Flashing the Arduino Dock

The Arduino Dock allows the Omega and the ATmega328P microcontroller interact with each other. Having an OS with the connectivity power of the Omega communicate easily with a microcontroller can be very powerful used effectively. We can do amazing things with the Omega communicating with the ATmega328P chip such as flashing the microcontroller wirelessly.

Programming and flashing a microcontroller mean the same thing, you are taking compiled code and uploading it to a microcontroller. The terms are often used interchangeably.

We'll first cover how to setup your computer and Omega, and then move on to cover how to actually flash your Arduino Dock.

Prerequisites

You'll need to first make sure that your Omega has connected to internet.

Then you'll want to ssh into the Omega's terminal in order to install the arduino dock package.

We've written a guide to connecting to your Omega's terminal via SSH in case you don't know how!

To install this package you'll need to use `opkg`. Enter the following commands on the command-line:

```
opkg update
opkg install arduino-dock-2
```

Accessing the Omega

The Omega must be accessible via its URL `http://omega-ABCD.local` where `ABCD` is your Omega's unique code.
The requirements vary depending on your Operating System:

Operating System	Actions Required
Windows	Install Apple's Bonjour Service
OS X	Nothing, good to go
Linux	Zeroconf services should already be installed, should be ok

Arduino IDE

This has to be done just once to enable flashing wirelessly.

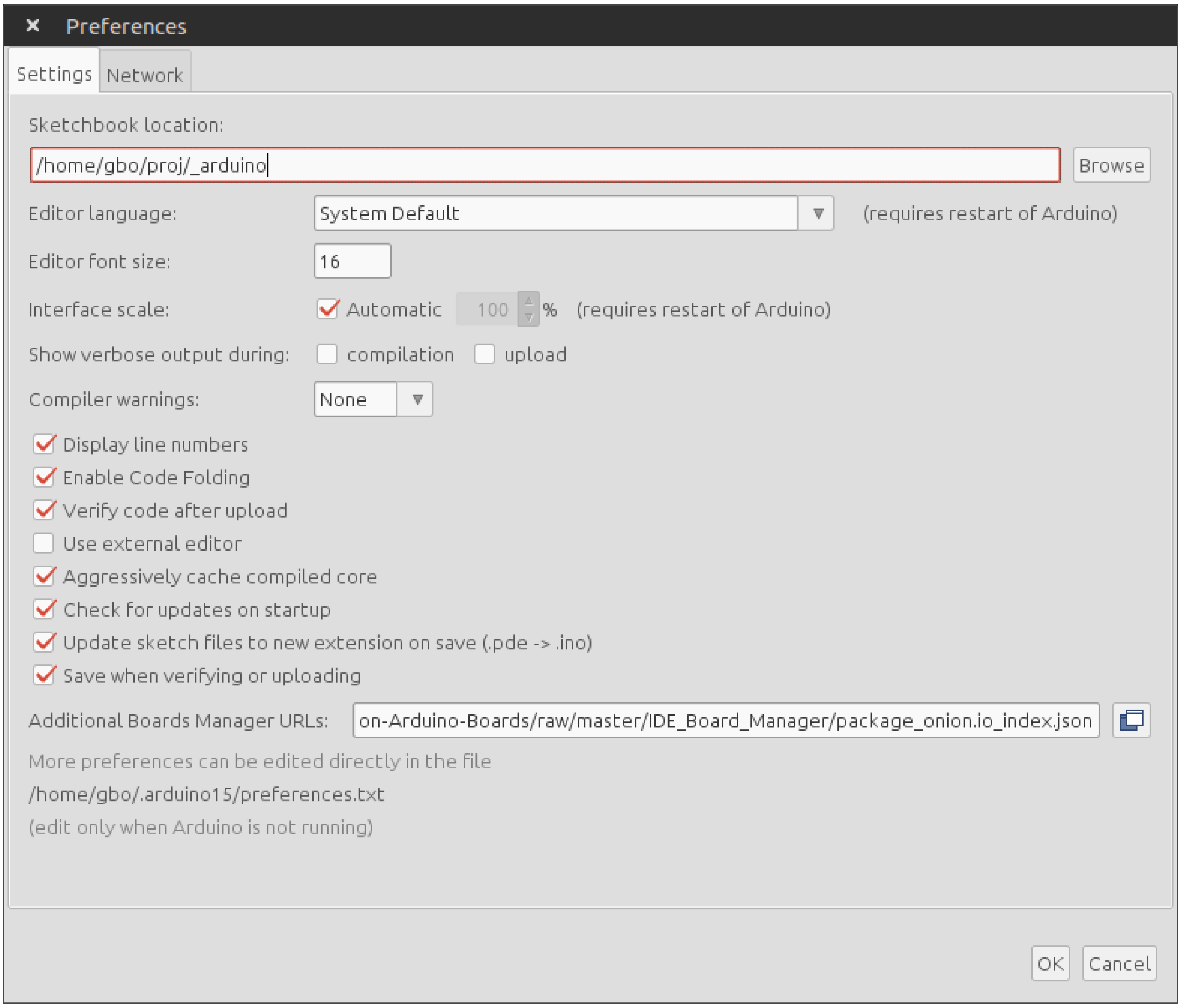
Install the latest Arduino IDE from the good folks over at Arduino. We did all of our testing using Version 1.8.0.

Installing the Arduino Dock Device Profile

Open the Arduino IDE and go to File -> Preferences. Copy this URL to our Arduino Dock device profile:

```
https://github.com/OnionIoT/Onion-Arduino-
Boards/raw/master/IDE_Board_Manager/package_onion.io_index.json
```

And paste it into the **Additional Boards Manager URLs** section near the bottom of the window.



Sketchbook location: /home/gbo/proj/_arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 16

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

Compiler warnings: None

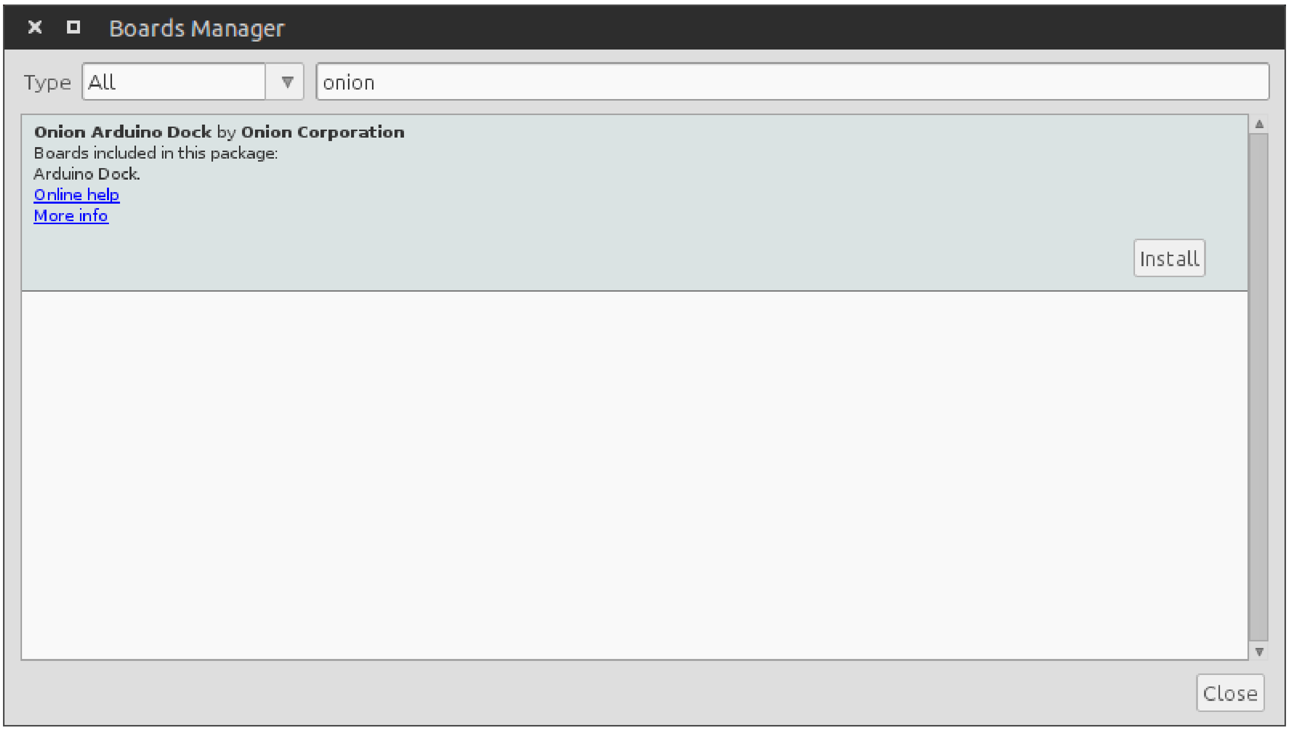
☑ Display line numbers
☑ Enable Code Folding
☑ Verify code after upload
□ Use external editor
☑ Aggressively cache compiled core
☑ Check for updates on startup
☑ Update sketch files to new extension on save (.pde -> .ino)
☑ Save when verifying or uploading

Additional Boards Manager URLs: on-Arduino-Boards/raw/master/IDE_Board_Manager/package_onion.io_index.json

More preferences can be edited directly in the file
/home/gbo/.arduino15/preferences.txt
(edit only when Arduino is not running)

If you already have links to other custom boards in your IDE, click on the button on the right of the text box. You can then add the URL in a new line.

Click OK, then go to Tools -> Boards -> Board Manager (at the top of the menu). In the search bar, type “Onion” and hit Enter. When the Onion Arduino Dock entry pops up, click on Install.



Onion Arduino Dock by **Onion Corporation**
Boards included in this package:
Arduino Dock.
Online help
More info

Click on Close to return to the IDE. The editor will now download the settings for the Arduino Dock and make it available as a board in the Tools -> Boards menu!

Doing the Actual Flashing

Now we get to the fun part, flashing sketches to the ATmega chip!

There are two methods for flashing the ATmega328P chip using the Omega:

- Using the Arduino IDE wirelessly
  - **We strongly recommend this option as it will work in almost all cases.**
- Compiling the file on your computer, copy it to the Omega, and then flash the chip from the command line
  - Only use this method as a backup plan in case you cannot upload using the IDE

Wireless Flashing with the Arduino IDE

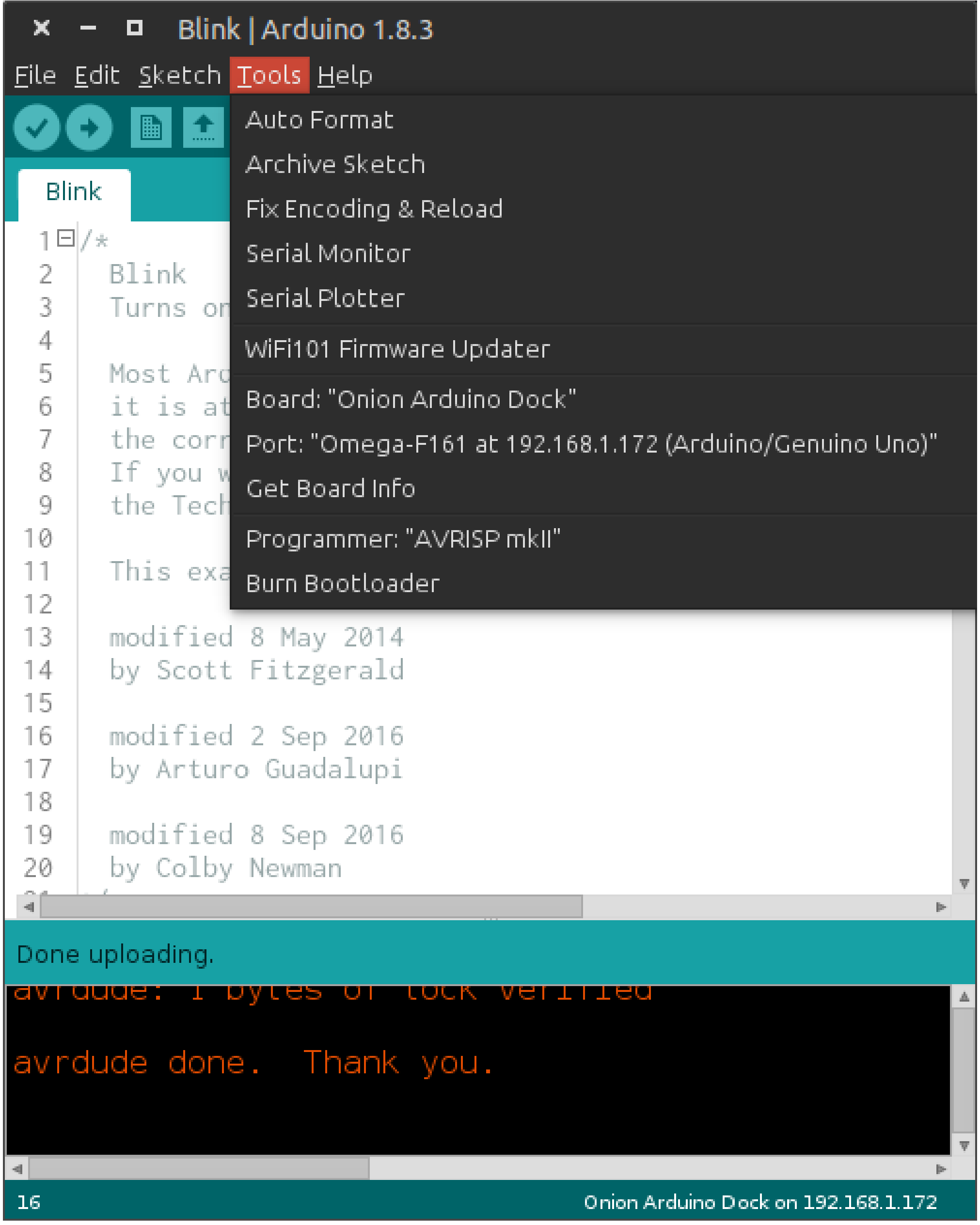
Thanks to the setup you did on your computer and the Arduino Dock, you can actually use the Arduino IDE on your computer to wirelessly flash Sketches to the Arduino Dock, so long as your computer and the Omega on your Arduino Dock are on the same WiFi network.

The process that takes place with this method:

1. Your computer and the Arduino IDE compile the Sketch
2. The compiled sketch is transferred to your Omega using SSH
3. The Omega will flash the microcontroller

The Steps:

In the Arduino Tools menu, select “Onion Arduino Dock” for the Board (near the bottom of the menu), and your Omega-ABCD hostname as the Port:



Board: "Onion Arduino Dock"

Port: "Omega-F161 at 192.168.1.172 (Arduino/Genuino Uno)"

Programmer: "AVRISP mkII"

```
/*
  Blink
  Turns on

  Most Ard
  it is at
  the corr
  If you w
  the Tech

  This exa

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi

  modified 8 Sep 2016
  by Colby Newman
```

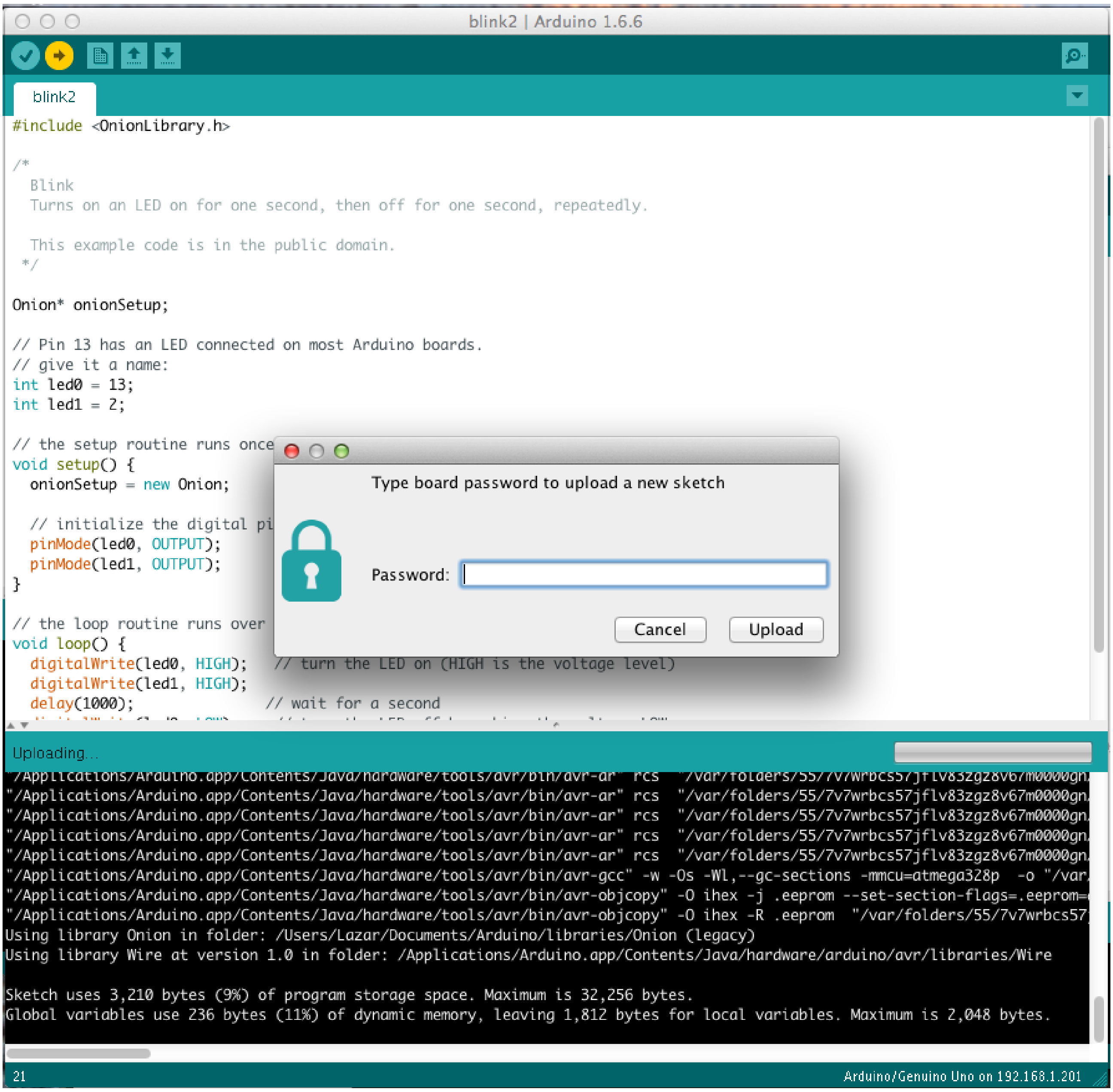
Done uploading.

```
avrdude: 1 bytes of lock verified

avrdude done.  Thank you.
```

16   Onion Arduino Dock on 192.168.1.172

If your Omega does not show up in the Port menu as a network port, restart the Arduino and wait for 30 seconds:

When your sketch is ready, hit the Upload button. Once the sketch is compiled, it will prompt you for your Omega password to upload the sketch. The password is `onioneer` by default:

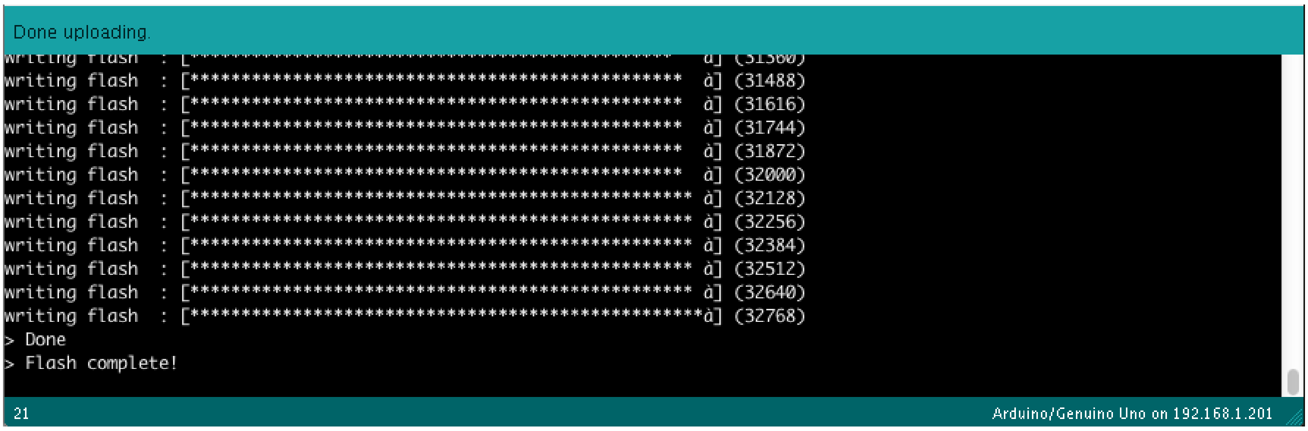

blink2 | Arduino 1.6.6
blink2
Type board password to upload a new sketch
Password:
Cancel
Upload
Uploading...
21
Arduino/Genuino Uno on 192.168.1.201

```
#include <OnionLibrary.h>

/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

Onion* onionSetup;

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led0 = 13;
int led1 = 2;

// the setup routine runs once
void setup() {
  onionSetup = new Onion;

  // initialize the digital pi
  pinMode(led0, OUTPUT);
  pinMode(led1, OUTPUT);
}

// the loop routine runs over
void loop() {
  digitalWrite(led0, HIGH);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(led1, HIGH);
  delay(1000);               // wait for a second
```

```
"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avr-ar" rcs  "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn
"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avr-ar" rcs  "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn
"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avr-ar" rcs  "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn
"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avr-ar" rcs  "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn
"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avr-ar" rcs  "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn
"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avr-gcc" -w -Os -Wl,--gc-sections -mmcu=atmega328p  -o "/var
"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avr-objcopy" -O ihex -j .eeprom --set-section-flags=.eeprom=
"/Applications/Arduino.app/Contents/Java/hardware/tools/avr/bin/avr-objcopy" -O ihex -R .eeprom  "/var/folders/55/7v7wrbcs57
Using library Onion in folder: /Users/Lazar/Documents/Arduino/libraries/Onion (legacy)
Using library Wire at version 1.0 in folder: /Applications/Arduino.app/Contents/Java/hardware/arduino/avr/libraries/Wire

Sketch uses 3,210 bytes (9%) of program storage space. Maximum is 32,256 bytes.
Global variables use 236 bytes (11%) of dynamic memory, leaving 1,812 bytes for local variables. Maximum is 2,048 bytes.
```

The IDE actually creates an SSH connection with the Omega to transfer the compiled hex file, and the Omega with then flash the ATmega microcontroller using 4 GPIOs.

Once the upload completes, the info screen will show something along the lines of:



```
writing flash  : [************************************************  à] (31360)
writing flash  : [*************************************************  à] (31488)
writing flash  : [*************************************************  à] (31616)
writing flash  : [*************************************************  à] (31744)
writing flash  : [*************************************************  à] (31872)
writing flash  : [*************************************************  à] (32000)
writing flash  : [**************************************************  à] (32128)
writing flash  : [**************************************************  à] (32256)
writing flash  : [**************************************************  à] (32384)
writing flash  : [**************************************************  à] (32512)
writing flash  : [**************************************************  à] (32640)
writing flash  : [***************************************************à] (32768)
> Done
> Flash complete!
```

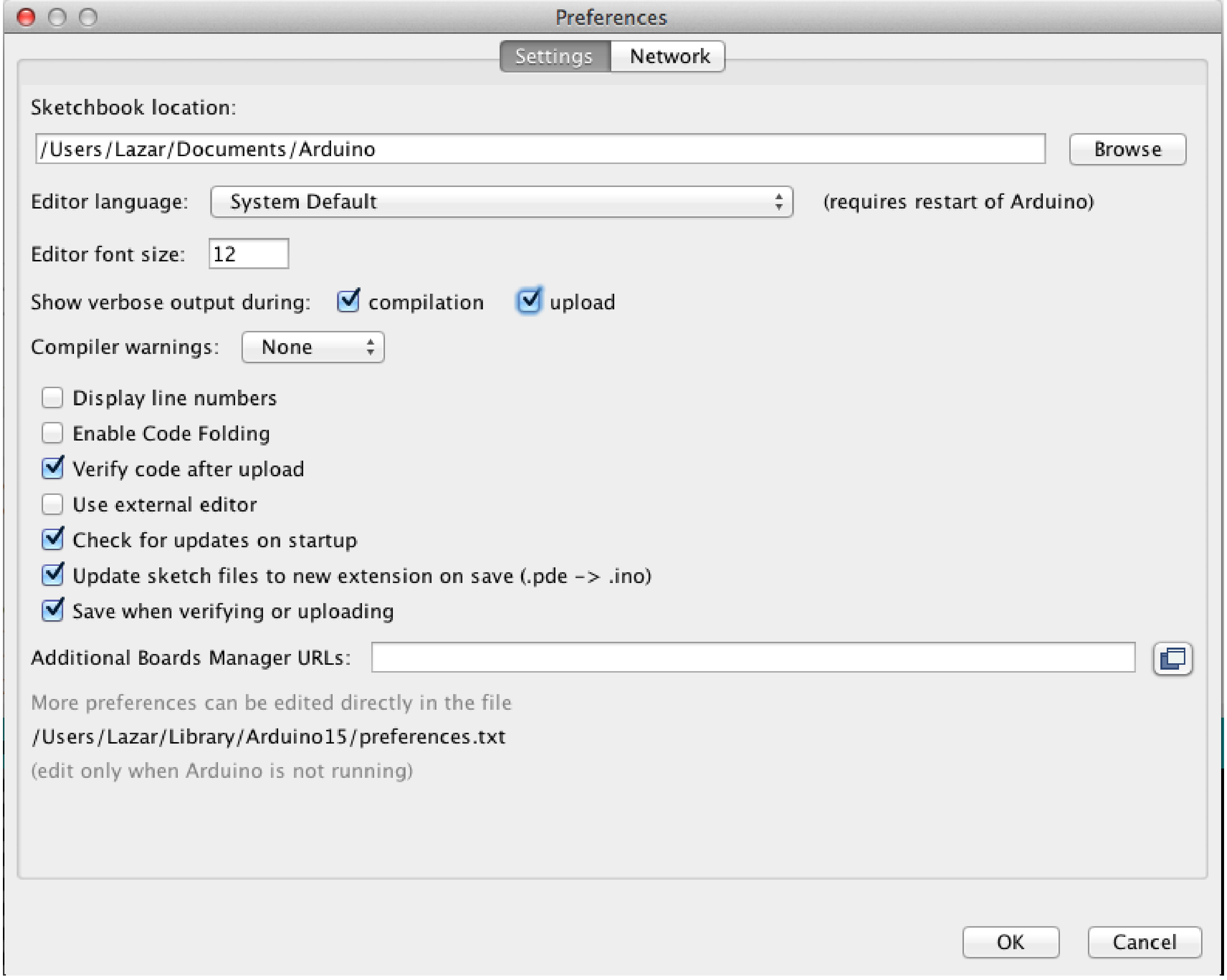
The ATmega chip is now running your sketch, enjoy!

Note: An orange message saying `ash: merge-sketch-with-bootloader.lua: not found` may appear in the info screen. You can **safely ignore this message**, it does not affect the sketch upload.

Manually Flashing on the Command line

Like we mentioned before this method should only be used as a backup to using the Arduino IDE. This is handy if the Arduino IDE cannot detect your Omega as a Network Port due to any connection/setup issues.

First, enable verbose output during compilation in the Arduino IDE Preferences:



Sketchbook location: /Users/Lazar/Documents/Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Show verbose output during: ☑ compilation ☑ upload

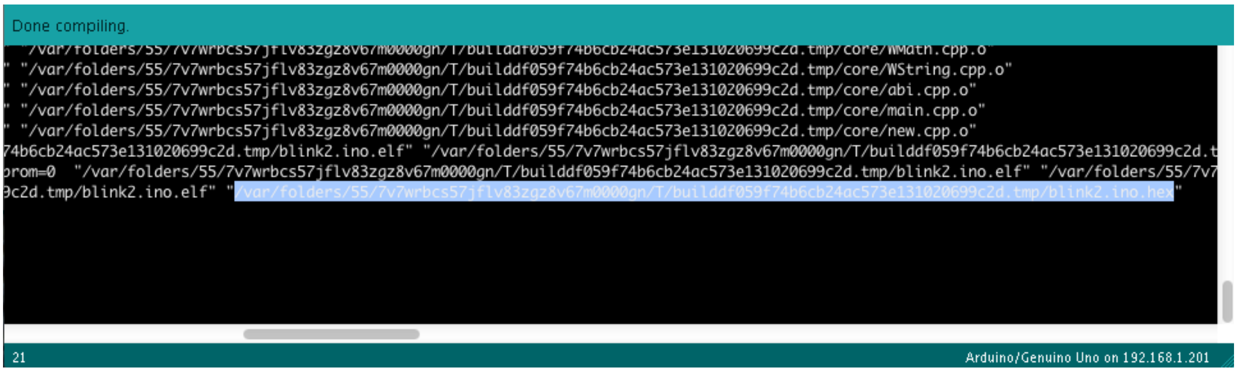
Compiler warnings: None

- ☐ Display line numbers
- ☐ Enable Code Folding
- ☑ Verify code after upload
- ☐ Use external editor
- ☑ Check for updates on startup
- ☑ Update sketch files to new extension on save (.pde -> .ino)
- ☑ Save when verifying or uploading

Additional Boards Manager URLs:

More preferences can be edited directly in the file
/Users/Lazar/Library/Arduino15/preferences.txt
(edit only when Arduino is not running)

Hit the verify button to compile the sketch, once it's complete you will have to scroll to the right to find the path to the compiled hex file:



Done compiling.

```
" "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn/T/builddf059f74b6cb24ac573e131020699c2d.tmp/core/WMath.cpp.o"
" "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn/T/builddf059f74b6cb24ac573e131020699c2d.tmp/core/WString.cpp.o"
" "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn/T/builddf059f74b6cb24ac573e131020699c2d.tmp/core/abi.cpp.o"
" "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn/T/builddf059f74b6cb24ac573e131020699c2d.tmp/core/main.cpp.o"
" "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn/T/builddf059f74b6cb24ac573e131020699c2d.tmp/core/new.cpp.o"
74b6cb24ac573e131020699c2d.tmp/blink2.ino.elf" "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn/T/builddf059f74b6cb24ac573e131020699c2d.t
prom=0  "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn/T/builddf059f74b6cb24ac573e131020699c2d.tmp/blink2.ino.elf" "/var/folders/55/7v7
9c2d.tmp/blink2.ino.elf" "/var/folders/55/7v7wrbcs57jflv83zgz8v67m0000gn/T/builddf059f74b6cb24ac573e131020699c2d.tmp/blink2.ino.hex"
```

Copy this path and then transfer the file to your Omega.

For more information on transferring files to your Omega from your computer you can check out our extensive guide to transferring files to your Omega

Now that the hex file is on your Omega, you can flash it to the ATmega chip from the Omega’s terminal:

```
sh /usr/bin/arduino-dock flash <hex file>
```

For example:

```
# sh /usr/bin/arduino-dock flash /root/blink2.hex
> Flashing application '/root/blink2.hex' ...
device          : /dev/i2c-0       (address: 0x29)
version         : TWIBOOTm328pv2.1??x (sig: 0x1e 0x95 0x0f => AVR Mega 32p)
flash size      : 0x7800 / 30720   (0x80 bytes/page)
eeprom size     : 0x0400 /  1024
writing flash   : [**************************************************#?] (3210)
verifying flash: [**************************************************#?] (3210)
> Done
```

The sketch has been flashing and is running on the Arduino Dock, enjoy!