

Project

Make Your Own GPS Transmitter with the HC-12 Transceiver

November 10, 2016 by [Mark Hughes](#)

Create a tracking device with an HC-12 transceiver, a GPS module, an Arduino, and Google Maps.

The first article in this two-part series, [Understanding and Implementing the HC-12 Wireless Transceiver Module](#), uses the HC-12 to create long-distance data transmission between two Arduino Unos. This article uses a pair of HC-12 transceivers, a GPS module, an Arduino, and Google Maps to create a very simple tracking device.

Item	Cost	More Information
HC-12 transceiver (x2)	\$4	Datasheet
GPS Receiver	\$16	Datasheet
or Adafruit GPS Logger Shield	\$45	Project Guide
Arduino Uno R3 (or compatible)	\$45	Reference

[Part one](#) of this two-part series discussed the HC-12 transceiver module and described how to hook it up to an Arduino and a power source.

In this article, we will create a remote GPS receiver that can be used to track nearby items without using a cellular network.

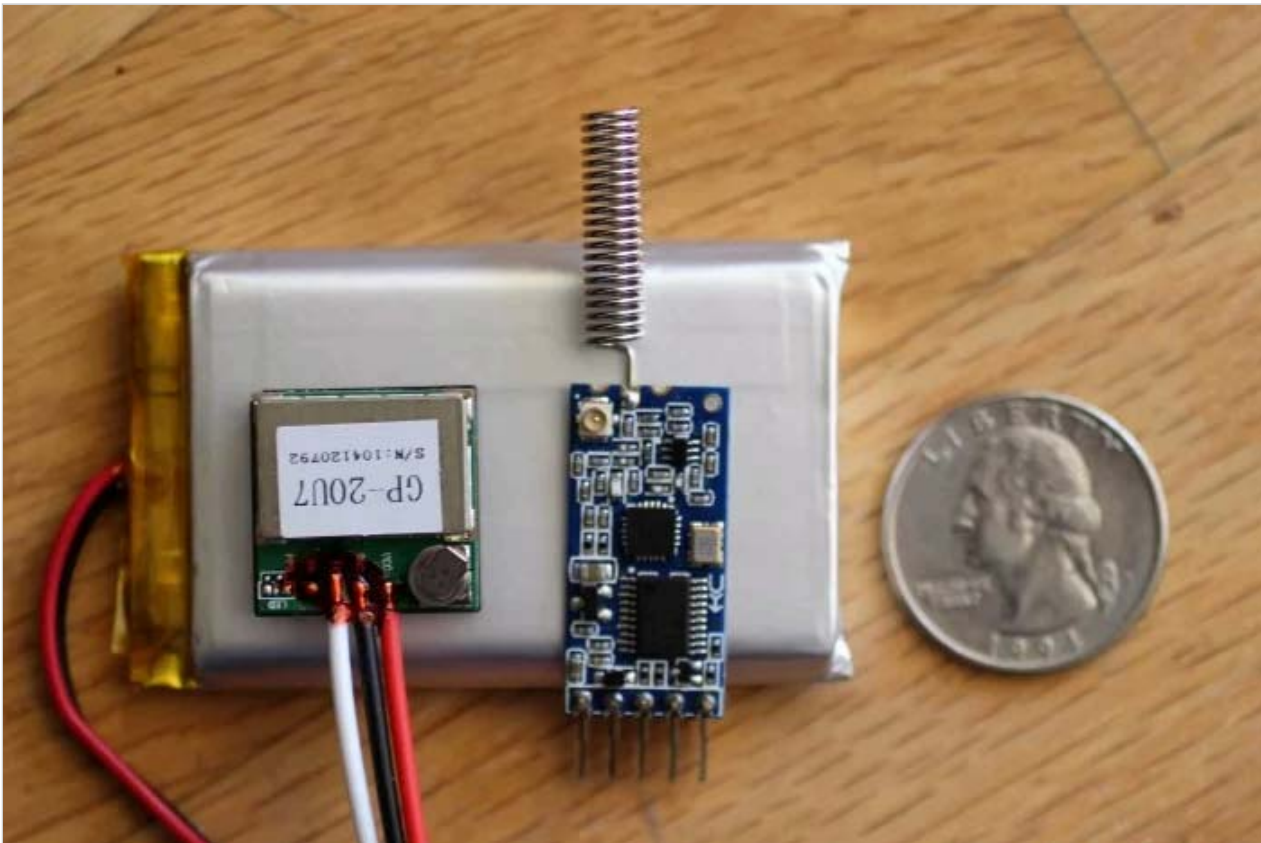
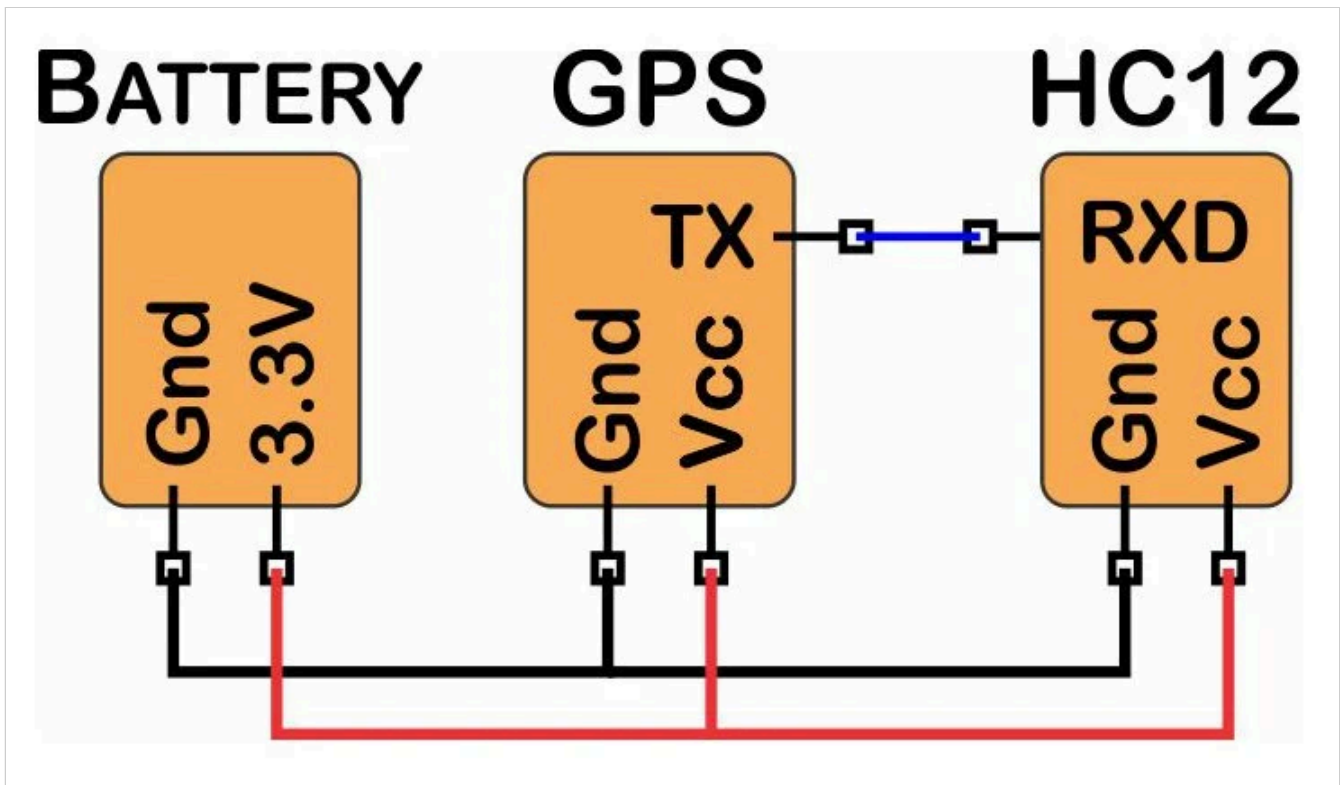
For further information on the transceiver module, please see the [HC-12 datasheet](#) (PDF).

Adding and Transmitting GPS

The Global Positioning System (GPS) allows users to accurately determine the location of objects on or above the surface of the Earth. Both of the GPS receivers listed at the top of this article transmit [National Marine Electronics Association \(NMEA\) sentences](#) that provide information that includes latitude and longitude, altitude, time, bearing, speed, and a great many other variables at 9600 baud.

The HC-12s can transmit the information from a GPS receiver with no additional programming or circuitry.

You can transmit GPS coordinates to remote locations with as little as a GPS receiver, an HC-12 transceiver, and a battery. Remotely transmitted coordinates would have to be received by another HC-12 transceiver and then processed with a microcontroller or computer.



A simple setup such as the one above would allow you to create a small remote object tracker; it will alert you if the object leaves a predefined area, and then you will have a certain amount of time to track the object before the transmitter goes out of range. You could use this with a vehicle, a pet, or even—if you are concerned about theft—the giant pumpkin that you are growing for the state fair.

If you are in an area with clear line of sight, the transmitters will broadcast up to one kilometer, which is a [15-minute walk](#) (or 5-minute run). The maximum range in urban areas will decrease but should remain adequate to alert you if your luggage is leaving the train station without you, or let you know where your dog ventures when he escapes from your yard.

We can refine this project by passing the GPS data to an Arduino instead of directly to the HC-12. Then, only selected strings can be transmitted to the HC-12. This is useful if you want to increase the range of the HC-12 communication by decreasing the over-the-air baud rate.

The \$16 SparkFun GPS module has a default transmission rate of 9600 baud, and this corresponds to an over-the-air rate of 15000 baud. Sending the 9600 baud GPS data to the Arduino and then transmitting only selected information to the HC-12 at 2400 baud will reduce the over-the-air rate to 5000 baud. According to the datasheet, this improves receiver sensitivity by at most 5 dB, which provides a modest increase in range.

The SparkFun GPS receiver provides six sentences at 9600 baud: [GPRMC](#), [GPVTG](#), [GPGGA](#), [GPGSA](#), [GPGSV](#), and [GPGLL](#).

The GPS shield from Adafruit, which is significantly more expensive, can be programmed to transmit select sentences at all standard baud rates.

This is an example data transmission from the receiver to an Arduino. You can decode your own strings with this [online tool](#).

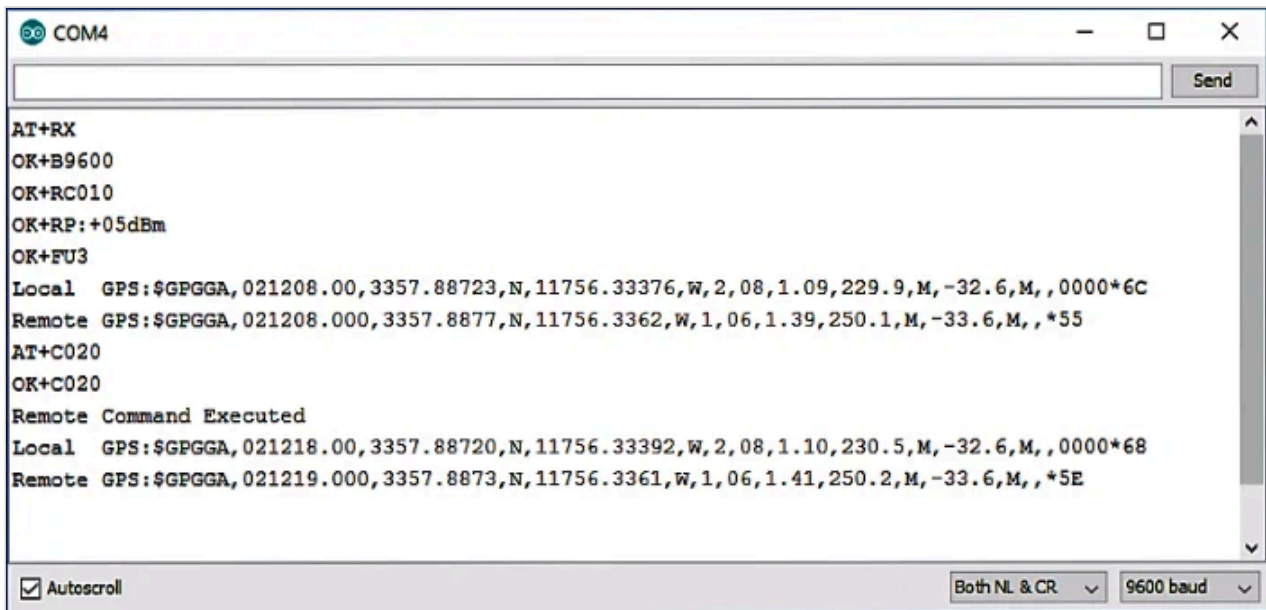
```
$GPRMC,210154.00,A,3358.88969,N,11756.33387,W,0.824,,200916,,,A*6F
$GPVTG,,T,,M,0.824,N,1.527,K,A*2C
$GPGGA,210154.00,3358.88969,N,11756.33387,W,1,05,1.67,254.3,M,-32.6,M,,*6E
$GPGSA,A,3,13,05,21,18,29,,,,,,,,,3.15,1.67,2.66*01
$GPGSV,3,1,11,05,20,044,23,10,10,223,,13,26,083,22,15,37,120,*70
$GPGSV,3,2,11,16,11,322,,18,45,224,23,20,76,043,,21,55,312,22*76
$GPGSV,3,3,11,25,23,195,17,26,27,298,,29,72,108,17*47
$GPGLL,3358.88969,N,11756.33387,W,210154.00,A,A*74
```

Arduino libraries exist that enable decoding of NMEA sentences into the latitude & longitude pairs below:

Type	UTC Time	Position	Speed	Altitude	HDOP, VDOP, PDOP	Satellites
RMC	2016-09-20T21:01:54Z	33°58'53.38"N, 117°56'20.03"W	0.824 knots			
VTG			0.824 knots			
GGA	2016-09-20T21:01:54Z	33°58'53.38"N, 117°56'20.03"W		254.3		5
GSA					1.67, 2.66, 3.15	5
GSV						11
GSV						11
GSV						11
GLL	2016-09-20T21:01:54Z	33°58'53.38"N, 117°56'20.03"W				

RMC (recommended minimum information), GGA (3D location and accuracy), and GLL (latitude and longitude) all include latitude, longitude, and time. GGA provides altitude, and GSA provides the [dilution of precision](#) of the reading (lower numbers indicate greater precision).

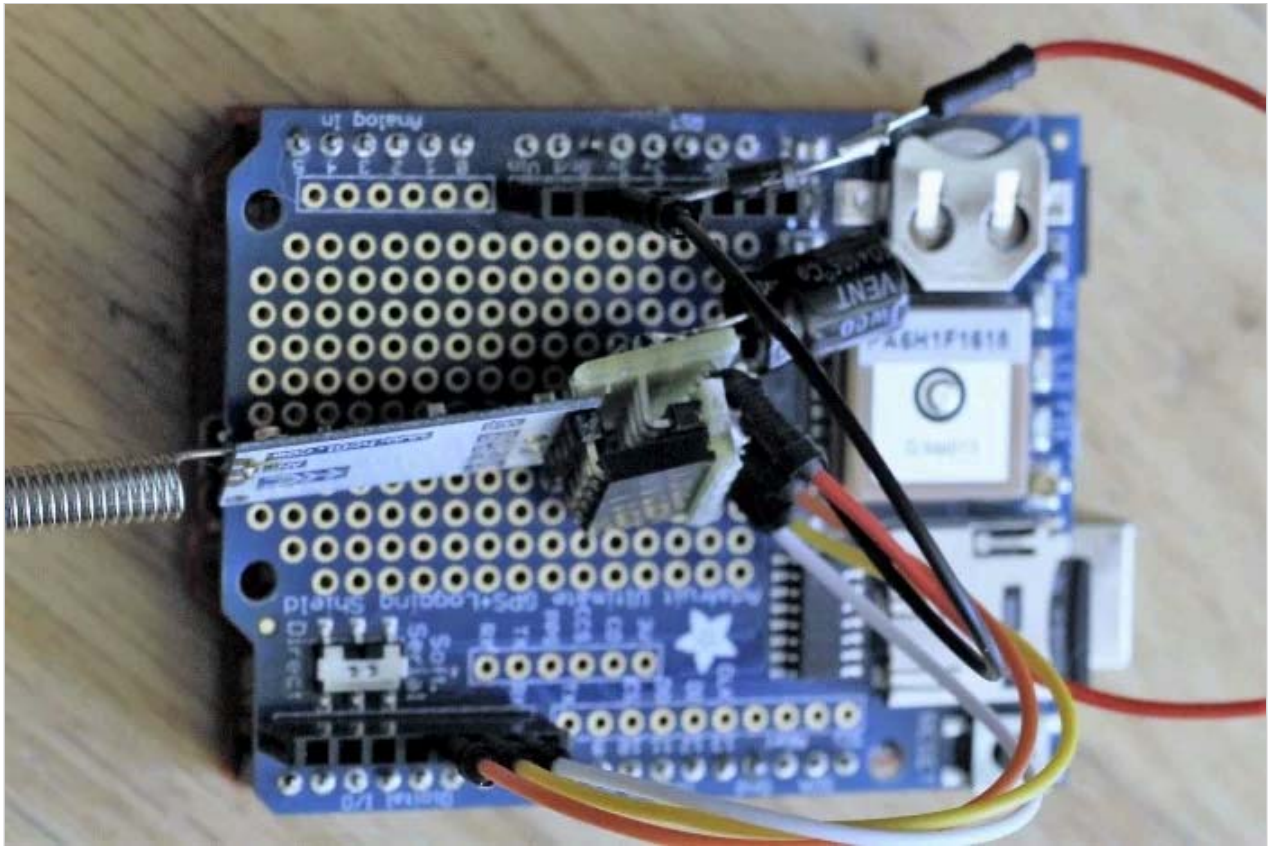
The following program is backward-compatible with the two programs presented in [part one](#). It reads the NMEA sentences sent by the GPS to the Arduino, discards all but the selected sentence, and transmits the selected sentence to a remote Arduino when requested. It works with either the SparkFun GPS receiver or the Adafruit GPS logger shield, as shown below. This program allows users to remotely "ping" distant transceivers to determine their location.



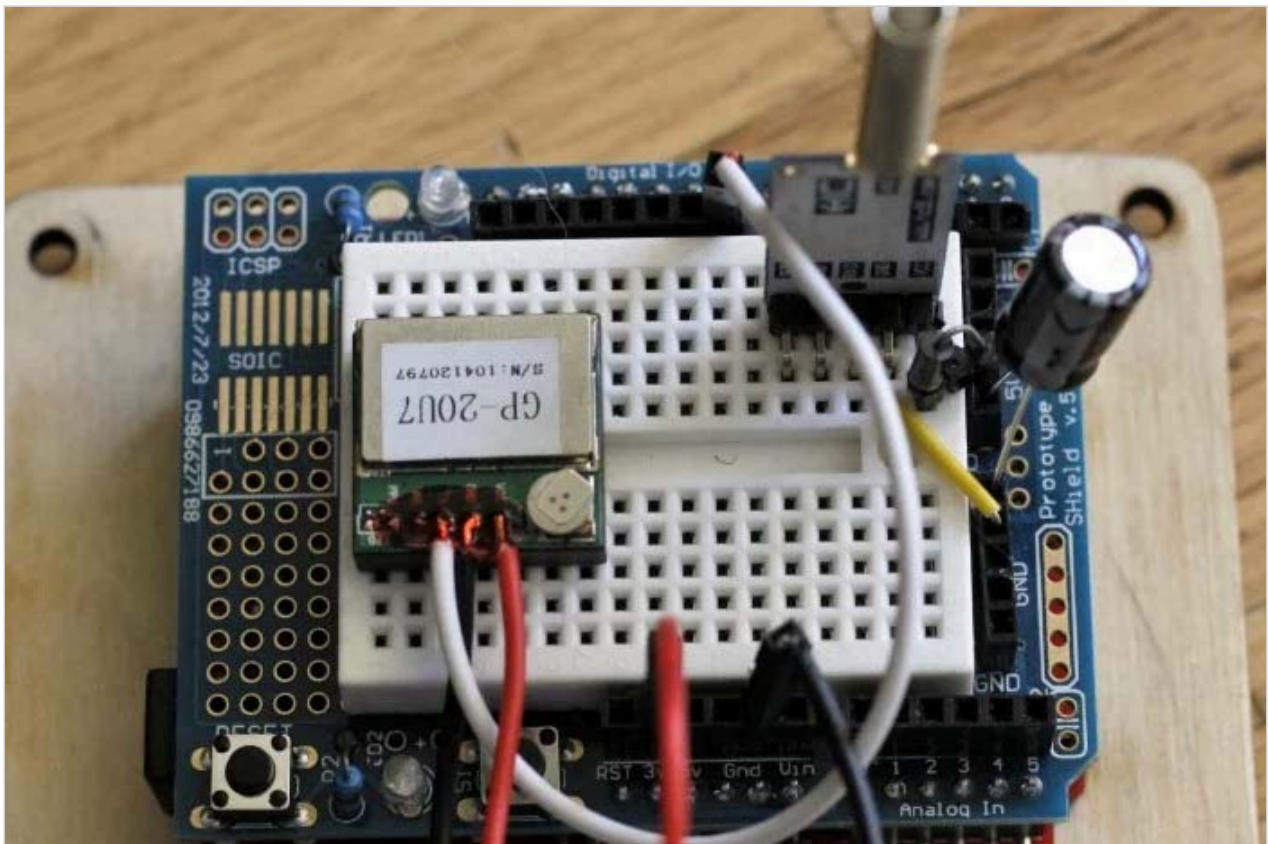
```
COM4
AT+RX
OK+B9600
OK+RC010
OK+RP:+05dBm
OK+FU3
Local  GPS:$GPGGA,021208.00,3357.88723,N,11756.33376,W,2,08,1.09,229.9,M,-32.6,M,,0000*6C
Remote GPS:$GPGGA,021208.000,3357.8877,N,11756.3362,W,1,06,1.39,250.1,M,-33.6,M,,*55
AT+C020
OK+C020
Remote Command Executed
Local  GPS:$GPGGA,021218.00,3357.88720,N,11756.33392,W,2,08,1.10,230.5,M,-32.6,M,,0000*68
Remote GPS:$GPGGA,021219.000,3357.8873,N,11756.3361,W,1,06,1.41,250.2,M,-33.6,M,,*5E
```

☒ Autoscroll Both NL & CR 9600 baud

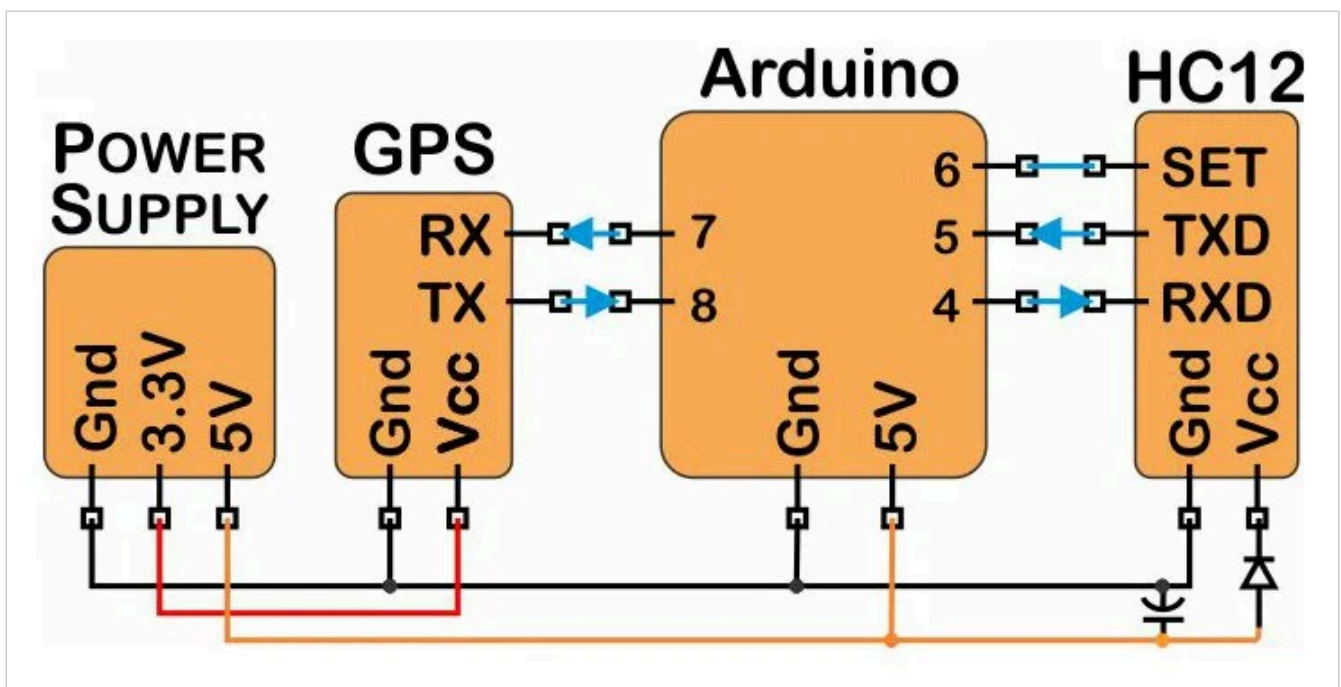
GPS data from a remote transmitter received, via a pair of HC-12 transceivers, by a local Arduino



HC-12 transceiver paired with an Adafruit GPS shield



Scroll to continue with content

HC-12 transceiver paired with a SparkFun GPS module

Connect the power supply, GPS, Arduino, and HC-12 as shown above.

The SparkFun GPS receiver has only three wires; the fourth signal (GPS RXD) is not needed for basic functionality and is not made available to the user. However, if you use the Adafruit shield, the GPS RX pin is enabled, and you can change the refresh rate and [which sentences the GPS transmits](#), eliminating the need for the portion of the Arduino code at the end of the file that simply deletes unwanted sentences.

One potential problem with using the Arduino UNO for this program is that the SoftwareSerial library can only "listen" to one serial port at a time—data sent to a serial port when the software isn't "listening" on

that port will be discarded. The program functions as intended during testing, but I would not consider it a robust solution. If you need multiple serial communication ports in your project, consider the Arduino Mega, or a separate chip such as the [ATSAMD21](#).

If you are tracking a single object near your house, it is sufficient to set upper and lower limits for expected latitude and longitude values. If you are trying to determine the distance between two GPS units, you might consider implementing [Vincenty's formula](#) on a 16-bit or [32-bit microcontroller](#).

```

/* HC12 Send/Receive Example Program 3
   By Mark J. Hughes
   for AllAboutCircuits.com

   This code will automatically detect commands as sentences that begin
   with AT and both write them and broadcast them to remote receivers
   when requested. Changing settings on a local transceiver will also
   change settings on a remote receiver.

   Connect HC12 "RXD" pin to Arduino Digital Pin 4
   Connect HC12 "TXD" pin to Arduino Digital Pin 5
   Connect HC12 "Set" pin to Arduino Digital Pin 6

   Connect GPS GND and 3.3V to Arduino or separate supply
   Connect GPS "RX" to Arduino Digital Pin 7 (optional)
   Connect GPS "TX" to Arduino Digital Pin 8

   Do not power over USB. Per datasheet, power the HC12
   with a supply of at least 100 mA current capability, and
   include a 22 uF - 1000 uF reservoir capacitor.

   Upload code to two Arduinos connected to two computers.

   Transceivers must be at least several meters apart to work in default mode.

*/

#include

//--- Begin Pin Declarations ---//
const byte HC12RxdPin = 4;           // "RXD" Pin on HC12
const byte HC12TxdPin = 5;           // "TXD" Pin on HC12
const byte HC12SetPin = 6;           // "SET" Pin on HC12
const byte GPSPRxdPin = 7;           // "RXD" on GPS (if available)
const byte GPSTxdPin = 8;            // "TXD" on GPS
//--- End Pin Declarations ---//

//--- Begin variable declarations ---//
char byteIn;                         // Temporary variable
String HC12ReadBuffer = "";          // Read/Write Buffer 1 -- Serial
String SerialReadBuffer = "";        // Read/Write Buffer 2 -- HC12
String GPSReadBuffer = "";           // Read/Write Buffer 3 -- GPS

boolean serialEnd = false;            // Flag for End of Serial String
boolean HC12End = false;              // Flag for End of HC12 String
boolean GPSEnd = false;               // Flag for End of GPS String
boolean commandMode = false;          // Send AT commands to remote receivers
boolean GPSLocal = true;              // send GPS local or remote flag
//--- End variable declarations ---//

// Create Software Serial Ports for HC12 & GPS
// Software Serial ports Rx and Tx are opposite the HC12 Rxd and Txd
SoftwareSerial HC12(HC12TxdPin, HC12RxdPin);
SoftwareSerial GPS(GPSTxdPin, GPSPRxdPin);

void setup() {

    HC12ReadBuffer.reserve(82);        // Reserve 82 bytes for message
    SerialReadBuffer.reserve(82);      // Reserve 82 bytes for message
    GPSReadBuffer.reserve(82);         // Reserve 82 bytes for longest NMEA sentence

    pinMode(HC12SetPin, OUTPUT);       // Output High for Transparent / Low for Command
    digitalWrite(HC12SetPin, HIGH);    // Enter Transparent mode
    delay(80);                          // 80 ms delay before operation per datasheet

```

```

Serial.begin(9600); // Open serial port to computer at 9600 Baud
HC12.begin(9600); // Open software serial port to HC12 at 9600 Baud
GPS.begin(9600); // Open software serial port to GPS at 9600 Baud
HC12.listen(); // Listen to HC12
}

void loop() {
  while (HC12.available()) { // If Arduino's HC12 rx buffer has data
    byteIn = HC12.read(); // Store each character in byteIn
    HC12ReadBuffer += char(byteIn); // Write each character of byteIn to HC12ReadBuffer
    if (byteIn == '\n') { // At the end of the line
      HC12End = true; // Set HC12End flag to true.
    }
  }

  while (Serial.available()) { // If Arduino's computer rx buffer has data
    byteIn = Serial.read(); // Store each character in byteIn
    SerialReadBuffer += char(byteIn); // Write each character of byteIn to SerialReadBuffer
    if (byteIn == '\n') { // At the end of the line
      serialEnd = true; // Set serialEnd flag to true.
    }
  }

  while (GPS.available()) {
    byteIn = GPS.read();
    GPSReadBuffer += char(byteIn);
    if (byteIn == '\n') {
      GPSEnd = true;
    }
  }

  if (serialEnd) { // Check to see if serialEnd flag is true
    if (SerialReadBuffer.startsWith("AT")) { // Check to see if a command has been sent
      if (SerialReadBuffer.startsWith("AT+B")) { // If it is a baud change command, delete it immediately
        SerialReadBuffer = "";
        Serial.print("Denied: Changing HC12 Baud does not change Arduino Baudrate");
      }
      HC12.print(SerialReadBuffer); // Send local command to remote HC12 before changing settings
      delay(100); //
      digitalWrite(HC12SetPin, LOW); // If true, enter command mode
      delay(100); // Delay before writing command
      HC12.print(SerialReadBuffer); // Send command to HC12
      Serial.print(SerialReadBuffer); // Send command to serial
      delay(500); // Wait 0.5s for reply
      digitalWrite(HC12SetPin, HIGH); // Exit command / enter transparent mode
      delay(100); // Delay before proceeding
    }
    if (SerialReadBuffer.startsWith("GPS")) {
      HC12.print(SerialReadBuffer);
      GPS.listen();
      GPSLocal = true;
    }
    HC12.print(SerialReadBuffer); // Send text to HC12 to be broadcast
    SerialReadBuffer = ""; // Clear buffer 2
    serialEnd = false; // Reset serialEnd flag
  }

  if (HC12End) { // If HC12End flag is true
    if (HC12ReadBuffer.startsWith("AT")) { // Check to see if a command was received
      digitalWrite(HC12SetPin, LOW); // If true, enter command mode
      delay(40); // Delay before writing command
      HC12.print(HC12ReadBuffer); // Send incoming command back to HC12
      Serial.println(HC12ReadBuffer); // Send command to serial
      delay(1000); // Wait 0.5s for reply
      digitalWrite(HC12SetPin, HIGH); // Exit command / enter transparent mode
      delay(80); // Delay before proceeding
      HC12.println("Remote Command Executed");
    }
    if (HC12ReadBuffer.startsWith("GPS")) {
      GPS.listen();
      HC12.print("Remote GPS Command Received");
      GPSLocal = false;
    }
    Serial.print(HC12ReadBuffer); // Send message to screen
    HC12ReadBuffer = ""; // Empty Buffer
    HC12End = false; // Reset Flag
  }
}

```



```

if (GPSEnd) {
  // Options include GPRMC, GPWGA, GPGLL, etc...
  if (GPSReadBuffer.startsWith("$GPWGA")) {      // Look for target GPS sentence
    if (GPSLocal) {
      Serial.print("Local GPS:");                // Send to local serial port
      Serial.print(GPSReadBuffer);              // Send local GPS
    } else {
      HC12.print("Remote GPS:");                 // Local Arduino responds to remote request
      HC12.print(GPSReadBuffer);                // Sends local GPS to remote
    }
    GPSReadBuffer = "";                          // Delete target GPS sentence
    HC12.listen();                               // Found target GPS sentence, start listening to HC12 again
  } else {
    GPSReadBuffer = "";                          // Delete unwanted strings
  }
  GPSEnd = false;                              // Reset GPS
}
}

```

Data that you collect can be converted into KML files for use in Google Maps using one of many [free online converters](#).



The above image shows variations in the logged GPS coordinates of a stationary object. The error is rather large, even for the low-cost setup used in this project—the GPS receiver sold by SparkFun, for example,

claims to provide positional accuracy of 2.5 m [CEP](#). It is likely that multipath interference made a significant contribution to the additional error.

Real-Time GPS Tracking in Google Earth

Now we will create a GPS tracker with the HC-12 and [Google Earth Pro](#). Through experimentation, I found that tracking worked if at least the \$GPGGA, \$GPGSA, and \$GPGLL strings were passed along to Google Earth.

The program below transmits GPS data to a remote receiver for tracking of remote objects. It receives all sentences at 9600 baud from the computer and then transmits just the \$GPRMC, \$GPGGA, and \$GPGGL sentences at 4800 baud through the HC-12. A separate HC-12/Arduino pair would be needed to receive the information and transmit it to the computer.

```

/* HC12 Send/Receive Example Program 4
   By Mark J. Hughes
   for AllAboutCircuits.com

   Connect HC12 "RXD" pin to Arduino Digital Pin 4
   Connect HC12 "TXD" pin to Arduino Digital Pin 5
   Connect HC12 "Set" pin to Arduino Digital Pin 6

   Connect GPS "TXD" to Arduino Digital Pin 7 (Optional)
   Connect GPS "RXD" to Arduino Digital Pin 8

   Do not power over USB. Per datasheet, power the HC12
   with a supply of at least 100 mA current capability, and
   include a 22 uF - 1000 uF reservoir capacitor.

   Upload code to two Arduinos connected to two computers.

   Transceivers must be at least several meters apart to work in default mode.

*/

#include <SoftwareSerial.h>

//--- Begin Pin Declarations ---//
const byte HC12RxdPin = 4;           // "RXD" Pin on HC12
const byte HC12TxdPin = 5;           // "TXD" Pin on HC12
const byte HC12SetPin = 6;           // "SET" Pin on HC12
const byte GPSTxdPin = 7;            // "TXD" on GPS (if available)
const byte GPSRxdPin = 8;            // "RXD" on GPS
//--- End Pin Declarations ---//

//--- Begin variable declarations ---//
char GPSbyteIn;                      // Temporary variable
String GPSBuffer3 = "";               // Read/Write Buffer 3 -- GPS

boolean debug = false;
boolean HC12End = false;              // Flag for End of HC12 String
boolean GPSEnd = false;              // Flag for End of GPS String
boolean commandMode = false;         // Send AT commands to remote receivers
//--- End variable declarations ---//

// Create Software Serial Ports for HC12 & GPS
// Software Serial ports Rx and Tx are opposite the HC12 Rxd and Txd
SoftwareSerial HC12(HC12TxdPin, HC12RxdPin);
SoftwareSerial GPS(GPSRxdPin, GPSTxdPin);

void setup() {
    buffer3.reserve(82);               // Reserve 82 bytes for longest NMEA sentence

    pinMode(HC12SetPin, OUTPUT);       // Output High for Transparent / Low for Command
    digitalWrite(HC12SetPin, HIGH);   // Enter Transparent mode
    delay(80);                         // 80 ms delay before operation per datasheet
    HC12.begin(4800);                  // Open software serial port to HC12
    GPS.begin(9600);                   // Open software serial port to GPS
    GPS.listen();

```

```

}

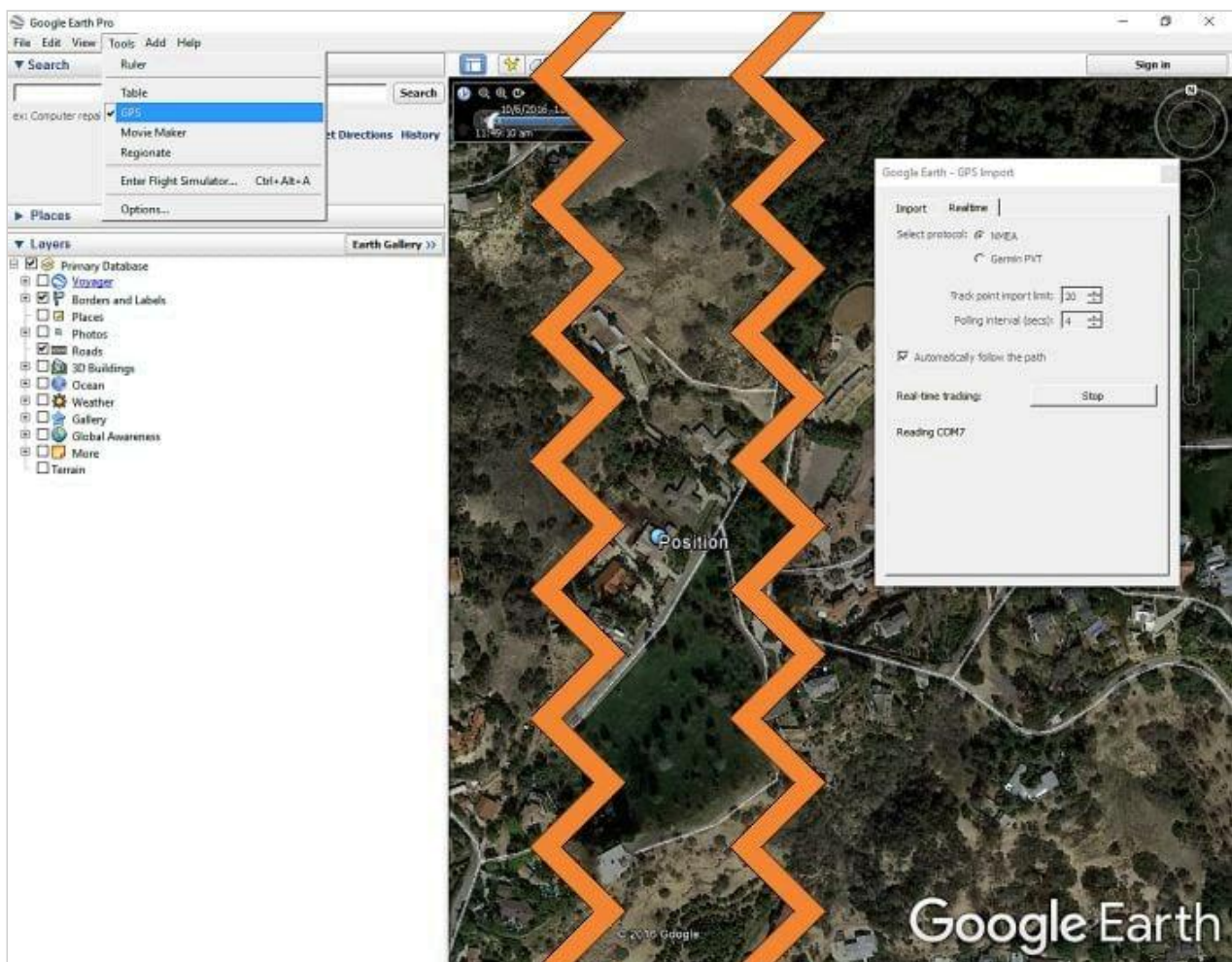
void loop() {
  while (GPS.available()) {
    byteIn = GPS.read();
    buffer3 += char(byteIn);
    if (byteIn == '\n') {
      GPSEnd = true;
    }
  }

  if (GPSEnd) {
    // GPRMC, GPVTG, GPGGA, GPGSA, GPGSV, GPGLL
    if (buffer3.startsWith("$GPRMC") || buffer3.startsWith("$GPGGA") || buffer3.startsWith("$GPGLL")) {
      HC12.print(buffer3);          // Transmit RMC, GGA, and GLL sentences
      buffer3 = "";                // Clear buffer
    } else {
      buffer3 = "";                // Delete GSA, GSV, VTG sentences
    }
    GPSEnd = false;               // Reset GPS flag
  }
}

```

Here are the steps you'll need to follow:

1. Open Google Earth
2. Select Tools -> GPS
3. Select the Realtime tab and click Start. Google Earth will cycle through available serial ports looking for NMEA sentences. When NMEA data is found, it will provide a location on the map.



Three separate screenshots were combined in order to show all the steps simultaneously

Note: An Arduino (or any other microcontroller) is not strictly required for this to work. A GPS module (such as the Adafruit logger shield) that can be programmed to transmit the desired sentences at 4800 baud can be connected directly to an HC-12. Data can then be sent to Google Earth using a separate HC-12 that is connected to a computer's serial port via a logic-level-to-RS232 converter.

Conclusion

This two-article series demonstrates that the HC-12 is a versatile and easy-to-use RF transceiver module. It is similar to the [nRF24L01](#), but it offers the important advantage of longer range. The straightforward UART interface facilitates integration into a wide variety of systems—microcontrollers and PCs can directly communicate with the HC-12.

As this article has shown, the HC-12 is a simple solution for logging GPS data and for real-time GPS tracking.

Featured image courtesy of [SpaceX](#).

- [← Previous Article](#)

Give this project a try for yourself! [Get the BOM](#).

[Load more comments](#)



Join Our Newsletter & Win!

Make sure to be notified of our upcoming Giveaways! Get cutting-edge technical insights and the latest engineering trends delivered weekly to your inbox.

Enter your email address *

☐ I agree to All About Circuit's [Privacy Policy](#).