Fix not avilable or invalid 1: GPS SPS Mode, fix valid 2: Differential GPS, SPS Mode, fix valid 3: GPS PPS Mode, fix valid
<7>	Number of satellites in use	07		Range 00 to 12
<8>	Horizontal dilution of precision	1.0		Range: 0.5-99.9
<9>	Sea level height	9.0	meters	Range: -9999.9-9999.9
<10>	Units		M	M stands for meter
<11>	Geoid Separation		meters	Range: -999.9-9999.9
<12>	Units		M	M stands for meter
<13>	Age of correction data		Seconds	Empty when no differentoal data is presented
<14>	Diff. Ref. Station ID	0000		Range: 0000-1023 (empty when no differential data is presented)
hh	Checksum	18		Checksum of all characters ASCII codes between $ and * (XOR each byte to get checksum, then convert it into ASCII characters in hexadecimal)

Num	Name	Example	Unit	Description
	CR LF			End of protocol frame

(2) $GPGLL (Geographic Position Information)

Num	Name	Example	Units	Description
	Message ID	$GPGLL		GLL protocol header
<1>	Latitude	3723.2475		ddmm.mmmm
<2>	Latitude Direction	N		N: north; S: south
<3>	Longitude	12158.3416		dddmm.mmmm
<4>	Longitude Direction	W		W: west; E: east
<5>	UTC Position	161229.487		hhmmss.sss
<6>	Data Statu	A		A=data valid; V=data invalid
hh	Checksum	2C		

(3) $GPGSA (Current Satellites Information)

Num	Name	Example	Unit	Description
	Message ID	$GPGSA		GSA Protocol Data header
<1>	Position Mode	A		M: manually; A: automatically
<2>	Posistion Type	3		1: fix not available
<3>	PRN code number of satellite used on Channel 1	07		PRN(Pseudo Random Noise) Range: 01~32, can receive 12 satellites information at most.
<4>	PRN code number of satellite used on Channel 2	02		Same as above
<5>	PRN code number of satellite used on Channel 3	26		Same as above
<6>	PRN code number of satellite used on Channel 4	27		Same as above

Num	Name	Example	Unit	Description
<7>	PRN code number of satellite used on Channel 5	09		Same as above
<8>	PRN code number of satellite used on Channel 6	04		Same as above
<9>	PRN code number of satellite used on Channel 7	15		Same as above
<10>	PRN code number of satellite used on Channel 8			Same as above
<11>	PRN code number of satellite used on Channel 9			Same as above
<12>	PRN code number of satellite used on Channel 10			Same as above
<13>	PRN code number of satellite used on Channel 11			Same as above
<14>	PRN code number of satellite used on Channel 12			Same as above
<15>	HDOP(Position Dilution Precision)	1.8		Range: 0.5-99.9
<16>	HDOP(Position Dilution Precision)	1.0		Range: 0.5-99.9
<17>	HDOP(Position Dilution Precision)	1.5		Range: 0.5-99.9
hh	Checksum	2C		
	CR LF			End of protocol frame

(4) $GPGSV (Satellites in View)

Num	Name	Example	Unit	Description
	Message ID	$GPGSV		GSV protocol header
<1>	Number of Messages	2		Range: 1 to 3
<2>	Message Serial Number	1		Range: 1 to 3
<3>	Satellites in View	07		Range: 00-12
<4>	Satellite ID	07		Range: 01-32
<5>	Elevation	79	Degree	Range: 00-90
<6>	Azimuth	Azimuth	Degree	Range: 000-359
<7>	SNR (C/No)	42	dBHz	Range 0 to 99, null when not tracking
<4>	Satellite ID	02		Range: 01-32
<5>	Elevation	51	Degree	Range: 00-90
<6>	Azimuth	062	Degree	Range: 000-359
<7>	SNR (C/No)	43	dBHz	Range 0 to 99
<4>	Satellite ID	26		Range: 01-32
<5>	Elevation	36	Degree	Range: 00-90
<6>	Azimuth	256	Degree	Range: 000-359
<7>	SNR (C/No)	42	dBHz	Range 0 to 99
<4>	Satellite ID	27		Range: 01-32
<5>	Elevation	27	Degree	Range: 00-90
<6>	Azimuth	138	Degree	Range: 000-359
<7>	SNR (C/No)	42	dBHz	Range: 0 to 99
hh	Checksum	71		
	CR LF			End of protocol frame

Num	Name	Example	Unit	Description
	CR LF			End of protocol frame

(5) $GPRMC (Minimum GNSS Data)

Num	Name	Example	Unit	Description
	Message ID	$GPRMC		RMC Protocol Header
<1>	UTC Position	161229.487		hhmmss.sss
<2>	Position Status	A		A: position; V: navigation
<3>	Latitude	3723.2475		ddmm.mmmm
<4>	Latitude Direction	N		N: north; S: south
<5>	Longitude	12158.3416		dddmm.mmmm
<6>	Longitude	W		W: west; E: east
<7>	Speed Over Ground	0.13	Knots	Range: 000.0-999.9
<8>	Course Over Ground	309.62	Degree	Taking due north as the reference datum, the two-dimensional direction points, which is equivalent to a two-dimensional compass
<9>	Data Status	A		A: data valid; V: data invalid
<10>	Magnetic Variation		Degree	Range: 000-180
<11>	Magnetic declination direction			E: east; W: west
hh	Checksum	10		
	CR LF			End of protocol frame

(6) $GPVTG (Ground Speed Information)

Num	Name	Example	Unit	Description

Num	Name	Example	Unit	Description
	Message	$GPVTG		VTG protocol header
<1>	Course Over Ground	309.62	Degree	Taking due north as the reference datum, the two-dimensional direction points, which is equivalent to a two-dimensional compass
<2>		T		True north reference system
<3>	Magnetic Variation		Degree	
<4>		M		Magnetic north reference system
<5>	Speed over ground	0.13	Knots	Range: 000.0-999.9
<6>		N		Knots
<7>	Horizontal velocity	0.2		
<8>		K		km/h
hh	Checksum	6E		
	CR LF			End of protocol frame

(7) Antenna Status Output

$GPTXT，01,01,01, ANTENNA OK*35

"Ok" means that the antenna has been detected, and "open" represents the antenna is disconnected.

(8) UTC time and Current time in Beijing

$GNGGA,**084852.000**,2236.9453,N,11408.4790,E,1,05,3.1,89.7,M,0.0,M,,*48

The numbers in bold represent UTC time. Its format is hhmmss.sss. The three digits after the decimal point should be omitted so the numbers above means that it is 08:48:52.

UTC + Time Zone Difference = Local Time

The time of eastern zone is positive, the western zone is negative. The time in Beijing follows the time offset of UTC+08:00 so the current time in Beijing is 16:48:52.

(9) Format of latitudes and longitudes

$GNRMC,084852.000,A,**2236.9453,N,11408.4790,E**,0.53,292.44,141216,,,A*7 5

Data format: ddd°mm.mmm′ Convert to the format of Google or Baidu Map. Latitude: ddmm.mmmm, Northern Latitude 2236.9453， 22+(36.9453/60)= 22.615755 Longitude: dddmm.mmmm, East Longitude 11408.4790， 114+(08.4790/60)=114.141317

(10) Description of hot start, warm start and cold start

The Cold Start refers to the process of starting GPS in an unfamiliar environment until it connects with the surrounding satellites and calculates coordinates.

The following three situations are all cold start:

1. Use for the first time;

2. Ephemeris information lost due to batterry depletion;

3. Move the receiver more than 1000km in power off state. That is to say, the cold start is a mandatory start-up through hardware. When the GPS has cleared the internal positioning information since the last operation, and the GPS receiver has lost satellites parameters, or the navigator cannot work properly because existing parameters because the existing parameters are too different from the actual received satellite parameters, it is necessary for GPS to obtain the new coordinate data provided by the satellite. A vehicle startting a navigation from a basement is cold start. This is also the reason why it takes a long time to search for satellites from the basement.

Warm start refers to the start-up more than 2 hours from the last positioning time. The positioning time is between cold start and hot start. If you have used GPS positioning one day ago, the first startup of next day belongs to warm start, and the last position information will be displayed after startup. The latitude, longitude and altitude of the last operation are known, but since the shutdown time is too long, the ephemeris has changed and the previous cannot accept it. Several satellittes in the parameters have lost contact with the GPS receiver and have to continue searching for additional position information. Therefore, searching time for warm start is longer than that of hot start and shorter than cold start.

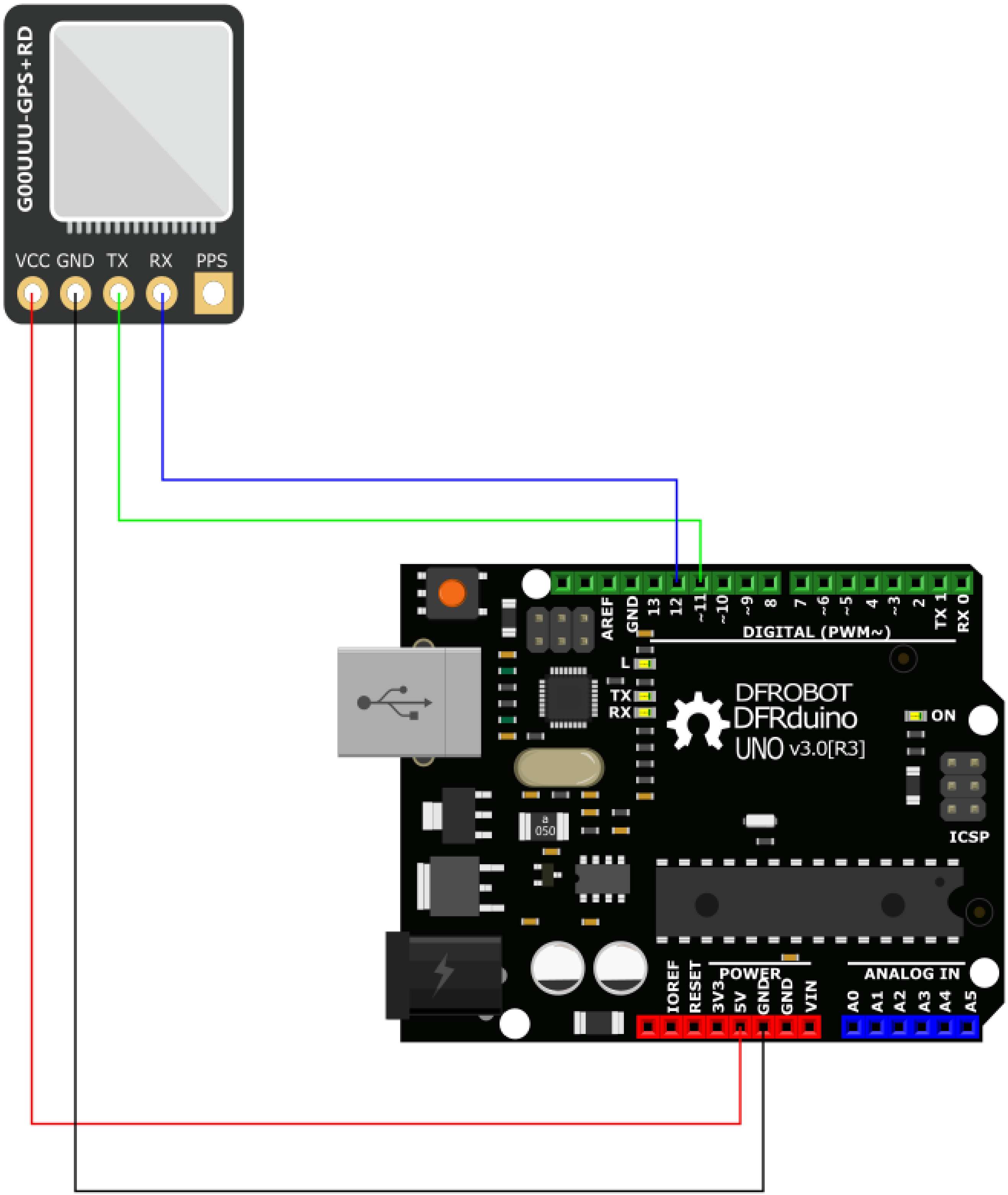
When starting GPS at the place where it was shut down last time, and the time from last positioning time is less than 2 hours, it is hot start. Some preparation work such as saving and closing can be done by software.

Tutorial

Requirements

- **Hardware**
  - Arduino UNO
  - GPS + BDS BeiDou Dual Module x1
  - Wires
- **Software**
  - Arduino IDE (https://www.arduino.cc/en/Main/Software)

Connection Diagram



Sample Code 1 (Read GPS Data)

```
#include <SoftwareSerial.h>
SoftwareSerial GpsSerial(12, 11); //RX,TX

void setup()
{
  Serial.begin(115200);  //Debug Serial
  GpsSerial.begin(9600);  //Gps Serial
}

void loop()
{  while (GpsSerial.available() > 0)
  {
    byte gpsData = GpsSerial.read();
   Serial.write(gpsData);
  }
}
```

Expected Results 1

```
14:49:53.504 -> $BDGSV,2,1,07,02,,,14,03,,,03,06,,,18,07,40,150,20*51
14:49:53.545 -> $BDGSV,2,2,07,09,60,331,26,10,,,24,16,60,021,20*62
14:49:53.587 -> $GNRMC,064953.000,A,3040.04847,N,10348.56476,E,5.18,148.47,291119,,,A*7C
14:49:53.668 -> $GNVTG,148.47,T,,M,5.18,N,9.58,K,A*25
14:49:53.712 -> $GNZDA,064953.000,29,11,2019,00,00*44
14:49:53.753 -> $GPTXT,01,01,01,ANTENNA OK*35
14:49:54.078 -> $GNGGA,064954.000,3040.04684,N,10348.56480,E,1,09,1.7,491.2,M,0.0,M,,*71
14:49:54.170 -> $GNGLL,3040.04684,N,10348.56480,E,064954.000,A,A*45
14:49:54.221 -> $GPGSA,A,3,10,14,20,31,32,34,,,,,,,4.5,1.7,4.1*32
14:49:54.269 -> $BDGSA,A,3,07,09,16,,,,,,,,,,4.5,1.7,4.1*28
14:49:54.314 -> $GPGSV,3,1,09,10,83,085,24,12,26,070,15,14,37,295,28,20,52,141,29*78
14:49:54.397 -> $GPGSV,3,2,09,25,,,22,31,37,226,20,32,54,329,32,33,,,24*7B
14:49:54.442 -> $GPGSV,3,3,09,34,56,093,22*4E
14:49:54.482 -> $BDGSV,2,1,07,02,,,14,03,,,03,06,,,18,07,40,150,20*51
14:49:54.561 -> $BDGSV,2,2,07,09,60,331,26,10,,,24,16,60,021,20*62
14:49:54.610 -> $GNRMC,064954.000,A,3040.04684,N,10348.56480,E,5.44,146.67,291119,,,A*76
14:49:54.690 -> $GNVTG,146.67,T,,M,5.44,N,10.08,K,A*1D
14:49:54.731 -> $GNZDA,064954.000,29,11,2019,00,00*43
14:49:54.773 -> $GPTXT,01,01,01,ANTENNA OK*35
14:49:55.075 -> $GNGGA,064955.000,3040.04529,N,10348.56511,E,1,10,1.0,491.5,M,0.0,M,,*75
14:49:55.174 -> $GNGLL,3040.04529,N,10348.56511,E,064955.000,A,A*49
14:49:55.212 -> $GPGSA,A,3,10,12,14,20,31,32,34,,,,,,2.3,1.0,2.1*30
14:49:55.297 -> $BDGSA,A,3,07,09,16,,,,,,,,,,2.3,1.0,2.1*29
14:49:55.337 -> $GPGSV,3,1,09,10,83,085,24,12,26,070,30,14,37,295,28,20,52,141,29*7F
```

14:49:55.378 -> $GPGSV,3,2,09,25,,,22,31,37,226,20,32,54,329,33,33,,,24*7A
14:49:55.470 -> $GPGSV,3,3,09,34,56,093,22*4E

Sample Code 2 ($GNRMC GPS Data Analysis)

```
#include <SoftwareSerial.h>
SoftwareSerial GpsSerial(12, 11);  //RX,TX

struct
{
  char GPS_DATA[80];
  bool GetData_Flag;      //Get GPS data flag bit
  bool ParseData_Flag;    //Parse completed flag bit
  char UTCTime[11];       //UTC time
  char latitude[11];      //Latitude
  char N_S[2];            //N/S
  char longitude[12];     //Longitude
  char E_W[2];            //E/W
  bool Usefull_Flag;      //If the position information is valid flag bit
} Save_Data;

const unsigned int gpsRxBufferLength = 600;
char gpsRxBuffer[gpsRxBufferLength];
unsigned int gpsRxLength = 0;

void setup()
{
  Serial.begin(115200);   //Debug Serial
  GpsSerial.begin(9600);  //Gps Serial

  Serial.println("DFRobot Gps");
  Serial.println("Wating...");

  Save_Data.GetData_Flag = false;
  Save_Data.ParseData_Flag = false;
  Save_Data.Usefull_Flag = false;
}

void loop()
{
  Read_Gps();         //Get GPS data
  parse_GpsDATA();    //Analyze GPS data
  print_GpsDATA();    //Output analyzed data
}

void Error_Flag(int num)
{
  Serial.print("ERROR");
  Serial.println(num);
```

```
  while (1)
  {
    digitalWrite(13, HIGH);
    delay(500);

    digitalWrite(13, LOW);
    delay(500);
  }
}

void print_GpsDATA()
{
  if (Save_Data.ParseData_Flag)
  {
    Save_Data.ParseData_Flag = false;

    Serial.print("Save_Data.UTCTime = ");
    Serial.println(Save_Data.UTCTime);

    if(Save_Data.Usefull_Flag)
    {
      Save_Data.Usefull_Flag = false;
      Serial.print("Save_Data.latitude = ");
      Serial.println(Save_Data.latitude);
      Serial.print("Save_Data.N_S = ");
      Serial.println(Save_Data.N_S);
      Serial.print("Save_Data.longitude = ");
      Serial.println(Save_Data.longitude);
      Serial.print("Save_Data.E_W = ");
      Serial.println(Save_Data.E_W);
    }
    else
    {
      Serial.println("GPS DATA is not usefull!");
    }
  }
}

void parse_GpsDATA()
{
  char *subString;
  char *subStringNext;
  if (Save_Data.GetData_Flag)
  {
    Save_Data.GetData_Flag = false;
    Serial.println("************************");
    Serial.println(Save_Data.GPS_DATA);

    for (int i = 0 ; i <= 6 ; i++)
    {
```

```
      if (i == 0)
      {
        if ((subString = strstr(Save_Data.GPS_DATA, ",")) == NULL)
          Error_Flag(1);    //Analysis error

      }
      else
      {
        subString++;
        if ((subStringNext = strstr(subString, ",")) != NULL)
        {
          char usefullBuffer[2];
          switch(i)
          {
            case 1:memcpy(Save_Data.UTCTime, subString, subStringNext - subString);br
            case 2:memcpy(usefullBuffer, subString, subStringNext - subString);break;
            case 3:memcpy(Save_Data.latitude, subString, subStringNext - subString);b
            case 4:memcpy(Save_Data.N_S, subString, subStringNext - subString);break;
            case 5:memcpy(Save_Data.longitude, subString, subStringNext - subString);
            case 6:memcpy(Save_Data.E_W, subString, subStringNext - subString);break;

            default:break;
          }
          subString = subStringNext;
          Save_Data.ParseData_Flag = true;
          if(usefullBuffer[0] == 'A')
            Save_Data.Usefull_Flag = true;
          else if(usefullBuffer[0] == 'V')
            Save_Data.Usefull_Flag = false;
        }
        else
        {
          Error_Flag(2);    //Analysis error
        }
      }
    }
  }
}

void Read_Gps()
{
  while (GpsSerial.available())
  {
    gpsRxBuffer[gpsRxLength++] = GpsSerial.read();
    if (gpsRxLength == gpsRxBufferLength)RST_GpsRxBuffer();
  }

  char* GPS_DATAHead;
  char* GPS_DATATail;
  if ((GPS_DATAHead = strstr(gpsRxBuffer, "$GPRMC,")) != NULL || (GPS_DATAHead = strs
```

```
    if ((GPS_DATAHead = strstr(gpsRxBuffer, "$GPRMC,")) != NULL || (GPS_DATAHead = strs
    {
      if (((GPS_DATATail = strstr(GPS_DATAHead, "\r\n")) != NULL) && (GPS_DATATail > GP
      {
        memcpy(Save_Data.GPS_DATA, GPS_DATAHead, GPS_DATATail - GPS_DATAHead);

        Save_Data.GetData_Flag = true;

        RST_GpsRxBuffer();
      }
    }
  }

  void RST_GpsRxBuffer(void)
  {
    memset(gpsRxBuffer, 0, gpsRxBufferLength);      //Clear
    gpsRxLength = 0;
```

Result 2

```
13:54:53.641 -> Save_Data.UTCTime = 055453.000
13:54:53.690 -> Save_Data.latitude = 3040.09146
13:54:53.724 -> Save_Data.N_S = N
13:54:53.773 -> Save_Data.longitude = 10348.55584
13:54:53.773 -> Save_Data.E_W = E
13:54:54.574 -> **************
13:54:54.574 -> $GNRMC,055454.000,A,3040.09175,N,10348.55593,E,0.00,0.00,251119,,,A*76
13:54:54.641 -> Save_Data.UTCTime = 055454.000
13:54:54.692 -> Save_Data.latitude = 3040.09175
13:54:54.741 -> Save_Data.N_S = N
13:54:54.741 -> Save_Data.longitude = 10348.55593
13:54:54.791 -> Save_Data.E_W = E
13:54:55.577 -> **************
13:54:55.577 -> $GNRMC,055455.000,A,3040.09204,N,10348.55601,E,0.00,0.00,251119,,,A*7A
13:54:55.671 -> Save_Data.UTCTime = 055455.000
13:54:55.671 -> Save_Data.latitude = 3040.09204
13:54:55.717 -> Save_Data.N_S = N
13:54:55.754 -> Save_Data.longitude = 10348.55601
13:54:55.790 -> Save_Data.E_W = E
13:54:56.570 -> **************
13:54:56.618 -> $GNRMC,055456.000,A,3040.09228,N,10348.55606,E,0.00,0.00,251119,,,A*70
13:54:56.664 -> Save_Data.UTCTime = 055456.000
13:54:56.705 -> Save_Data.latitude = 3040.09228
13:54:56.738 -> Save_Data.N_S = N
13:54:56.773 -> Save_Data.longitude = 10348.55606
13:54:56.822 -> Save_Data.E_W = E
13:54:57.582 -> **************
13:54:57.582 -> $GNRMC,055457.000,A,3040.09259,N,10348.55615,E,0.00,0.00,251119,,,A*75
13:54:57.687 -> Save_Data.UTCTime = 055457.000
13:54:57.687 -> Save_Data.latitude = 3040.09259
```

```
13:54:57.721 -> Save_Data.N_S = N
13:54:57.772 -> Save_Data.longitude = 10348.55615
13:54:57.819 -> Save_Data.E_W = E
```

FAQ

More Documents

Get **GPS +BDS BeiDou Dual Module** (https://www.dfrobot.com/product-2051.html) from DFRobot Store or **DFRobot Distributor**. (https://www.dfrobot.com/index.php?route=information/distributorslogo)

Turn to the Top