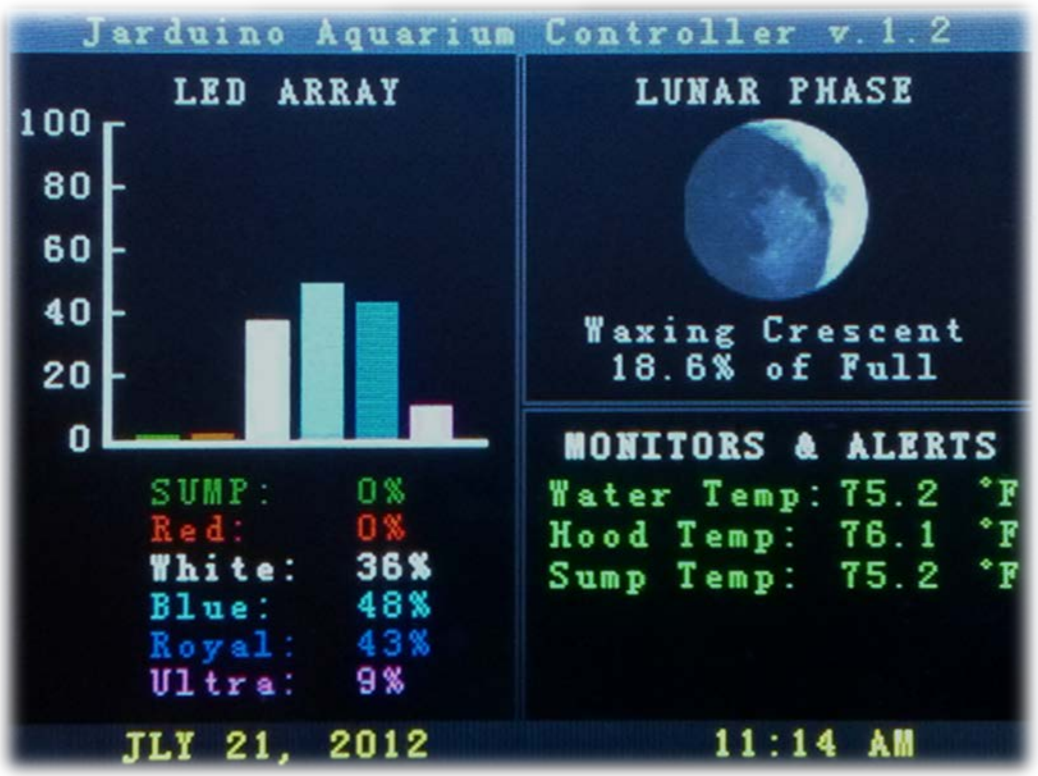
JARDUINO AQUARIUM CONTROLLER

USER MANUAL

version 1.2 beta


Jarduino Aquarium Controller v.1.2

LED ARRAY

SUMP: 0%
Red: 0%
White: 36%
Blue: 48%
Royal: 43%
Ultra: 9%

LUNAR PHASE

Waxing Crescent
18.6% of Full

MONITORS & ALERTS

Water Temp: 75.2 °F
Hood Temp: 76.1 °F
Sump Temp: 75.2 °F

JLY 21, 2012 11:14 AM

BY JAMIE JARDIN

LICENCE

DOCUMENT PURPOSE

This User Manual presents a brief description of the required hardware needed to build the main part of the Jarduino and how to assemble it. Also included is a step-by-step tutorial on how upload the program sketch to an Arduino MEGA 2560 and a detailed description of the Jarduino Aquarium Controller functionality.

CHANGE HISTORY

VERSION	DATE	AUTHOR	SUMMARY OF CHANGES
1.2 beta	September 2012	Jamie Jardin	Initial Draft

STANDARD LEGAL DISCLAIMER

As with any project that uses items that cut, burn, chop, fall, rotate, flog, zap, blind, etc., caution is most important. Please be advised that I take absolutely no responsibility for your actions regarding your use of any material provided here.

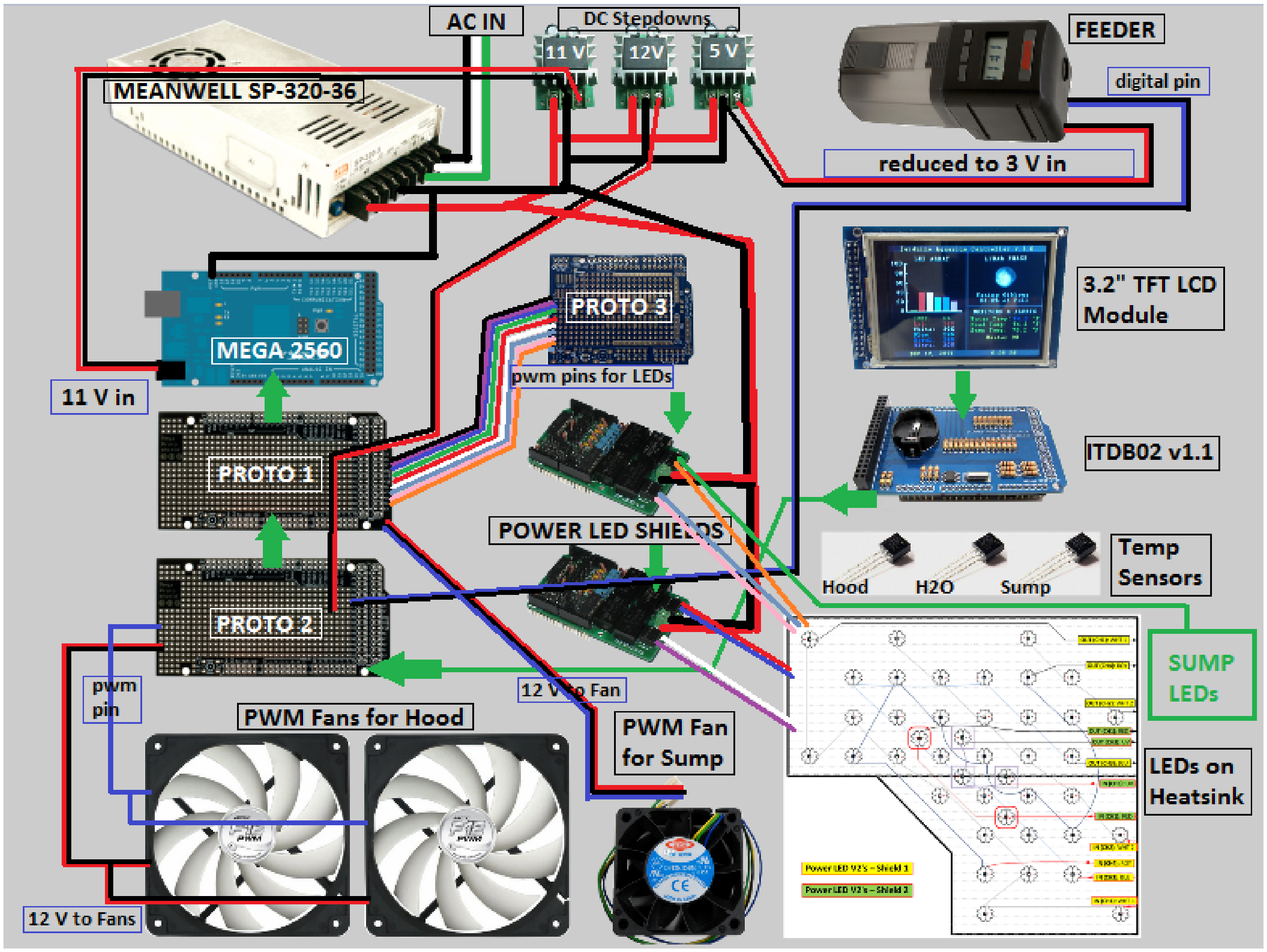
Articles and information provided are for educational purposes only. There is no substitution for official manufacturer's instructions and professional advice. Please contact the product manufacturer before modifying any devices or software. Please contact a licensed professional before attempting anything physical or following any advice given here. Neither the author, Jamie Jardin, nor any other contributors are responsible whatsoever for any damage incurred by following any instructions or advice provided.

Although it would be difficult to do, if you follow these articles and it leads to an electrical short that burns your house down - it's not my problem. If your wife divorces you shortly after, not my problem. If you lose your job and end up on the street - not my problem. If your bank account is empty - not my problem. If your credit cards are maxed out, I may share your pain, but it's still not my problem. If you follow through with this, any and all consequences are your own problem and I will not be held responsible in any way.

TABLE OF CONTENTS

1. INTRODUCTION

The Jarduino Aquarium Controller is an Arduino based controller with a 3.2" color Touch Screen LCD Display. Its user-friendly graphical interface is capable of dynamically controlling & monitoring LEDs, fans, and powerheads, as well as controlling a heater, a chiller, an alarm, and an automatic fish feeder. It's also capable of being expanded with relative ease. The code was originally inspired by the Krusduino and Stilo projects. An *example* of the electronics and main accessories the Jarduino can control is diagrammatically depicted below:



NOTE: The above diagram is a rough depiction of my personal setup. It leaves out a few peripherals that the Jarduino controls & monitors, and this USER MANUAL does NOT cover how to hook up everything that's pictured. For more "complete" details on my build (how I got started, my reasons for doing what I did, how I hooked everything up, and various other details), please refer to my build thread available on UKreefs. My "handle" on the forums is "TheDOdblG."

MAIN FEATURES (Jarduino Aquarium Controller v.1.1):

- Control up to 5 separate LED Channels for White, Blues, Royal Blues, UV, & Red (or any color choices)
- Control a 6th Channel for a Sump/Refugium LED Light
- Control a 7th Channel that mirrors a Real Time Lunar Cycle
- Advanced LED testing features & simulations
- Water & LED Heatsink Temperature Sensors
- Control of a Heater and a Chiller
- Audible & Visual Alarm Notices for Defined Temperature Variances
- Dynamic Speed Control for Heatsink Fans (Display & Sump Lights)
- Wave Maker / Powerhead control (Various Synchronous & Alternating Pulse modes)
- Automatic Fish Feeder (Schedule up to 4 feedings a day OR Feed on Demand)
- User-Selectable Formatting
    - Celsius & Fahrenheit Temperature Scales
    - 24HR & 12HR Time Formats
    - Month DD, YYYY & DD/MM/YYYY Date Formats
- Settings & Preferences stored in EEPROM (Arduino Memory)

MAIN FEATURES – UPDATES (Jarduino Aquarium Controller v.1.2 beta)

- Added support for Arduino 1.0 IDE (Must use my patched version)
- Fully compatible with Arduino 1.0.1 IDE (No need to use my patch)
- Can choose between ITDB02_Graph16.h Library or UTFT.h Library by Henning Karlsen
    - All compatible controllers (and modules) are now included within the sketch
    - Approximately 40 different LCD modules to choose from!
- Replaced Matt Joyce's DS1307 Library with Henning Karlsen's DS1307 Library
    - Modified formatting in Karlsen's DS1307 Library
    - Rewrote all RTC coding in sketch, based on Karlsen's "ITDB02 Analog Clock"
    - Added "Day of the Week" to the "Date & Time Bar."
    - Fixed setting a date that does not exist possibility (ie. FEB 31, 2011)
    - Fixed a minor bug with the Time & Date Bar
- Added 24 Hour Time formatting to the "Test LED Array Output Settings" screen
- Modified "View/Change Moon LED Max Output"
    - Added descriptive pics and the ability to set/save the Min & Max Illuminations
- Replaced LED Output Testing "Widget" with "Slider Bars"
- Upgraded "Change LED Output Values" page with "Slider Bars"
- Included more Built-In Protections and Redundancies
- Added an additional page of user settings including:
    - Choice of showing the Day of the Week
    - Setting the Fan(s) startup temperature
    - Added four additional user-defined options to the Screensaver
- Added an Auto-Dimming Feature (with options accessible through the General Settings)
    - Can choose to Dim the LEDs at a user defined temperature
    - User defined dimming percentage
    - Visual notification on Main Screen if feature was automatically utilized
- Changed the look of some buttons as well as made some other visual tweaks

EXPECTED UPDATE (Jarduino v.1.3)

- WiFi and possibly Android/iPhone App

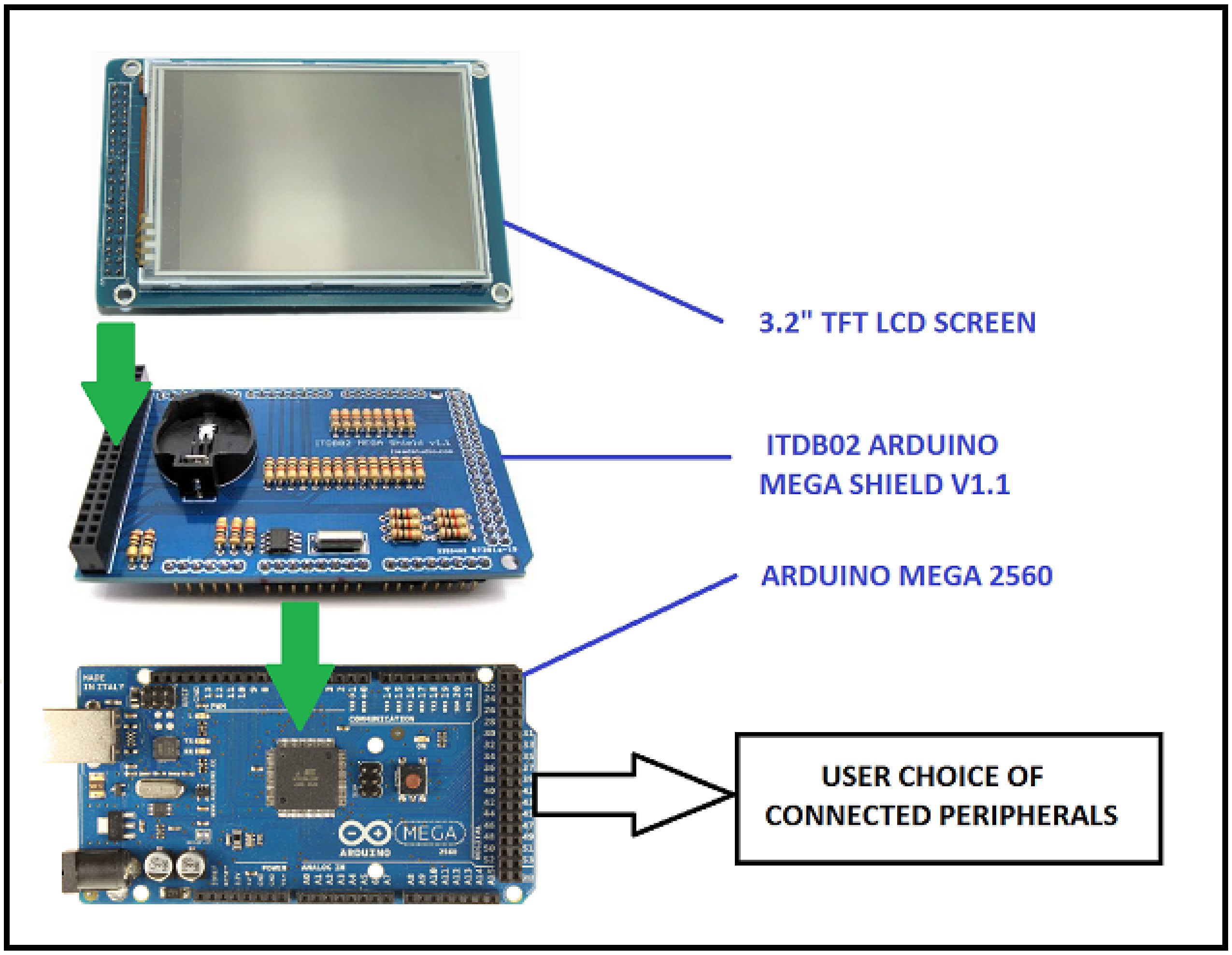
POSSIBLE FUTURE DEVELOPMENTS

- Improved User Interfaces (Main Screen Panels & Shortcuts)
- Replace the ITDB02_Touch.h Library with the latest release (when available)
- Random Weather (Clouds, Thunderstorms, etc. along with Increased Wave Action)
- Real Weather (Mirror the real weather conditions from user-selected locations)
- Modification of Wave Maker to include Tidal Forces with Lunar Cycle
- Automatic Water Changer & Top Off (AWC & ATO)
- Inclusion of Other Sensors:
  - pH
  - Salinity
  - Calcium
  - Orp
- Automatic Dosing
- Wireless Peripherals
- Video Monitoring (check out your tank from any computer or smartphone)

*** PLEASE NOTE – I do not have a specific timeframe for the release of *any* of the upgrades listed. I'm a one man show and a new dad, so please be patient!**

2. JARDUINO BUILD INSTRUCTIONS

When it comes to building a Jarduino Aquarium Controller, you have several options. In the INTRODUCTION to this manual, you were greeted with a rough layout of my own personal design. On the surface, it may look complicated, but I think that impression may be made based off of the number of peripherals I've included with my setup. The fact is, you can use the Jarduino Aquarium Controller to control and monitor as few peripherals as you like (ie – use it to control a single string of LEDs), or as many as I currently do and more! When you look at the basics, I believe you will come to the realization that it is actually quite simple. Really, all that is needed to get started is a few key components, as laid out in the diagram below:


3.2" TFT LCD SCREEN
ITDB02 ARDUINO MEGA SHIELD V1.1
ARDUINO MEGA 2560
USER CHOICE OF CONNECTED PERIPHERALS

HARDWARE

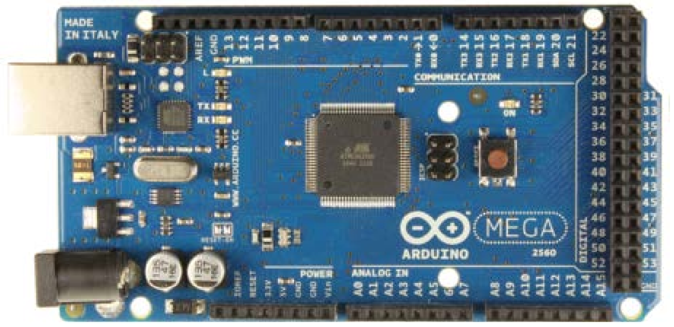
Above, you have three basic components that all get pressed together. At the base, there's an Arduino MEGA 2560. It is a microcontroller board that is the "brains" behind the operations. Immediately above the Arduino is an ITDB02 Arduino MEGA Shield v1.1 with a Built-In Real Time Clock (RTC). This shield serves as an interface between the Arduino and the 3.2" Touch Screen LCD that's pictured at the top of the diagram. Keep in mind that there are a few alternatives for each of the above hardware components, which I'll touch upon soon. But essentially, those three components (or alternatives of those three), are all that is needed to build a Jarduino Aquarium Controller. The only thing left after you have built the Jarduino Controller is decide what components you want it to control / monitor, and then hook them up. Ok, I admit it can be a little tricky interfacing various peripherals, but once you figure out how to hook the first one up, then the next thing you interface becomes a little bit easier, then more easier still for a third component, then by the end of your build you can sit back and wait for the top technology companies to start a bidding war over signing you and your newly acquired expertise in electronics to a long-term lucrative work contract. ☺

PARTS CHECKLIST

In this section I present you with a shopping list. While it doesn't cover every nut and bolt needed, it does list all the main components and provides several recommendations. I tried to include hyperlinks to some places from which I've made a purchase from in the past and trust. If you are trying to save some money, a lot of these parts can be acquired relatively cheap on eBay or elsewhere online. You can simply follow any link I provide, get the specs from that site, and try to find them cheaper elsewhere. Google is your friend when it comes to finding the parts you need. *Shopping tip:* sometimes it's a good idea to get several parts from one place, as your total cost may be cheaper considering shipping costs.

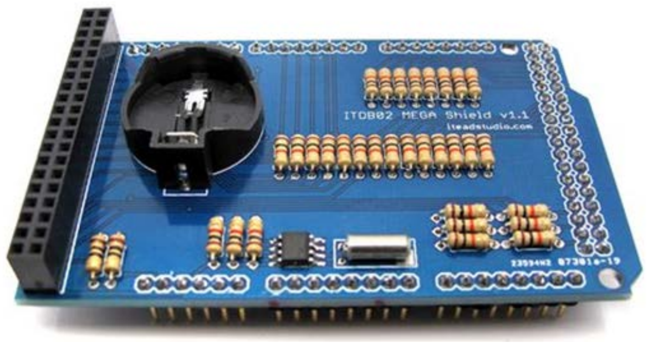
REQUIRED PARTS for the Jarduino Aquarium Controller:

- ☐ **Arduino MEGA 2560 -** eBay (pay attention to "seller feedback" ratings!)

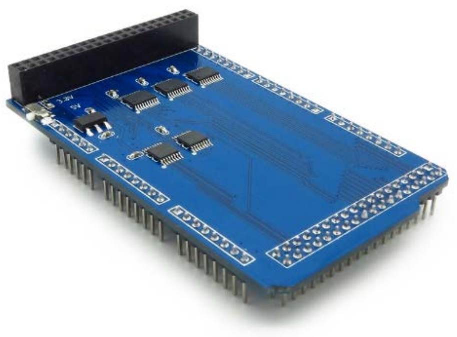


NOTE: There are several versions / variations of the Arduino MEGA 2560. Any of them should be compatible with the Jarduino.

- ☐ **ITDB02 Arduino MEGA Shield Kit v1.1** – iTeadStudio.com / iMall (The link I provided is to a kit that you have to assemble yourself. There are some places that sell it fully assembled)



NOTE: There's a new version of this shield available: ITDB02 Arduino MEGA Shield v2.0. This newest release DOES NOT have a built-in Real Time Clock (RTC), so if you decide to get that version, you will need to add an RTC circuit to it yourself in order to make it compatible (which I will show you how in the ASSEMBLY section below). There are also a few other shields out there by other makers. None of these shields I've seen currently have a built in RTC, which the Jarduino does REQUIRE. And for some reason, the ITDB02 v1.1 is less expensive! I only recommend getting a version other than the ITDB02 Shield v1.1 in the event you cannot find the ITDB02 Shield v1.1.



Pictured on the left is an ITDB02 MEGA Shield v2.0, on the right is version made by "ELECFreaks." Neither version has an RTC!

☐ **3.2" TFT LCD** – iTeadStudio.com / iMall (I went with iTead for this one because I was able to save on shipping by getting this and the ITDB02 Shield Kit v1.1 together)

*NOTE: Below is a truncated list of **Compatible TFT LCD Modules** that was originally compiled by Henning Karlsen, who wrote the ITDB02_graph16 and the UTFT Arduino Libraries. I use the word "compatible" here loosely. **The Jarduino was written explicitly for a screen size of 320 x 240 pixels**. Therefore, while the LCD Modules listed below may be "compatible," they might not necessarily display correctly if the screen isn't exactly 320 x240. For instance, if you wanted to use the 5 inch ITDB02-5.0, which is 720 x 480 pixels, then the Jarduino fill only part of the screen (the first 320 pixels across followed by a 720-320=**400** pixel wide black bar to its right, and the first 480-240=**240** pixels going down followed by a 240 pixel wide black bar below it). If you really want to use that particular screen (and properly fill it up), or any other non-320 x240, then you will have to go through the Jarduino Sketch and change all the lines in the code that have to do with screen formatting. While this is not impossible, keep in mind that my code is nearly 6000 lines long!*

The following LCD MODULES have been tested and works with the ITDB02_graph16 Library. This library is FASTER than the UTFT Library but has LESS compatibility. Other modules should work as long as they have one of the supported controllers. I highly recommend using one of the following 3.2 inch modules (highlighted in yellow) due to the screen size (320 x 240):

Supplier: ITead Studio

Module	Model for ITDB02()	Controller	Notes
ITDB02-3.2	ITDB32	HX8347-A	
ITDB02-3.2WC	ITDB32WC	ILI9327	
ITDB02-3.2S	ITDB32S	SSD1289	

Supplier: ElecFreaks

Module	Model for ITDB02()	Controller	Notes
TFT01-2.4	TFT01_24	ILI9325D	
TFT01-3.2	TFT01_32	SSD1289	
TFT01_3.2W	TFT01_32W	ILI9327	

The following LCD MODULES have been tested and works with the UTFT Library. This library is SLOWER than ITDB02_graph16 but has MORE compatibility. Other modules should work as long as they have one of the supported controllers:

Supplier: ITead Studio

Module	Model for ITDB02()	Controller	Notes
ITDB02-1.8SP	ITDB18SP	ST7735	
ITDB02-2.2SP	ITDB22SP	HX8340-B(N)	
ITDB02-2.2	ITDB22	HX8340-B(T)	
ITDB02-2.4	ITDB24	ILI9325C	Retired
ITDB02-2.4D	ITDB24D	ILI9325D	
ITDB02-2.4DWOT	ITDB24DWOT	ILI9325D	
ITDB02-2.4E	ITDB24E_8	S6D1121	8bit mode
ITDB02-2.4E	ITDB24E_16	S6D1121	16bit mode
ITDB02-2.5H	ITDB25H	S1D19122	
ITDB02-2.8	ITDB28	ILI9325D	Retired
ITDB02-3.2	ITDB32	HX8347-A	Retired
ITDB02-3.2WC	ITDB32WC	ILI9327	
ITDB02-3.2S	ITDB32S	SSD1289	
ITDB02-3.2WD	ITDB32WD	HX8352	
ITDB02-4.3	SSD1963	SSD1963_480	
ITDB02-5.0	SSD1963	SSD1963_800	

UTFT Library compatible LCD MODULES (continued):

Supplier: ElecFreaks

Module	Model for ITDB02()	Controller	Notes
TFT01-2.4	TFT01_24_8	ILI9325D	8bit model
TFT01-2.4	TFT01_24_16	ILI9325D	16bit model
TFT01-3.2	TFT01_32	SSD1289	
TFT01_3.2W	TFT01_32W	ILI9327	Retired
TFT01_3.2WD	TFT01_32WD	HX8352	

Supplier: NKC Electronics

Module	Model for ITDB02()	Controller	Notes
RGB LCD 65K color module	LPH9135	PCF8833	

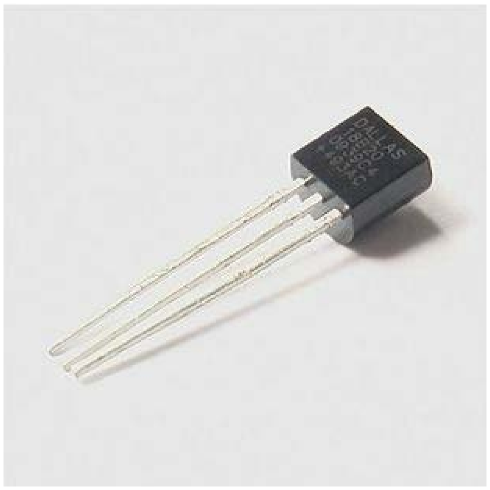
If you are unsure of which LCD Module you have, you can try the following module codes for the supported controllers instead with the UTFT Library:

Controller	Model for UTFT()	Supported mode		
		8bit	16bit	Serial
HX8340-B(N)	HX8340B_S			✓
HX8340-B(T)	HX8340B_8	✓		
HX8347-A	HX8347A		✓	
HX8352-A	HX8352A		✓	
ILI9325C	ILI9325C	✓		
ILI9325D	ILI9325D_8	✓		
ILI9325D	ILI9325D_16		✓	
ILI9327	ILI9327		✓	
PCF8833	PCF8833			✓
S1D19122	S1D19122		✓	
S6D1121	S6D1121_8	✓		
S6D1121	S6D1121_16		✓	
SSD1289	SSD1289		✓	
SSD1963	SSD1963_480		✓	
SSD1963	SSD1963_800		✓	
ST7735	ST7735			✓

RECOMMENDED PARTS for the Jarduino Aquarium Controller:

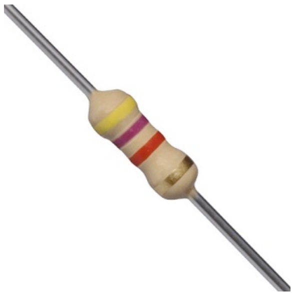
❖ TEMPERATURE SENSOR PARTS

- ☐ **DS18B20 Temperature Sensors** – eBay (There's some seller's that offer these in "lots." I recently purchased 10 for around $7 USD. There are even some vendors that sell them attached to a cable and sealed in water-proof casings, like the one pictured on the right).



NOTE: The Jarduino code is explicitly written to utilize these particular temperature sensors. I used three of these in my build. You can use the same amount, more, or less in yours. With a few modifications to the code & pin hookups, alternative temperature sensors (such as analog LM35 temp sensors) can be substituted. With LM35's, you would however be limited to the number of analog pins that are available.

- ☐ **4.7 kΩ resistors** – Any Local Electronics Store or Online



NOTE: Needed for the DS18B20 temperature sensors; you need only 1 of these no matter how many DS18B20's you decide to use.

❖ REAL TIME CLOCK (RTC) MODULE

***** *NOTE: This is ONLY if you utilize a shield other than the ITDB02 Arduino MEGA v1.1.*

- ☐ **DS1307 RTC Module** – eBay or elsewhere online (I purchased this particular module on eBay for $2.85 USD. It had free shipping and came with the battery.)

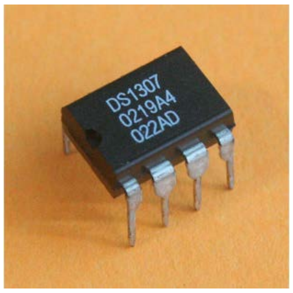
TOP VIEW

BOTTOM VIEW

❖ REAL TIME CLOCK (RTC) PARTS

***** *NOTE: This is ONLY if you utilize a shield other than the ITDB02 Arduino MEGA v1.1. You can use a pre-assembled RTC module like the one above, or you can build your own. Below, I list the parts you need to build your own.*

☐ **DS1307 RTC Chip** – eBay or elsewhere online



NOTE: This is a Serial Real Time Clock chip

☐ **32.768kHz Crystal** – eBay or elsewhere online

NOTE: This is the clock source for the RTC chip

☐ **10 kΩ resistors** – Local Electronics Store or Online

NOTE: You will need two of these for the RTC circuit.

☐ **CR2032 Battery** – Available in Most Stores that Sell Batteries

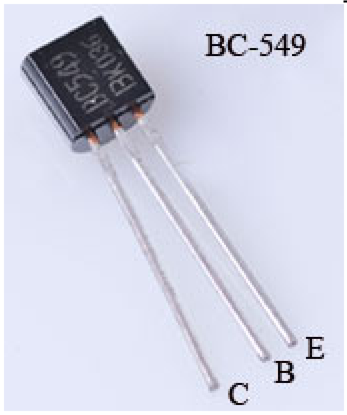
NOTE: This is for RTC Battery Backup, and like most batteries, there are several brands available. The Jarduino should still work fine

☐ **CR2032 Battery Holder** – eBay or Online

NOTE: This is a nice little holder for your RTC battery backup.

❖ MISCELLANEOUS PARTS

☐ **BC549CNPN Transistors** – Local Electronics Store or Online

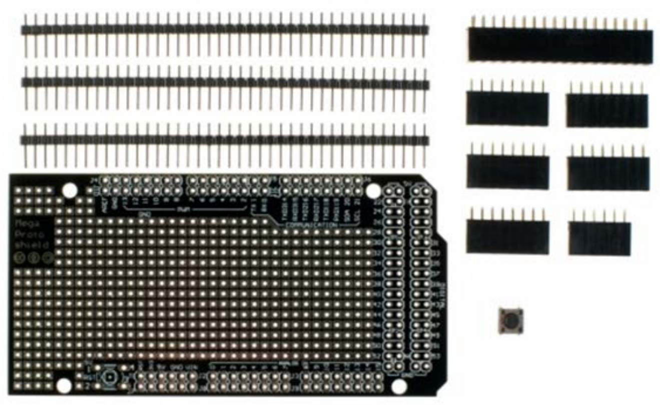


NOTE: Used in various circuits for some of the peripherals (fans, feeder, and the alarm); get one of these for each peripheral you plan on implementing. You can easily substitute Transistors that have similar specs, such as the BD137 NPN Transistor.

☐ **1 kΩ resistors** – Local Electronics Store or Online

NOTE: Used in various circuits for peripherals, such as fans, feeder, alarm, etc.; get one for each peripheral except for LEDs. Use one of these every time you use of the transistors from above.

☐ **Arduino MEGA Prototyping Shields** – DFRobot (I went with DFRobot because their Arduino MEGA protoboard seems to offer the largest amount of prototyping real estate when compared to others I've seen)

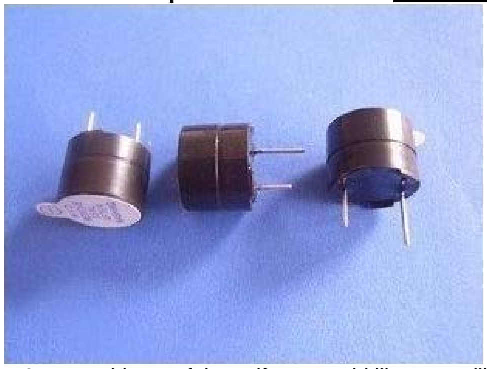


NOTE: Used for interfacing all the accessories and what not to the Arduino. This is by no means the only option. But it gets the job done, and provides a lot of room to put all your circuits. I used two of these in my build. ***Potential alternatives include:*** *direct connections to the Arduino underside, connections to the ITDB02 shield, perfboard, various other proto shields, and screw shields to name a few). If you decide to use prototyping shield(s), I highly recommend using ones intended for the Arduino MEGA.*

OPTIONAL PARTS for the Jarduino Aquarium Controller:

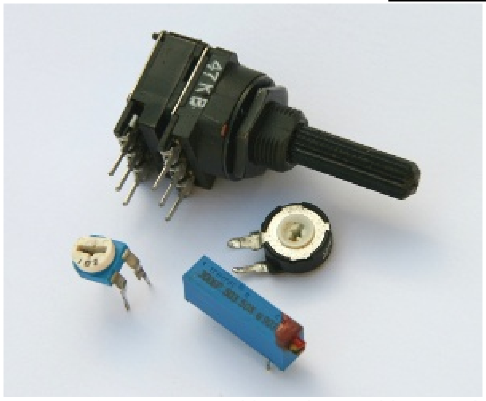
❖ AUDIO ALARM PARTS

- ☐ **5V Passive Speaker / Buzzer** – Local Electronics Store or Online



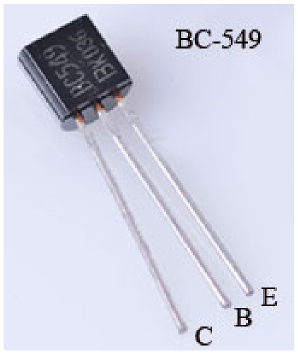
NOTE: Add one of these if you would like an audible Alarm. The Jarduino will sound an alarm if your water temperature goes above or below user-define temperature thresholds. This same alarm can be used to sound other user-defined alarms with a few lines of coding.

- ☐ **Multi-Turn Variable Resistor** – Local Electronics Store or Online



NOTE: If you want to add an Alarm, add one of these to control the volume. A variable resistor / potentiometer with a maximum value of 10 kΩ should suffice. There are many of different types from which to choose.

- ☐ **BC549CNPN Transistor & 1 kΩ Resistor** – Local Electronics Store or Online



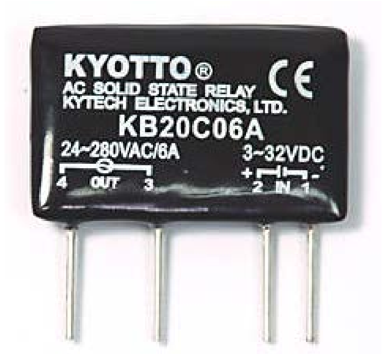
NOTE: Both of these were already listed above under Recommended Miscellaneous Parts. I simply listed them here again to remind you that they are part of the Alarm Circuit.

❖ WAVEMAKER PARTS

☐ **Electrical Outlet / Outlet Box / Wall Plate** – Local Hardware Store

NOTE: Pictured on the left is a 15 amp Duplex Electrical Outlet that is typically found in US homes. The other items pictured are some options available to encase the outlet (single-gang outlet box & wall plate). Substitute these items for whatever outlets are native to your country. You are not limited to using these particular types of outlets. You can even modify wall strips using the same parts and circuit.

☐ **KYOTTO KB20C06A Solid State Relay (SSR)** – eBay or Elsewhere Online



NOTE: These particular relays have a very low leakage current, so these aren't necessarily ideal for running powerheads. I however use them in my build, and they work great with the Hydor Koralia Powerheads. I went with SSR's because they are completely silent. You can easily substitute mechanical relays with similar specs. They are much less expensive, but make noticeable clicking sounds when activated.

☐ **ULN2803A Darlington Transistor** – eBay or Elsewhere Online

NOTE: This transistor will accommodate up to 8 individual outlets.

☐ **18-Pin DIP IC Socket** – eBay or Elsewhere Online

NOTE: This is completely optional. They do come in handy if you every need to remove / replace the ULN2803A.

☐ **X2 Capacitors (0.22µF / 270VAC)** – eBay or Elsewhere Online

NOTE: Here's another optional part. These caps simply serve as a snubber, and I put them in place to be in "compliance" with the EMC Directive. But it's DIY, so who really cares? If you do, then get one per outlet.

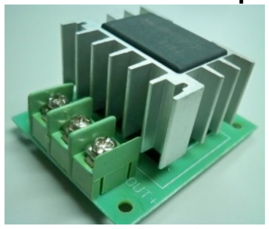
Want to control other AC components such as a Heater, a Chiller, T5 Lights, Calcium Reactor, or something else? Then you can use the exact same parts I listed above for the Wavemaker to control them. Just add an additional outlet for each additional AC peripheral. In addition to the Wavemaker, the Jarduino is already coded to control a Heater & a Chiller. Other AC peripherals are fairly easy to code for and a cinch to control.

❖ MORE MISCELLANEOUS PARTS

☐ **IN4001 Diode** – Local Electronics Store or Online

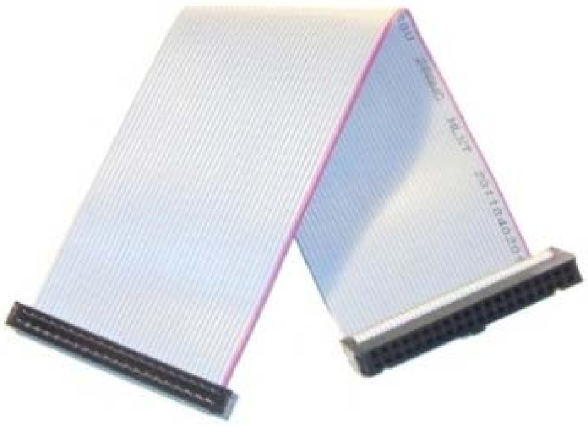
NOTE: I used this particular diode for circuit protection in my fans. You can get away with not using them. You can also substitute with diodes that have similar specs. Do whatever makes you comfortable.

☐ **HRD Series DC Step Down Converters (Adjustable 36V-5V)** – eBay



NOTE: I used 3 of these to lower the voltage from my main power supply to whatever voltage I needed for the various peripherals. In particular, I used it to lower the 36 Volts from the PSU down to 11 V for my Arduino, 12 V for my fans, and 3 V for my Automatic Feeder (I added a few diodes in series to drop the voltage to 3 V, since the converter only dropped it down to 5v). I used these so I didn't need additional power supplies for my peripherals. Everything runs off of one main power supply!

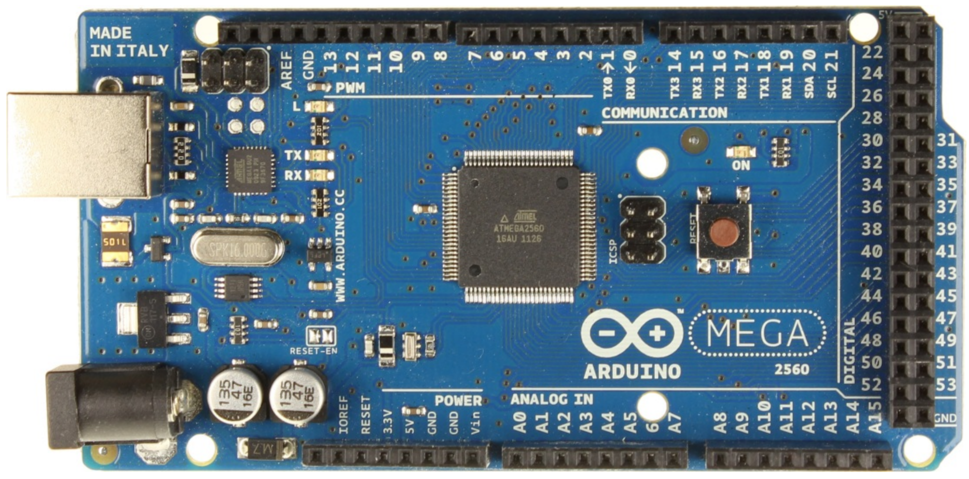
☐ **40-Pin IDE Ribbon Cable (Male-Female)** – Local Electronics Store or Online



NOTE: You can use this to extend the LCD Screen away from the ITDB02 Arduino MEGA Shield. Make sure you get a 40-pin IDE cable that has one end that's FEMALE, and the other end MALE. Also, make sure there are no "blocks" in the connectors. These connectors are commonly used to connect CD/DVD ROMs to a Motherboard. IT IS UNKOWN how long of a cable you can use. I've successfully tested up to 30 cm (1 ft.) without any issues.

JARDUINO HARDWARE ASSEMBLY

There are a few things I want to make sure you understand before you start assembling your aquarium controller. First, let's talk a little bit about the Arduino MEGA 2560. I'm not going to bore you with all the specs and features. I simply want to make sure you understand where to "plug" things in when I instruct you to do so. So again, here's a picture of an Arduino MEGA 2560:

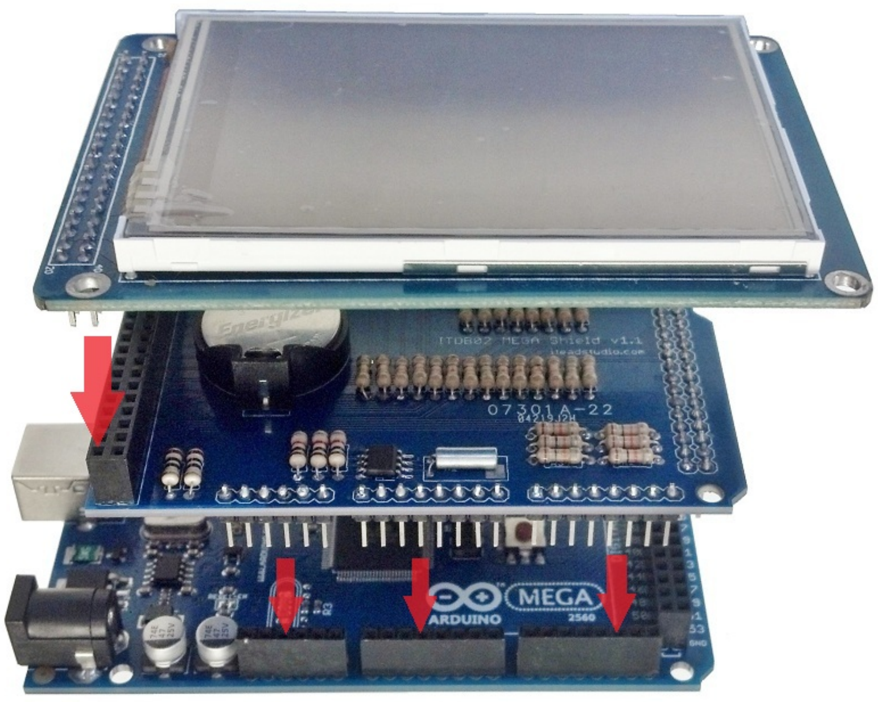


On the board itself, you should be able to see that each pin is clearly labeled with a number or a brief description of what the pin is. Further, the pins are grouped into categories "PWM" for pins 2-13, "DIGITAL" on the right for pins 22-53, etc. In the Jarduino Program sketch, I defined what pin each peripheral gets hooked up to. This is a pin assignment. You can change some of the pin assignments if you like, but many of them you cannot change. In particular, the pins that are used by the TFT LCD Screen cannot be changed. This is how the pins are currently defined in the sketch:

PIN ASSIGNMENTS

Controller Peripheral / Component	Pin Assignment(s)
TFT LCD Shield	2, 3, 4, 5, 6 38, 39, 40, 41
ITDB02 Shield	22-37
LEDs: Sump	7
LEDs: Regular Blue	8
LEDs: White	9
LEDs: Royal Blue	10
LEDs: Red	11
LEDs: Ultraviolet (UV)	12
LEDs: Moonlight	13
Real Time Clock (DS1307 RTC)	20, 21
WaveMaker One	42
WaveMaker Two	43
Fans: For Hood Lights (PWM)	44
Fans: For Sump Lights (PWM)	45
Fans (ON/OFF): Hood Lights	47
Fans (ON/OFF): Sump Lights	48
Alarm for Temperature Thresholds	49
Automatic Feeder	50
Temperature Sensors (DS18B20)	51
Heater	52
Chiller	53

If you've been paying attention up to now, you should already know how to assemble the main components of the Jarduino. But to be sure, with the Arduino at the base "press" on the ITDB02 Arduino MEGA Shield, then at the top "press" on a 3.2" LCD Screen, like so:



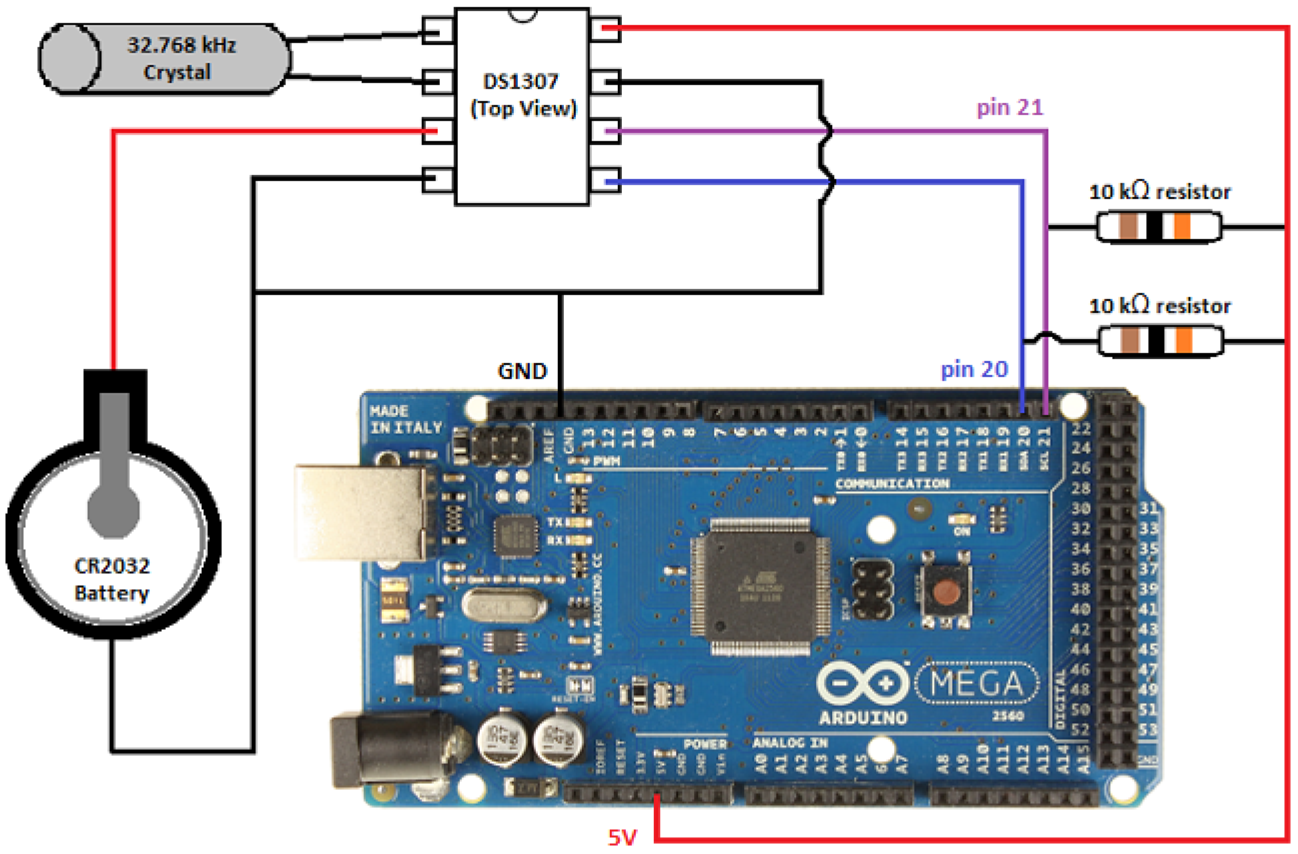
With it all sandwiched together, it should look like this:

How easy was that! At this point you may be eager to see the Jarduino up and running on the controller. The good news is you *can* upload the sketch right now. Refer to LOADING THE JARDUINO ONTO THE ARDUINO below to learn how. Before you do, however, there are a few things you may want to consider. First, if you didn't use an ITDB02 v1.1 Arduino MEGA LCD Expansion Shield with the built in RTC, you will need to connect an RTC circuit. Also, you may want to connect the DS18B20 Temperature Sensor(s). Leaving out an RTC and Temp Sensor(s) will not prevent you from loading up the sketch. It will simply leave you with some errors on the screen until you get them properly connected. So if you want to load up the Jarduino, go ahead and do it now. Later, once you get the Temp Sensors (and RTC if needed) hooked up, you can simply re-upload the Jarduino sketch.

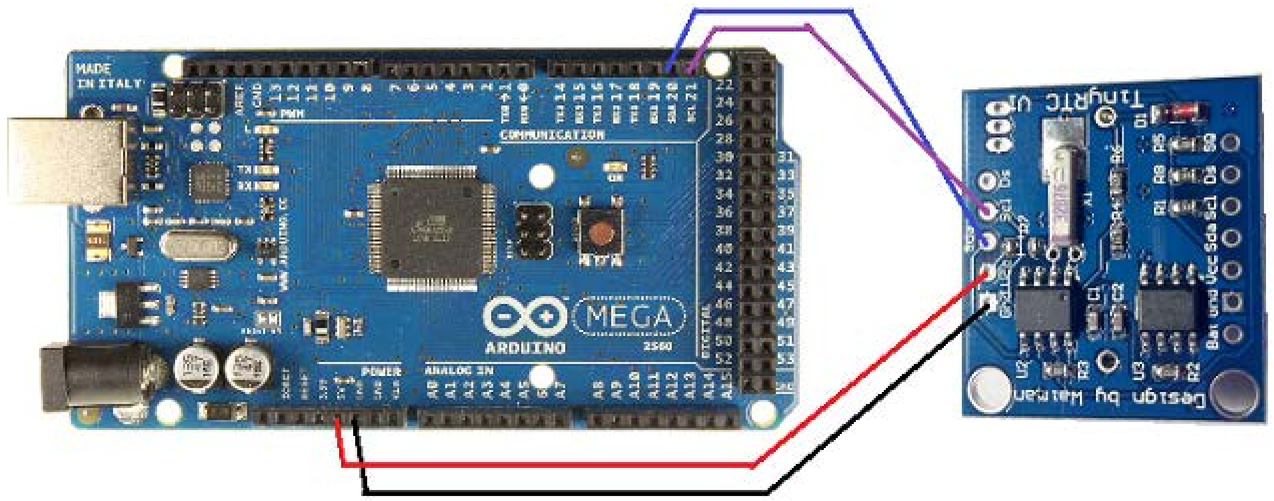
RTC HOOKUP

Go ahead and skip this section if you are using an ITDB02 v1.1 Arduino MEGA shield with built in RTC.

If you're still reading this, then for whatever reason, you don't have an ITDB02 v1.1 Arduino MEGA shield with built in RTC. Oh well, that just means you'll have to add your own RTC. Here's a rather non-abstract RTC circuit design:



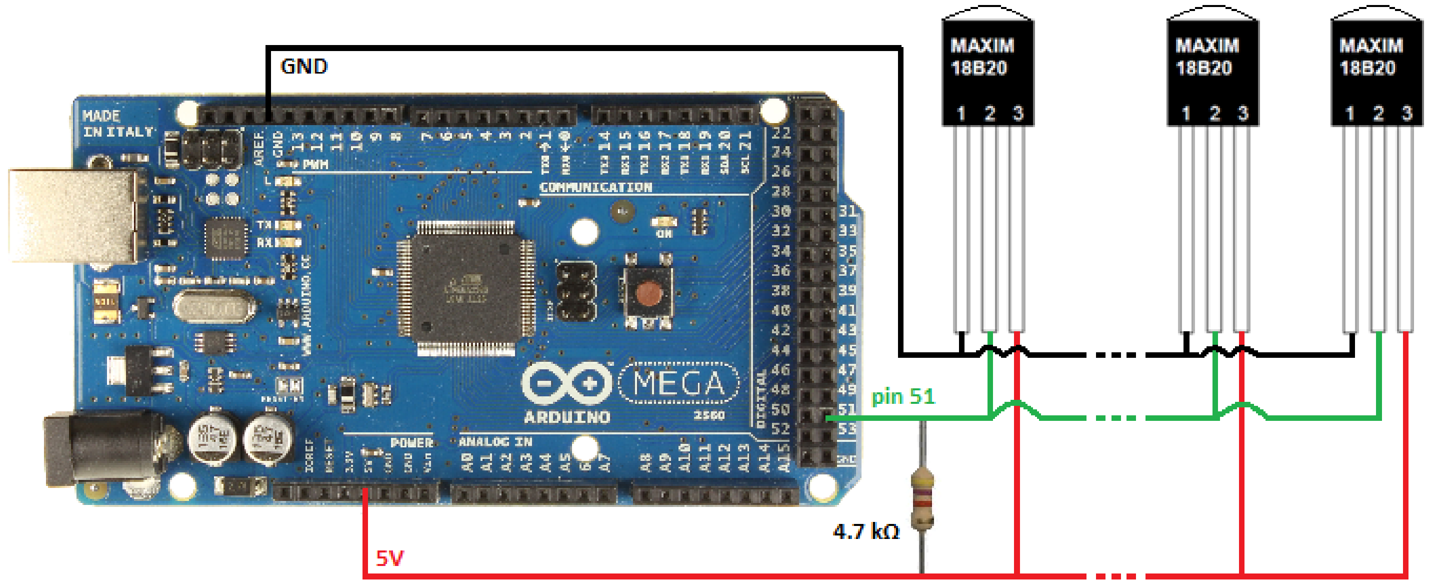
So building this circuit is not too overly complicated. Simply wire up and solder everything up as I have it characterized above. You can put it on a perfboard, prototyping board, screw shield, or directly solder it to the bottom of the Arduino (or to the solder joints on top of the LCD expansion shield. You have plenty of freedom here. As long as you have it hooked up right, it should work. Even easier to hook up would be a pre-assembled RTC module (just 4 wires):



NOTE: Depending on the RTC Module, the connection positions may vary slightly.

TEMPERATURE SENSOR(S) HOOKUP

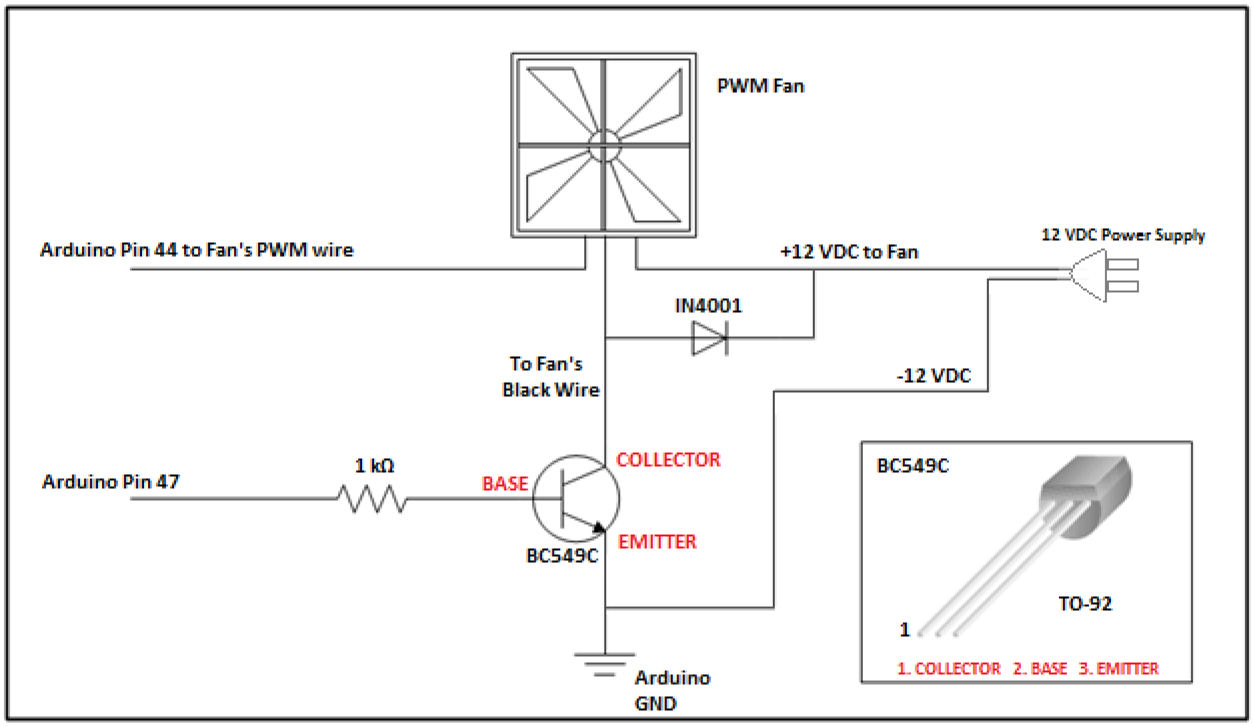
When you first download the Jarduino, you are getting the code in the exact same state I left it in when I uploaded it to my Arduino. Therefore, it is specific to my setup. Although it isn't necessary to make any major changes to the code, you will need to determine the addresses specific to your temperature sensors. Once you get those addresses, you will need to enter them into the Jarduino code. In the Software section below, I show you how to do both. But first, in order to get those addresses, you will need to follow this diagram to hook up the temperature sensors to the Arduino:



Notice the two sensors on the far right. I depicted the wires "broken" here to indicate you can hook up more sensors to the Arduino than just three. Notice, however, there is only one 4.7 kΩ resistor. No matter how many sensors you decide to add, you use only one of these resistors. I think it is most convenient to connect the resistor somewhere close to the Arduino itself, then from that point run the cables with temp sensors from that point to the particular location you want to monitor. I think phone cables work well here, as they have at least three wires inside. Make sure you somehow waterproof any sensor(s) you plan on submersing in the water. As I already mentioned, several vendors sell these encased in a water-proof stainless steel casing. They cost slightly more, but can save you some work.

FAN HOOKUP

The Jarduino provides dynamic cooling of LED heatsink(s) through the use of Pulse Width Modulated (PWM) fans. The hotter the heatsink gets (as measured by DS18B20 Temp Sensors), the faster they run. If they are cool enough, they shut off conserving power. They run very quiet, operating at a frequency output of 25 kHz, which is outside our hearing range. Here is how I connected them to the Arduino. I'm assuming by now you probably understand a more symbolic schematic.



LED HOOKUP

As you probably already know, there are many different ways in which you can hook up your LEDs, what kind of LEDs you can use, and what type of LED drivers you can use. To complicate things, you start wondering what colors you should use, how many of each, how powerful, so on and so forth. I'm not even going to try to write a guide on this, as there is just too many possible variations that rely too heavily on user preference. Since I feel there's a lot of good information readily available on helping you make a decision on what's best for your needs, I'll simply assume you have already made your decision when it comes to the LEDs you want to use, and what drivers you want to use to run them.

The following are the main things you need to know about the Jarduino and the LEDs it controls:

- The Jarduino uses PWM signals to control the brightness of the LEDs, so you need to make sure your LED drivers are capable of being controlled via PWM signals. It doesn't matter if the LED driver requires a separate power supply or is an all-in-one LED driver; Jarduino can control them so long as they can be controlled via PWM.
- If you want to control a commercially available LED fixture, the drivers need to be controlled via PWM. If they aren't, you can always replace the drivers with ones that are. ☺
- Currently, the Jarduino can control up to 5 separate channels of LEDs. You can devote a different color to each channel. You can also add as many LEDs as you like to each channel (as long as you can provide the necessary power to run them).
- The Lunar Phase is on its very own channel, so you will need dedicated LEDs along with its own LED driver. You shouldn't need too many LEDs for this feature however. (I use two CREE XP-E Royal Blue LED's on my 44 Gallon and it is more than bright enough).

The only wires you need to connect from the Jarduino to your drivers (and whatever other circuits your drivers may require) are a ground (**GND**) pin and the **PWM** pins for each respective color. You of course will need to power the LEDs, but for the sake of this manual, I will assume that you already know how or have figured out how from somewhere else. Without any modification to the Jarduino code, these are the Colors the Jarduino controls along with the respective Arduino Pin numbers:

- Pin 7: SUMP/Refugium LED Light Fixture (White)
- Pin 8: Regular Blue
- Pin 9: White
- Pin 10: Royal Blue
- Pin 11: Red
- Pin 12: UV
- Pin 13: Moon Lights

Note: You can change the colors and pin assignments in the Jarduino sketch to whatever you like. Those are simply from my setup.

3. JARDUINO SOFTWARE INSTRUCTIONS

SOFTWARE

The Arduino development environment (IDE) can be obtained from here: http://arduino.cc/en/Main/Software

All versions of the Jarduino Code can be obtained from here: http://code.google.com/p/jarduino-aquarium-controller/

Various Libraries from Henning Karlsen: http://www.henningkarlsen.com/electronics/library.php

Here are some links to a few File Archivers: WinZip, WinRAR, 7-Zip (you don't have to use these, but you will need a File Archiver)

JARDUINO VERSIONS

Jarduino v1.1 – April 8, 2012
This archive contains a modified version of the Arduino Environment 0022. It is modified to patch a bug that surfaces when trying to upload large sketches. Also included is the Jarduino Aquarium Controller v1.1, along with all the necessary libraries needed to successfully upload the sketch to the Arduino MEGA 2560. The patched Arduino IDE 0022 is the *only* IDE that is known to successfully upload the Jarduino v1.1.

Jarduino v1.2 (beta1) – May 19, 2012
This archive contains the Jarduino Aquarium Controller v1.2, beta release, along with all the necessary libraries needed to successfully upload the sketch to the Arduino MEGA 2560. For a full list of features, refer to page 3 of this document. Also included in this archive is a modified version of Arduino 1.0. It is modified to patch a bug that surfaces when trying to upload large sketches. The patched Arduino IDE 1.0 will successfully upload the Jarduino v1.2 beta, as well as all subsequent releases of the Jarduino. Arduino IDE 1.0.1 (available from Arduino's website) should successfully upload Jarduino v1.2 beta and all subsequent releases without the need to use my patched version of the IDE. There have been a few reports of not being able to use Arduino 1.0.1. If you experience the same issue, you should revert back to using the patched version of Arduino IDE 1.0 included in this release.

Jarduino v1.2 (beta1 with ITDB02_Graph16) – May 24, 2012
This archive contains only one change from the previous release of the Jarduino Aquarium Controller v1.2 (beta1). It replaces the newer UTFT Library with the older ITDB02_Graph16 library. The UTFT library has more compatibility, while the ITDB02_Graph16 library is faster. More will be covered on which one you should choose later in this document.

Jarduino v1.2 (beta2) – June 28, 2012
This archive contains a modified version of the Jarduino Aquarium Controller v1.2, beta release. This release of v1.2 Beta fixes the Auto-Dimming when LEDs reach a user-defined threshold feature. Also a few other tweaks & fixes have been made.

LOADING THE JARDUINO ONTO THE ARDUINO

This tutorial is aimed at Windows users, but other OS's will closely follow the same procedure. More complete instructions on how to hook up the Arduino to your computer is readily available on the Arduino website.

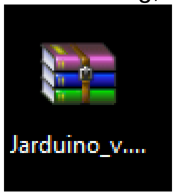
STEP 1: Install Archiver Software

Although you may likely have this on your computer already, the first thing you will need is "archive" software. Examples include but are certainly not limited to WinZip, WinRAR, and 7-Zip. Some of these programs cost money, but those ones usually come with a free trial. 7-Zip and many others are freeware. You can use whatever archiver best suites your needs. You will need archiving software to "unpack" the files from my download page, and also the files from Arduino.

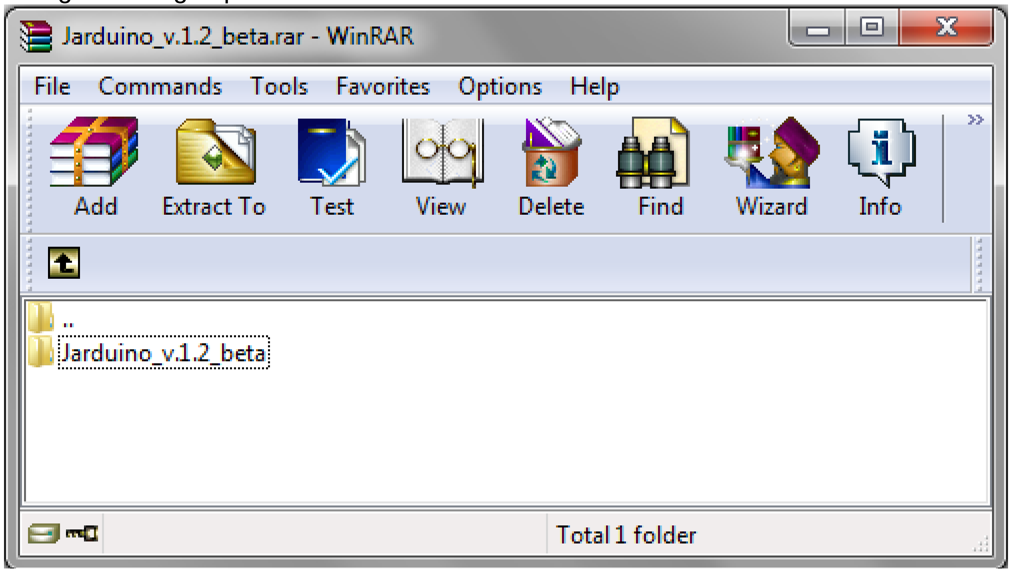
STEP 2: Download & Install the Arduino Development Software

Assuming you obtained a copy of the Arduino IDE you need, the next thing you need to do is "install" it on your computer. For Jarduino v1.1, use my patched version of Arduino 0022. For Jarduino v1.2 and later, you will install either my patched version of Arduino 1.0, or use the official version of Arduino 1.0.1 (available on Arduino's website).

For this example, I'm choosing to download Jarduino v1.2 (beta1) because it has my patched version of Arduino 1.0 along with all the libraries I need and of course the Jarduino sketch. I chose to "save" it to my desktop. Once the file finished downloading, I "double-clicked" on the archive folder "Jarduino_v.1.2_beta.rar" that was on my desktop.

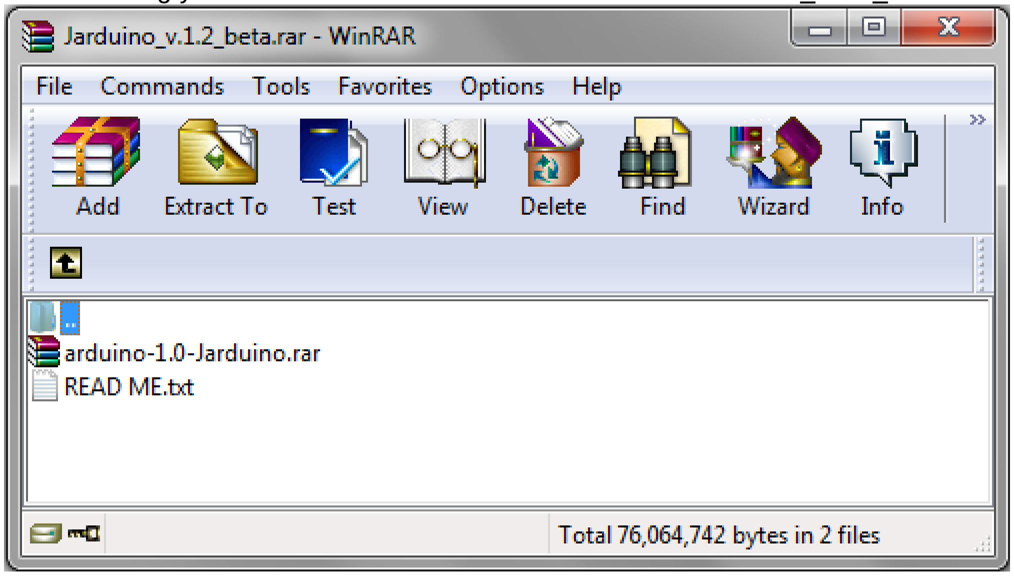


Doing this brings up this screen:


Jarduino_v.1.2_beta.rar - WinRAR
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info
..
Jarduino_v.1.2_beta
Total 1 folder

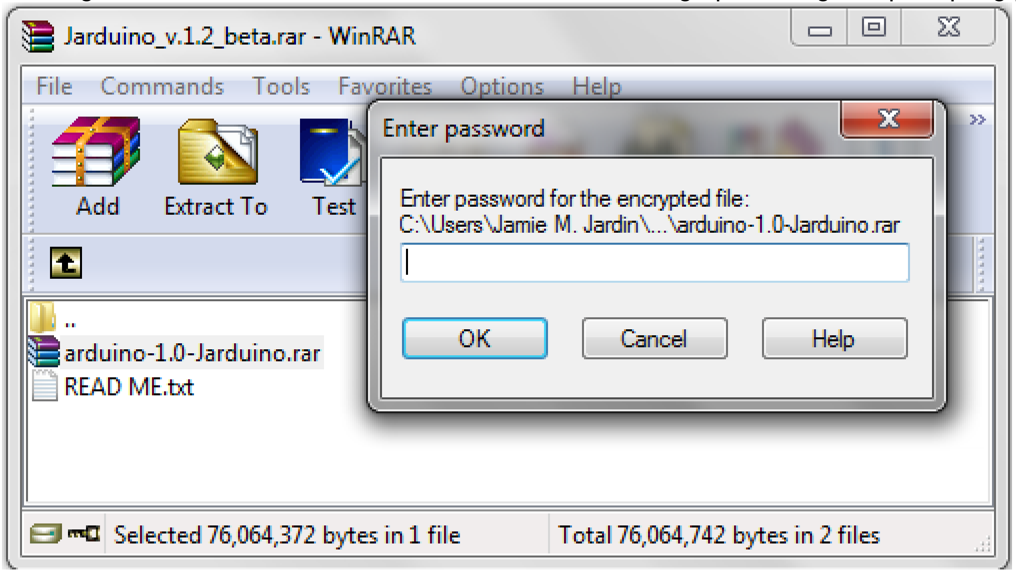
NOTE: The archiver pictured above is WinRAR, and thus may look slightly different depending on your archiver.

The next thing you want to do is double click on the folder "Jarduino_v.1.2_beta." When you do that you'll then see this:


Jarduino_v.1.2_beta.rar - WinRAR
File Commands Tools Favorites Options Help
Add Extract To Test View Delete Find Wizard Info
..
arduino-1.0-Jarduino.rar
READ ME.txt
Total 76,064,742 bytes in 2 files

NOTE: The "READ ME" file contains some information I included on this particular release.

Clicking on the archive “Arduino-1.0.Jarduino.rar” will bring up a dialog box prompting you to enter a password:

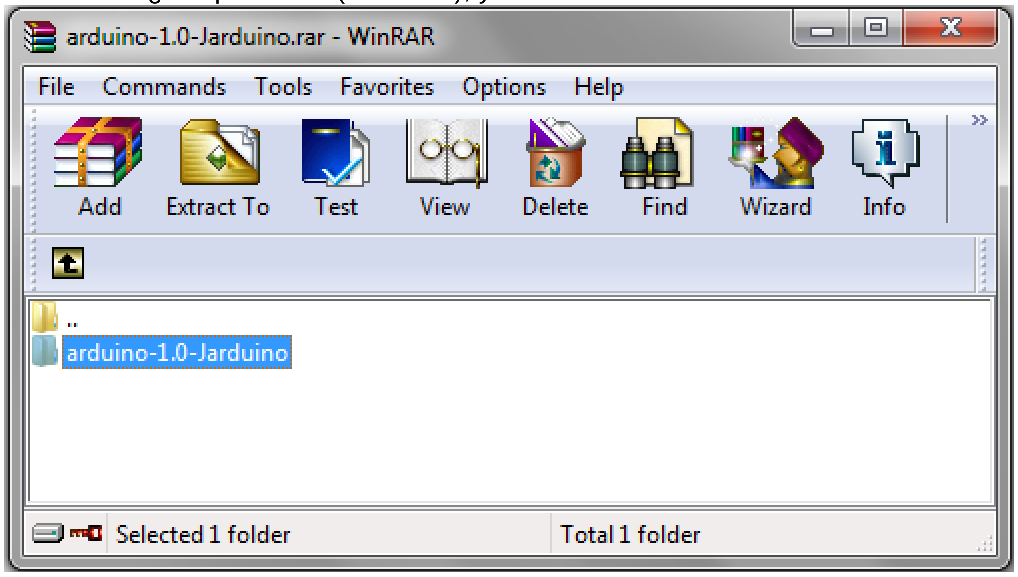


Enter password for the encrypted file:
C:\Users\Jamie M. Jardin\...\arduino-1.0-Jarduino.rar

Selected 76,064,372 bytes in 1 file
Total 76,064,742 bytes in 2 files

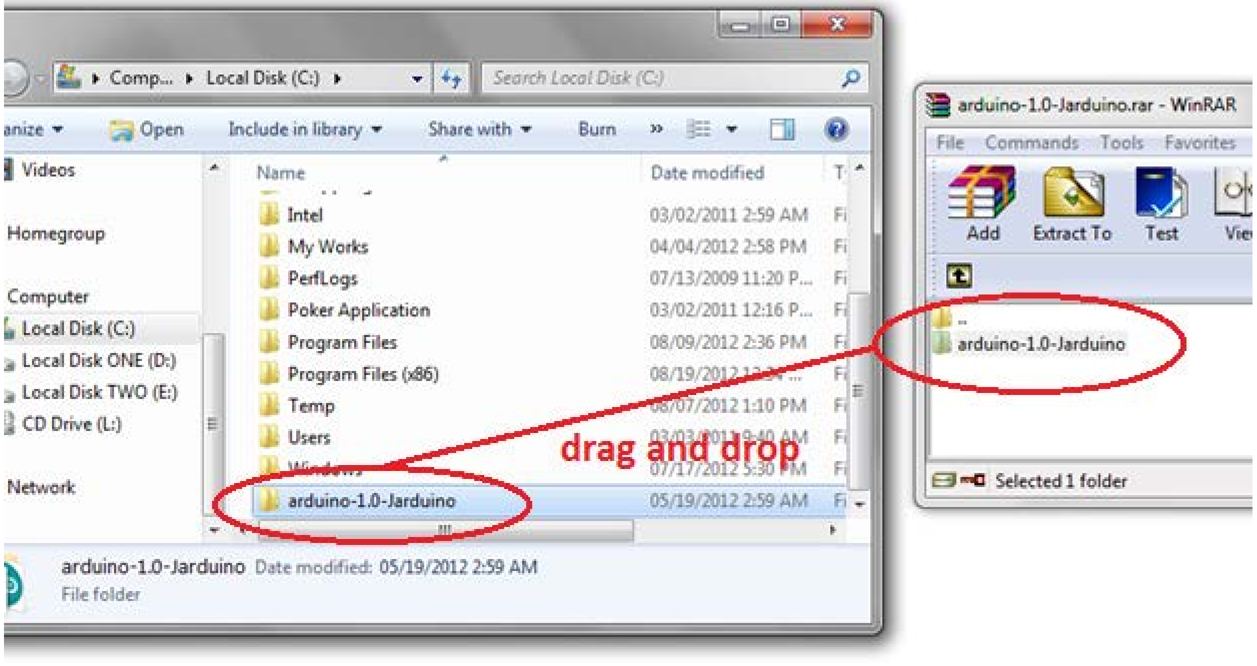
NOTE: If you are using Jarduino v1.1, no password is necessary. If you made a reasonably sized donation, I put you on my Beta Testers list, and emailed you a password to open the archive.

After entering the password (if needed), you will then see this:

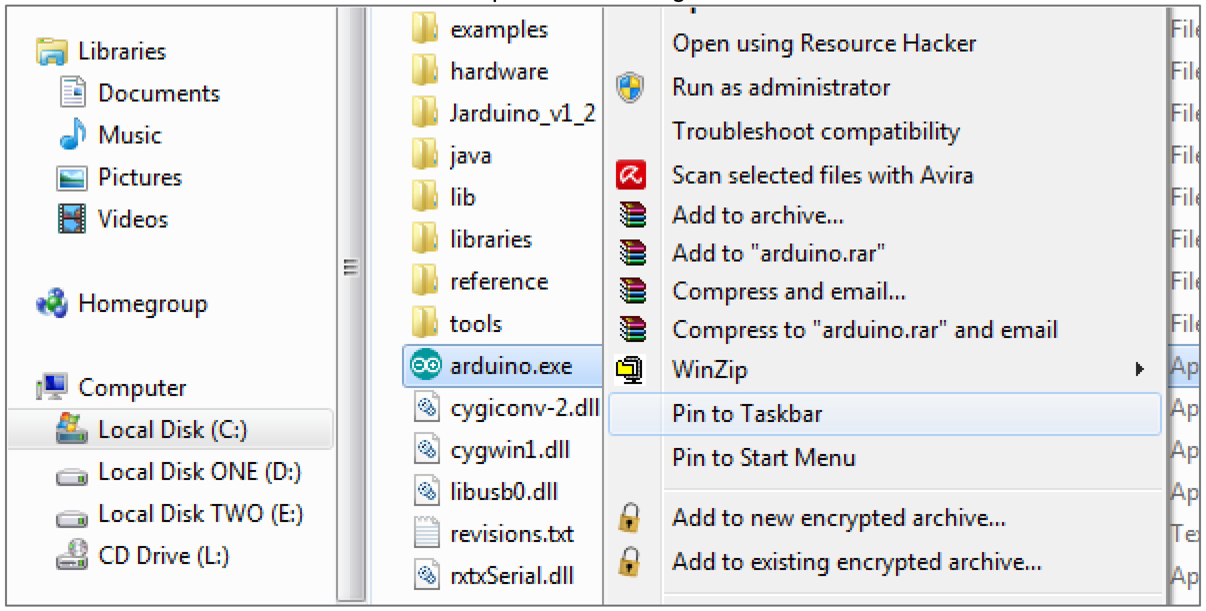

arduino-1.0-Jarduino.rar - WinRAR
File Commands Tools Favorites Options Help
Add
Extract To
Test
View
Delete
Find
Wizard
Info
..
arduino-1.0-Jarduino

Selected 1 folder
Total 1 folder

Next, all you need to do is “drag and drop” this folder to wherever it is you would like the Arduino “installed.” I chose to place it on my C:/ drive. When you do this, it will take a little time for the archiver to “extract” the files:


Comp... ▸ Local Disk (C:) ▸
Search Local Disk (C:)
Open
Include in library
Share with
Burn
Videos
Homegroup
Computer
Local Disk (C:)
Local Disk ONE (D:)
Local Disk TWO (E:)
CD Drive (L:)
Network
Name
Date modified
Intel 03/02/2011 2:59 AM
My Works 04/04/2012 2:58 PM
PerfLogs 07/13/2009 11:20 P...
Poker Application 03/02/2011 12:16 P...
Program Files 08/09/2012 2:36 PM
Program Files (x86) 08/19/2012
Temp 08/07/2012 1:10 PM
Users 03/03/2011
Windows 07/17/2012 5:30 PM
arduino-1.0-Jarduino 05/19/2012 2:59 AM
drag and drop
arduino-1.0-Jarduino Date modified: 05/19/2012 2:59 AM
File folder
arduino-1.0-Jarduino.rar - WinRAR
File Commands Tools Favorites
Add
Extract To
Test
arduino-1.0-Jarduino
Selected 1 folder

I found it to be very handy to make a “Quick Launch” button for the Arduino IDE. To do this, “double-click” on the newly created “Arduino-1.0-Jarduino” folder to open it. Then “right-click” on “arduino.exe” and click on “Pin to Taskbar.”



NOTE: This procedure is for Windows 7, so if you’re using something else, you may need to take different steps to make a shortcut.

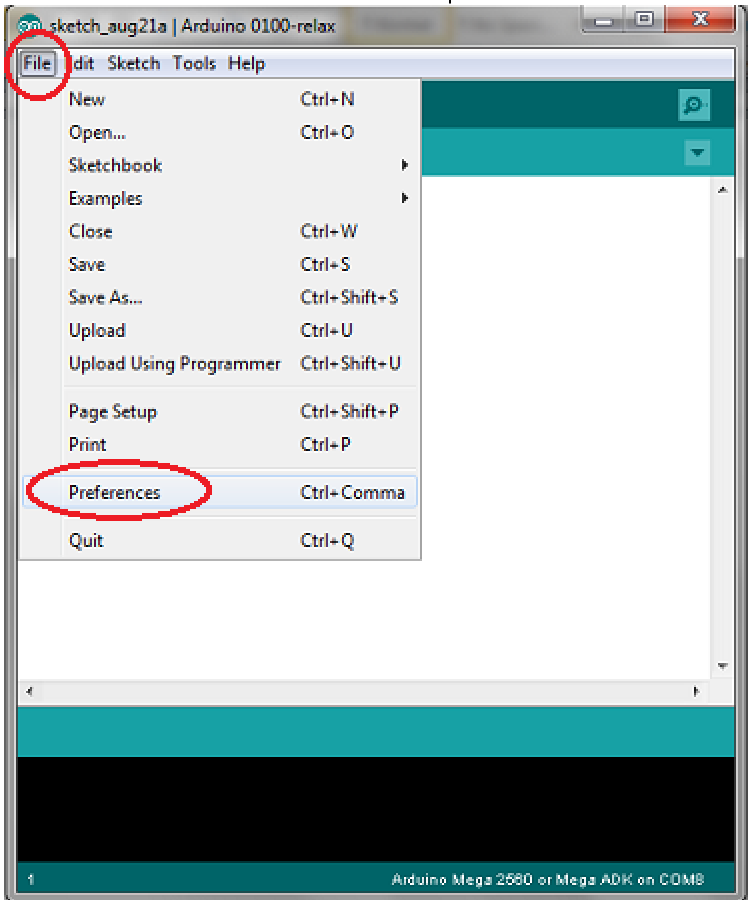
Now you can easily access the Arduino program from your taskbar.

STEP 3: Plug in your Arduino MEGA 2560

At this point, go ahead and plug one end of a USB cable to an open USB port in your computer, and the other end into the Arduino MEGA 2560. If you’re using Windows (and it’s hooked up to the internet), it should automatically download and install the necessary drivers. If you are having trouble with this step (ie – you got errors, your drivers didn’t install correctly, or you are using a different OS), please refer to Arduino’s website for installation troubleshooting tips).

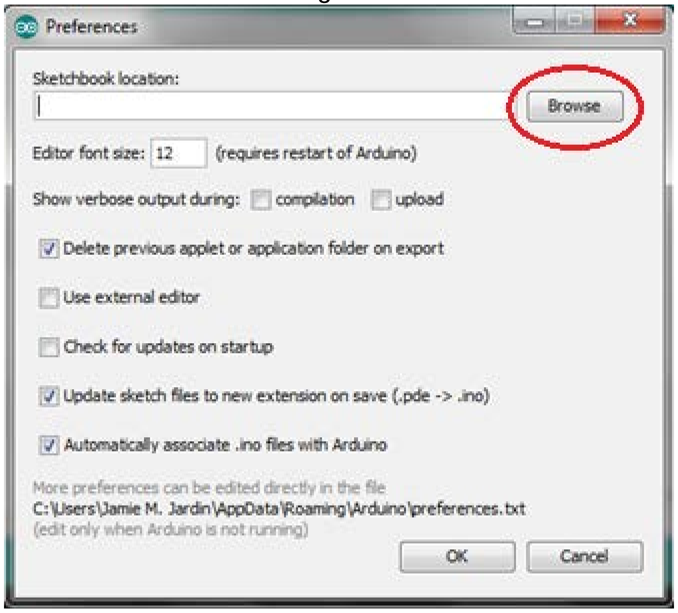
STEP 4: Setup the Arduino IDE

First, click on the Arduino icon in the taskbar (if you didn't pin it to your taskbar, then click on "arduino.exe" in the Arduino folder. The Arduino IDE should load up within a few seconds. When it does, click on "File," then click on "Preferences."



NOTE: When loading for the first time, it may ask about a "sketchbook location." Don't worry about it; just click "ok."

You will then see this dialog box:



Sketchbook location:

Editor font size: 12 (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

☑ Delete previous applet or application folder on export

☐ Use external editor

☐ Check for updates on startup

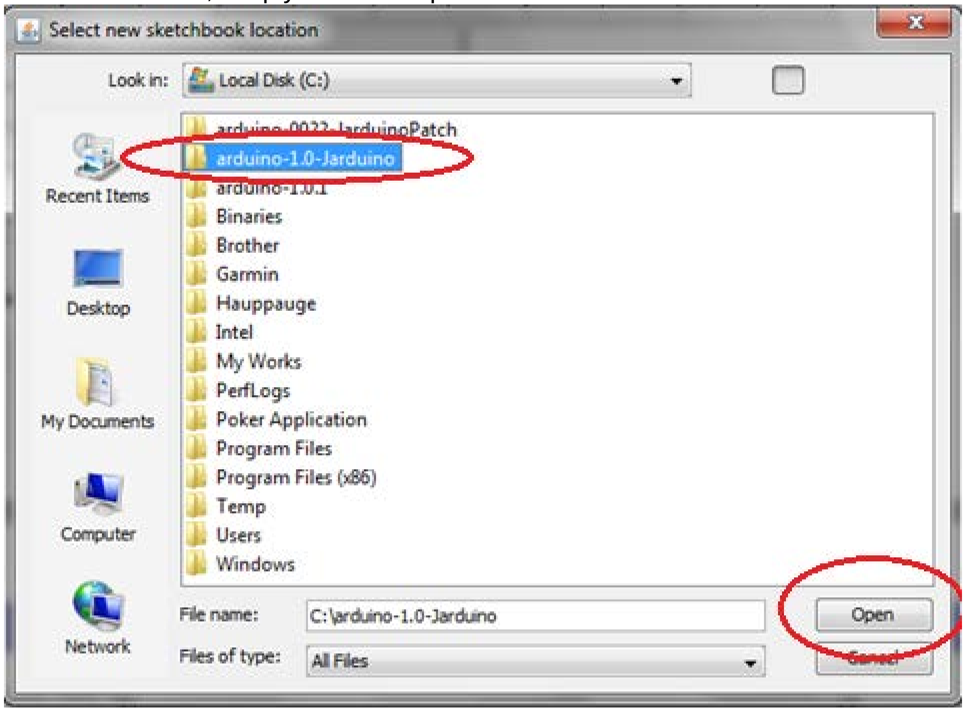
☑ Update sketch files to new extension on save (.pde -> .ino)

☑ Automatically associate .ino files with Arduino

More preferences can be edited directly in the file
C:\Users\Jamie M. Jardin\AppData\Roaming\Arduino\preferences.txt
(edit only when Arduino is not running)

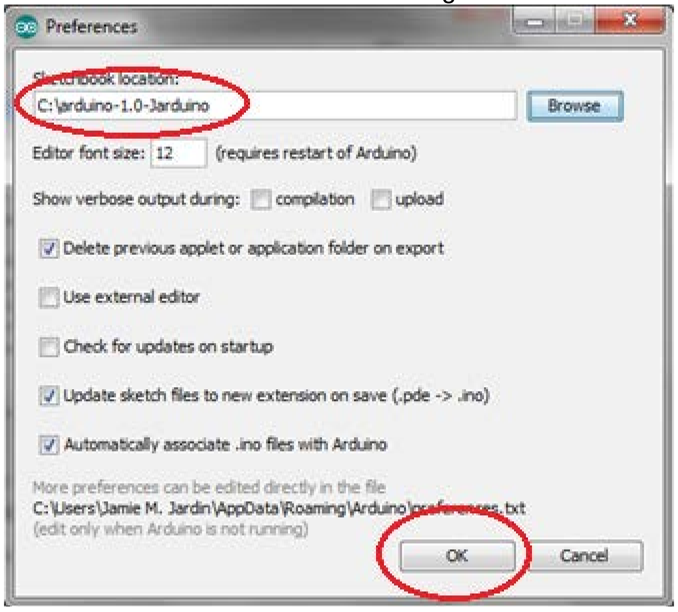
NOTE: And don't worry if the "sketchbook location" has something filled in there alreadyr. You're going to promptly change it anyway…

Next thing you need to do is set the “Sketchbook location.” To do that, click on “Browse” as I’ve indicated above. The explorer box should open up. What you want to do is navigate to where it is you put your Arduino folder. Recall that I put mine on the C drive, so I’ll go ahead and navigate to there and single-click on the Arduino folder so it’s highlighted like it is below. Once it is, simply click on “Open.”



File name: C:\arduino-1.0-Jarduino
Files of type: All Files

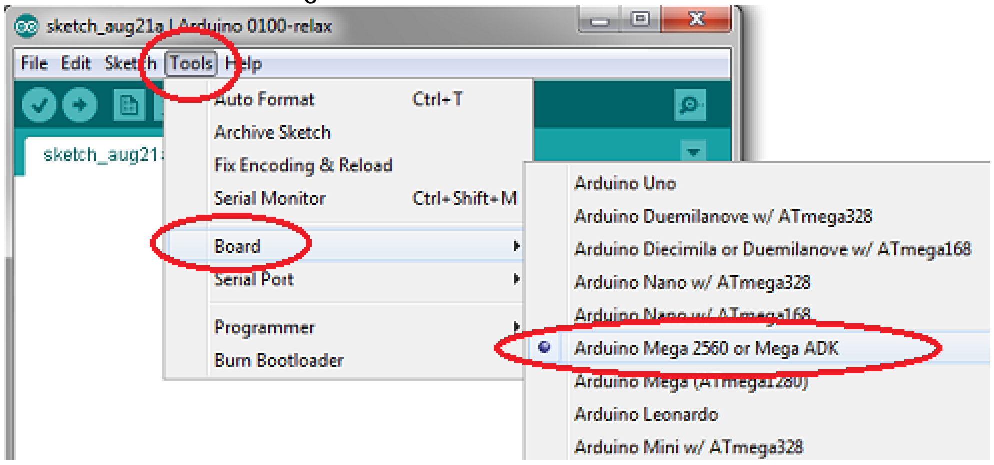
Clicking on “Open,” you will be brought back to the “Preferences” screen of the Arduino, but now you’ll see the “Sketchbook location” has been changed to where I want it to be:



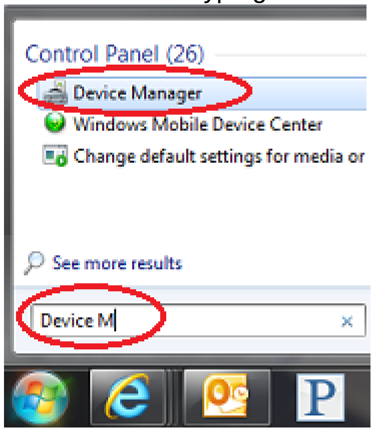
Sketchbook location: C:\arduino-1.0-Jarduino
Editor font size: 12 (requires restart of Arduino)
Show verbose output during: ☐ compilation ☐ upload
☑ Delete previous applet or application folder on export
☐ Use external editor
☐ Check for updates on startup
☑ Update sketch files to new extension on save (.pde -> .ino)
☑ Automatically associate .ino files with Arduino
More preferences can be edited directly in the file
C:\Users\Jamie M. Jardin\AppData\Roaming\Arduino\preferences.txt
(edit only when Arduino is not running)

Simply click “OK” and this dialog box will close.

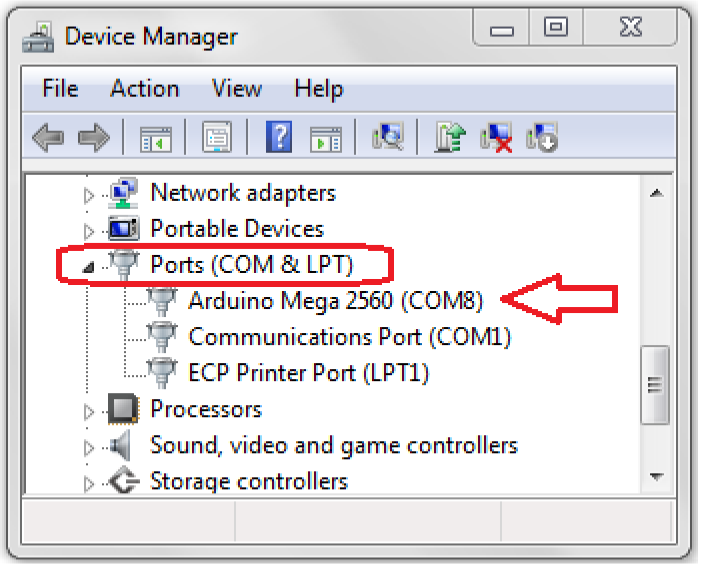
Next click on “Tools” and go to “Board” and select “Arduino MEGA 2560 or MEGA ADK.”


sketch_aug21a | Arduino 0100-relax
File Edit Sketch Tools Help
sketch_aug21a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
Programmer
Burn Bootloader
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Mini w/ ATmega328

Now you will want to find out which COM port your Arduino is plugged into. If you didn’t already, go ahead and plug your Arduino into your computer. Then open the “Device Manger.” For windows 7, click on the blue start button orb, and in the search bar start typing “Device Manager.” When it pops up, go ahead and click on it.

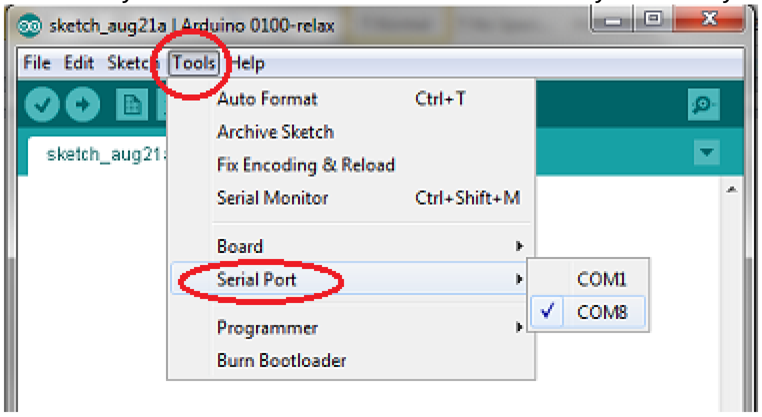

Control Panel (26)
Device Manager
Windows Mobile Device Center
Change default settings for media or
See more results
Device M

The Device Manager should open up. Scroll down to where it says “Ports (COM & LPT) and expand it. You should see your Arduino MEGA 2560 listed. Beside it you will see in parentheses the COM number. Mine says (COM8).



Either take a mental note or write down what COM port number your Arduino is utilizing.

Then back to the Arduino IDE, click on "Tools" again, but this time navigate on down to where it says "Serial Port." A box will slide out where you can then select the COM port the Arduino is plugged into. For me, mine is on "COM8." In all likelihood yours will be on a different COM. Once you select your COM, you should see a checkmark next to it.



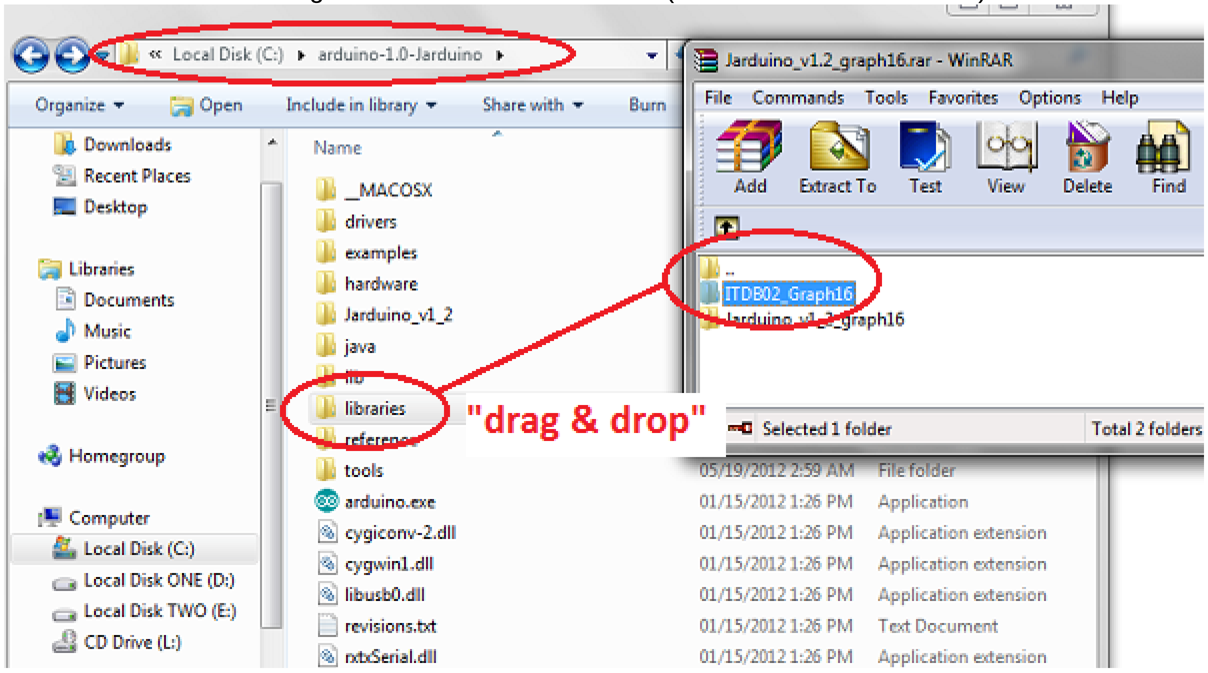
*NOTE: Finally, and this is very important: **CLOSE DOWN THE ARDUINO IDE**. Make sure no Arduino windows are open.*

STEP 5: Put all the Files in the Right Places

Arduino can be very finicky when it comes to file placement, and is probably the main reason why most people initially get errors when using Arduino. As long as files are where the IDE expects them to be, there shouldn't be too many issues.

Now, if you have been following this tutorial verbatim, then you should have no problems with file locations. Everything from that particular download (Jarduino v1.2 (beta1)) is an exact copy of my Arduino 1.0 (patched version) folder exactly as it was on my computer. The same is true for Jarduino v1.1 and the Arduino 0022 IDE. But for subsequent uploads of the Jarduino releases, I did NOT include the IDE, and in some of them I attached an additional library here and there. Therefore, those files will all need to be placed in the correct spots in ordered for them to upload properly.

For the various LIBRARIES, you will need to place them in the "arduino-1.0-Jarduino" "libraries" folder (or whatever version of the IDE it is you're using). Here's an example of placing the "ITDB02_Graph16" library in the proper place. Start by Downloading the Jarduino v1.2 (beta1 with ITDB02_Graph16). When you get to the part when you open the archive, simply "drag & drop" the "ITDB02_Graph16" folder into the "libraries" sub-folder of the "arduino-1.0-Jarduino" main folder. Make sure it goes INTO the libraries folder (and not some other folder):



tools	05/19/2012 2:59 AM	File folder
arduino.exe	01/15/2012 1:26 PM	Application
cygiconv-2.dll	01/15/2012 1:26 PM	Application extension
cygwin1.dll	01/15/2012 1:26 PM	Application extension
libusb0.dll	01/15/2012 1:26 PM	Application extension
revisions.txt	01/15/2012 1:26 PM	Text Document
rxtxSerial.dll	01/15/2012 1:26 PM	Application extension

NOTE: This is the same procedure you will need to follow for all subsequent libraries I (or you) may add in the future.

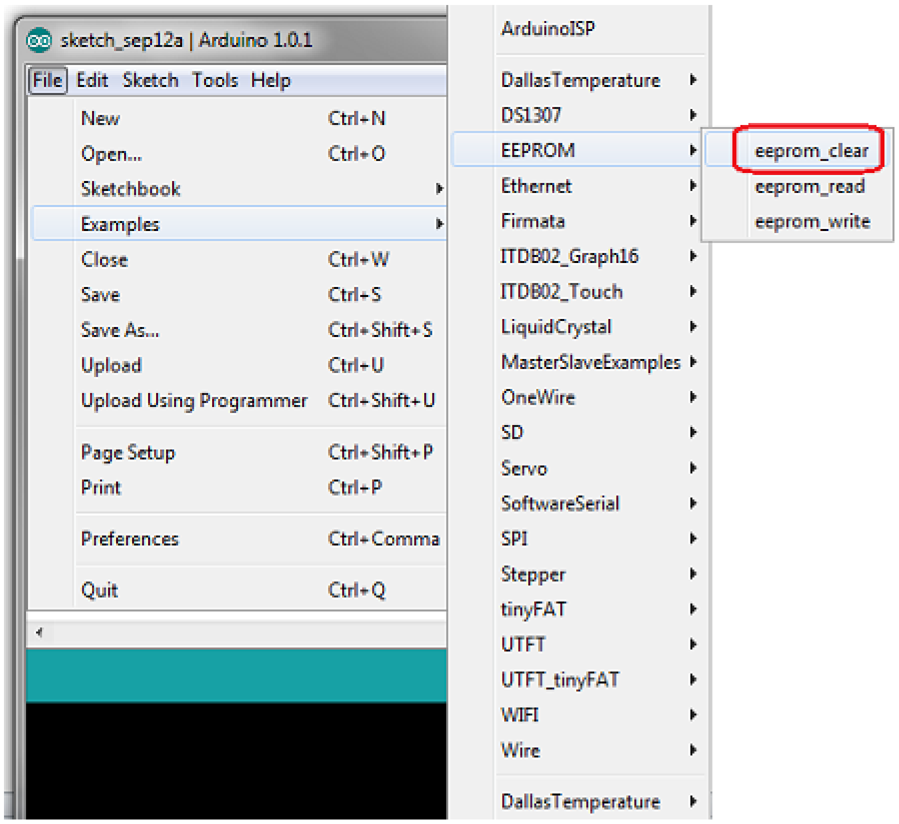
For PROGRAM SKETCHES (ie – various releases of the Jarduino), you can also place these files in the "libraries" sub-folder. I however prefer to place them directly in the Main folder of the Arduino. You can already see in the above picture that this is the location of "Jarduino_v1_2." This is where I would extract "Jarduino_v1_2_graph16" if I wanted to upload that particular sketch to my board.

ANOTHER VERY IMPORTANT NOTE: Whenever you make a change to the Main Folder of the Arduino, such as placing a new sketch in there, or putting in a new Library Folder, make sure the Arduino IDE is CLOSED! Even though you can make changes while the IDE is open, the changes you make will not be recognized until the library has been rebuilt. In order for that to happen, the Arduino IDE needs to "build" it. It does this when it loads up (or in the case of being open, upon closing and reopening the IDE, it will "re-build" the library).

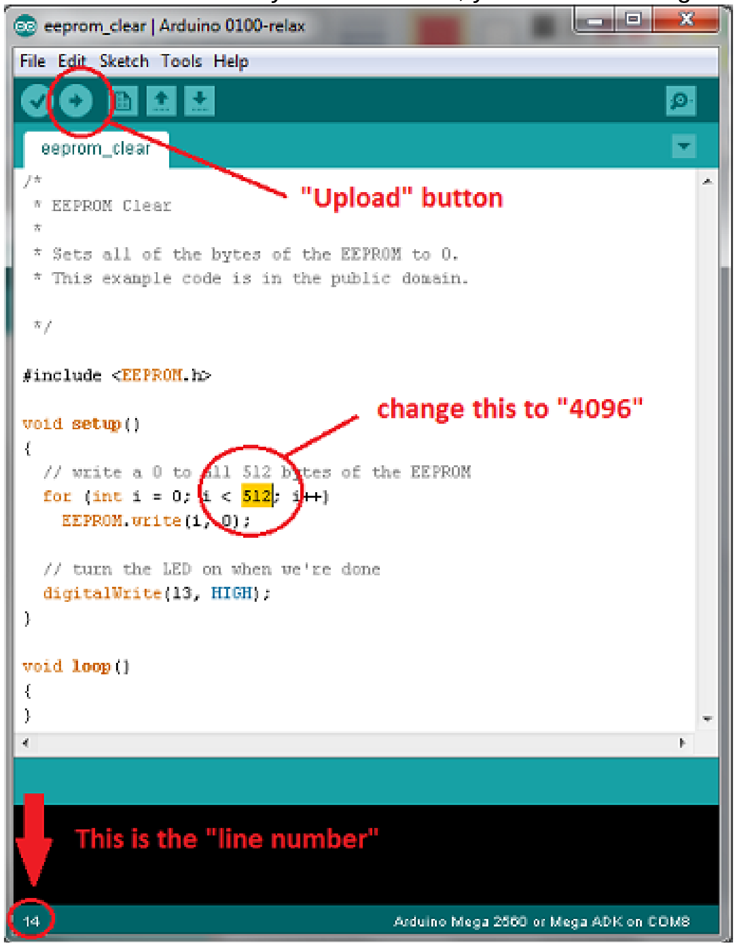
STEP 6: Clear the EEPROM

Now what you need to do is make sure the EEPROM (aka Arduino's Memory) is cleared. **You will need to clear the memory each time you decide to change versions of the Jarduino.** If you are simply re-uploading the same version, you don't need to perform this step (although it is perfectly fine if you do).

First, open the Arduino IDE. Click on "File," move down to "Examples," then on over to "EEPROM," then finally click on 'eeprom_clear.'



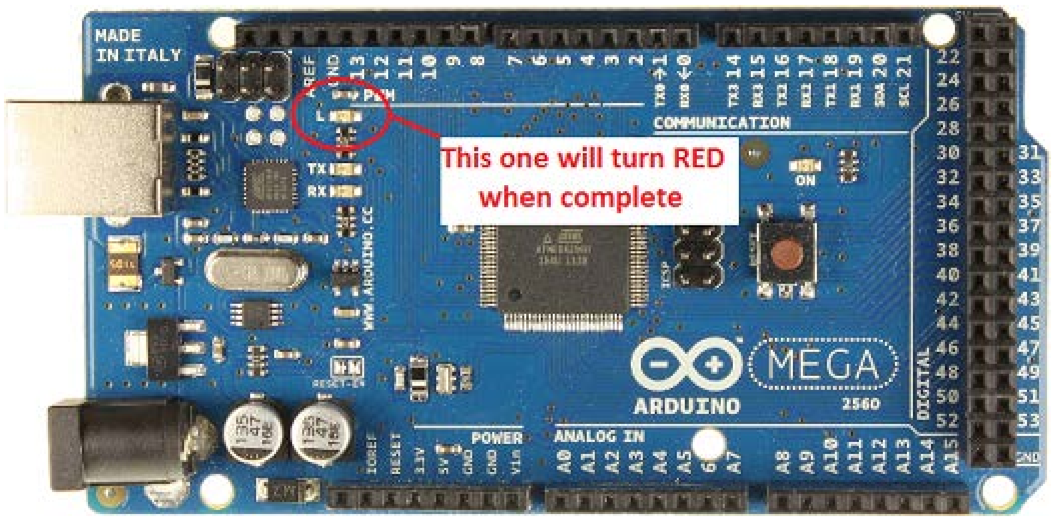
Clicking on “eeprom_clear” will open up that “example sketch.” By default, this sketch has a value of “512” entered in. That is meant for the Arduino UNO / Duemilanove which only has 512 bytes of memory. The Arduino MEGA 2560 on the other hand has 4096 bytes. Therefore, you need to change that “512” to “4096.”


eeprom_clear | Arduino 0100-relax
File Edit Sketch Tools Help
eeprom_clear
"Upload" button
change this to "4096"
This is the "line number"
14
Arduino Mega 2560 or Mega ADK on COM8

```
/*
 * EEPROM Clear
 *
 * Sets all of the bytes of the EEPROM to 0.
 * This example code is in the public domain.

 */

#include <EEPROM.h>

void setup()
{
  // write a 0 to all 512 bytes of the EEPROM
  for (int i = 0; i < 512; i++)
    EEPROM.write(i, 0);

  // turn the LED on when we're done
  digitalWrite(13, HIGH);
}

void loop()
{
}
```

HINT: clicking anywhere within a sketch will reveal the line number at the bottom left of the Arduino IDE.

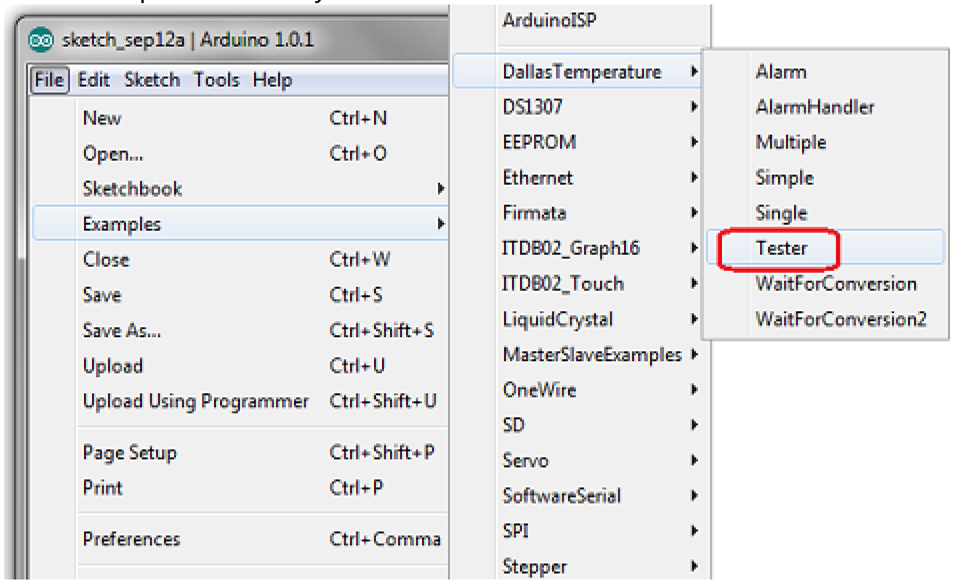
After changing the “512” to “4096,” proceed to click on the “Upload” button, which is indicated above. It should only take a few seconds for the “clear_eeprom” sketch to load to your Arduino. While it uploads, the “RX,” “TX,” and “L” LEDs will flash rapidly. After the upload completes, the LEDs will stop flashing and the program sketch will run until it has “cleared” all the memory. You will know that it has finished the job when the Red “L” LED lights up on the board and stays on. The whole process should only take around 15 seconds or so:



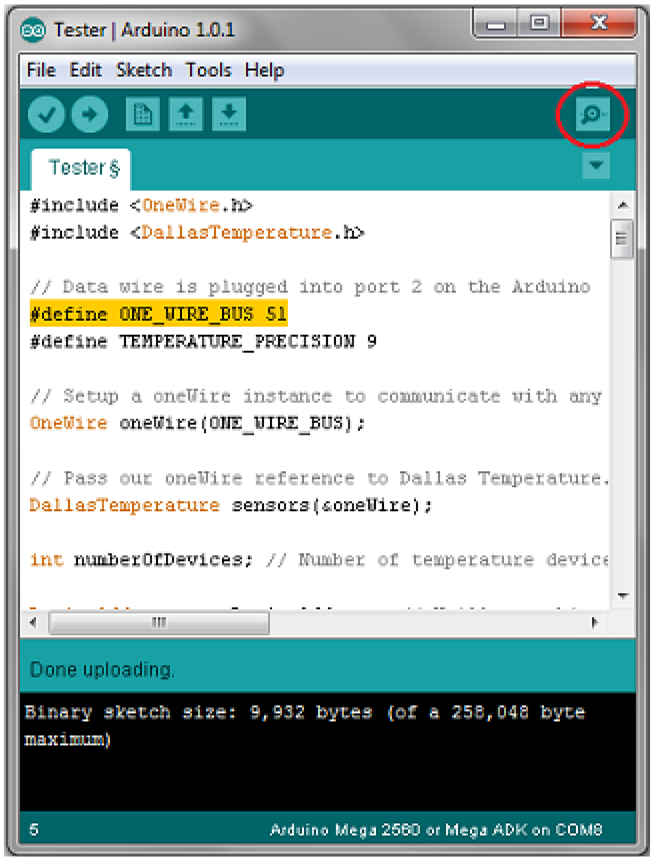
STEP 7: Obtain Temperature Sensor Addresses

The Jarduino uses three temperature sensors. In particular, it uses Dallas Temperature DS18B20 digital thermometers. Each of these sensors has their very own unique address. Because of this, you can put as many sensors as you want on one wire. The Jarduino simply calls the address of whatever sensor it wants temperature info from, and the sensor will give the Jarduino the information. Refer to this section to see how to connect these sensors to the Arduino.

OK, with your sensors properly hooked up to your Arduino, open up the example sketch “Tester” located in the “DallasTemperature” library:



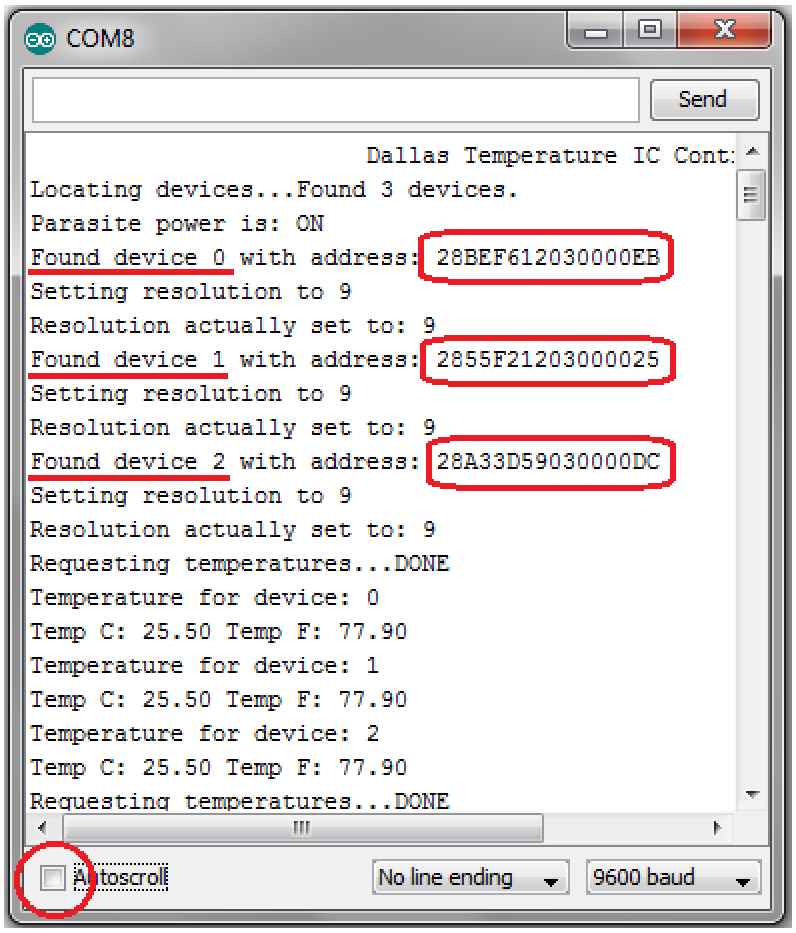
When the sketch opens, change line 5 “#define ONE_WIRE_BUS **2**” to “#define ONE_WIRE_BUS **51**” and go ahead and upload it to your Arduino (the same way you uploaded the “eeprom_clear” sketch. After the upload completes, click on the “Serial Monitor” Magnifying Glass icon as indicated below:


Tester | Arduino 1.0.1
File Edit Sketch Tools Help
Tester §
Done uploading.
5
Arduino Mega 2560 or Mega ADK on COM8

```
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is plugged into port 2 on the Arduino
#define ONE_WIRE_BUS 51
#define TEMPERATURE_PRECISION 9

// Setup a oneWire instance to communicate with any
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

int numberOfDevices; // Number of temperature device
```

```
Binary sketch size: 9,932 bytes (of a 258,048 byte
maximum)
```

Clicking on the “Serial Monitor” will bring up this:



```
                        Dallas Temperature IC Cont:
Locating devices...Found 3 devices.
Parasite power is: ON
Found device 0 with address: 28BEF612030000EB
Setting resolution to 9
Resolution actually set to: 9
Found device 1 with address: 2855F21203000025
Setting resolution to 9
Resolution actually set to: 9
Found device 2 with address: 28A33D59030000DC
Setting resolution to 9
Resolution actually set to: 9
Requesting temperatures...DONE
Temperature for device: 0
Temp C: 25.50 Temp F: 77.90
Temperature for device: 1
Temp C: 25.50 Temp F: 77.90
Temperature for device: 2
Temp C: 25.50 Temp F: 77.90
Requesting temperatures...DONE
```

The bottom left has “Autoscroll” checked by default. For the time being, go ahead and remove the tick. Make sure you scroll to the top of the box. You can see it found all three of my Temp Sensors, and gave me the address for each. You can “copy” and “paste” these addresses into a text file and save it for later because you will need them for the Jarduino sketch.

So, these are the three addresses I found for mine:

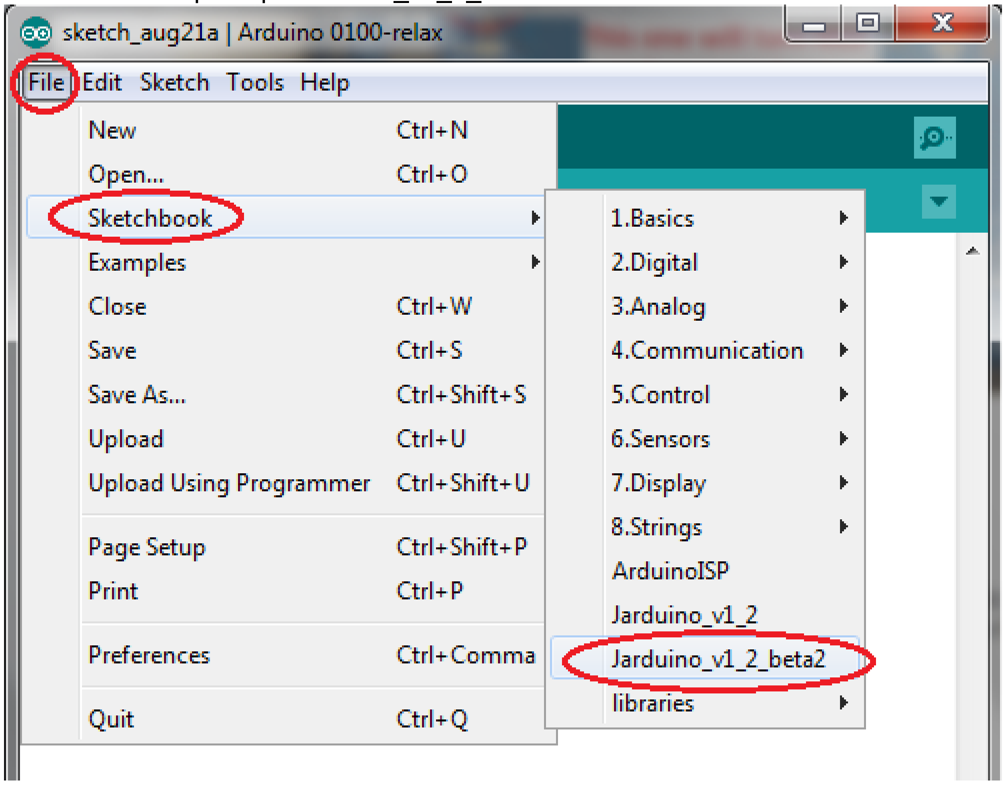
- 28BEF612030000EB
- 2855F21203000025
- 28A33D59030000DC

HINT: Since you have multiple sensors hooked up simultaneously, you may think that it might be a little challenging determining what address belongs to which “device.” You could always do one at a time. Alternatively, you can simply re-tick the “Autoscroll” feature. Then, squeeze one of the temperature sensor in between your fingers. Wait a few seconds, then while still holding the sensor, un-tick the “Autoscroll.” One of the “devices” should be significantly different from the rest. Note what the “device” number it is, scroll back to the top, and then you know the address for that sensor. Label the sensor with a piece of tape, and then set it aside. Repeat this procedure for the other sensors.

STEP 8: Upload the Jarduino to the Arduino

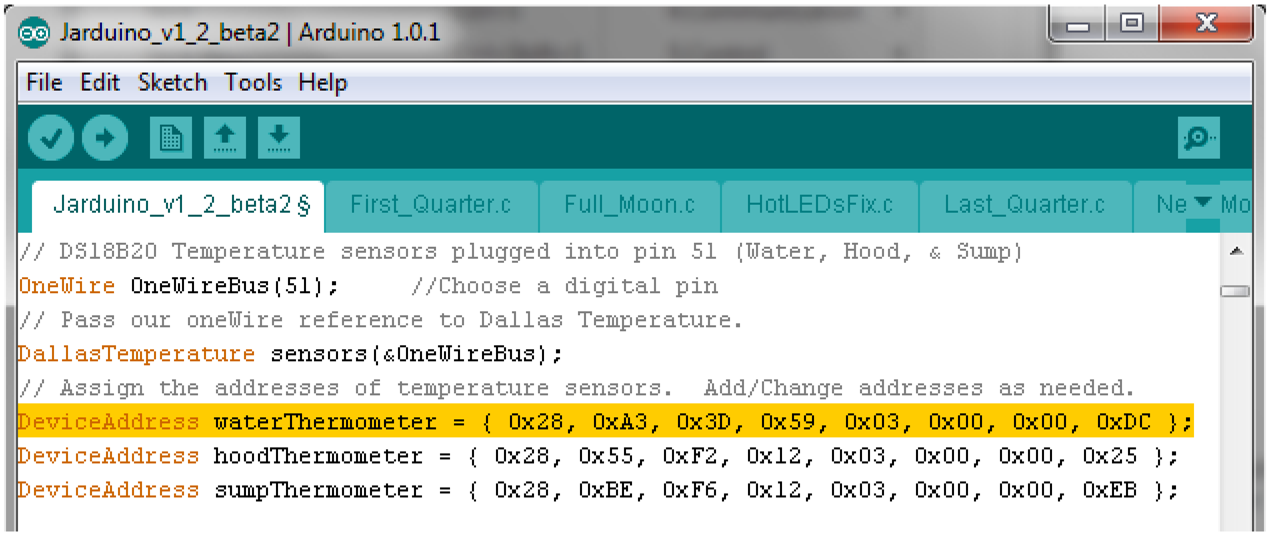
So now that the memory has been cleared, and you have the addresses of your temperature sensor(s), your Arduino is now ready to accept the Jarduino sketch. For this example, I'm going to upload Jarduino v1.2 (beta2). At this point, you should now know where to place all the files.

Go ahead and open up "Jarduino_v1_2_beta2."



NOTE: This particular release utilizes the "ITDB02_Graph16" library. You can "easily" make it work with the "UTFT Library" if needed. Simply change lines 131 & 147 of this sketch to look like the same line numbers in "Jarduino_v1_2."

Once you have the Jarduino sketch open, you will need to make a few changes. Since you just figured out the addresses for your temp sensors, you minus well go ahead and start by plugging those values into the sketch. Scroll on down to lines 216-218:

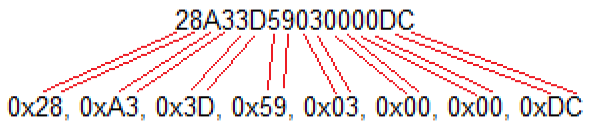


```
// DS18B20 Temperature sensors plugged into pin 51 (Water, Hood, & Sump)
OneWire OneWireBus(51);      //Choose a digital pin
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&OneWireBus);
// Assign the addresses of temperature sensors.  Add/Change addresses as needed.
DeviceAddress waterThermometer = { 0x28, 0xA3, 0x3D, 0x59, 0x03, 0x00, 0x00, 0xDC };
DeviceAddress hoodThermometer = { 0x28, 0x55, 0xF2, 0x12, 0x03, 0x00, 0x00, 0x25 };
DeviceAddress sumpThermometer = { 0x28, 0xBE, 0xF6, 0x12, 0x03, 0x00, 0x00, 0xEB };
```

At first glance, these don't look like the addresses I just got, but let's have a closer look. Recall that the address I got for my sensors were:

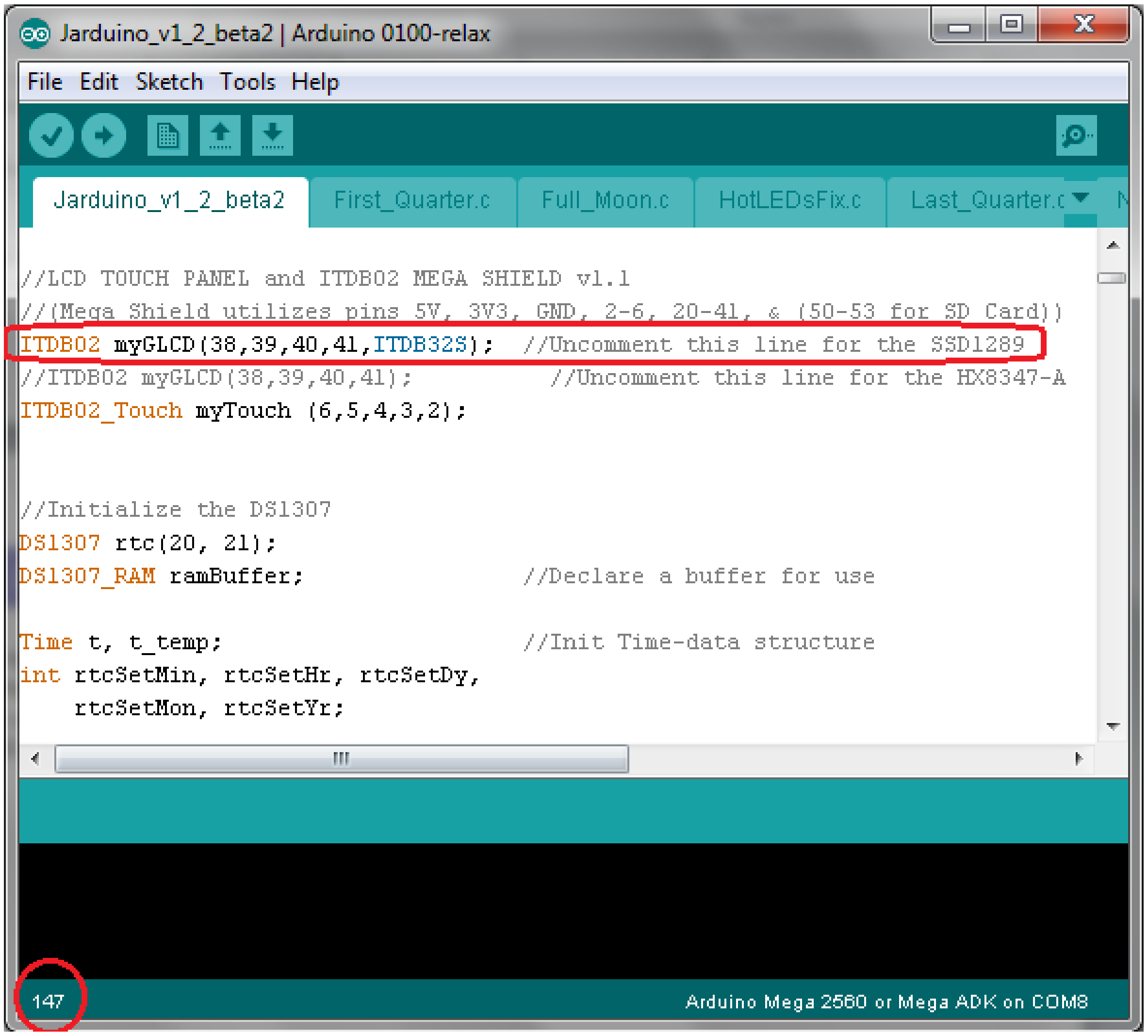
- 28BEF612030000EB
- 2855F21203000025
- 28A33D59030000DC

Focusing on the last address (which I shall designate as my water thermometer), I will change it into a format that the IDE can understand. I do this by grouping them into pairs of two and add a "0x" in front of each pair, separated by a comma:



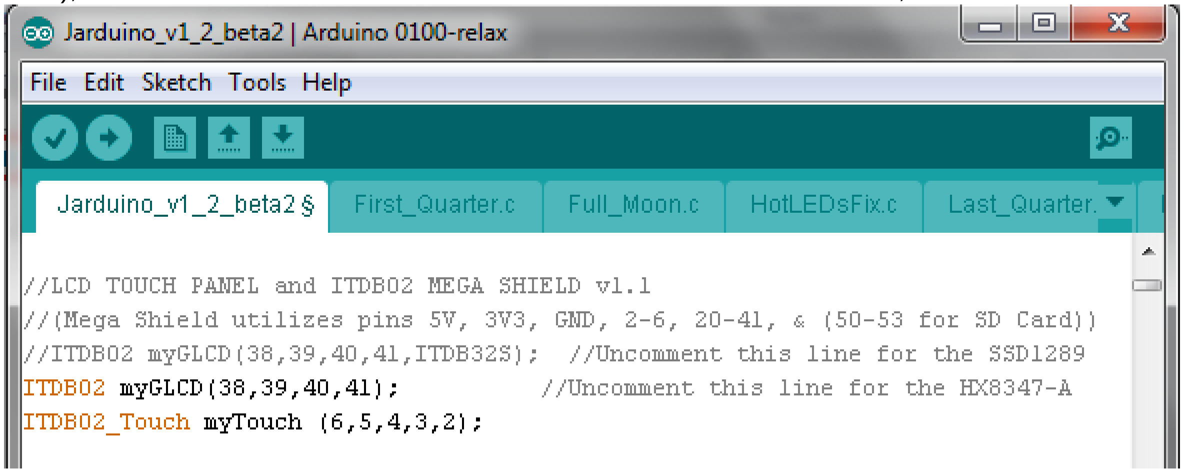
Now all you have to do is put the addresses you personally obtained from the DallasTemperature "Tester" sketch into the correct format, and copy them on over to the Jarduino.

So now that you have the addresses for the Temperature Sensors entered in, you will need to make sure you have the correct LCD module selected. For this example, I will show you how the code should read for a 3.2" TFT LCD Display with a SSD1289 controller module (line 147).


Jarduino_v1_2_beta2 | Arduino 0100-relax
File Edit Sketch Tools Help
Jarduino_v1_2_beta2
First_Quarter.c
Full_Moon.c
HotLEDsFix.c
Last_Quarter.c
147
Arduino Mega 2560 or Mega ADK on COM8

```
//LCD TOUCH PANEL and ITDB02 MEGA SHIELD v1.1
//(Mega Shield utilizes pins 5V, 3V3, GND, 2-6, 20-41, & (50-53 for SD Card))
ITDB02 myGLCD(38,39,40,41,ITDB32S);  //Uncomment this line for the SSD1289
//ITDB02 myGLCD(38,39,40,41);          //Uncomment this line for the HX8347-A
ITDB02_Touch myTouch (6,5,4,3,2);


//Initialize the DS1307
DS1307 rtc(20, 21);
DS1307_RAM ramBuffer;                //Declare a buffer for use

Time t, t_temp;                      //Init Time-data structure
int rtcSetMin, rtcSetHr, rtcSetDy,
    rtcSetMon, rtcSetYr;
```

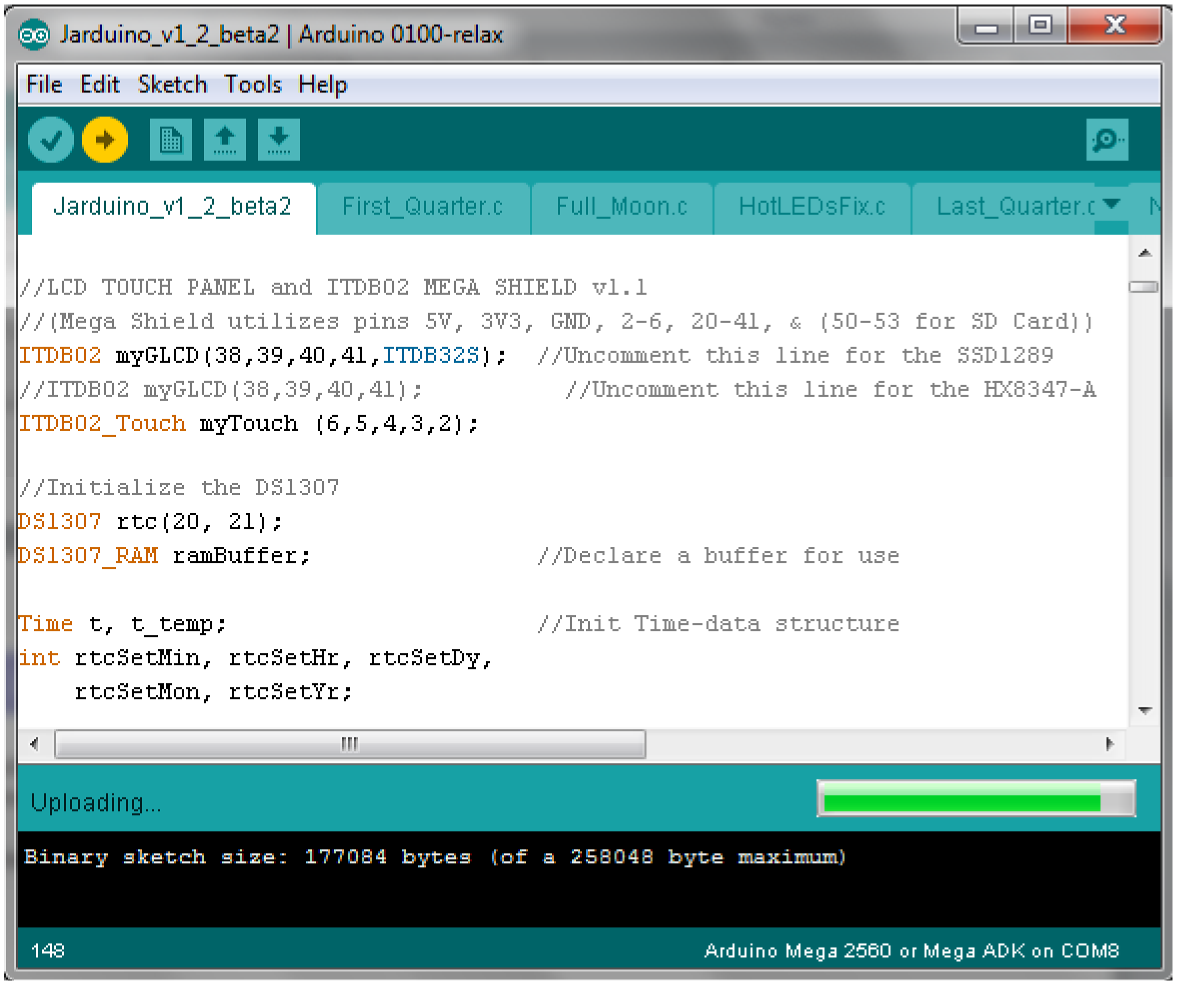
As you can see, I indeed do have the right one selected. The other option on line 148 is for the "HX8347-A" module. You can see that particular line is "greyed" out. In the Arduino IDE, putting a "//" in front of a line makes is "invisible" to the Arduino. If I happened to be using an "HX8347-A," I would simply put "//" in front of the code on line 147 (which will "grey" it out), and I would delete the "//" in front of the code on line 148. If I did that, it would then look like this:



```
//LCD TOUCH PANEL and ITDB02 MEGA SHIELD v1.1
//(Mega Shield utilizes pins 5V, 3V3, GND, 2-6, 20-41, & (50-53 for SD Card))
//ITDB02 myGLCD(38,39,40,41,ITDB32S);  //Uncomment this line for the SSD1289
ITDB02 myGLCD(38,39,40,41);           //Uncomment this line for the HX8347-A
ITDB02_Touch myTouch (6,5,4,3,2);
```

The ITDB02_graph16 library has the HX8347-A module as the default Model, and therefore doesn't require you specifying the Model number (ie – ITDB32S for the SSD1289). For a controller other than one of those two, you may need to refer to the Compatibility Chart in the Hardware Section to get the necessary code. The code you may need is listed under the heading "Model for ITDB02()." Selecting the wrong Model will result in the Jarduino displaying a WHITE SCREEN.

Back to the example; so now that I have selected the correct module that I'm going to use, I'm going to go ahead and click the "Upload" button. If everything compiles correctly, then your IDE should look something like this:



```
//LCD TOUCH PANEL and ITDB02 MEGA SHIELD v1.1
//(Mega Shield utilizes pins 5V, 3V3, GND, 2-6, 20-41, & (50-53 for SD Card))
ITDB02 myGLCD(38,39,40,41,ITDB32S);  //Uncomment this line for the SSD1289
//ITDB02 myGLCD(38,39,40,41);          //Uncomment this line for the HX8347-A
ITDB02_Touch myTouch (6,5,4,3,2);

//Initialize the DS1307
DS1307 rtc(20, 21);
DS1307_RAM ramBuffer;                //Declare a buffer for use

Time t, t_temp;                      //Init Time-data structure
int rtcSetMin, rtcSetHr, rtcSetDy,
    rtcSetMon, rtcSetYr;
```

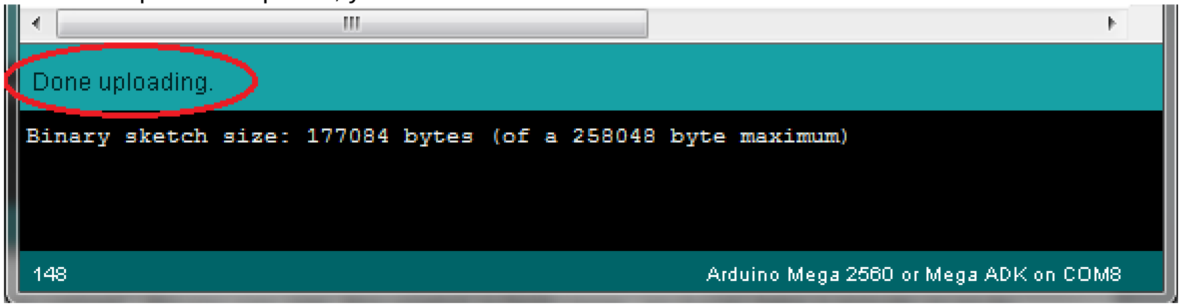
Uploading...

```
Binary sketch size: 177084 bytes (of a 258048 byte maximum)
```

148 Arduino Mega 2560 or Mega ADK on COM8

NOTE: Don’t worry if it takes a while to upload. As you can see, this sketch is fairly large, so it may take a minute or so to upload. If it goes much longer than that, then you may have some issues that need tending to. Most likely, “stuck” uploads are usually the fault of the bootloader or the IDE. Please contact me if this happens to you, and I’ll try to help you through it. It may be worth mentioning that this issue can arise if using Arduino 1.0.1, so take note.

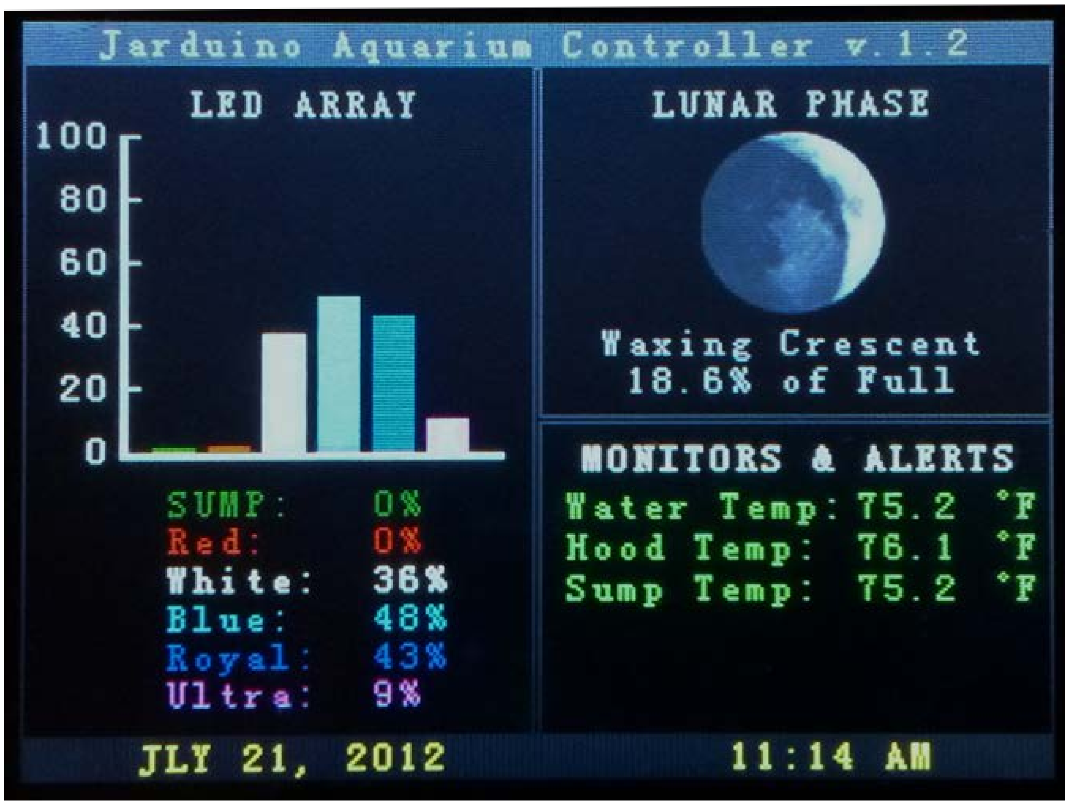
When the upload completes, you should see this at the bottom of the IDE:



```
Done uploading.
Binary sketch size: 177084 bytes (of a 258048 byte maximum)
```

Hopefully, all went well and you see the Main Screen of the Jarduino on your LCD screen (as pictured on the Cover of this manual).

4. JARDUINO FUNCTIONALITY



Waxing Crescent
18.6% of Full

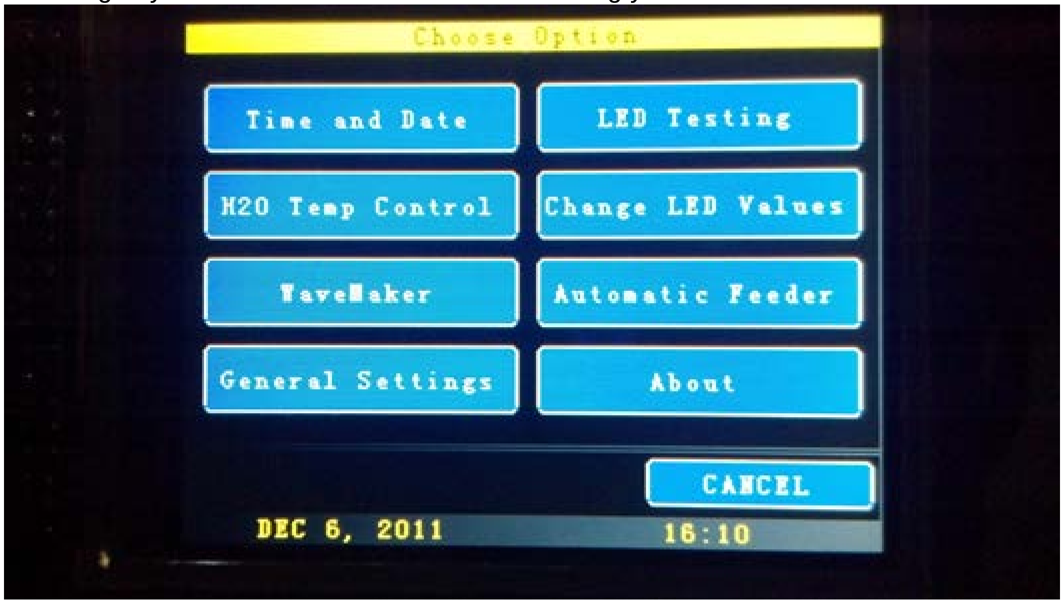
SUMP:	0%
Red:	0%
White:	36%
Blue:	48%
Royal:	43%
Ultra:	9%

Water Temp: 75.2 °F
Hood Temp: 76.1 °F
Sump Temp: 75.2 °F

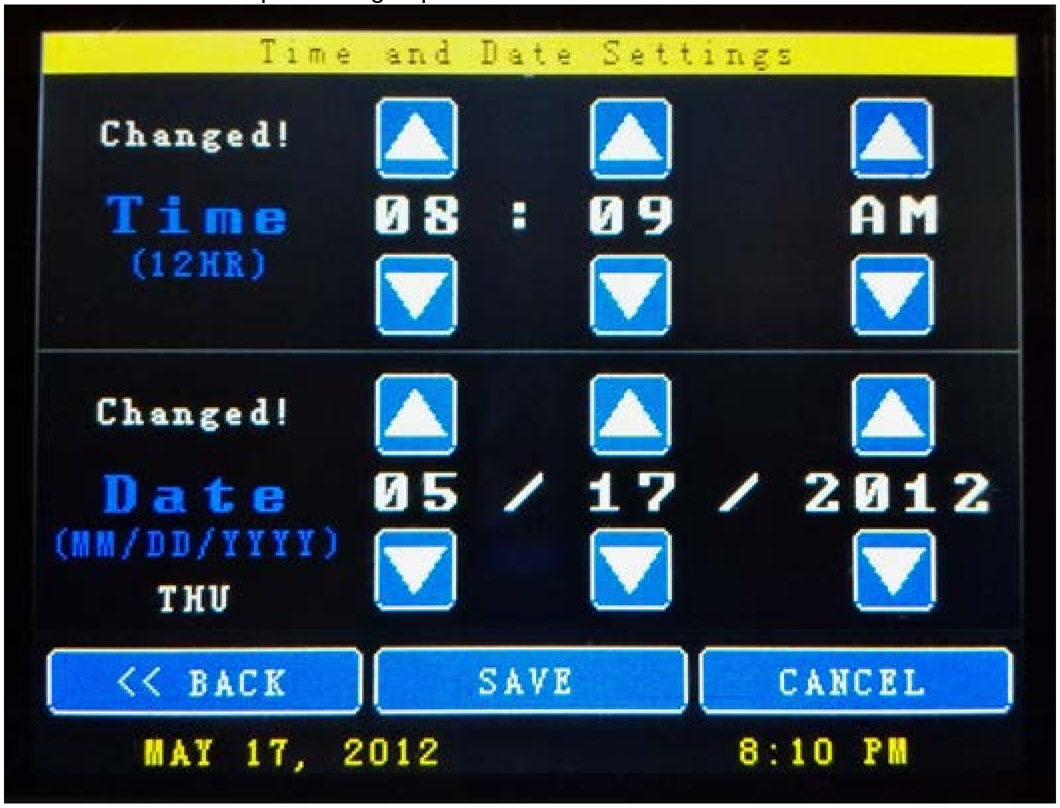
JLY 21, 2012 11:14 AM

As you can see, the **MAIN SCREEN** displays each LED color's current percent output. Also included is the Lunar Phase complete with a picture of the current phase, a description of said phase, and the current percentage of a Full Moon. Under the MONITORS & ALERTS section, the water temperature, and the temps of the Main Hood and the Sump LED fixture are monitored. Alerts will also flash and change colors, etc if/when the temperatures go outside of user defined ranges. If a temp sensor isn't connected (or there's a bad connection), an Error message will be displayed in place of the temperature. You may also briefly encounter this message upon returning to this screen. The status bar at the bottom of the screen will carry out to all the other pages and is fully functional on each subsequent page.

Touching anywhere on the **MAIN SCREEN** will bring you to the **MAIN MENU**:



The **Time and Date** option brings up this screen:

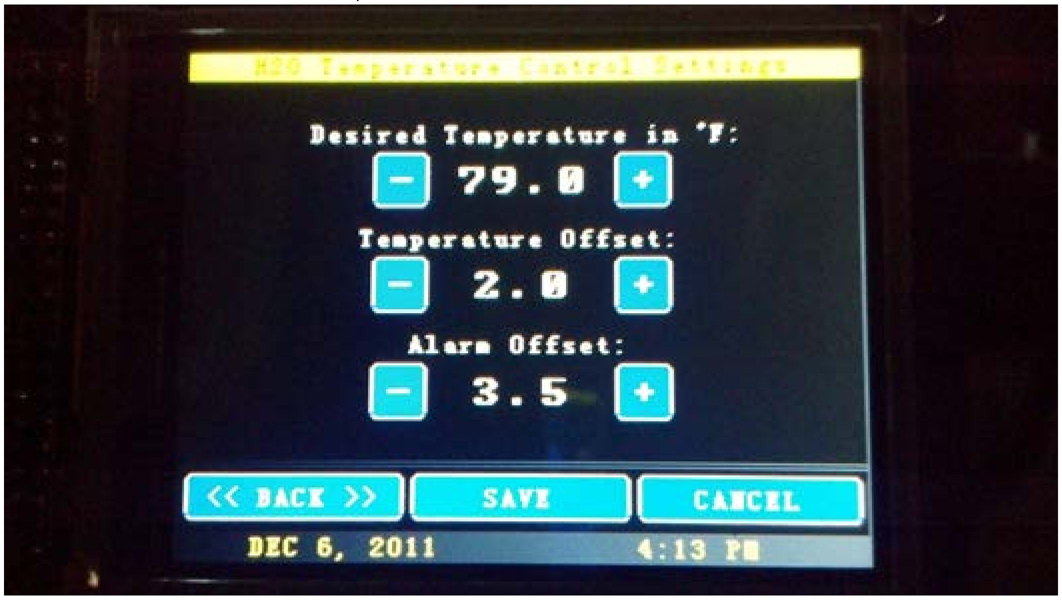

Time and Date Settings

Changed!
Time (12HR) 08 : 09 AM

Changed!
Date (MM/DD/YYYY) 05 / 17 / 2012
THU

MAY 17, 2012 8:10 PM

The **H2O Temp Control** option is for setting your desired water temperature, and also allows you to set offsets to turn on and off heaters and/or chillers, and also an audio alarm:

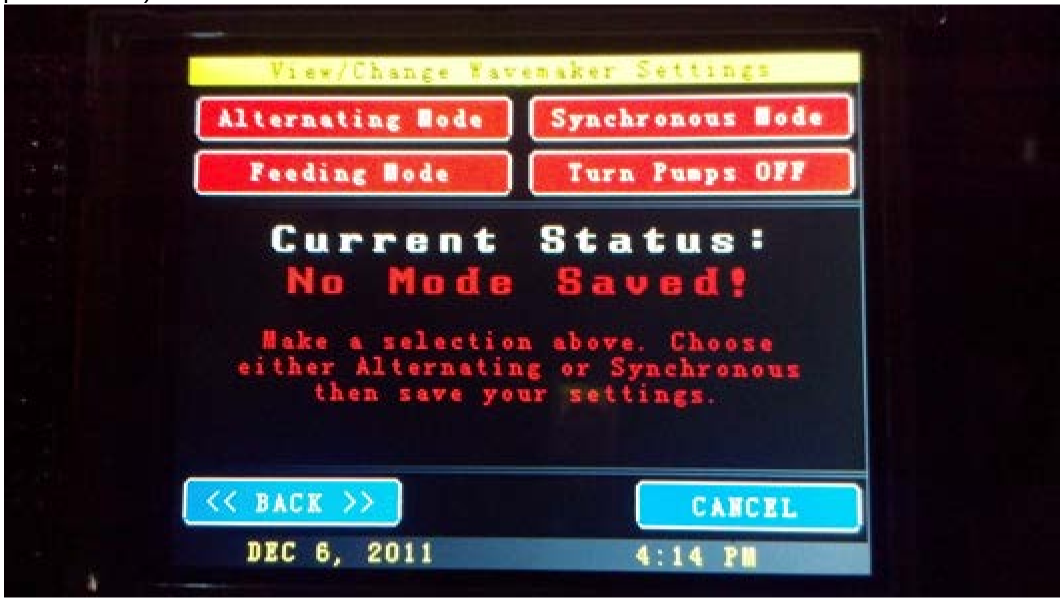

H2O Temperature Control Settings

Desired Temperature in °F: 79.0
Temperature Offset: 2.0
Alarm Offset: 3.5

DEC 6, 2011 4:13 PM

This simply says that if the temperature goes 2 degress above 79 degress F, it will turn on a chiller. If it goes below by 2 degrees a heater will kick on. +/- 3.5 degrees and an alarm will sound. This alarm will also flash on the **MAIN SCREEN** and turn the temperature readout from green to blue for too cold or red for too hot.

Next up is the **WaveMaker** options screen. There are several sub-menus for this particular option. This is what it looks like when you enter into the **WaveMaker** section for the first time (the information presented will change after you enter your preferences):

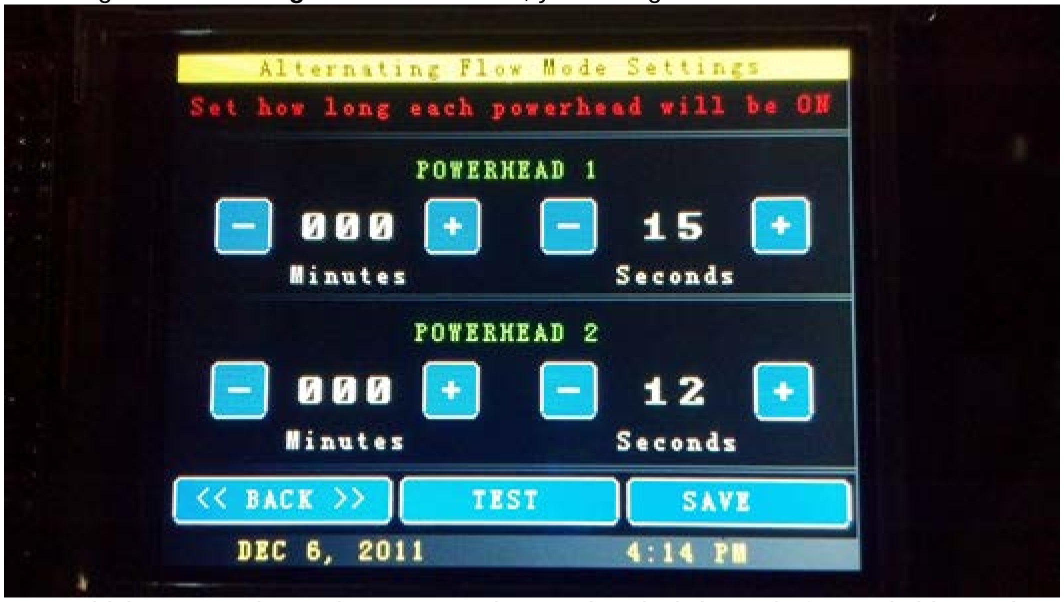


Current Status:
No Mode Saved!

Make a selection above. Choose either Alternating or Synchronous then save your settings.

DEC 6, 2011 4:14 PM

Selecting the **Alternating Mode** from above, you then get this screen:



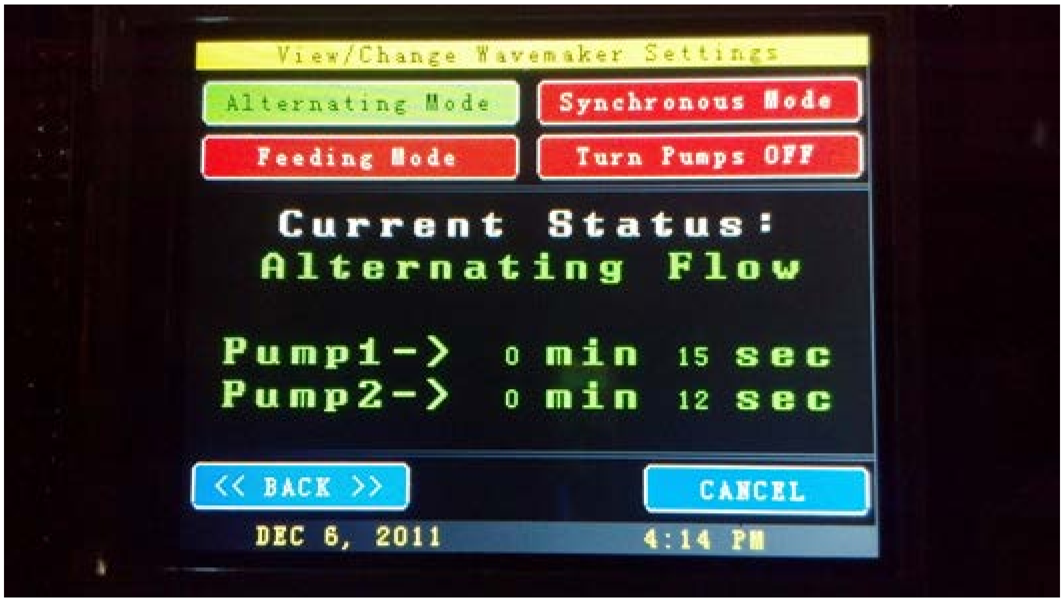
Set how long each powerhead will be ON

POWERHEAD 1
000 Minutes 15 Seconds

POWERHEAD 2
000 Minutes 12 Seconds

DEC 6, 2011 4:14 PM

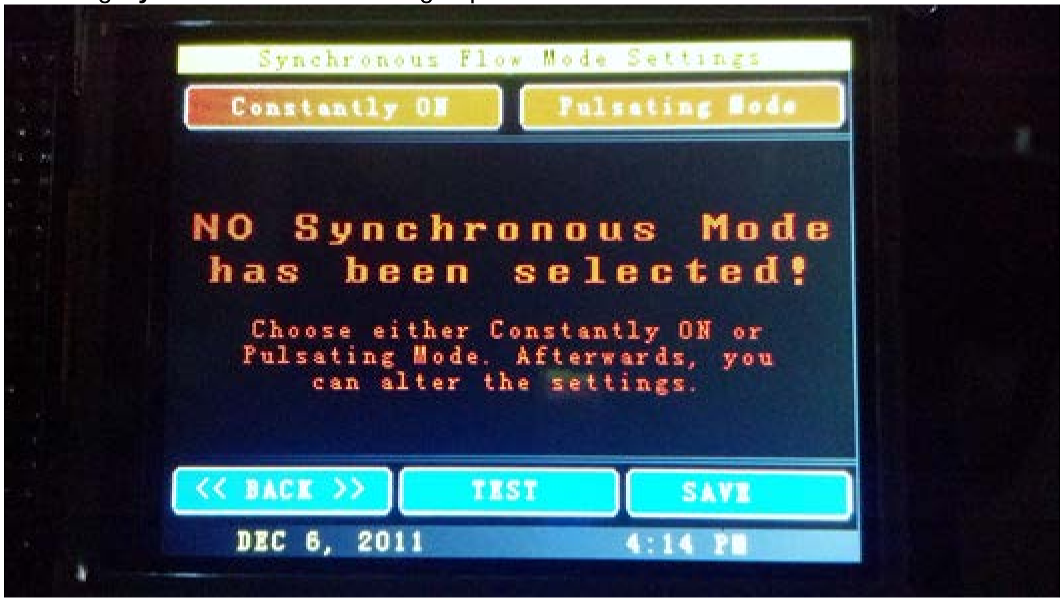
From which you can the set how long each powerhead will be on before switching to the other. You can test your settings directly from this screen. Pressing the **TEST** button will turn it green, and the powerheads will run according to the time you set. If you like it, you can save it to EEPROM (Arduino's memory). After saving this, and going back to the main **WaveMaker** screen, you now see the following:


View/Change Wavemaker Settings

Current Status:
Alternating Flow

Pump1-> 0 min 15 sec
Pump2-> 0 min 12 sec

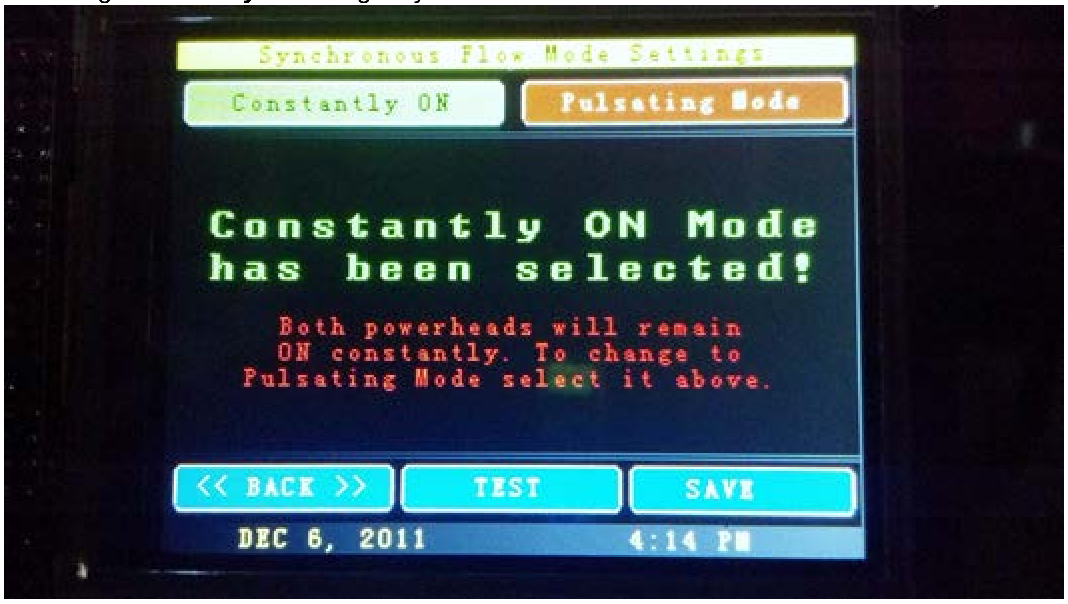
Selecting **Synchronous Mode** brings up this screen:


Synchronous Flow Mode Settings

NO Synchronous Mode
has been selected!

Choose either Constantly ON or
Pulsating Mode. Afterwards, you
can alter the settings.

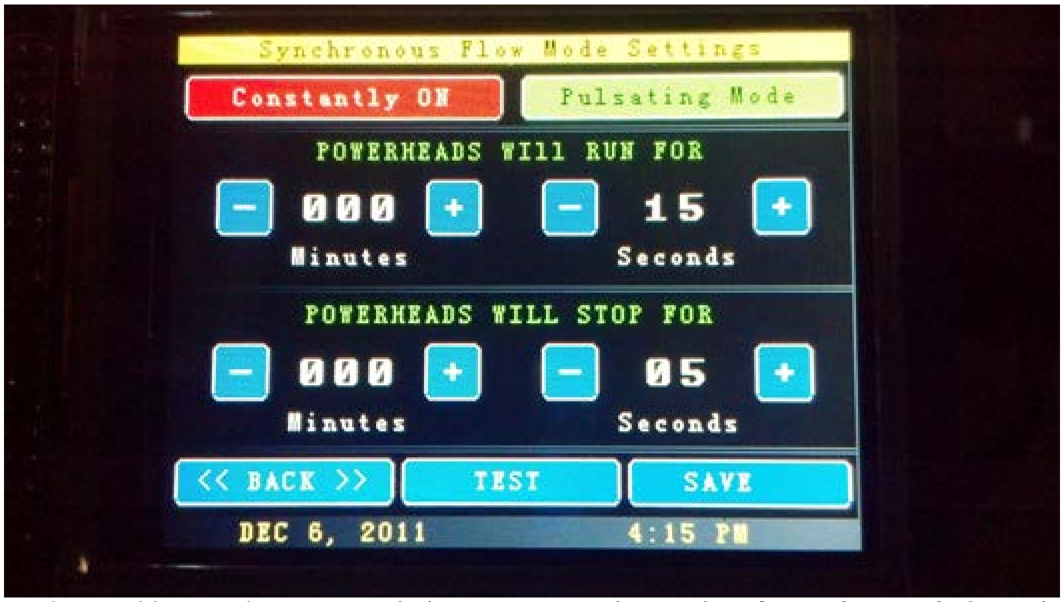
Selecting **Constantly ON** will give you this:



Constantly ON Mode has been selected!

Both powerheads will remain ON constantly. To change to Pulsating Mode select it above.

And selecting the **Pulsating Mode** option will bring you to this page, allowing you to set how long the Powerheads will be ON and OFF:



POWERHEADS WILL RUN FOR

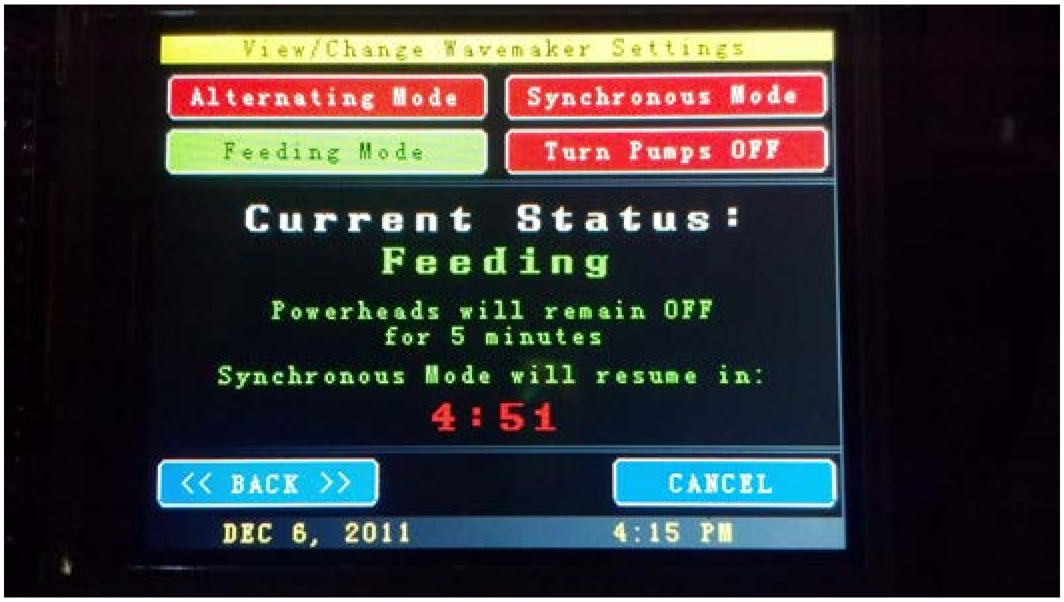
000 Minutes 15 Seconds

POWERHEADS WILL STOP FOR

000 Minutes 05 Seconds

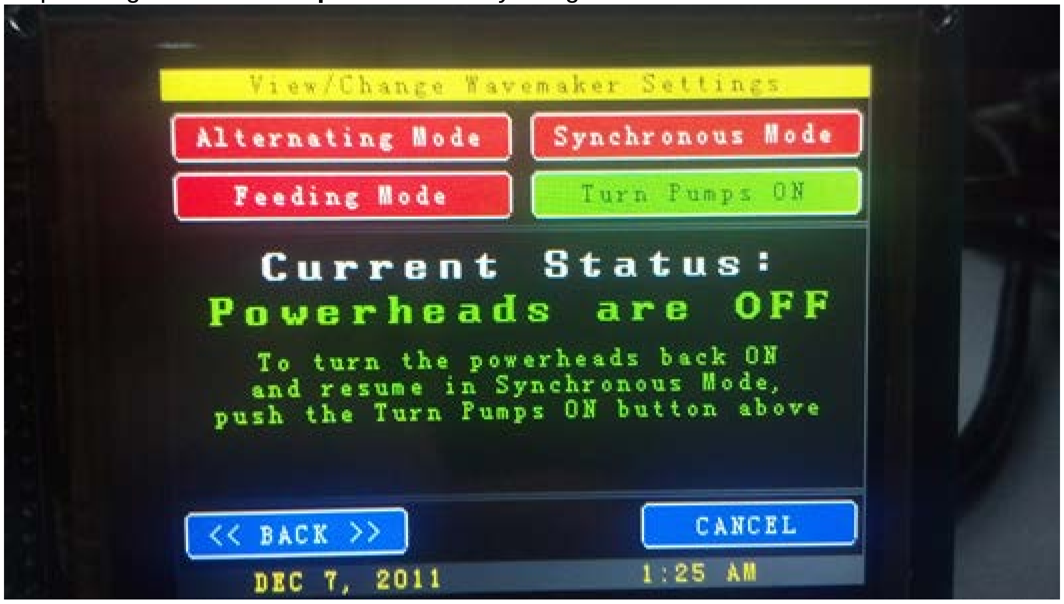
Setting up this page is pretty much the same as setting up the **Alternating Mode** from above.

And back to the main **WaveMaker** screen, if you select **Feeding Mode**, you will see this message along with a countdown.



Current Status:
Feeding
Powerheads will remain OFF for 5 minutes
Synchronous Mode will resume in:
4:51

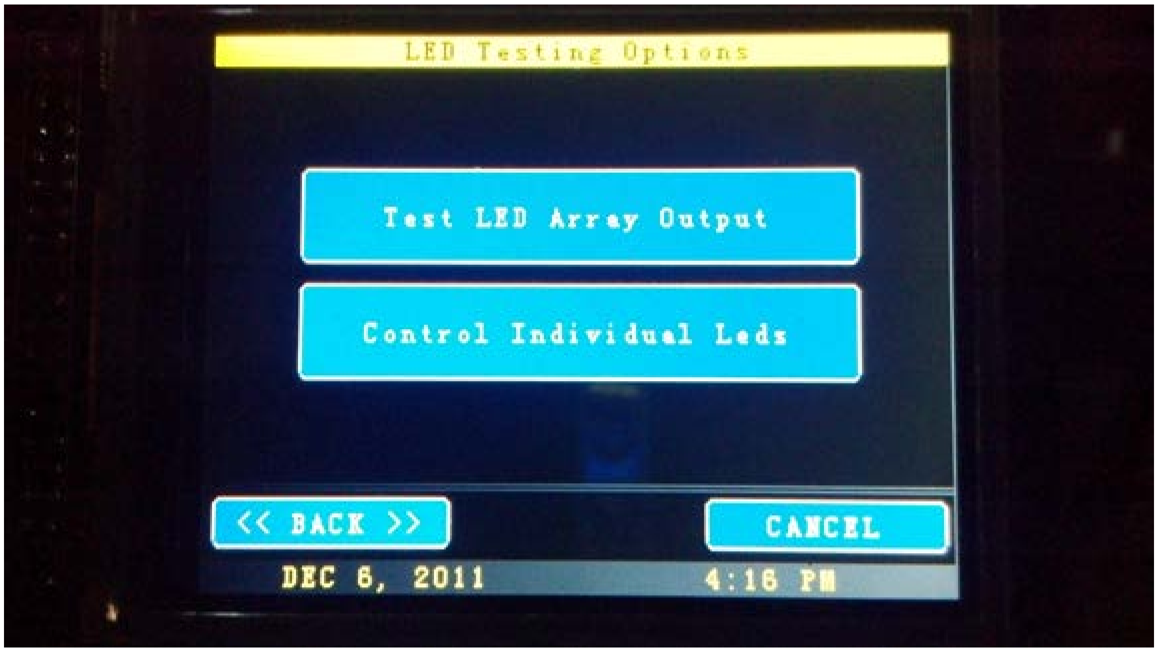
Or pressing the **Turn Pumps OFF** button you'll get:



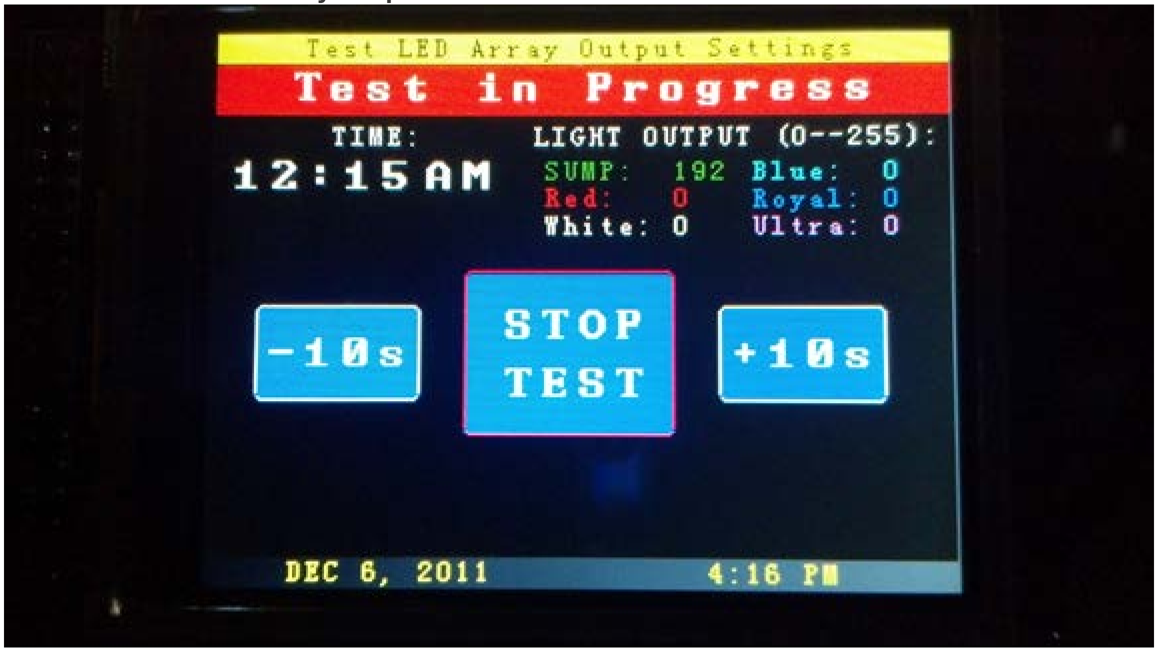
Current Status:
Powerheads are OFF
To turn the powerheads back ON and resume in Synchronous Mode, push the Turn Pumps ON button above

Pressing **CANCEL** or **BACK** will also resume the saved wave mode.

Next up on the **MAIN MENU / CHOOSE OPTIONS** screen we have the **LED Testing** button. Pressing it will give you more choices:

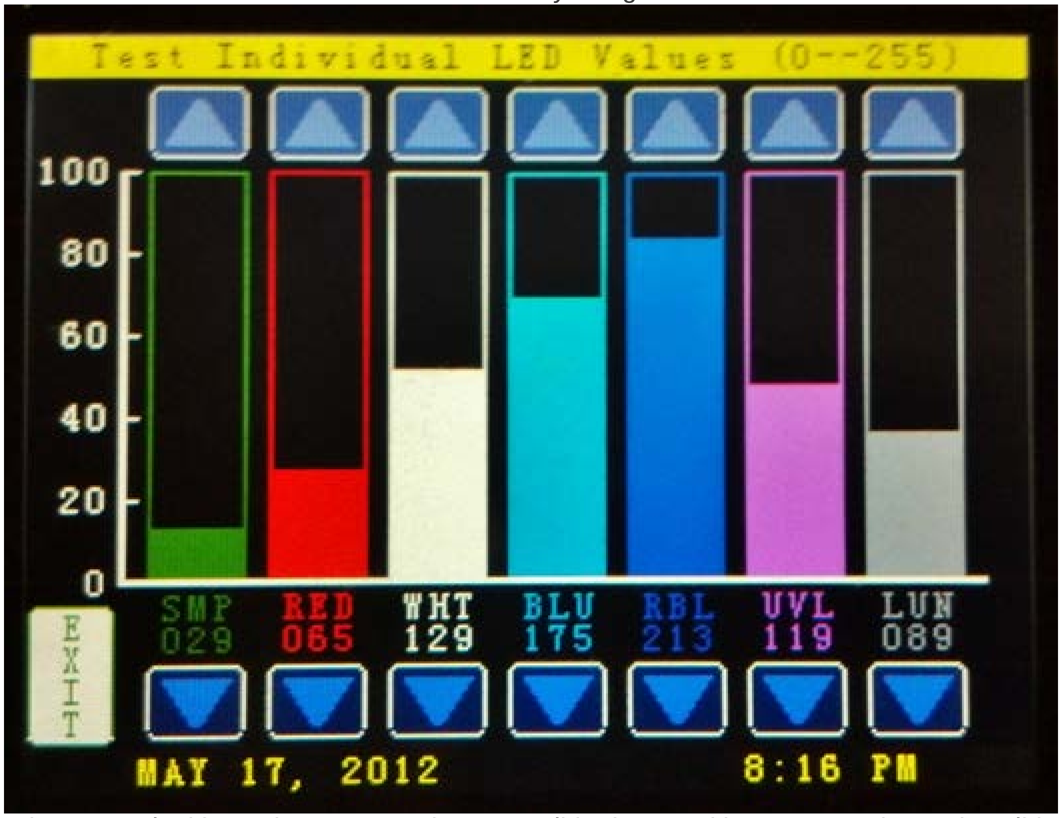


Here is the **Test LED Array Output** screen:



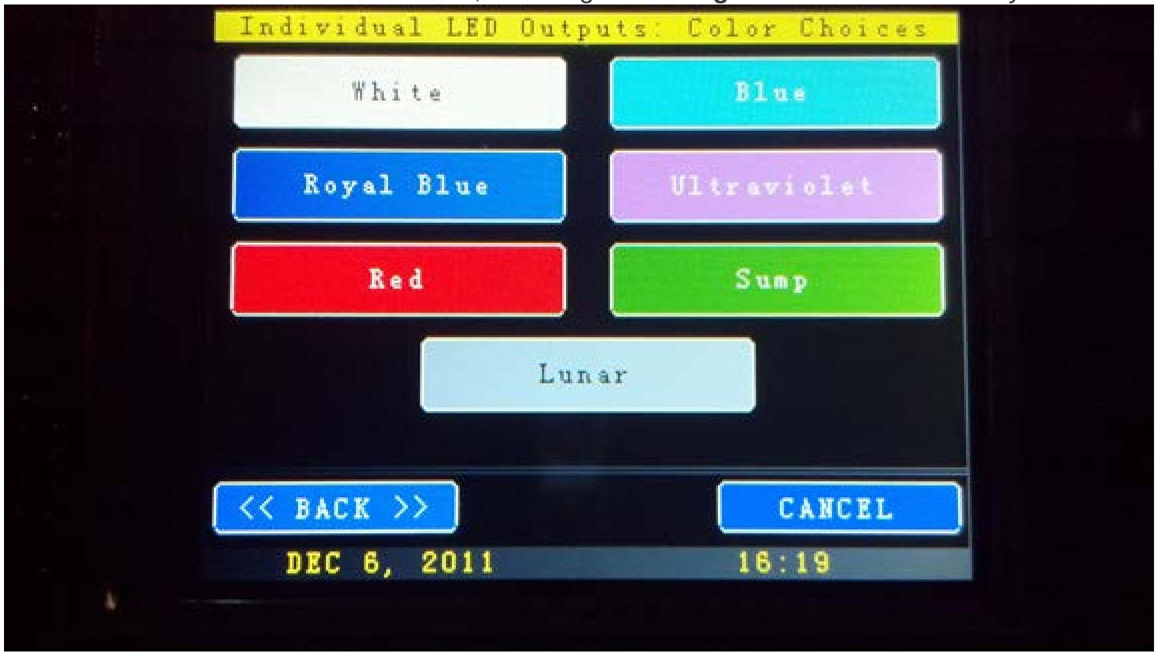
This test runs through the entire day quickly from midnight on. In other words, this allows you to watch your lighting change intensities in fast-forward.

You can **Control Individual LEDs** with ease by using this screen.

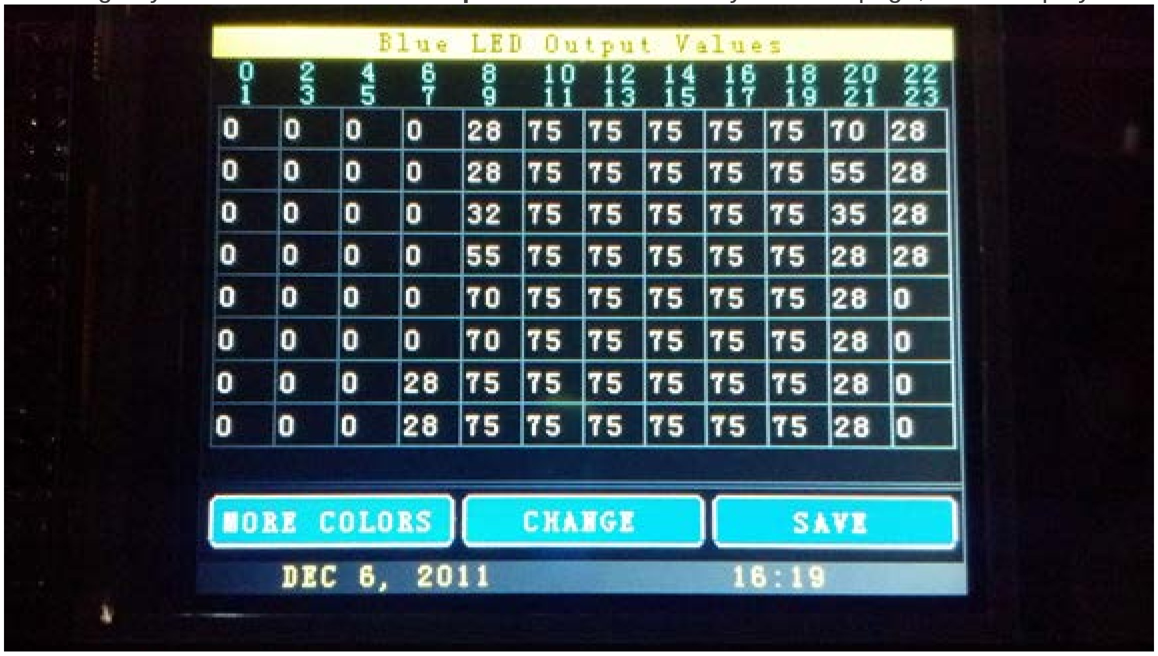


What you're looking at here are touch screen slider bars and buttons. You just point, slide, or drag your finger up and down to quickly get a value, and the results are immediate. You can even slide your finger in any pattern and get that result across all channels. It takes all of 1 second to make ALL those colors go from 0 to the LEDs shining at the values depicted. For further fine tuning, you can use the up and down buttons above each color. Once you hit "EXIT," you will leave that screen and the LEDs will resume their regular schedule.

Back to the **CHOOSE OPTIONS** screen, selecting the **Change LED Values** will take you to this menu:



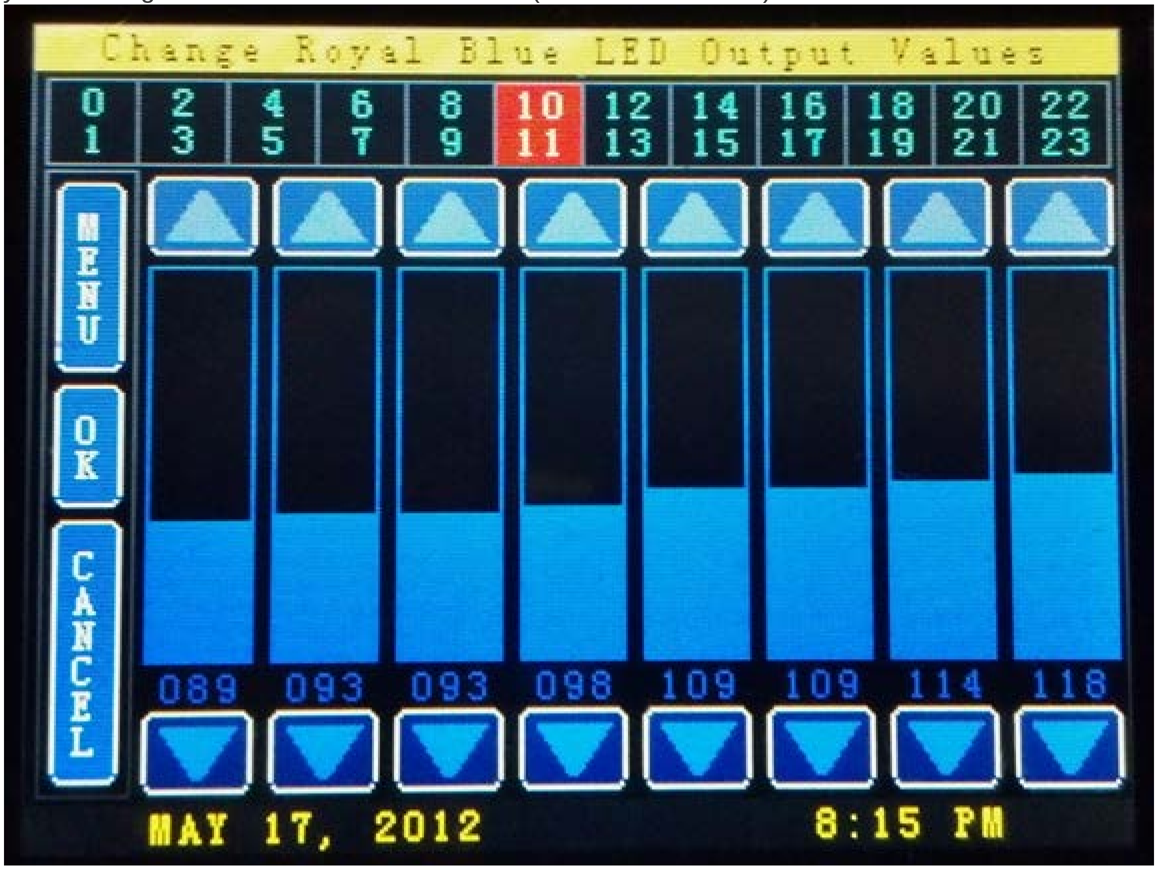
Selecting any of the Colors or the **Sump** button will then take you to this page, which displays the LED values over 24 hrs:



Blue LED Output Values

0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15	16 17	18 19	20 21	22 23
0	0	0	0	28	75	75	75	75	75	70	28
0	0	0	0	28	75	75	75	75	75	55	28
0	0	0	0	32	75	75	75	75	75	35	28
0	0	0	0	55	75	75	75	75	75	28	28
0	0	0	0	70	75	75	75	75	75	28	0
0	0	0	0	70	75	75	75	75	75	28	0
0	0	0	28	75	75	75	75	75	75	28	0
0	0	0	28	75	75	75	75	75	75	28	0

Pressing the **MORE COLORS** button will allow you to select from the other colors, pressing the **CHANGE** button will allow you to change the value for each time frame (15 minute intervals). Selected below are the intervals from 10-11 AM:



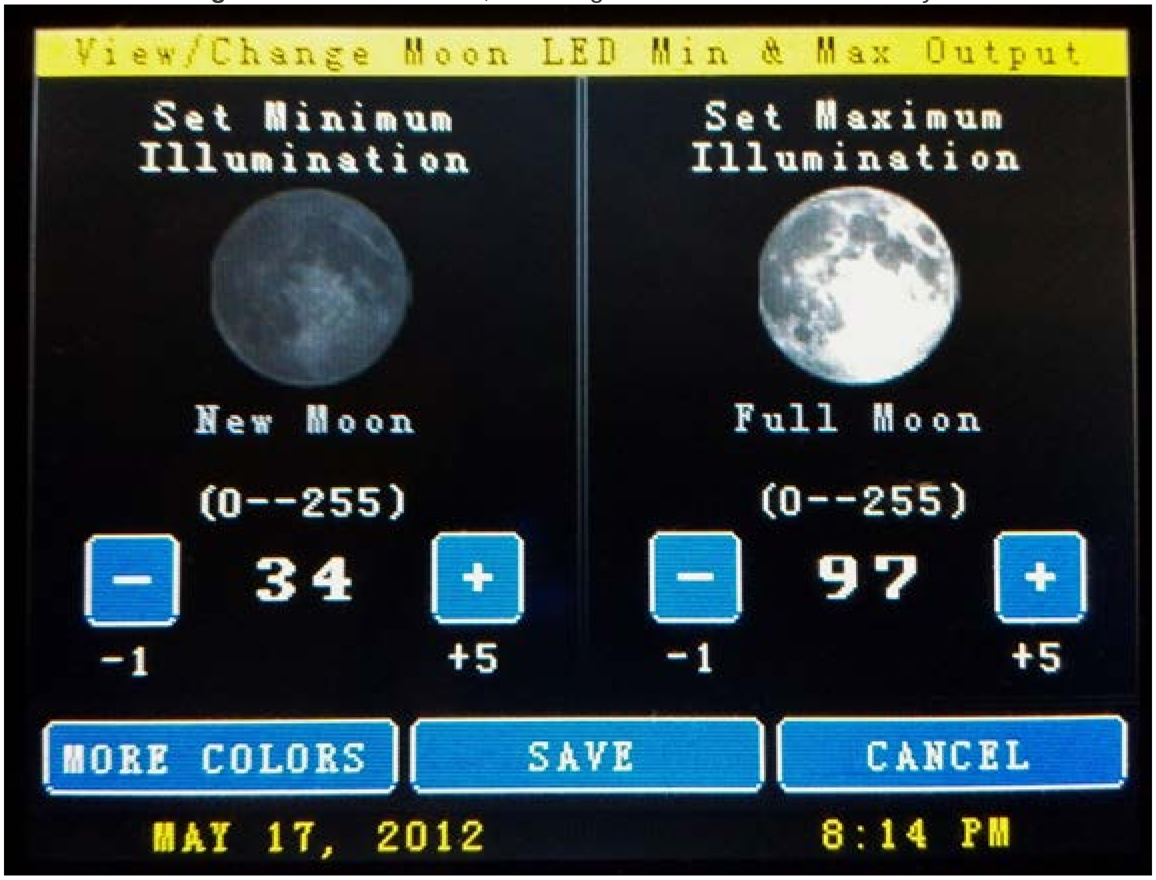
Change Royal Blue LED Output Values

0 1	2 3	4 5	6 7	8 9	10 11	12 13	14 15	16 17	18 19	20 21	22 23

089	093	093	098	109	109	114	118

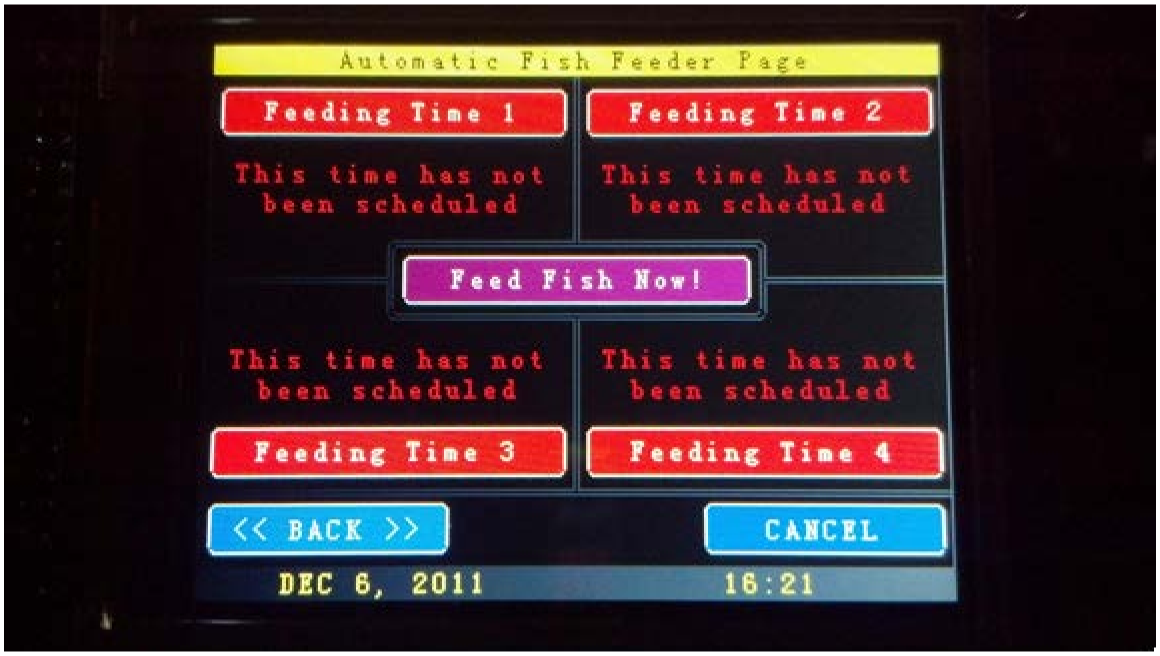
This page also utilizes sliders and buttons. When you select the color, the slider bars and numbers will also be that color. Pressing **OK** will take you back a page with your new values. Pressing **SAVE** will write these values to the EEPROM, ready to resume your choices after recovering from a power failure.

Back to the **Change LED Values** screen, choosing the **Lunar** button will take you to this screen:



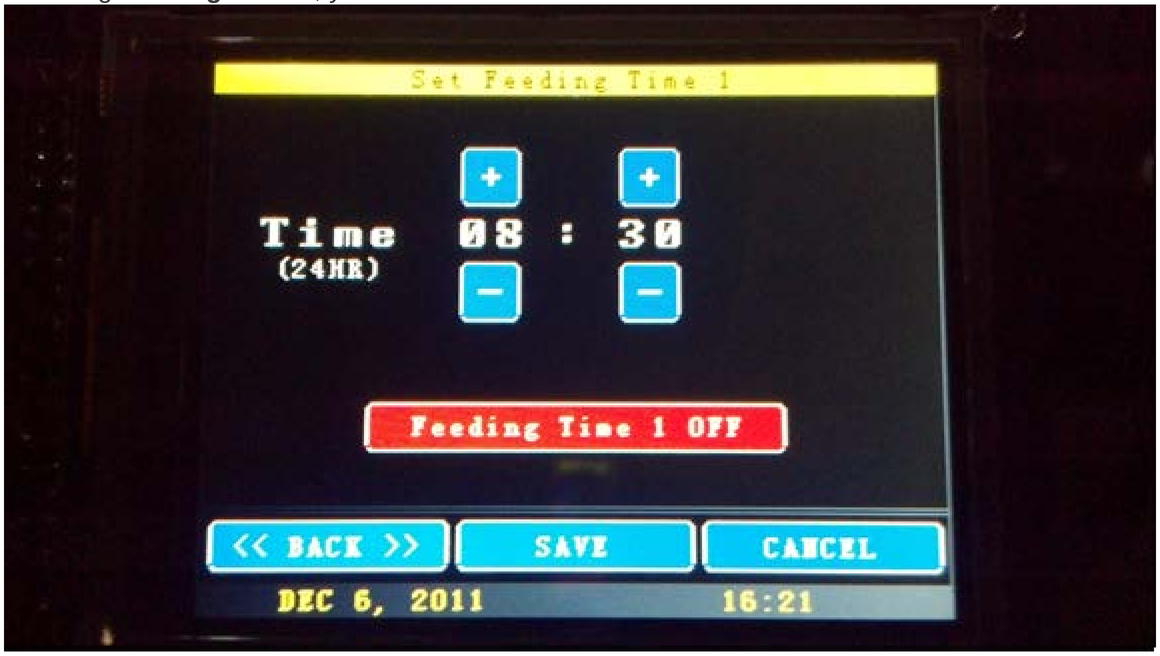
You have the ability to change the **maximum** value of the Lunar Cycle (aka Full Moon) as well as the **minimum** value (aka New Moon). Your New Moon doesn't have to be completely dark if you don't want it to be.

Next on the **CHOOSE OPTIONS** screen you will see the **Automatic Feeder** option. This is another menu that has several different choices:

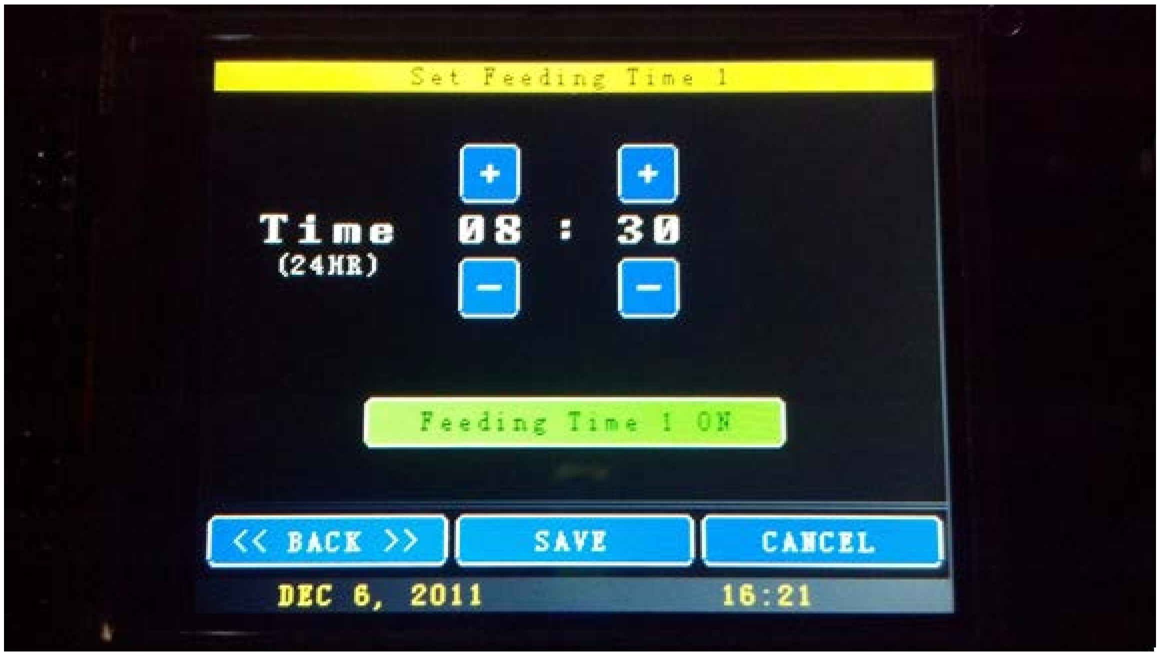

Automatic Fish Feeder Page
Feeding Time 1
Feeding Time 2
This time has not been scheduled
This time has not been scheduled
Feed Fish Now!
This time has not been scheduled
This time has not been scheduled
Feeding Time 3
Feeding Time 4
<< BACK >>
CANCEL
DEC 6, 2011
16:21

This is what it looks like without any feeding times saved.

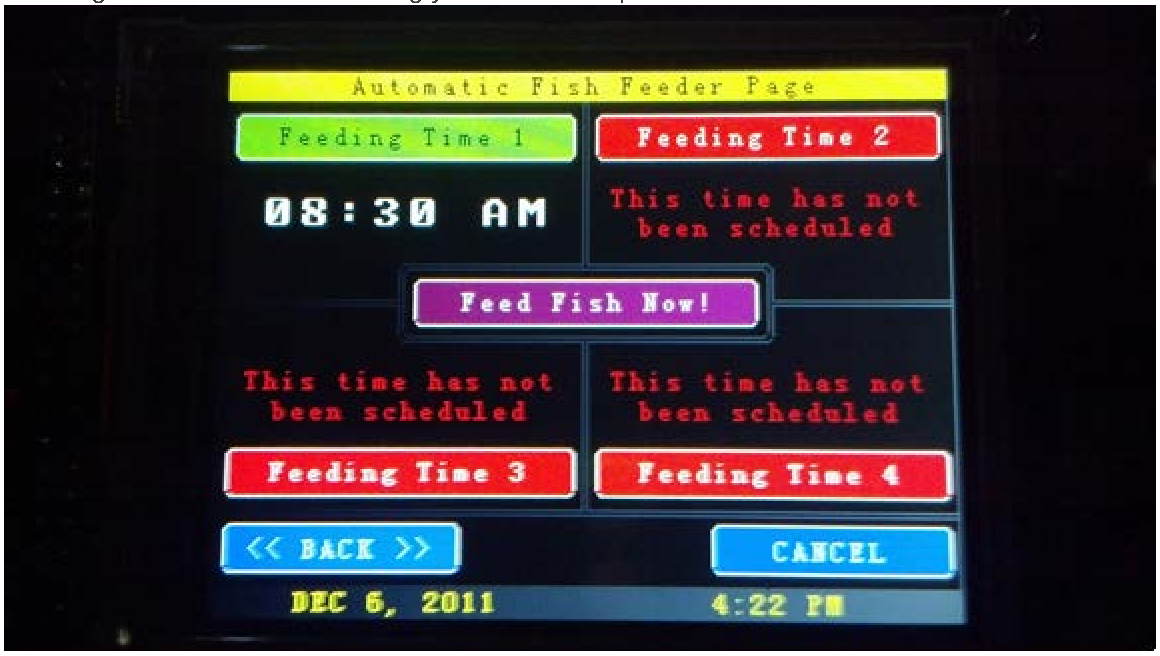
Selecting **Feeding Time 1**, you will see this screen:



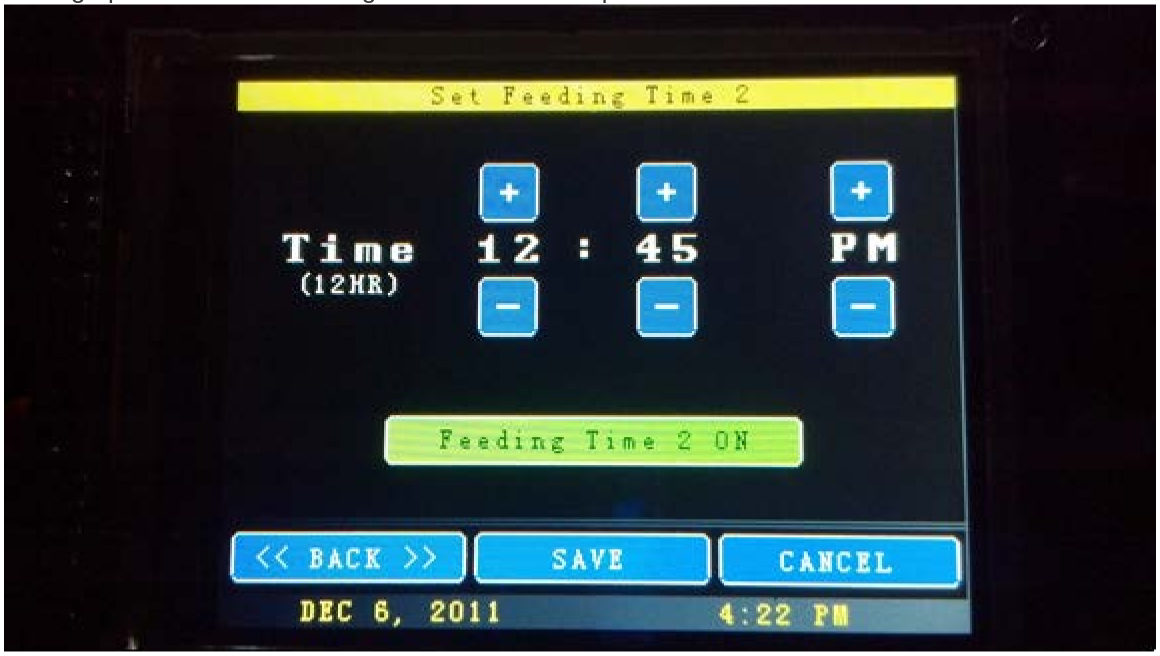
So after setting the time, you then decide if you want it ON or OFF. When you select the red button to turn it ON you see this:



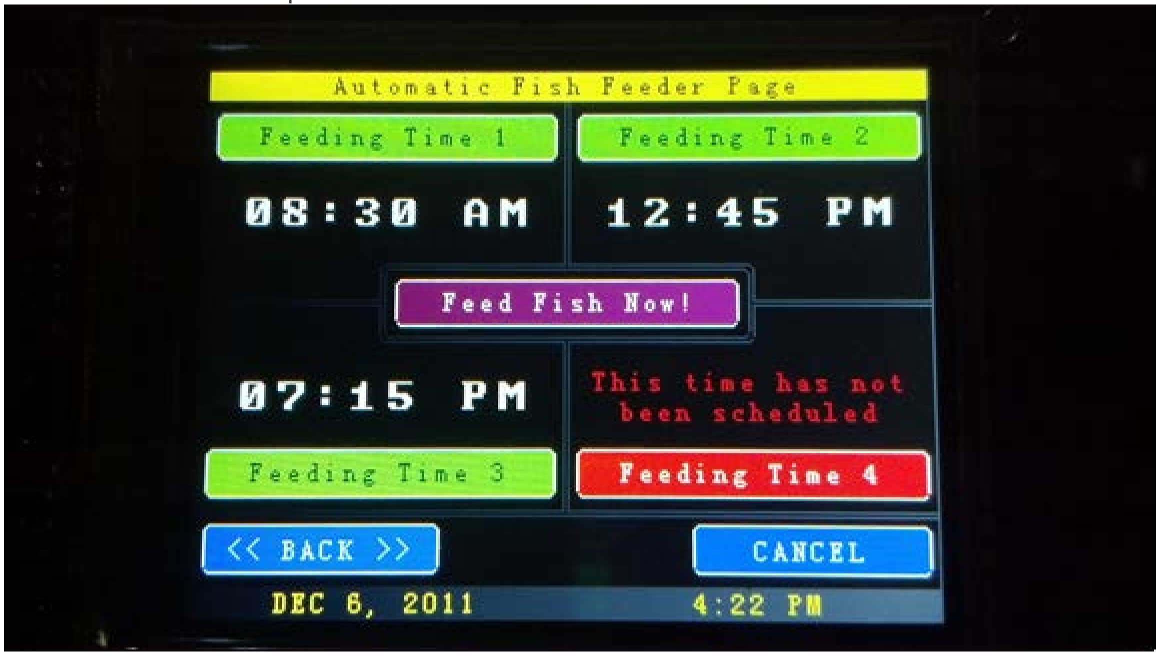
Pressing the **SAVE** button will bring you back to the previous screen:


Automatic Fish Feeder Page
Feeding Time 1
08:30 AM
Feeding Time 2
This time has not been scheduled
Feed Fish Now!
This time has not been scheduled
This time has not been scheduled
Feeding Time 3
Feeding Time 4
<< BACK >>
CANCEL
DEC 6, 2011
4:22 PM

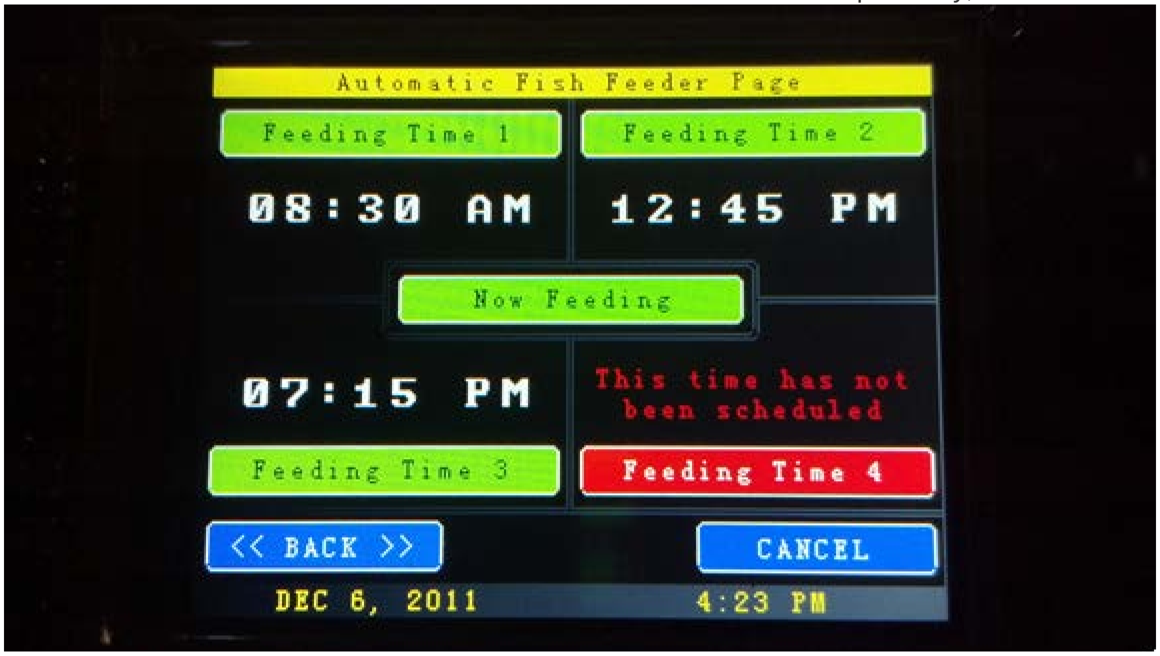
Setting up an additional feeding time is identical in procedure:



And another shot of the previous screen with additional times saved:

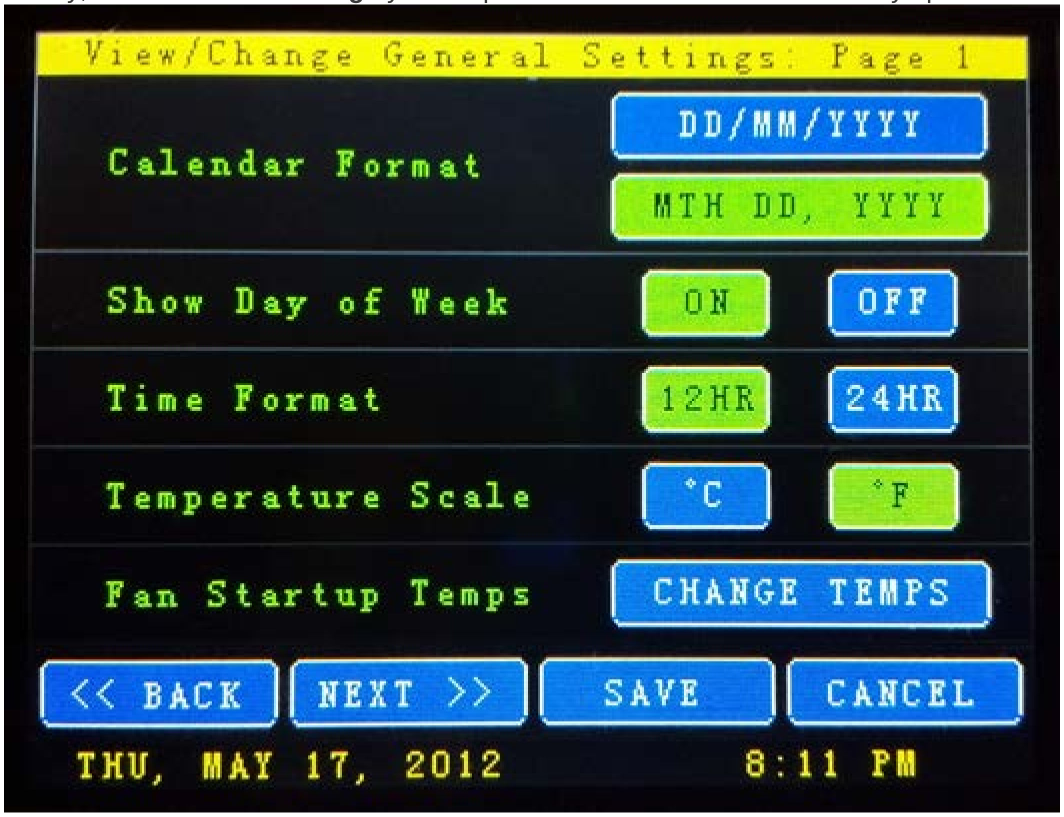


So that red-violet **Feed Fish Now!** Button in the center should be self-explanatory, here it is in action:



It stays green for a few seconds before going back to the red-violet color. If you like, you can have the powerheads shut off automatically for 5 minutes during the feeding times when you press the **Feed Fish Now!** button. This can easily be programmed by selecting it in the **General Settings** page (covered below). You could alternatively suspend the powerheads by turning them off in the **WaveMaker** page.

Finally, in the **General Settings** you are presented with several user-friendly options. Here's Page 1:



View/Change General Settings: Page 1

Calendar Format: DD/MM/YYYY | MTH DD, YYYY

Show Day of Week: ON | OFF

Time Format: 12HR | 24HR

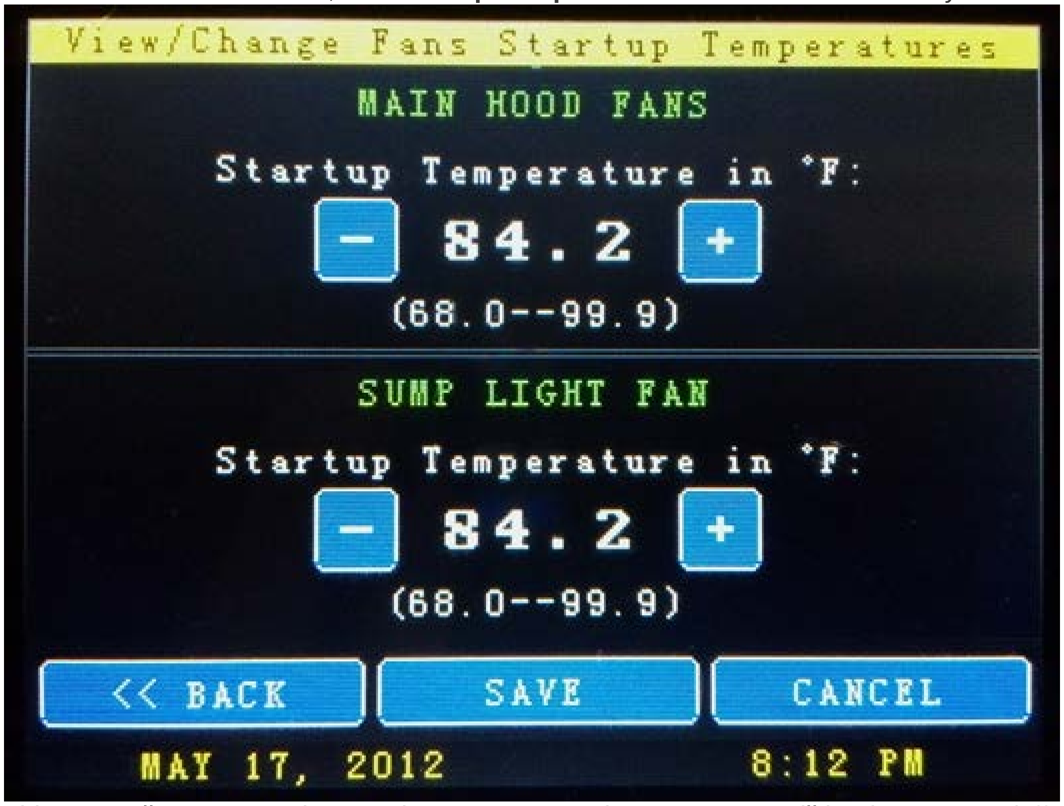
Temperature Scale: °C | °F

Fan Startup Temps: CHANGE TEMPS

<< BACK | NEXT >> | SAVE | CANCEL

THU, MAY 17, 2012   8:11 PM

Most of the stuff on here should be self-explanatory. Simply select the options you like, and when you do they are highlighted in green. Press "SAVE" and the Jarduino will remember your selections. One thing that might not be so obvious is the last selection, **Fan Startup Temps**. Here's a screenshot of what you'll see when you touch that button:



View/Change Fans Startup Temperatures

MAIN HOOD FANS

Startup Temperature in °F:

- 84.2 +

(68.0--99.9)

SUMP LIGHT FAN

Startup Temperature in °F:

- 84.2 +

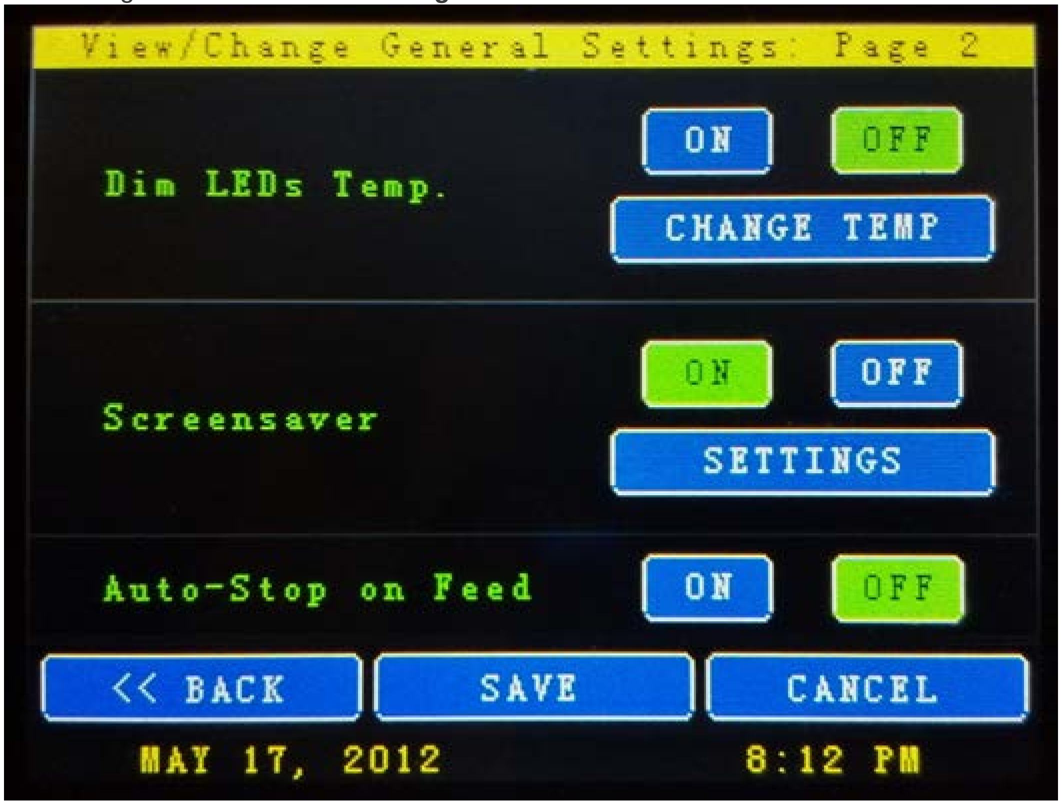
(68.0--99.9)

<< BACK | SAVE | CANCEL

MAY 17, 2012   8:12 PM

This page allows you to choose what temperature the PWM Fans will begin to turn on/off. It is from that temperature that for every degree of temperature increase, the PWM fans will increase in speed by 10%.

Here’s Page 2 of the **General Settings**:



View/Change General Settings: Page 2

Dim LEDs Temp. ON [✓ OFF] CHANGE TEMP

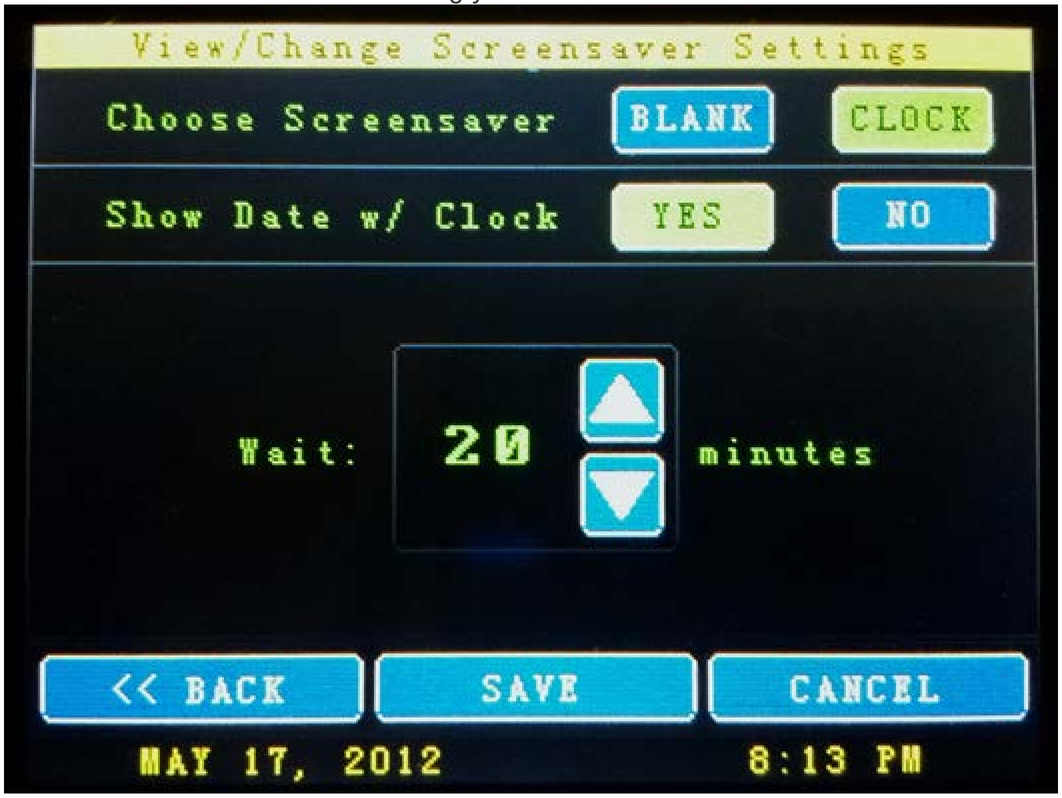
Screensaver [✓ ON] OFF SETTINGS

Auto-Stop on Feed ON [✓ OFF]

<< BACK SAVE CANCEL

MAY 17, 2012 8:12 PM

There are three options on this page. Now you can turn a **Screensaver** “ON” or “OFF” from this page. Selecting the “SETTINGS” of **Screensaver** will bring you here:



View/Change Screensaver Settings

Choose Screensaver BLANK [✓ CLOCK]

Show Date w/ Clock [✓ YES] NO

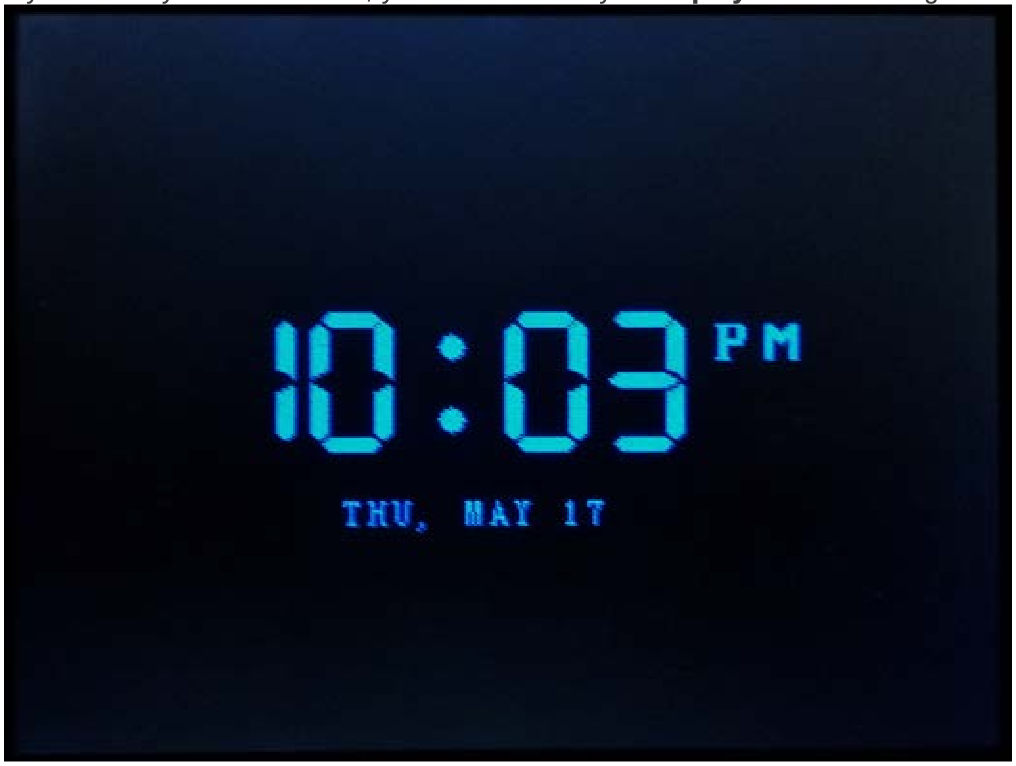
Wait: 20 minutes

<< BACK SAVE CANCEL

MAY 17, 2012 8:13 PM

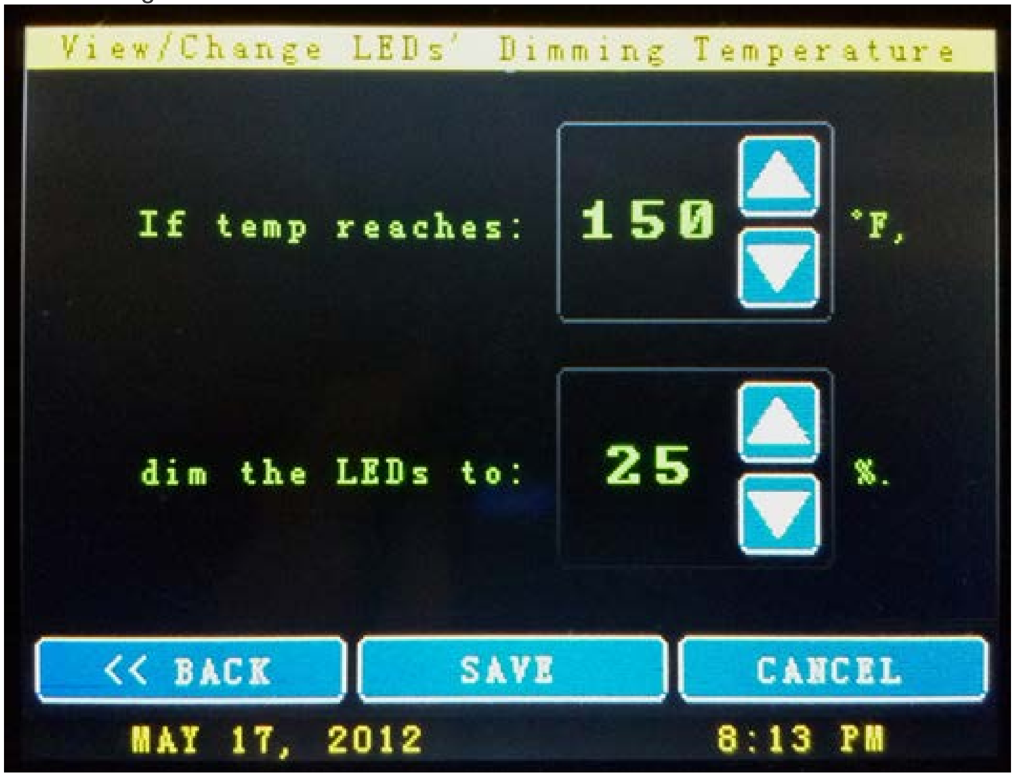
From this screen, you can customize the **screensaver**. First off, at the bottom section you can now choose **how long** it will take for the screensaver to come on. You can select anywhere from one minute to 99 minutes.

If you decided you just want a **blank screen** as a screensaver instead of a **Digital Clock**, no problem. Just select it. But if you decide you like the clock, you have the ability to **display the date** along with the time. Check it out:



Of course, whatever **Calendar** and **Time Formats** you choose will also be carried over to this screen.

The **Dim LEDs Temp** is a built in safeguard that automatically dims the LEDs whenever your heatsink reaches a defined temperature. This option is to prevent your LEDs from getting too hot, which will lead to premature failure. Like the **Screensaver**, you can choose whether or not to use this feature by selecting **ON** or **OFF**. Selecting **Change Temp**, you will be brought to this screen:

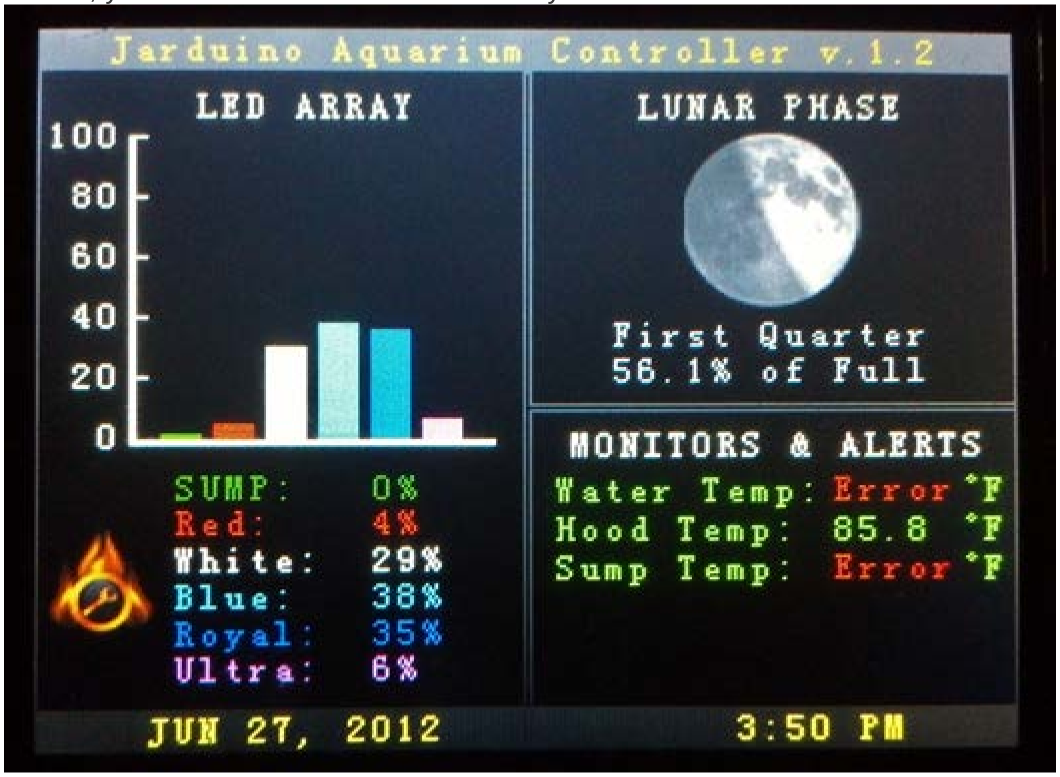


View/Change LEDs' Dimming Temperature

If temp reaches: 150 °F,

dim the LEDs to: 25 %.

It should go without saying by now, but in case you're wondering, the **temperature scale** you chose from the **General Settings: Page 1** will be translated. What you do here is **select the temperature** at which the failsafe will kick in. For illustrative purposes, 150 °F was chosen (out of a possible range of 0-255). If and only if the temperature of the heatsink reaches the chosen value (The heatsink is the temperature being monitored, and is labeled "Hood Temp" on the **Main Screen**), then **ALL the LEDs will dim** to 25% of their respective scheduled values (or whatever value you choose, anywhere from 100% which of course would do nothing, all the way down to 0% which is completely off). The LEDs will continue to remain dimmed until the temperature falls 5° below the set temp, at which time they will resume their normal schedule. That is unless of course they reach that defined temp, then the whole cycle would repeat again. If this feature kicks in, you will see a little wrench on fire on your **Main Screen**:


Jarduino Aquarium Controller v.1.2

LED ARRAY

SUMP: 0%
Red: 4%
White: 29%
Blue: 38%
Royal: 35%
Ultra: 6%

LUNAR PHASE

First Quarter
56.1% of Full

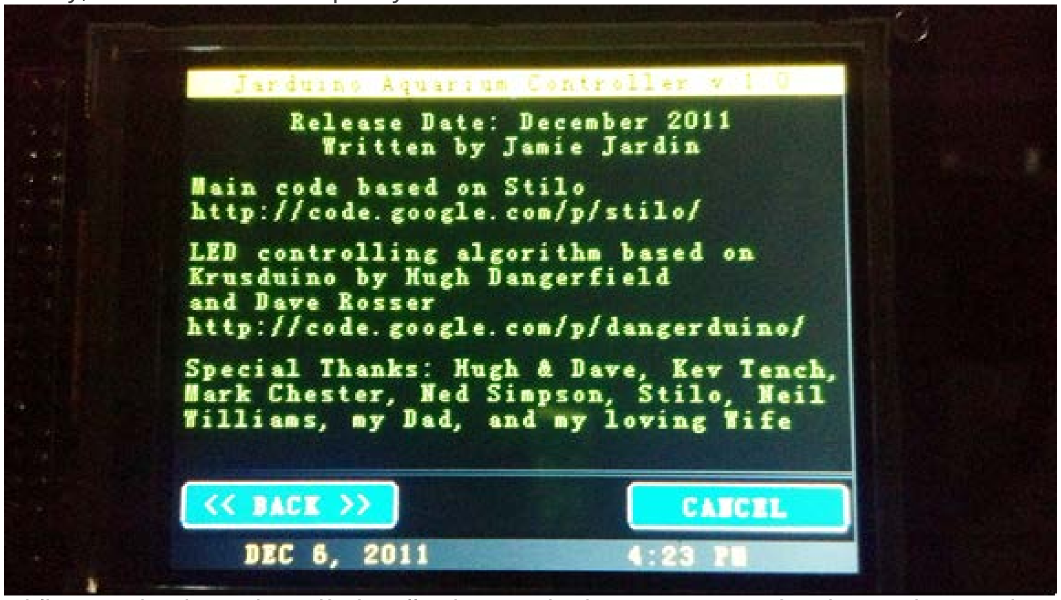
MONITORS & ALERTS

Water Temp: Error °F
Hood Temp: 85.8 °F
Sump Temp: Error °F

JUN 27, 2012 3:50 PM

The fire will turn to a blue color after the 5° cooling off period, at which time if you touch it, you can dismiss the "message."

Finally, the **About** button is purely informational:


Jarduino Aquarium Controller v 1.0

Release Date: December 2011
Written by Jamie Jardin

Main code based on Stilo
http://code.google.com/p/stilo/

LED controlling algorithm based on
Krusduino by Hugh Dangerfield
and Dave Rosser
http://code.google.com/p/dangerduino/

Special Thanks: Hugh & Dave, Kev Tench,
Mark Chester, Ned Simpson, Stilo, Neil
Williams, my Dad, and my loving Wife

DEC 6, 2011 4:23 PM

Obligatory thanks to the Wife for allowing me the huge amounts of patience, time, and understanding to complete this project!