GPS Tracker - Kit

A super simple and small GPS logger with the ability to attach to GoPro Mounts, Straps, Collars or anything else you wish by swapping out its interchangeable mounting adapters.

Difficulty: Easy

Platforms: TinyCircuits, noads

Estimated time: 2 hours

License: GNU General Public License, version 3 or later (GPL3+)

Items used in the project

Hardware components

- **1x** GPS TinyShield
- **1x** Flash Memory TinyShield
- **3x** M3 6.0mm Screw Zinc Plated Steel
- **1x** TinyDuino Processor Board
- **1x** USB TinyShield Side Mount
- **1x** Lithium Ion Polymer Battery - 3.7V 70mAh
- **1x** USB Cable A/MicroB - 3ft

Software apps & online services

- Arduino IDE

Hand tools & fabrication machines

- 3D Printer

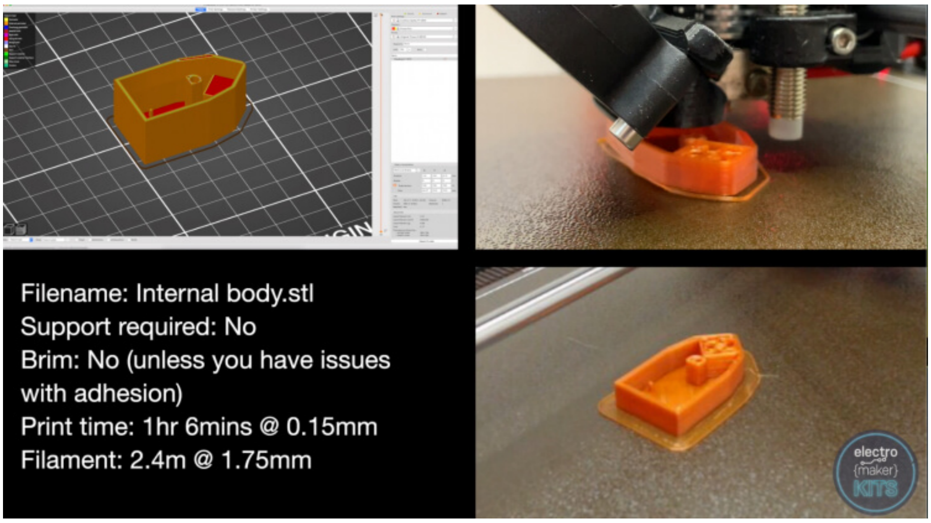
Digital files

The schematics, CAD and code files are available via the Electromaker website at: https://www.electromaker.io/kits/gps-tracker/guide

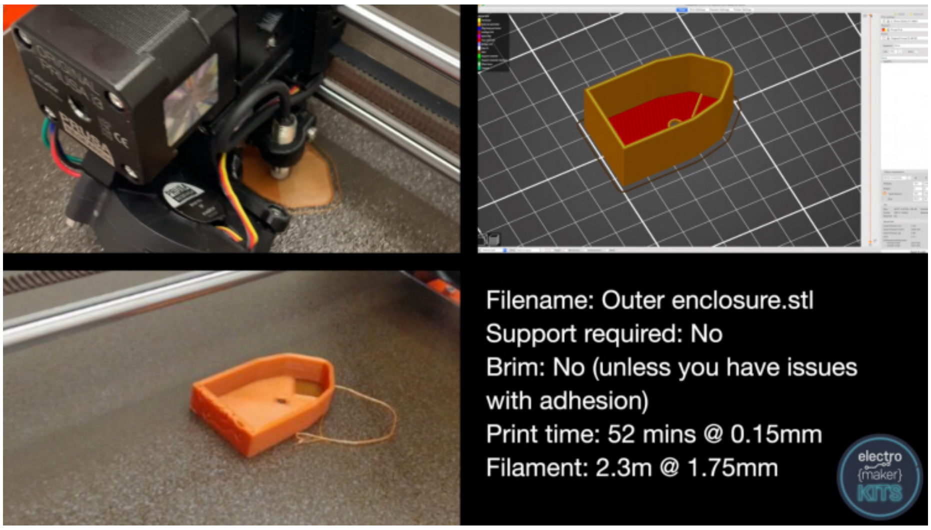
Instructions

The 3D prints

There are two parts you will need to print for the main housing. Neither of them are very large so I printed them both with 0.15mm layer height and my slicers 'quality' profile selected. If you have problems with adhesion then add a wide brim to the object.



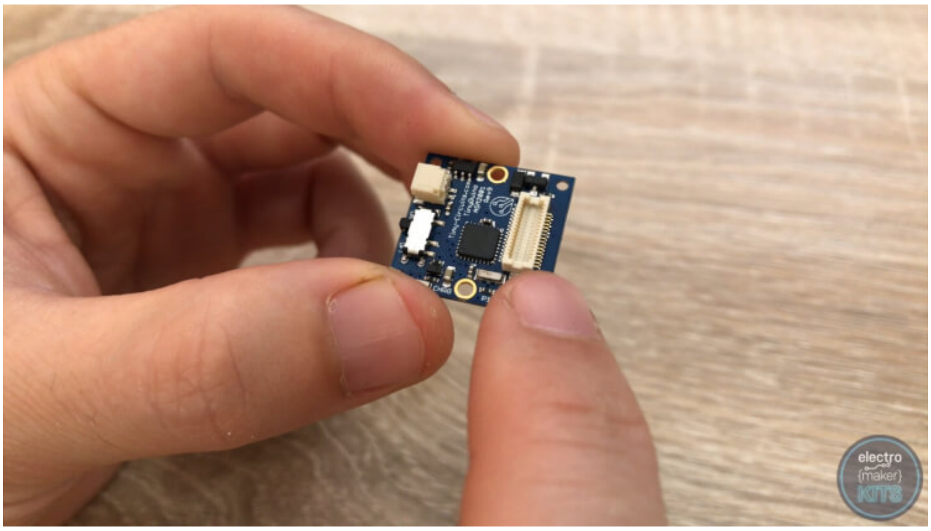
Filename: Internal body.stl
Support required: No
Brim: No (unless you have issues with adhesion)
Print time: 1hr 6mins @ 0.15mm
Filament: 2.4m @ 1.75mm



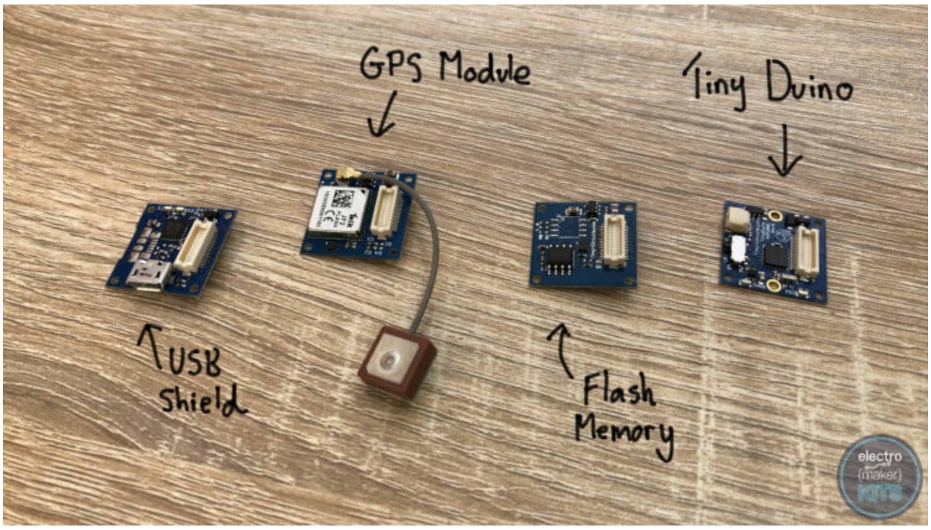
Filename: Outer enclosure.stl
Support required: No
Brim: No (unless you have issues with adhesion)
Print time: 52 mins @ 0.15mm
Filament: 2.3m @ 1.75mm

Assembling the electronics

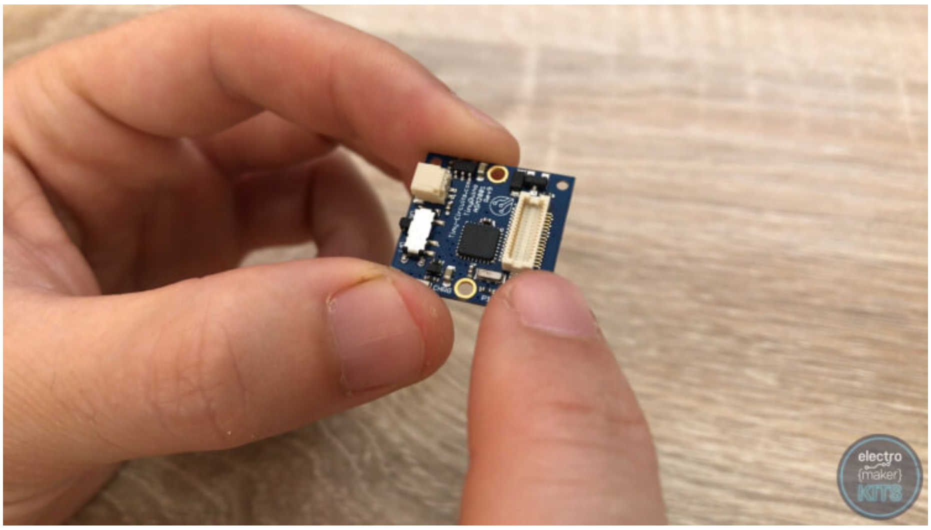
These boards have very easy to use push fittings. These are the large ivory coloured connectors.



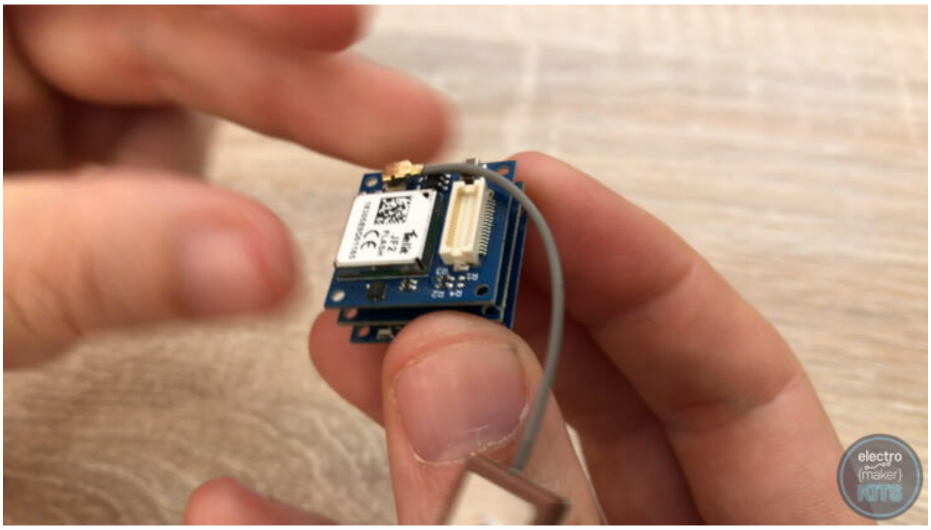
The order we stack the boards together does not affect how they operate, however, the housing that you are printing has been designed to accommodate the boards when stacked in a certain order.



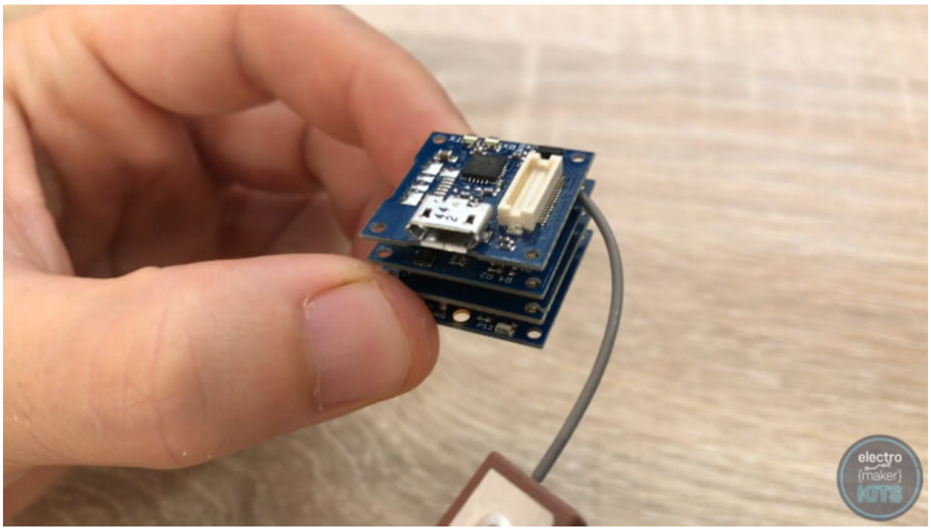
The boards need to be stacked with the main Tiny Duino processor board (shown on the far right) at the bottom. The flash memory board is then stacked on top of the main processor board.



This is then followed by the GPS module.

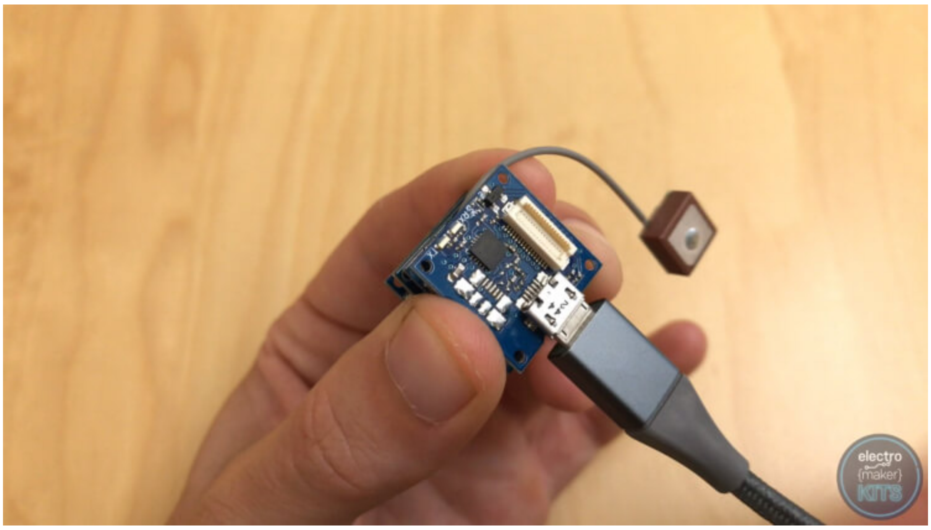


We will also temporarily add the USB board to the top of the stack. We only require this board to be included in the stack whilst we are communicating with the processor board via USB. When we get to placing it inside the 3D printed housing, we will remove the USB board so as to create as small a finished device as we can to aid with portability.



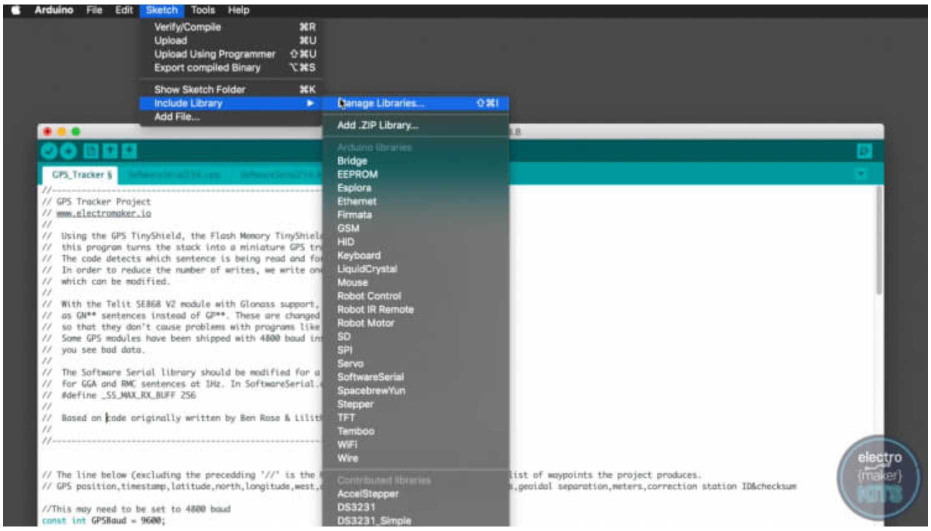
Programming the board

Connect the included USB cable to the USB shield and the other end to your PC.



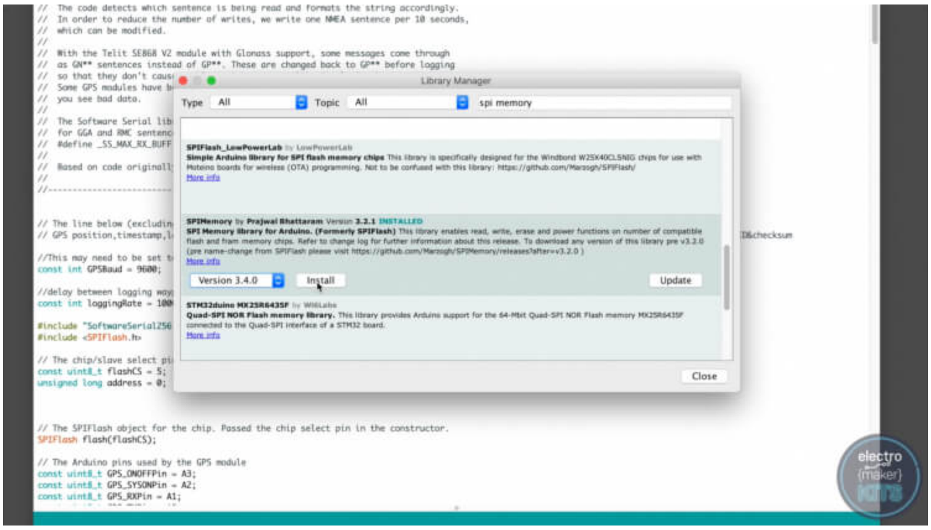
Launch the Arduino IDE. You can download it from www.arduino.cc if you do not already have it.

Before we can upload the code you will need the SPI Flash library. To get this, open the Library Manager by heading to 'Sketch' >> "Include Library" >> "Manage Libraries..." in the IDE.



```
//-----------------------------------------
// GPS Tracker Project
// www.electromaker.io
//
//  Using the GPS TinyShield, the Flash Memory TinyShiel
//  this program turns the stack into a miniature GPS tr
//  The code detects which sentence is being read and fo
//  In order to reduce the number of writes, we write on
//  which can be modified.
//
//  With the Telit SE868 V2 module with Glonass support,
//  as GN** sentences instead of GP**. These are changed
//  so that they don't cause problems with programs like
//  Some GPS modules have been shipped with 4800 baud in
//  you see bad data.
//
//  The Software Serial library should be modified for a
//  for GGA and RMC sentences at 1Hz. In SoftwareSerial.
//  #define _SS_MAX_RX_BUFF 256
//
//  Based on code originally written by Ben Rose & Lilit
//
//-----------------------------------------


// The line below (excluding the precedding '//' is the       list of waypoints the project produces.
// GPS position,timestamp,latitude,north,longitude,west,      ,geoidal separation,meters,correction station ID&checksum

//This may need to be set to 4800 baud
const int GPSBaud = 9600;
```

Search for 'SPI Memory' using the search box and then scroll down through the results until you find 'SPIMemory by Prajwal Bhattaram". Press the Install button to install it.



```
//  The code detects which sentence is being read and formats the string accordingly.
//  In order to reduce the number of writes, we write one NMEA sentence per 10 seconds,
//  which can be modified.
//
//  With the Telit SE868 V2 module with Glonass support, some messages come through
//  as GN** sentences instead of GP**. These are changed back to GP** before logging
//  so that they don't caus
//  Some GPS modules have b
//  you see bad data.
//
//  The Software Serial lib
//  for GGA and RMC sentenc
//  #define _SS_MAX_RX_BUFF
//
//  Based on code originall
//
//-------------------------


// The line below (excludin
// GPS position,timestamp,l                                                        ID&checksum

//This may need to be set t
const int GPSBaud = 9600;

//delay between logging way
const int loggingRate = 100

#include "SoftwareSerial256
#include <SPIFlash.h>

// The chip/slave select pi
const uint8_t flashCS = 5;
unsigned long address = 0;


// The SPIFlash object for the chip. Passed the chip select pin in the constructor.
SPIFlash flash(flashCS);

// The Arduino pins used by the GPS module
const uint8_t GPS_ONOFFPin = A3;
const uint8_t GPS_SYSONPin = A2;
const uint8_t GPS_RXPin = A1;
```

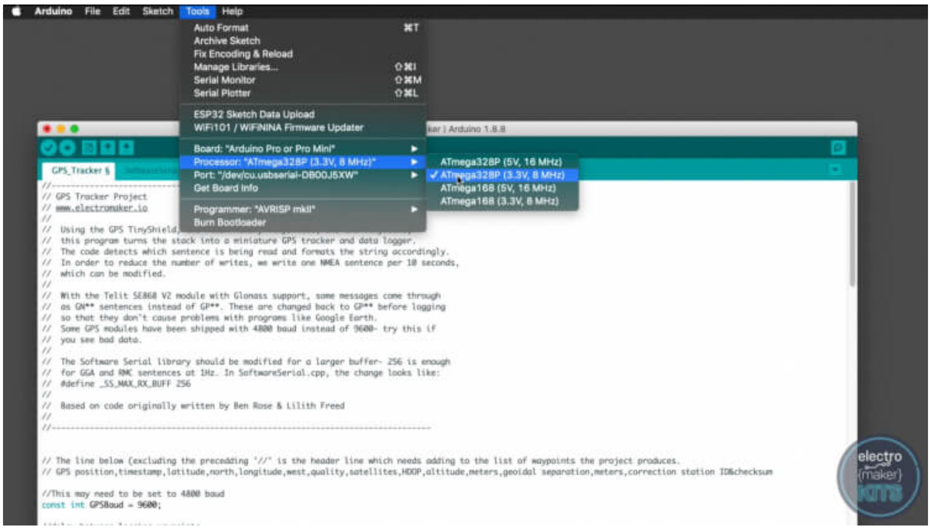
SPIFlash_LowPowerLab by LowPowerLab
Simple Arduino library for SPI flash memory chips This library is specifically designed for the Windbond W25X40CLSNIG chips for use with Moteino boards for wireless (OTA) programming. Not to be confused with this library: https://github.com/Marzogh/SPIFlash/
More info

SPIMemory by **Prajwal Bhattaram** Version **3.2.1** INSTALLED
SPI Memory library for Arduino. (Formerly SPIFlash) This library enables read, write, erase and power functions on number of compatible flash and fram memory chips. Refer to change log for further information about this release. To download any version of this library pre v3.2.0 (pre name-change from SPIFlash please visit https://github.com/Marzogh/SPIMemory/releases?after=v3.2.0)
More info

STM32duino MX25R6435F by WI6Labs
Quad-SPI NOR Flash memory library. This library provides Arduino support for the 64-Mbit Quad-SPI NOR Flash memory MX25R6435F connected to the Quad-SPI interface of a STM32 board.
More info

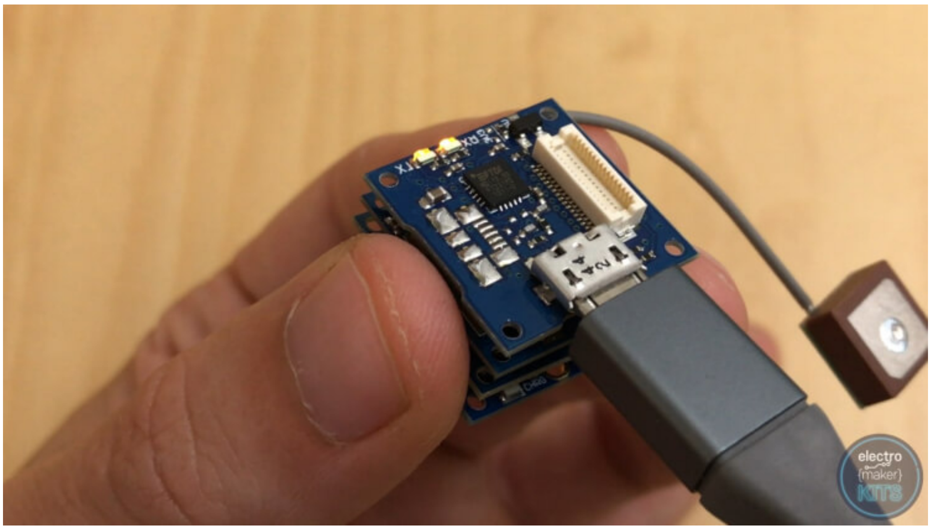
Once this has completed installing you can close the Library Manager window using the button in the bottom right of the window.

Next, ensure that you have the correct board type and processor selected. Do this by going to "Tools" >> "Board" and selecting "Arduino Pro or Pro Mini". For the processor select "ATmega 328p (3.3v, 8mhz)"

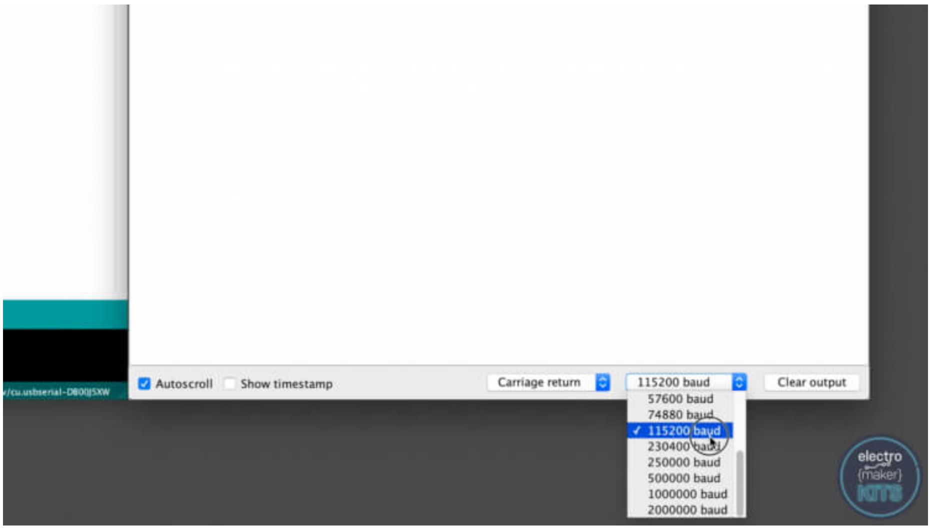


```
//-------------------------
// GPS Tracker Project
// www.electromaker.io
//
//  Using the GPS TinyShield,
//  this program turns the stack into a miniature GPS tracker and data logger.
//  The code detects which sentence is being read and formats the string accordingly.
//  In order to reduce the number of writes, we write one NMEA sentence per 10 seconds,
//  which can be modified.
//
//  With the Telit SE868 V2 module with Glonass support, some messages come through
//  as GN** sentences instead of GP**. These are changed back to GP** before logging
//  so that they don't cause problems with programs like Google Earth.
//  Some GPS modules have been shipped with 4800 baud instead of 9600- try this if
//  you see bad data.
//
//  The Software Serial library should be modified for a larger buffer- 256 is enough
//  for GGA and RMC sentences at 1Hz. In SoftwareSerial.cpp, the change looks like:
//  #define _SS_MAX_RX_BUFF 256
//
//  Based on code originally written by Ben Rose & Lilith Freed
//
//------------------------------------------------------------------------------


// The line below (excluding the preceding '//' is the header line which needs adding to the list of waypoints the project produces.
// GPS position,timestamp,latitude,north,longitude,west,quality,satellites,HDOP,altitude,meters,geoidal separation,meters,correction station ID&checksum

//This may need to be set to 4800 baud
const int GPSBaud = 9600;
```

Once this has been verified you can download the code for the project from the Electromaker website (from this projects page) and then upload it to the TinyDuino. Two LEDs, marked TX and RX, will flash during the upload to show that you are communicating via USB with the board.



After the upload has completed open the serial monitor within the Arduino IDE and set the baud speed to 115200.

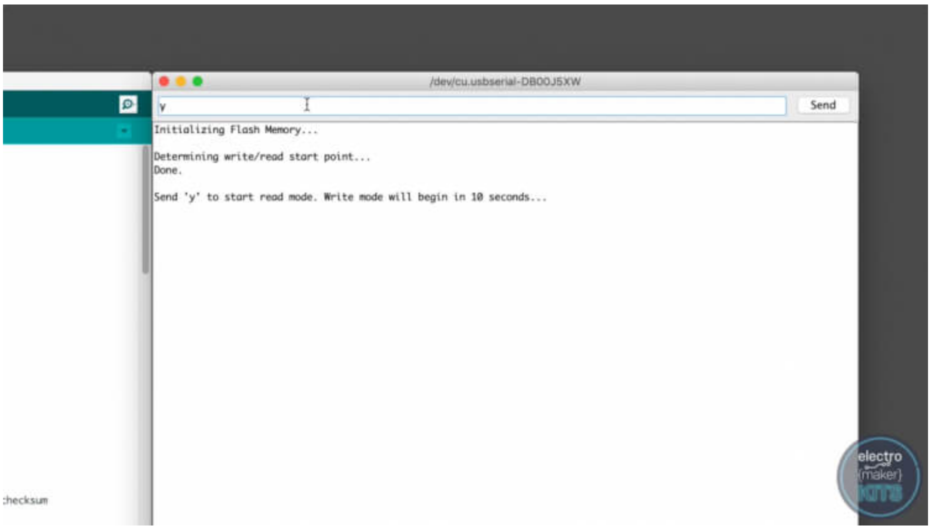


/cu.usbserial-DB00J5XW

☑ Autoscroll □ Show timestamp Carriage return 115200 baud Clear output

If you reset the TinyDuino or re-upload the code you will see the Arduino IDE will offer us the option to enter 'Read mode' by sending a letter 'y' through the serial monitor. Send the letter 'y'.

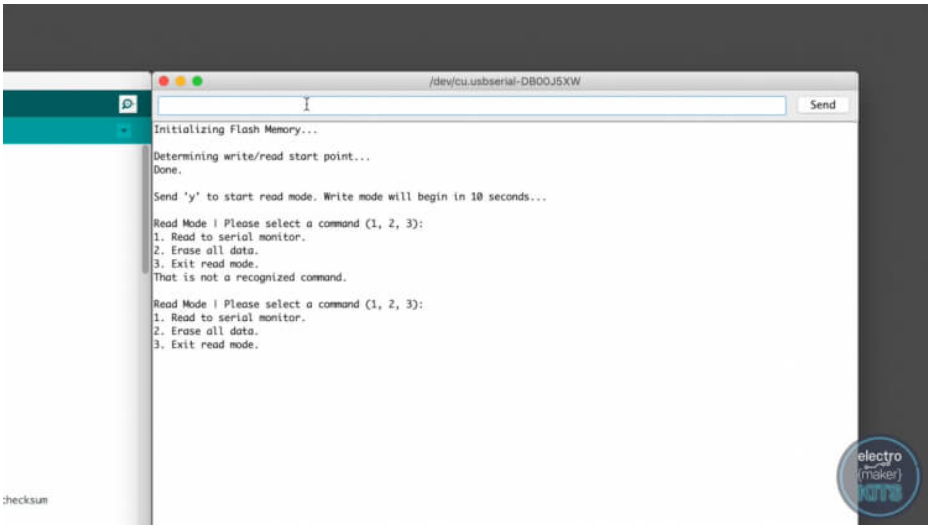
If you do nothing for 10 seconds then the board will default to entering write mode and start logging GPS coordinates. If this happens just restart the TinyDuino again.



```
Initializing Flash Memory...

Determining write/read start point...
Done.

Send 'y' to start read mode. Write mode will begin in 10 seconds...
```

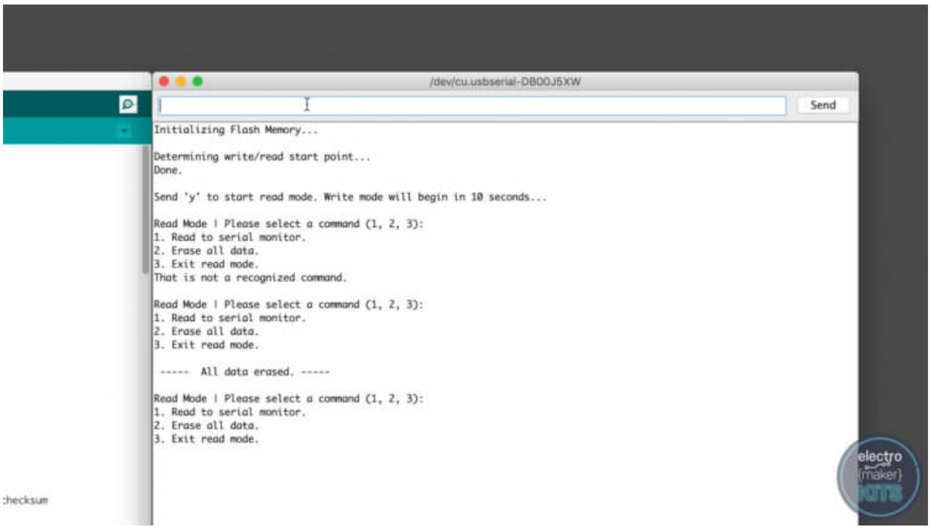
After you have entered read mode you will be presented with three more options:

1. **Read to serial monitor:** This will print all the data waypoints that have been stored on the flash memory to the serial monitor of the Arduino IDE. We will use this after we have tracked a trip to download the data and plot it onto a graphical map.
2. **Erase all data:** Once you have downloaded all the data we will use this option to clear the memory.
3. **Exit read mode:** If you select this the tracker will enter write mode and begin storing GPS waypoints.



```
Initializing Flash Memory...

Determining write/read start point...
Done.

Send 'y' to start read mode. Write mode will begin in 10 seconds...

Read Mode | Please select a command (1, 2, 3):
1. Read to serial monitor.
2. Erase all data.
3. Exit read mode.
That is not a recognized command.

Read Mode | Please select a command (1, 2, 3):
1. Read to serial monitor.
2. Erase all data.
3. Exit read mode.
```

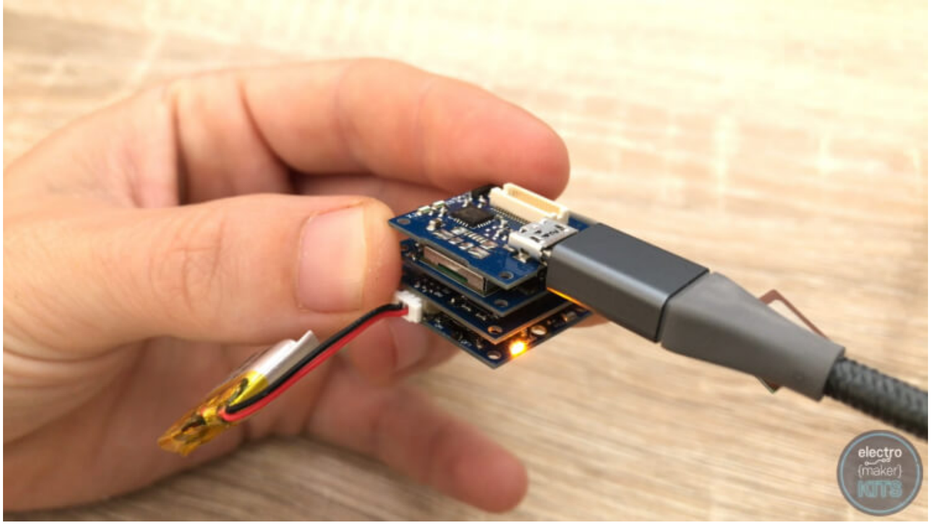
Let's start by erasing the memory to clear any waypoints which may have been logged whilst we assembled and programmed our tracker. To do this send the number '2' to the TinyDuino via the serial monitor. If successful you will receive confirmation that all the data has been erased and then be returned back to the read mode menu.



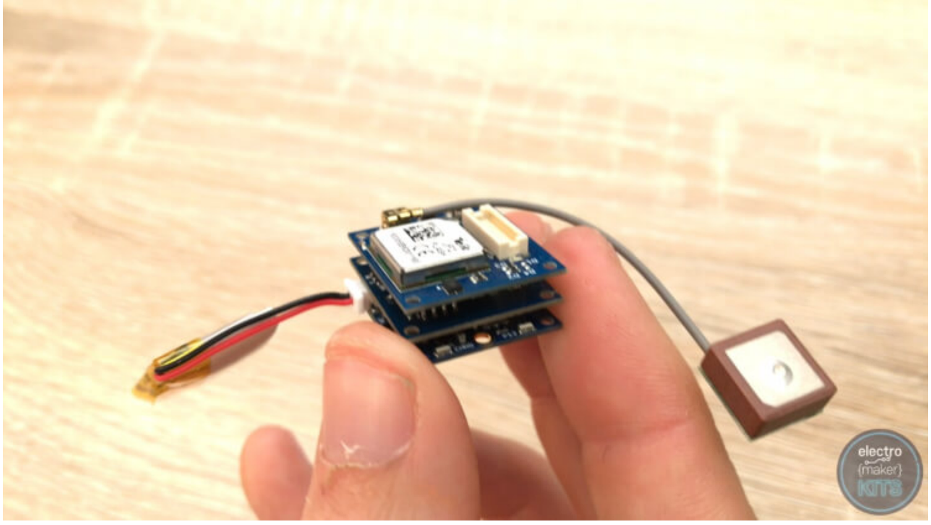
```
Initializing Flash Memory...

Determining write/read start point...
Done.

Send 'y' to start read mode. Write mode will begin in 10 seconds...

Read Mode | Please select a command (1, 2, 3):
1. Read to serial monitor.
2. Erase all data.
3. Exit read mode.
That is not a recognized command.

Read Mode | Please select a command (1, 2, 3):
1. Read to serial monitor.
2. Erase all data.
3. Exit read mode.

 -----  All data erased. -----

Read Mode | Please select a command (1, 2, 3):
1. Read to serial monitor.
2. Erase all data.
3. Exit read mode.
```

Connecting and charging the Lithium Battery

Ensure that the power switch on the side of the main TinyDuino processor board is off (the lowest board in the stack) and insert the battery lead into the white connector adjacent to the power switch. Connect the project to a powered USB port via the USB connector and the orange charge light should illuminate indicating that the battery is currently charging.

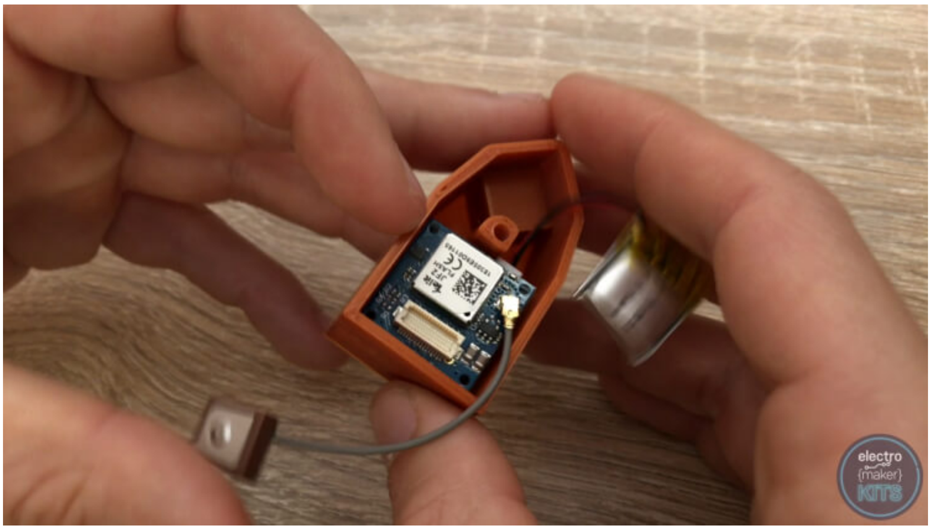


Once the battery has been fully charged the orange light will turn off. We can now remove the USB cable and we'll also remove the USB board from the top of the stack before moving onto installing it into the housing.

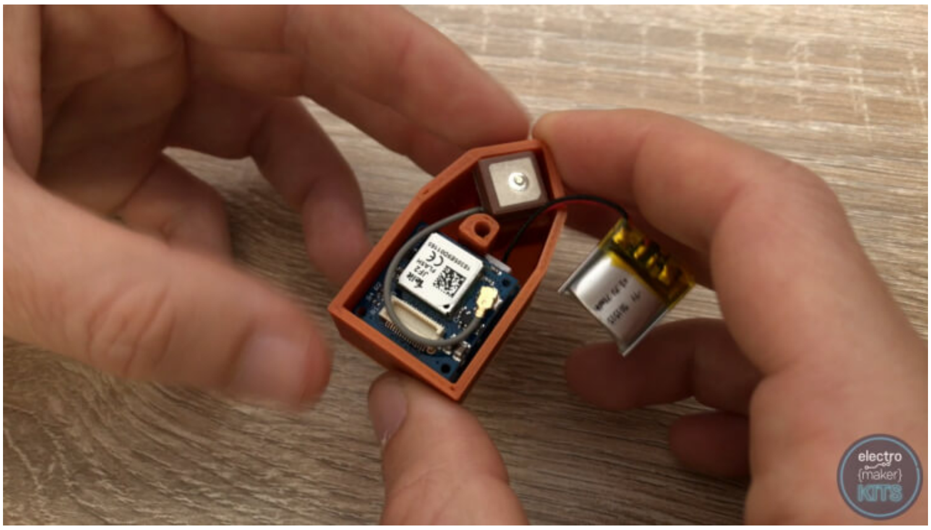


Fitting the enclosure

Although the stack of electronic boards has a square footprint their orientation inside the enclosure matters. The white stacking clips need to go against the flat rear wall as shown below:



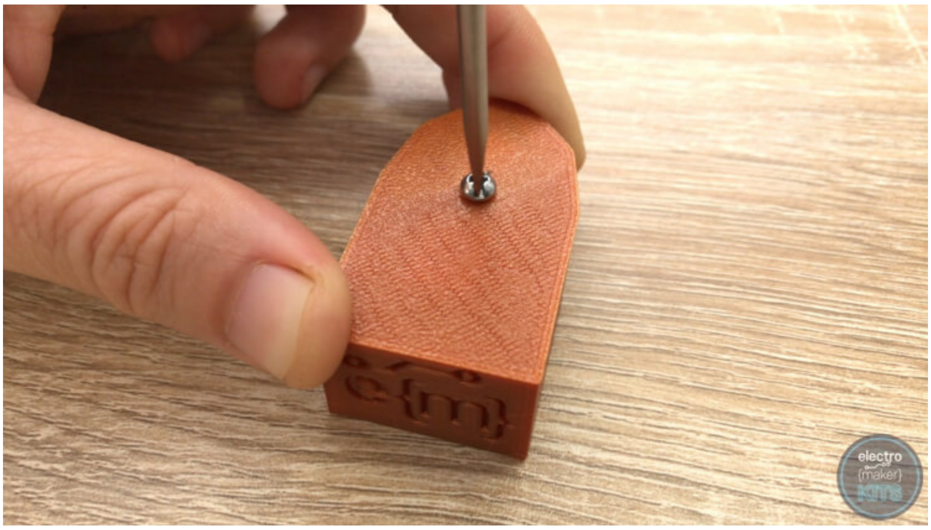
Once gently pushed down into position, carefully curl the GPS antennas cable around the print and position the receiver onto the small pedestal at the pointed end of the print.



The lithium battery is then positioned on top of the stack of electronics boards.

The lid of the enclosure features a very thin layer of plastic over the area that the GPS receiver sits beneath. This is designed to help protect it whilst still allowing it to receive GPS signals. This does not mean that the enclosure is 100% waterproof though without additional precautions being taken.

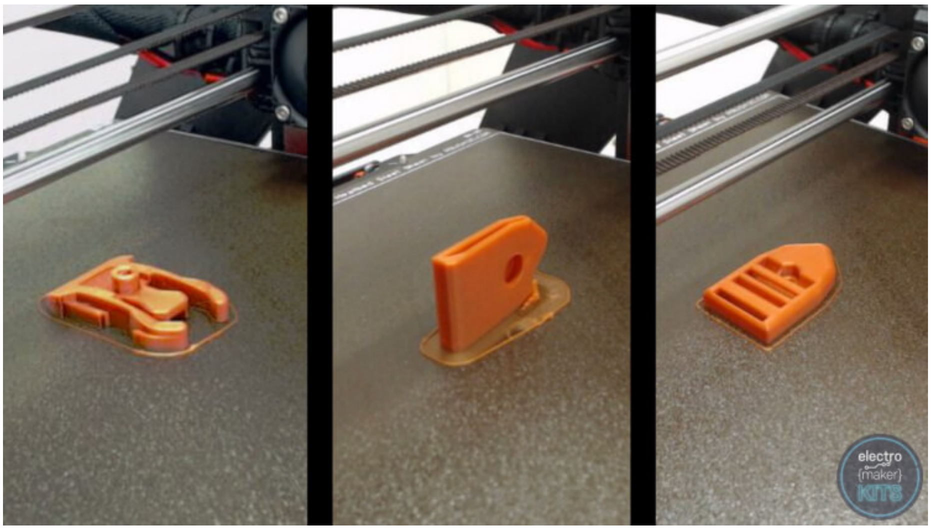
This lid is then slid over the top of the lower housing and its electronics. It's secured in place using one of the included M3 x 6mm screws.



Attachment options

There are three different attachment options which I have already made for you to download and print. They are easily interchanged as they are secured to the main assembly with a single M3 x 6mm screw. If you would like to design your own you can find a dimensioned drawing in the schematics section at the end of this write-up.

I printed all three attachments in PLA with a layer height of 0.15mm. The buckle clip was also printed on its side with a brim and support to help adhesion.

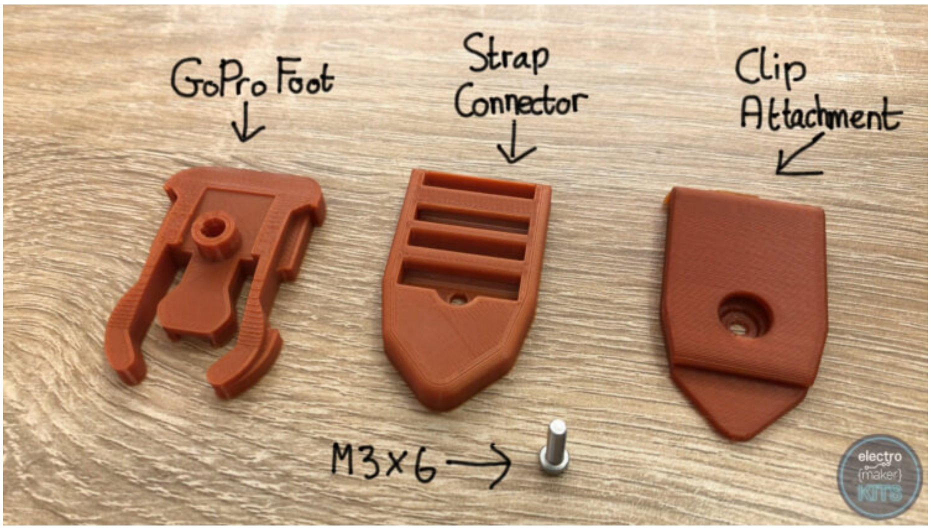


The three ready-made attachment options are:

GoPro Foot

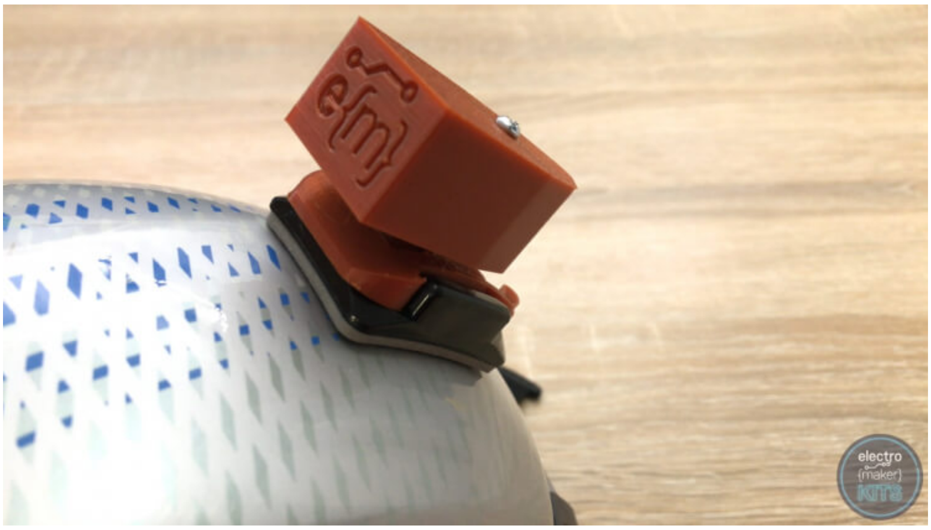
Strap connector

Clip attachment



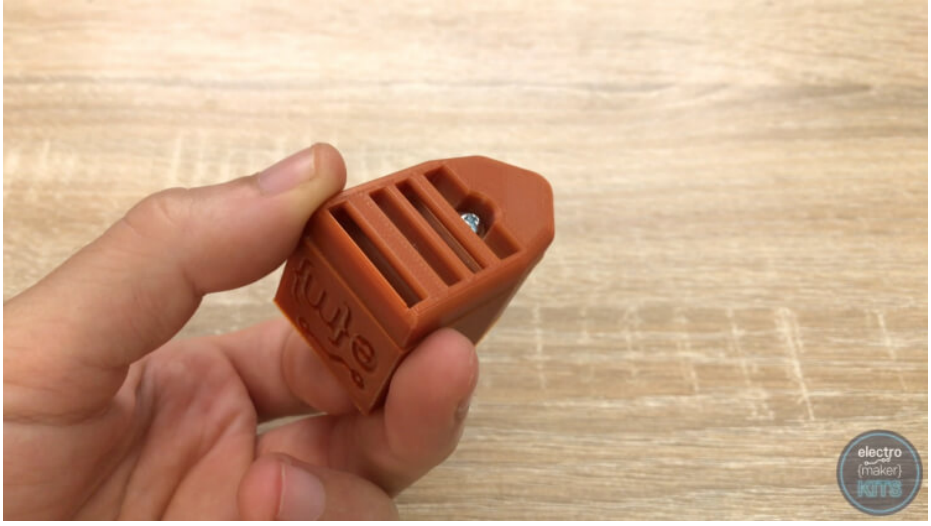
GoPro foot:

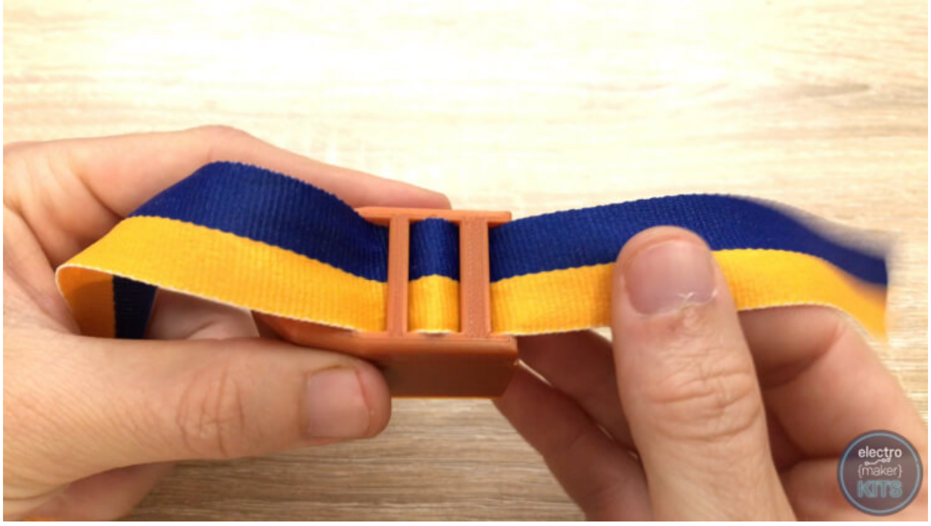
This attachment is designed to be compatible with GoPro mountings. Once added it simply slides onto the GoPro base. To remove, pinch the end of the buckle and push backward.



Strap connector

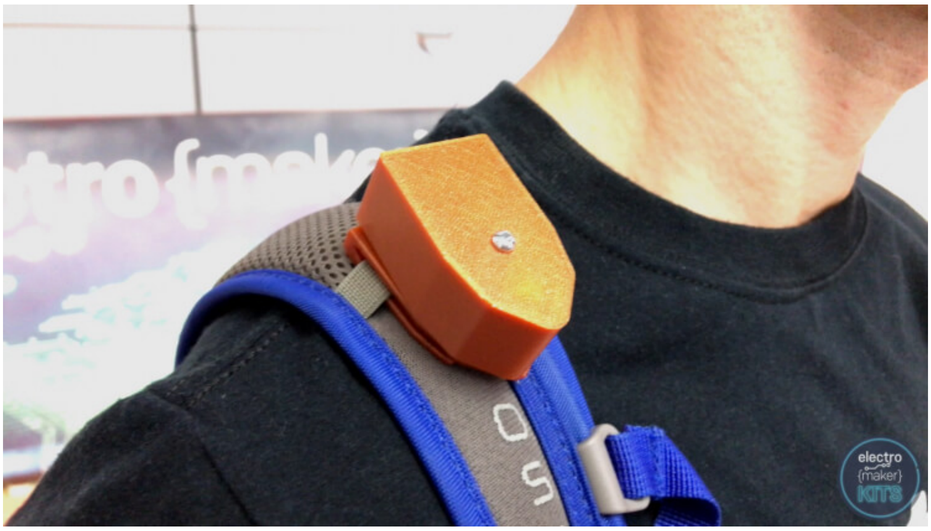
Great for attaching to materials such as the collar of a pet or a belt.





Clip attachment

Good for easily attaching to clothing such as a pocket or rucksack.



Tracking a journey

To begin tracking a journey switch on the tracker using the sliding power switch and then fit it inside its housing. Now add an attachment accessory of your choosing. By the time you have done this the tracker will have defaulted to entering write mode and will begin collecting waypoint information every ten seconds.

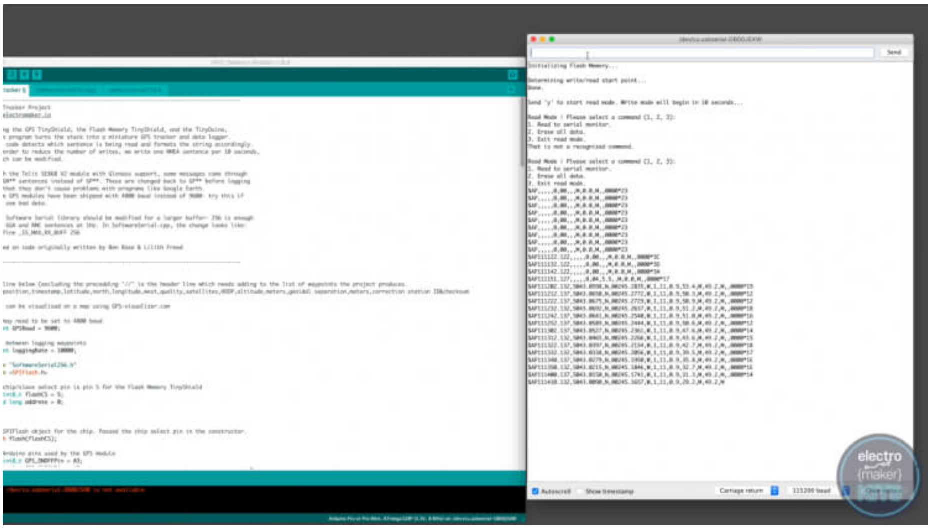
To stop tracking just disconnect the battery or slide the power switch to off.

Mapping the data collected

Once you have collected a set of waypoint data we can go through the steps of retrieving the data from the tracker and plotting it onto a map.

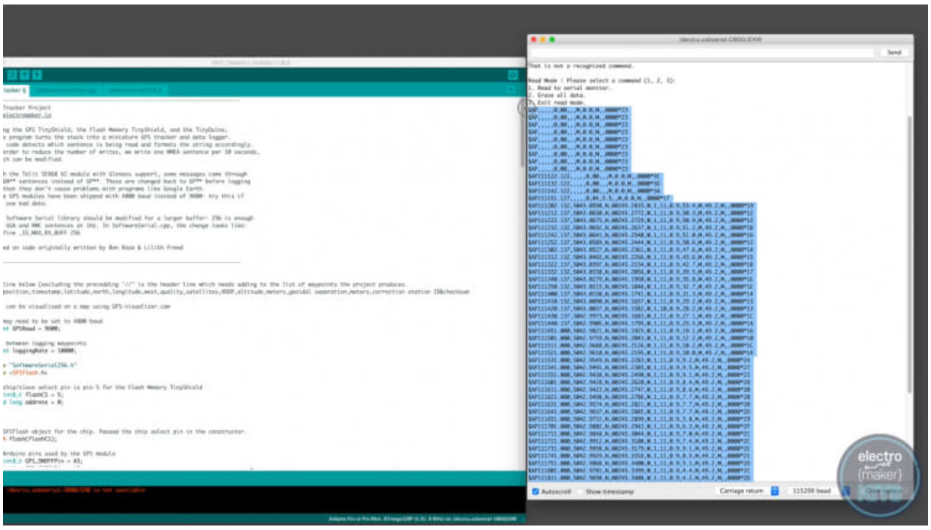
Remove the tracker from the housing and add the USB board back onto the top of the stack. Use a USB cable to connect it to your PC. Open the Arduino IDE and then switch on your tracker with the serial monitor open and set to 115200 baud rate. If you do not see the options menu show up as we did earlier then just re-upload the project code to the processor. The GPS waypoint data is saved to the flash memory and so will not be overwritten.

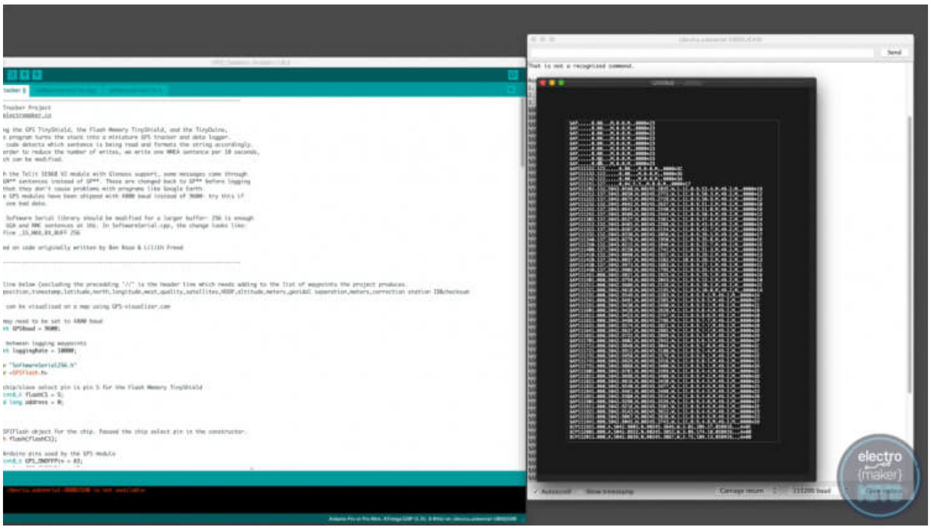
When the menu does show, enter 'y' to enter read mode and then a '1' to read the data to the serial monitor.



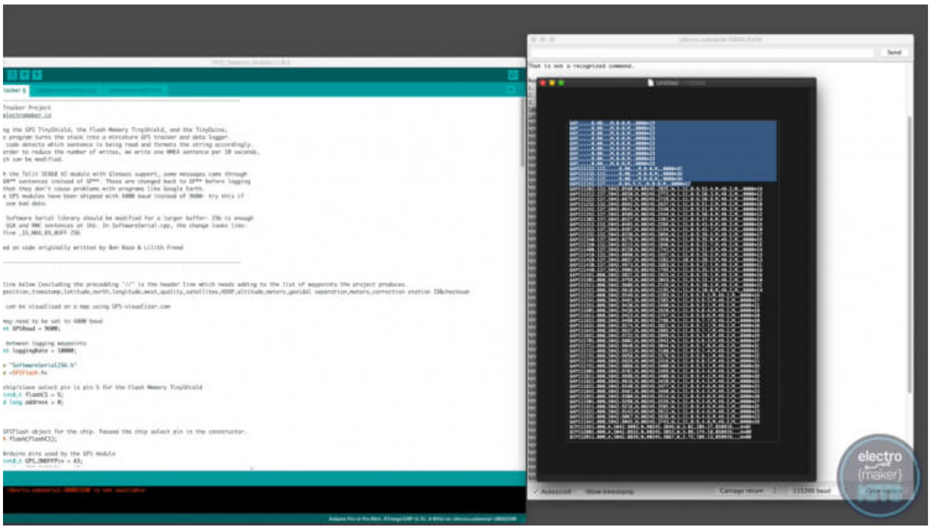
```
Initializing Flash Memory...

Determining write/read start point...
Done.

Send 'y' to start read mode. Write mode will begin in 10 seconds...

Read Mode : Please select a command (1, 2, 3):
1. Read to serial monitor.
2. Erase all data.
3. Exit read mode.
That is not a recognized command.

Read Mode : Please select a command (1, 2, 3):
1. Read to serial monitor.
2. Erase all data.
3. Exit read mode.
[illegible]
```

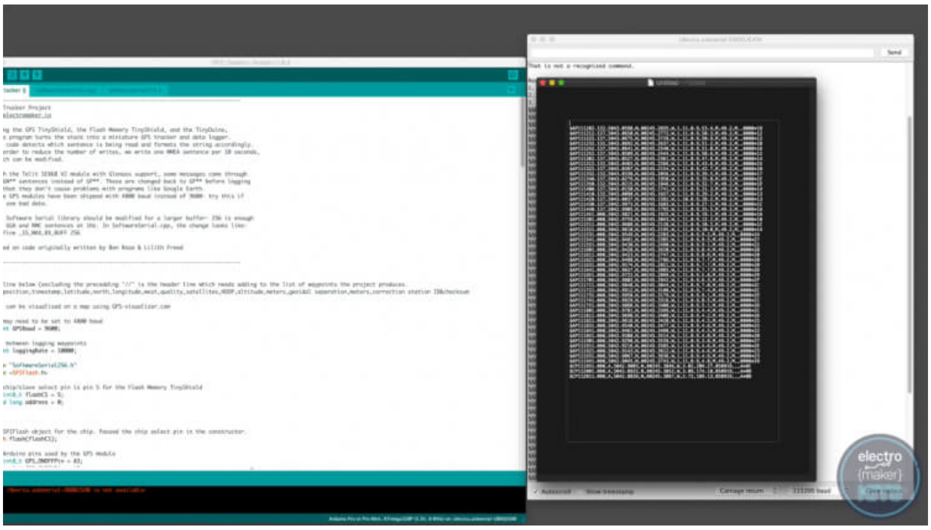
Highlight all the rows of data in the serial monitor and copy this into a new plain text document.





In my data, you can see that a few of the lines at the beginning were recorded to the flash memory when we did not have sufficient GPS data yet to create a complete entry. This is because the device had only just been powered on and was still acquiring enough satellite signals. If you have similar lines in your recording then remove them from the plain text document as I have done.





```
Tracker Project
electromaker.io

ng the GPS TinyShield, the Flash Memory TinyShield, and the TinyDuino,
s program turns the stack into a miniature GPS tracker and data logger.
code detects which sentence is being read and formats the string accordingly.
order to reduce the number of writes, we write one NMEA sentence per 10 seconds,
ch can be modified.

h the Telit SE868 V2 module with Glonass support, some messages come through
GN** sentences instead of GP**. These are changed back to GP** before logging
that they don't cause problems with programs like Google Earth.
e GPS modules have been shipped with 4800 baud instead of 9600- try this if
 see bad data.

 Software Serial library should be modified for a larger buffer- 256 is enough
 GGA and RMC sentences at 1Hz. In SoftwareSerial.cpp, the change looks like:
fine _SS_MAX_RX_BUFF 256

ed on code originally written by Ben Rose & Lilith Freed

line below (excluding the preceding "//" is the header line which needs adding to the list of waypoints the project produces.
position,timestamp,latitude,north,longitude,west,quality,satellites,HDOP,altitude,meters,geoidal separation,meters,correction station ID&checksum

 can be visualised on a map using GPS-visualizer.com

may need to be set to 4800 baud
nt GPSBaud = 9600;

 between logging waypoints
nt loggingRate = 10000;

e "SoftwareSerial256.h"
e <SPIFlash.h>

chip/slave select pin is pin 5 for the Flash Memory TinyShield
int8_t flashCS = 5;
d long address = 0;

SPIFlash object for the chip. Passed the chip select pin in the constructor.
h flash(flashCS);

Arduino pins used by the GPS module
int8_t GPS_ONOFFPin = A3;
```

```
That is not a recognized command.
```

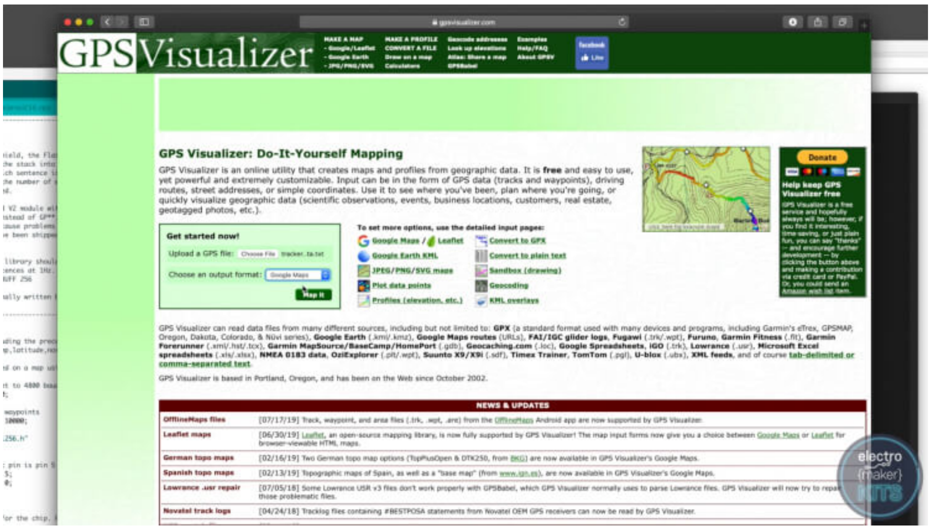
Next, you need to add a new header line to the top of the document which will explain to the mapping software how our data has been formatted. I have put a copy of this line near the top of the project's code (highlighted in yellow below). Copy and paste this so that it is the first line of the document. The same line is also written out here:

GPS position,timestamp,latitude,north,longitude,west,quality,satellites,HDOP,altitude,meters,geoidal separation,meters,correction station ID&checksum

Save the plain text file as a document with the extension .txt.

To map these co-ordinates onto a graphical map we will use a free tool provided at www.gpsvisualizer.com.

On the front page is a tool to upload the text document we just created. You can then choose an output format. Please do have a look at and try the various options available. For this tutorial select 'Google Maps' as the output.



GPS Visualizer: Do-It-Yourself Mapping

GPS Visualizer is an online utility that creates maps and profiles from geographic data. It is **free** and easy to use, yet powerful and extremely customizable. Input can be in the form of GPS data (tracks and waypoints), driving routes, street addresses, or simple coordinates. Use it to see where you've been, plan where you're going, or quickly visualize geographic data (scientific observations, events, business locations, customers, real estate, geotagged photos, etc.).

Get started now!

Upload a GPS file: Choose File tracker..ta.txt

Choose an output format: Google Maps

Map it

To set more options, use the detailed input pages:

Google Maps / Leaflet · Convert to GPX · Google Earth KML · Convert to plain text · JPEG/PNG/SVG maps · Sandbox (drawing) · Plot data points · Geocoding · Profiles (elevation, etc.) · KML overlays

Help keep GPS Visualizer free

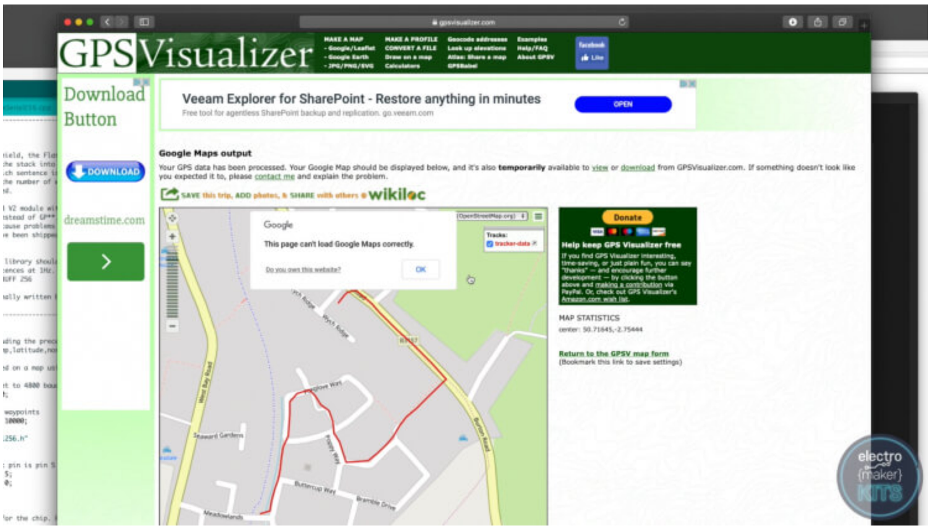
GPS Visualizer is a free service and hopefully always will be; however, if you find it interesting, time-saving, or just plain fun, you can say "thanks" -- and encourage further development -- by clicking the button above and making a contribution via credit card or PayPal. Or, you could send an Amazon wish list item.

GPS Visualizer can read data files from many different sources, including but not limited to: **GPX** (a standard format used with many devices and programs, including Garmin's eTrex, GPSMAP, Oregon, Dakota, Colorado, & Nüvi series), **Google Earth** (.kml/.kmz), **Google Maps routes** (URLs), **FAI/IGC glider logs**, **Fugawi** (.trk/.wpt), **Furuno**, **Garmin Fitness** (.fit), **Garmin Forerunner** (.xml/.hst/.tcx), **Garmin MapSource/BaseCamp/HomePort** (.gdb), **Geocaching.com** (.loc), **Google Spreadsheets**, **iGO** (.trk), **Lowrance** (.usr), **Microsoft Excel spreadsheets** (.xls/.xlsx), **NMEA 0183 data**, **OziExplorer** (.plt/.wpt), **Suunto X9/X9i** (.sdf), **Timex Trainer**, **TomTom** (.pgl), **U-blox** (.ubx), **XML feeds**, and of course tab-delimited or comma-separated text.

GPS Visualizer is based in Portland, Oregon, and has been on the Web since October 2002.

NEWS & UPDATES	
OfflineMaps files	[07/17/19] Track, waypoint, and area files (.trk, .wpt, .are) from the OfflineMaps Android app are now supported by GPS Visualizer.
Leaflet maps	[06/30/19] Leaflet, an open-source mapping library, is now fully supported by GPS Visualizer! The map input forms now give you a choice between Google Maps or Leaflet for browser-viewable HTML maps.
German topo maps	[02/16/19] Two German topo map options (TopPlusOpen & DTK250, from BKG) are now available in GPS Visualizer's Google Maps.
Spanish topo maps	[02/13/19] Topographic maps of Spain, as well as a "base map" (from www.ign.es), are now available in GPS Visualizer's Google Maps.
Lowrance .usr repair	[07/05/18] Some Lowrance USR v3 files don't work properly with GPSBabel, which GPS Visualizer normally uses to parse Lowrance files. GPS Visualizer will now try to repair those problematic files.
Novatel track logs	[04/24/18] Tracklog files containing #BESTPOSA statements from Novatel OEM GPS receivers can now be read by GPS Visualizer.

Press 'Map It' and if all goes as planned the website will return a map of the route traveled.



Google Maps output

Your GPS data has been processed. Your Google Map should be displayed below, and it's also **temporarily** available to view or download from GPSVisualizer.com. If something doesn't look like you expected it to, please contact me and explain the problem.

This page can't load Google Maps correctly.

Help keep GPS Visualizer free

If you find GPS Visualizer interesting, time-saving, or just plain fun, you can say "thanks" — and encourage further development — by clicking the button above and making a contribution via PayPal. Or, check out GPS Visualizer's Amazon.com wish list.

MAP STATISTICS
center: 50.71645,-2.75444

Return to the GPSV map form
(Bookmark this link to save settings)

Don't forget to erase the data on your tracker when you are finished. This way it is ready for the time you want to track something.

Have you enjoyed this Electromaker Kit?

Why not try one of the other kits in the range such as the Disco Helmet - Kit or the LED Mood Lamp - Kit?

Visit the Electromaker Kits section on our website to find out more:

https://www.electromaker.io/kits