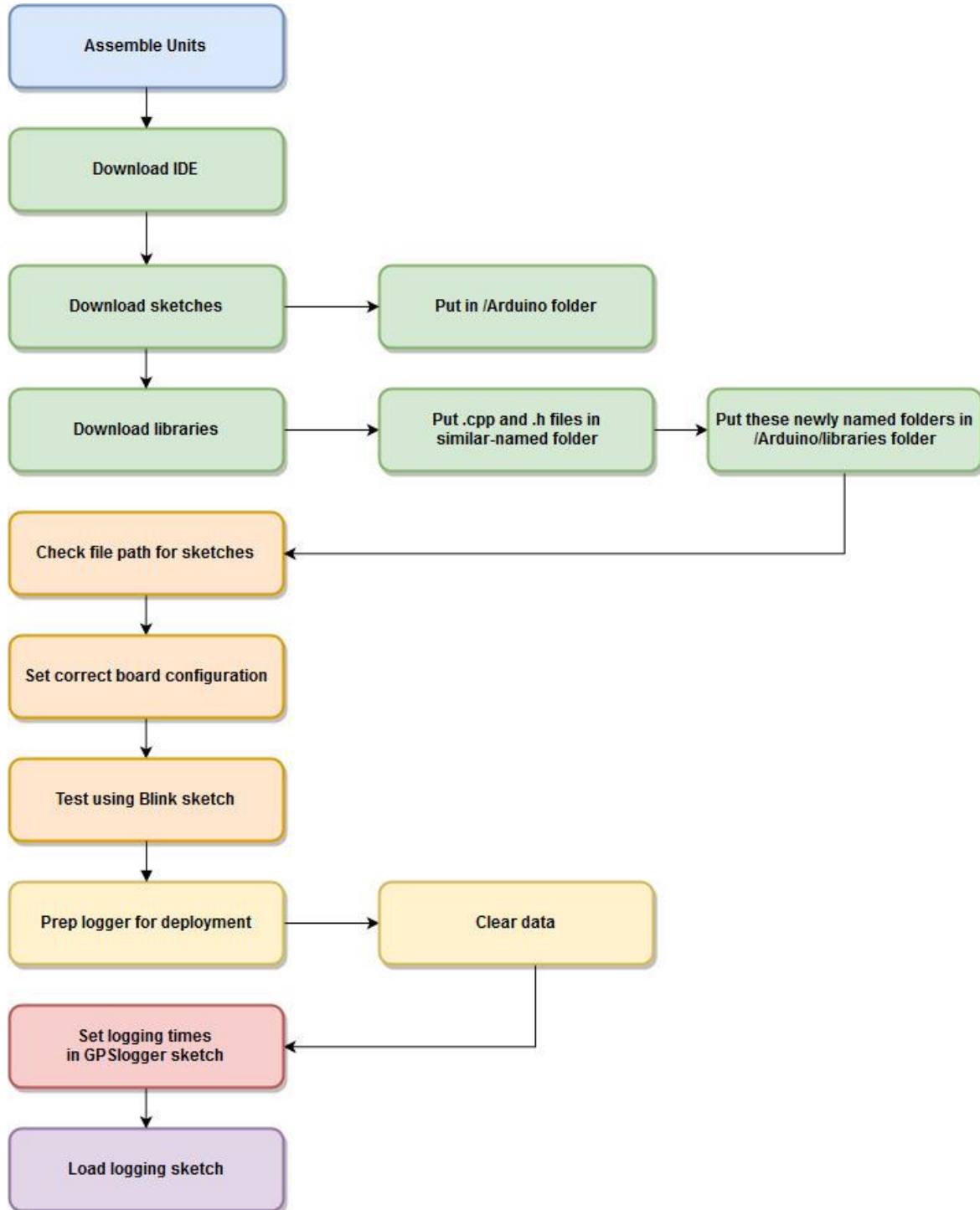


Arduino Setup

The following contains a list of steps to download and locate the files necessary to collect data on the GPS loggers.



Arduino

The Arduino IDE can be found at <https://www.arduino.cc/>

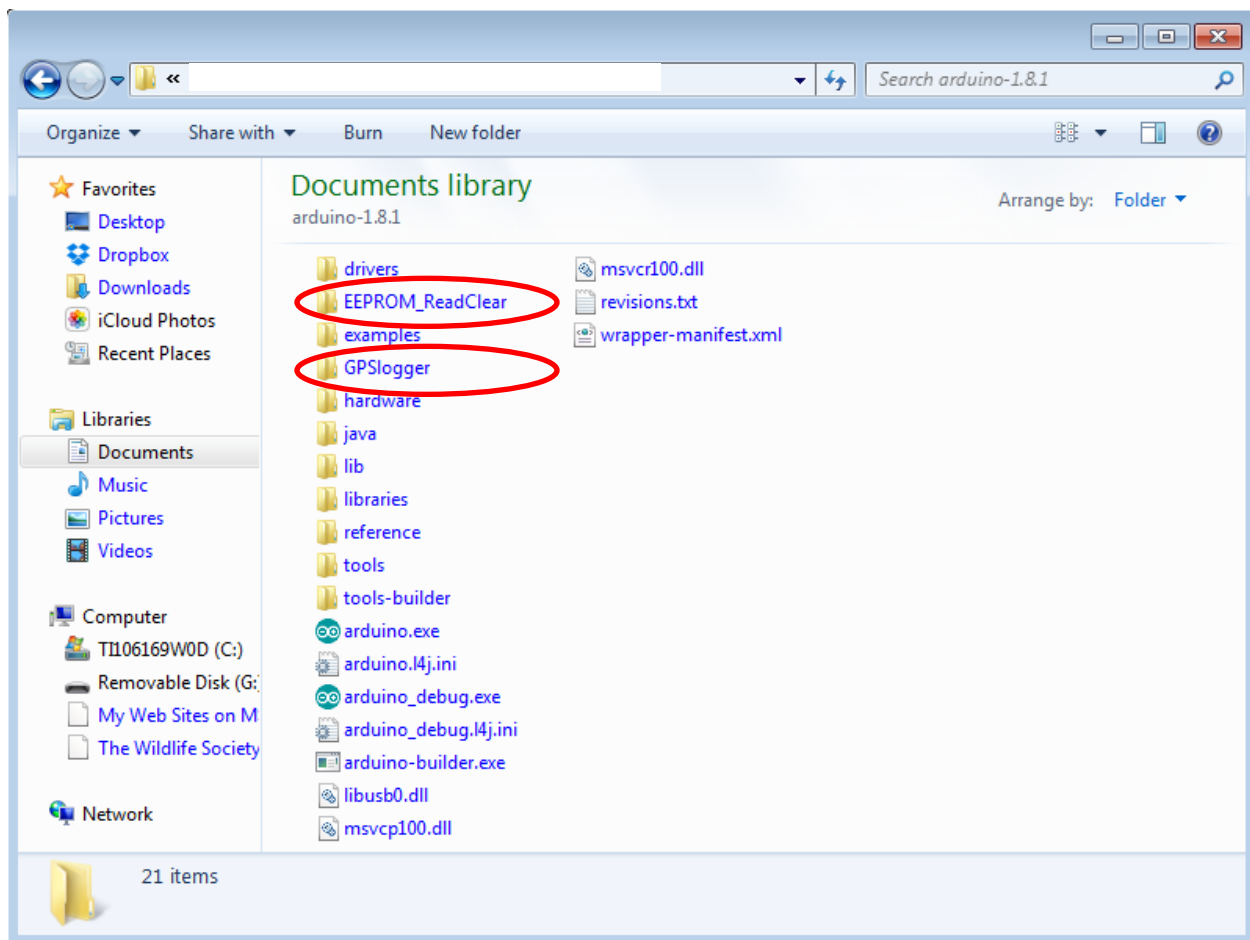
Software -> Download the IDE -> Choose the correct platform (e.g., Windows, Mac OS, etc.)

Unzip the file and double click “arduino.exe” to start the installation process.

Sketches

Download the “TOS_logger”, “TOS_ReadClear”, “TNG_logger”, and “TNG_ReadClear” sketches from: <https://bitbucket.org/Splat01/gpslogger> to the “Arduino” folder.

Download these files into the same Arduino folder that contains the “libraries” folder. Also, put each of the sketches (.ino files) into a folder of the same name.



Libraries

Info for installing Arduino libraries can be found in the “GPSlogger” sketch.



```
GPSlogger | Arduino 1.6.5
File Edit Sketch Tools Help

GPSlogger

/*****
 *
 *      Code for a turtle GPS receiver and logger
 *
 *      March 2016
 *
 *      P. Cain, Indiana State University, Terre Haute, IN
 *
 *      -and-
 *
 *      M. Cross, Bowling Green State University, Bowling Green, OH
 *
 * Microcontroller is the Arduino Pro Mini 3v3: https://www.sparkfun.com/products/11114
 * GPS Receiver: https://www.sparkfun.com/products/13740
 *
 * Wiring:
 * -GPS-
 * Power pin to 3v3
 * GND pin transistor to GND
 * Tx pin to D4
 * PN2222 Transistor: base (middle pin) to Pin D2
 *
 *****/

// Libraries and links to download.
//For info on installing Arduino libraries, see https://www.arduino.cc/en/Guide/Libraries
#include <SoftwareSerial.h>
#include <TinyGPS++.h>           // http://arduiniiana.org/libraries/tinygpsplus/
#include <LowPower.h>           // https://github.com/rockscream/Low-Power
#include <EEPROMex.h>           // http://this.elenbaas.net/2012/07/extended-EEPROM-library-for-arduino
```

Download the “EEPROMex”, “TinyGPS++”, “LowPower”, “eepromi2c” libraries to the “Libraries” folder in the “Arduino” folder. The links to the libraries are listed in the sketches or here:

TinyGPS++ <http://arduiniiana.org/libraries/tinygpsplus/>

Click on the download link and download the most recent version

Arduiniana

Arduino wisdom and gems by Mikal Hart

TinyGPS++

A *NEW* Full-featured GPS/NMEA Parser for Arduino

TinyGPS++ is a new Arduino library for parsing NMEA data streams provided by GPS modules.

Like its predecessor, TinyGPS, this library provides compact and easy-to-use methods for extracting position, date, time, altitude, speed, and course from consumer GPS devices.

However, TinyGPS++'s programmer interface is considerably simpler to use than TinyGPS, and the new library can extract arbitrary data from any of the myriad NMEA sentences out there, even proprietary ones.

Download and Installation

To install this library, download here, unzip the archive into the Arduino "libraries" folder, and restart Arduino. You should rename the folder "TinyGPSPlus".



LowPower <https://github.com/rocketscream/Low-Power>

Click on the "Clone or download" button in the upper right corner, then "Download ZIP"

[Code](#) [Issues 5](#) [Pull requests 4](#) [Projects 0](#) [Pulse](#) [Graphs](#)

Low Power Library for Arduino <http://www.rocketscream.com>

26 commits 1 branch 3 releases 3 contributors

Branch: master New pull request Find files **Clone or download**

rocketscream Update library.properties ...

| | | |
|--------------------|---|------------|
| Examples | updated example for ATmega256rfr2 | |
| LowPower.cpp | Update LowPower.cpp | |
| LowPower.h | Merged with upstream, added comments | |
| README.md | Update README.md | a year ago |
| keywords.txt | Added support for ATSAM21G18A and library format compliant with Ardu... | a year ago |
| library.properties | Update library.properties | a year ago |

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

<https://github.com/rocketscream/Low-Powe>

Open in Desktop Download ZIP

EEPROMex <http://thijs.elenbaas.net/2012/07/extended-eeeprom-library-for-arduino>

Click the “download EEPROMex library for Arduino – zip” link

Extended EEPROM library for Arduino

July 22, 2012 by Thijs Elenbaas | 90 Comments

For my ongoing clock project, I want to persistently store some data. That is, store data that is retained after turning off the Arduino. The processor on the Arduino board comes with on-board EEPROM. In the case of the Arduino Uno, the processor is the Atmega328, equipped with 1 glorious KByte of EEPROM memory.

The AVR libraries that come with the ATmega implements a relatively broad set of functions for reading, writing and management of the EEPROM (for a description see the [AVR user manual](#)). However, the Arduino standard EEPROM library exposes only functionality for reading and writing a single byte, as described [here](#).

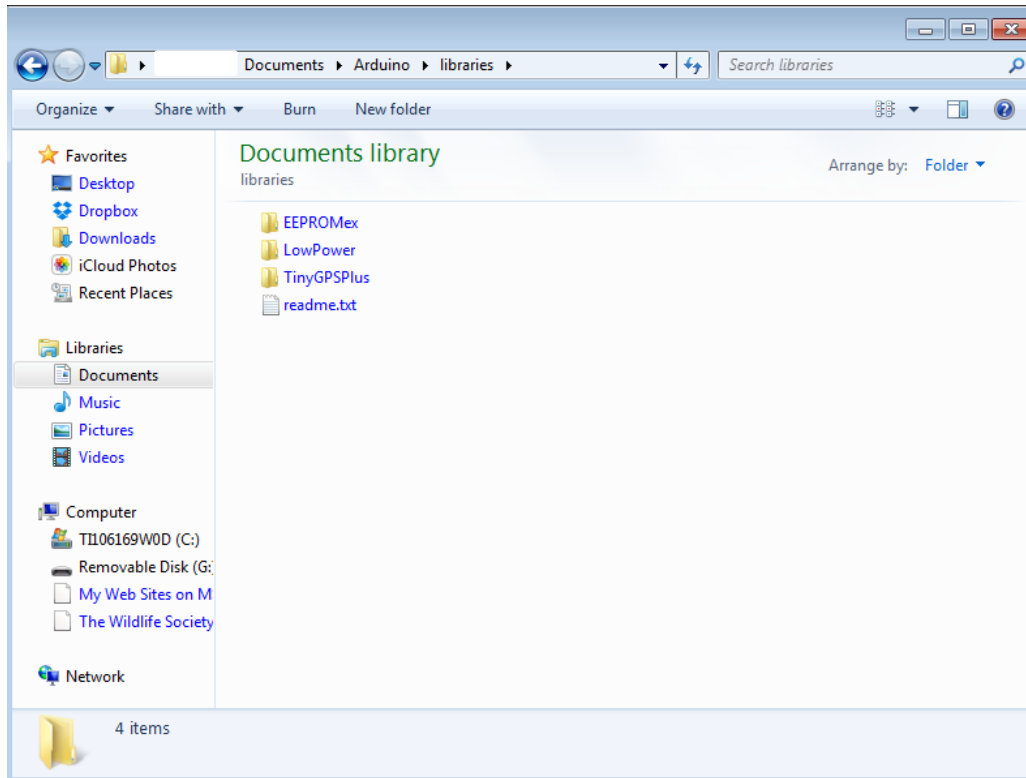
This is why I wrote the EEPROMex library, an extension of the standard Arduino EEPROM library. It writes and reads basic types like bytes, longs, ints, floats & doubles. It can also read/write single bits, arbitrary data-formats and arrays. It adds debug functionality to identify and stop writing outside of the EEPROM memory size and excessive writing to prevent memory wear.

The library

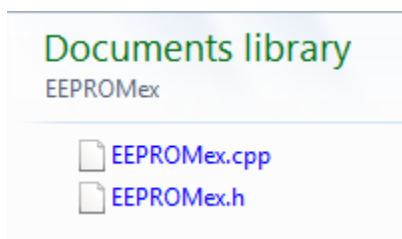
First of all, the library can be downloaded here:

[Download EEPROMex library for Arduino - zip](#)

Create folders in the Arduino libraries folder that match the names of the sketches. Do **not** use “+” signs when naming the TinyGPS++ folder, instead name it “**TinyGPSPlus**”.



Copy and paste the corresponding “.h” and “.cpp” from the downloads into the appropriate newly named folders.

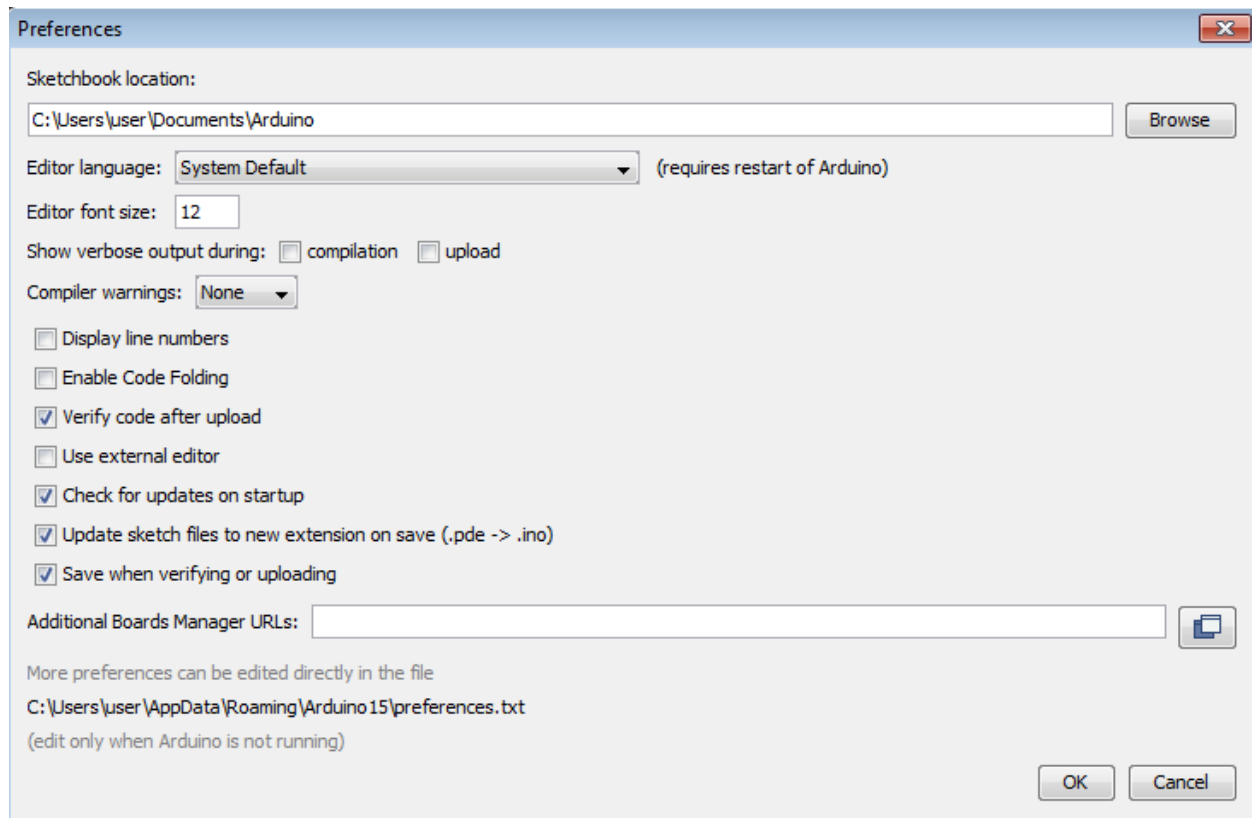


Using the Arduino IDE

Open the Arduino IDE

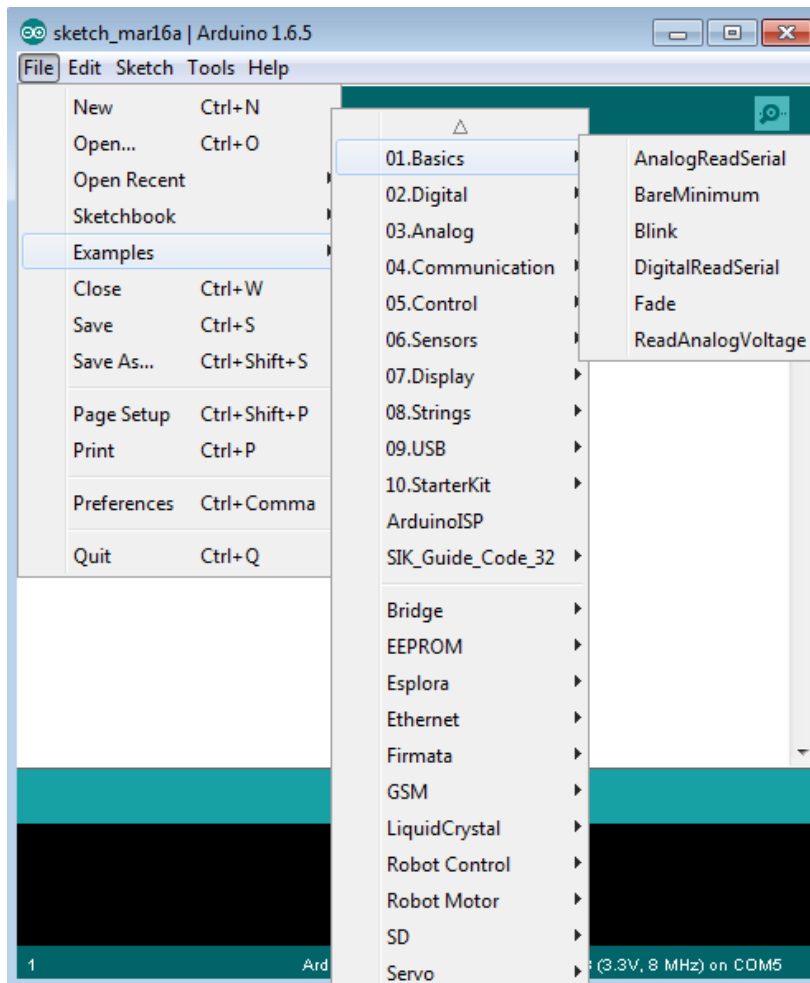
Check the file path associated with the Arduino download and folders you created

File -> Preferences -> change the directory as needed



Test the loggers by uploading the “Blink” sketch.

File -> Examples -> Basics -> “Blink”

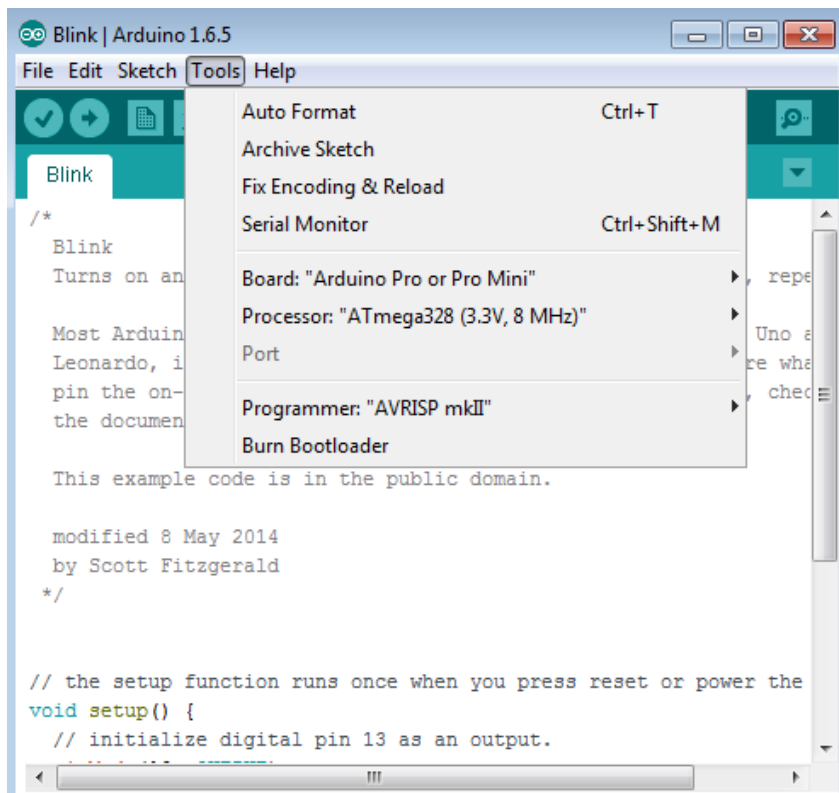


Under the “Tools” menu, check the following and make the appropriate changes:

Board: Arduino Pro or Pro Mini

Processor: ATmega328 (3.3V, 8MHz)

Port: May change frequently, but will usually be a number higher than 1 or 2. For example, COM5. If not sure, unplug and plug back in to see which COM changes.



Click the check mark to check the sketch for errors and compile it.

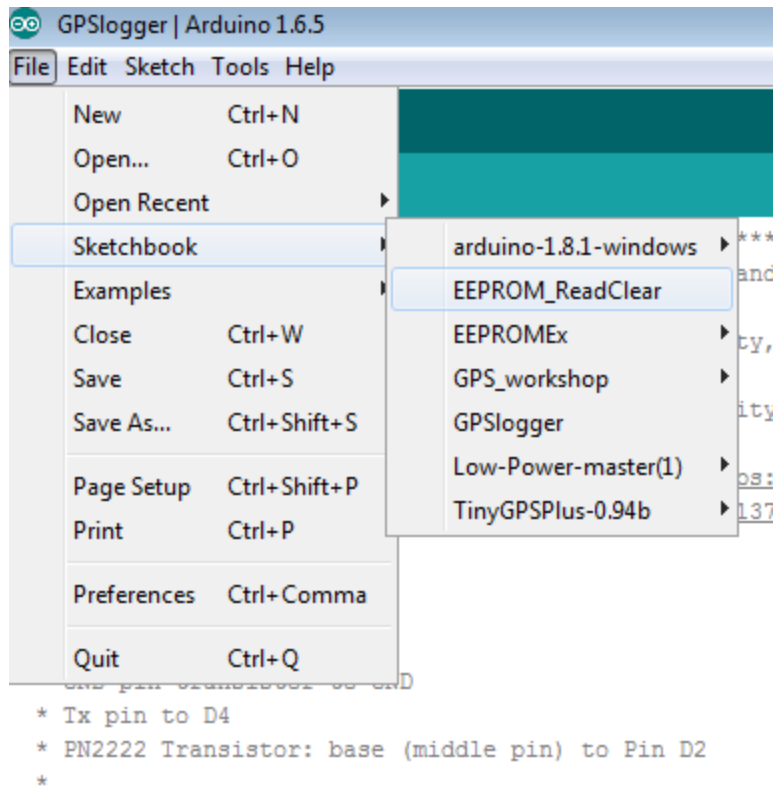
Once complete, click the right arrow button to load the sketch to the device. This should cause the LED on pin 13 to blink on for a second and then turn off for a second.

Remember, the microcontroller can only hold one sketch at a time.

Prepping Logger

Load TOS_ReadClear sketch to the loggers.

File -> Sketchbook -> TOS_ReadClear

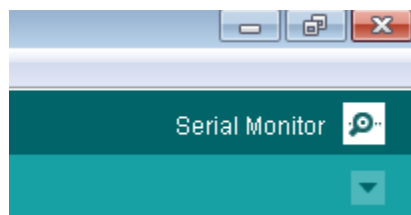


All of the settings should still be the same as those used in the “Blink” test, but double check these, especially the port, in the Tools menu.

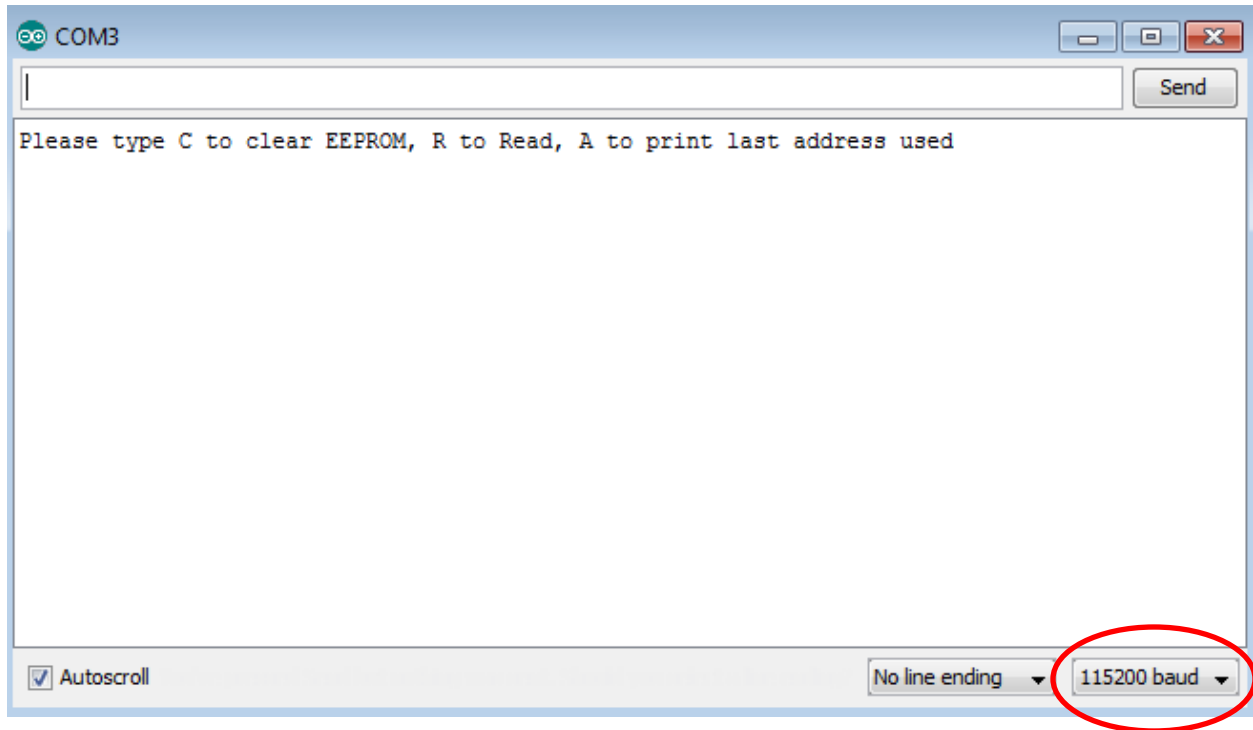
Click the upload button.

The LED on the logger unit should blink rapidly for a few seconds to let you know the sketch was uploaded correctly.

Once the light is done blinking, click the “Serial Monitor” button in the upper right hand corner.



This will open a window with a command prompt



Make sure the baud rate is set to 115200

R – will read all nonzero values, Z – will read all values, including zeros, C – will clear everything.

Type “C” to clear the memory and remove any previous data.

Uploading GPS logger sketch

File -> Sketchbook -> TOS_logger

Set the amount of time the GPS is allowed to acquire satellites. Use more time for low visibility areas and less time for higher visibility areas (e.g, 2-3 minutes for low visibility areas [dense canopy cover] and 1 minute for high visibility areas [open field]).

Set the logger_interval to desired number of hours. Must be an integer value (i.e., no decimals, for now).

```

/***** ( USER DEFINED VARIABLES ) *****/
Two variables to define are how long the GPS will stay on to acquire
up time with battery life. The other variable is how often the unit
This occurs in hour intervals, starting on power up. Twelve hours w:
will give you three, etc. */

// how long will GPS receiver stay on
int stay_on = 2; /*minutes*/

// how long is the data logging interval? (time between readings)
int logger_interval = 5; /*hours*/
/*****/

```

To test, set the logger interval to “0”, this will have it read every minute.

Click the upload sketch button.

Insert battery and switch on.

Collect data.

Wander.

Data download

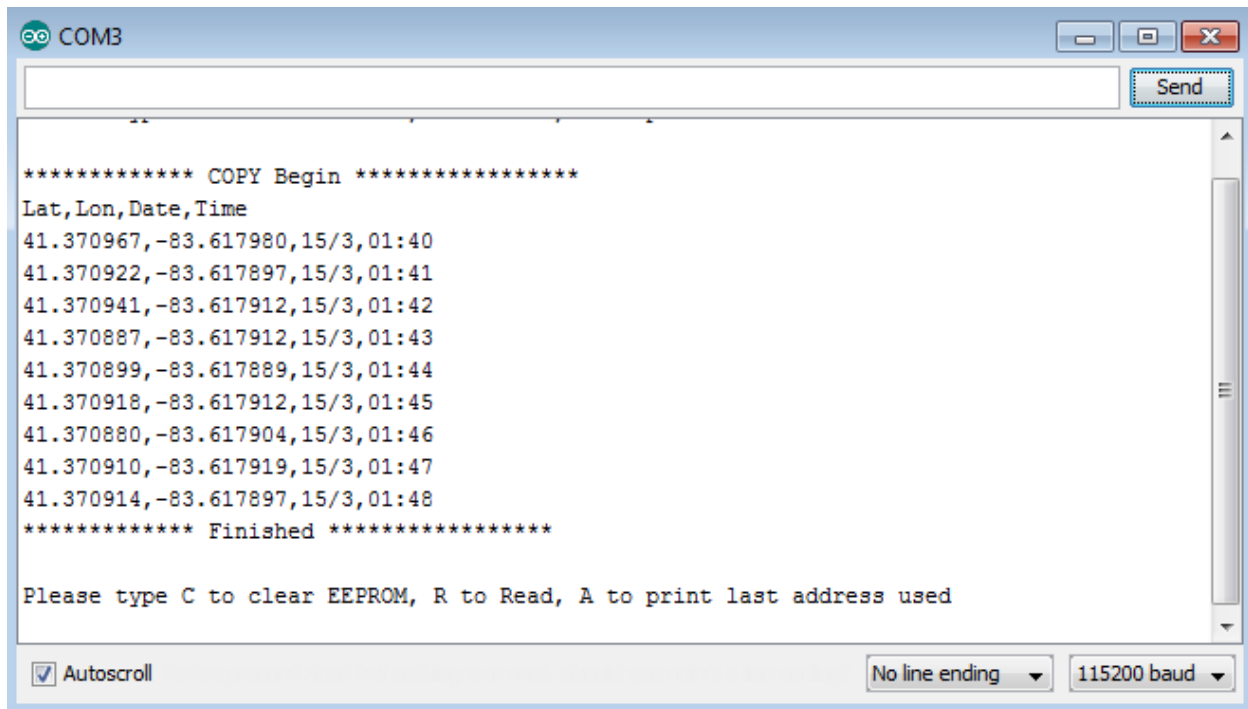
Turn the unit off or unplug the battery.

Plug unit back into the FTDI reader.

Load the TOS_ReadClear sketch and follow the instructions from earlier to bring up the command prompt.

Type “R” to read, but also try “Z” in case the unit missed recording the initial point.

The output should look like:



A screenshot of a serial terminal window titled 'COM3'. The window has a title bar with standard Windows controls (minimize, maximize, close) and a 'Send' button. The main text area displays the following output:

```
***** COPY Begin *****  
Lat,Lon,Date,Time  
41.370967,-83.617980,15/3,01:40  
41.370922,-83.617897,15/3,01:41  
41.370941,-83.617912,15/3,01:42  
41.370887,-83.617912,15/3,01:43  
41.370899,-83.617889,15/3,01:44  
41.370918,-83.617912,15/3,01:45  
41.370880,-83.617904,15/3,01:46  
41.370910,-83.617919,15/3,01:47  
41.370914,-83.617897,15/3,01:48  
***** Finished *****  
  
Please type C to clear EEPROM, R to Read, A to print last address used
```

At the bottom of the window, there is a status bar with the following controls:

- ☒ Autoscroll
- No line ending (dropdown menu)
- 115200 baud (dropdown menu)

This output can be copied into notepad and then saved as a .csv file. Use CTRL or CMD + C to copy; right-clicking will not bring up a menu.

.CSV files can be loaded into the DNR GPS (Formerly DNR Garmin) program, displayed in ArcGIS, or converted to a .kml/.kmz file and opened in Google Earth. There are also .csv to .kml converters available online.