*Projects and Stuff*

Pong.0

Project Log

Contents

[2012/04/21 3](#_Toc325213642)

[2012/04/22 3](#_Toc325213643)

[2012/04/24 3](#_Toc325213644)

[2012/04/25 3](#_Toc325213645)

[2012/04/28 4](#_Toc325213646)

[2012/04/29 5](#_Toc325213647)

[2012/05/01 6](#_Toc325213648)

[2012/05/02 6](#_Toc325213649)

[2012/05/04 8](#_Toc325213650)

[2012/05/05 9](#_Toc325213651)

[2012/05/06 10](#_Toc325213652)

[2012/05/19 11](#_Toc325213653)

# 2012/04/21

Restarted work on the project after a long hiatus (military deployments make engineering difficult).

# 2012/04/22

Began working BOM based on parametric information from Atmel and my past QTouch experience. BOM will focus on parts to be used in both the test board and final design.

BPS needs a better name. Going to have to brainstorm. Beer Pong Sense works well as a code name of sorts, like Microsoft's Longhorn, but it doesn't have much of a ring to it.

# 2012/04/24

Today began fleshing out the BOM based on known needs. For instance, I know I will require the following basic parts:

* **Battery Controller (currently looking for a rechargeable Li-Ion battery at around 5V and 2000mAh)**
* **AVR Microcontroller capable of QMatrix/QTouch and with a good amount of memory and peripherals for hacking**
* **Display drivers for the 47+ RBG LEDs used in this project**
* **A Voltage Regulator that can source enough current for all of the LEDs and other components**

I used the parametric search tools on Digikey to start finding components that would meet my needs. For instance, if I assume each LED (3x47=141 LEDs due to 3 LEDs per RGB) is 10mA, that’s 141\*0.01= 1.41A – A lot of current. Now in reality, we won’t allow all the LEDs to be on at once, and we’ll use PWN or other methods to adjust their brightness, so the end result will be much less than 1.41Amps, but we’ll still be using a lot of current.

Once these parts have been identified and I’ve gone through the datasheets to verify that everything meets my needs and will be compatible, I’ll move on toward starting the basic schematics in KiCad. I prefer KiCad over Cadsoft Eagle for several reasons, including the fact that KiCad is open source, and it doesn’t limit users to a specific board size.

# 2012/04/25

The plan right now is to use the Atmel AVR ATMEGA164/324/644 Series Microcontrollers along with Atmel’s QTouch Library (utilizing QMatrix technology) to sense the liquid in cups placed on the board, and then use three AS1107WL display drivers to light up LEDs corresponding to the location of each cup.

There are two possible configurations for using these display drivers

1. Each AS1107WL will control one color of all RGB LEDs

Pro: Should be pretty simple to program.

Con: Each RGB LED is driven from 3 different display drivers (one driver for each of the 3 LEDs in an RGB LED). This will likely greatly increase the complexity of PCB routing.

1. Utilize 6 of the 8 segments on each AS1107WL, controlling 2 sets of 8 RGB LEDs with each display driver.

Pro: Would likely make PCB routing much simpler, since each RGB LED is controlled from only one display driver. Likely pretty easy to program.

Con: Would have to add additional external pots to each drive line to adjust the current for the different colors of LEDs, increasing cost. Leaves 2 segments of each driver unutilized (though this could also prove to be a pro).

I didn’t realize Atmel Studio 6 Beta is now out. Once I start coding, I may use this, since (from what I’m reading) it already has the Touch libraries integrated. May make things easier, though Beta can be a bit scary. Still, I’m not in production with this project yet, so maybe taking the leap and trying Atmel Studio 6 is the way to go. I’m concerned, though, about the fact that version 6, like version 5, doesn’t support Linux. That makes it difficult for a large portion of the Open Hardware community to work on or adapt this project ☹.

# 2012/04/28

I’ve been communicating with various battery manufacturers in China and elsewhere via Alibaba. I’m working on getting quotes for Li-Ion and Li-Pol batteries. The specifications I’m aiming for:

Voltage: 6-7.2V

Capacity: 1500-2500mAh

Power Capability: around 300mA-500mA (or very roughly, about 0.2C)

Thickness: < 7mm (or roughly 0.27”)

So far the most promising quote has come from Shenzhen Puchuangyuan Technology Co. Ltd, who have answered all of my questions, and promise the ability to provide Li-Pol batteries thin enough to meet the needs of this project. They’ve also sent datasheets for each, though I have a couple questions I’m waiting for answers on. Samples will allow me to make real-world tests, rather than hoping that a similar batter will act the same in production, but the expedited shipping from China is expensive!

A couple other companies have also offered quotes, and I’m following up with them to find the best battery for my needs at a reasonable price.

# 2012/04/29

One interesting thing you’ll learn if you buy batteries from the manufacturer is that you can buy them with or without a charge management PCB. In the quotes I’ve received so far, the difference in cost with the PCB vice without it comes to about $1.50 or so. Looking at the charge management ICs on the market, in bulk, quality chips themselves would cost about $0.80 to $1.00. When you take into account the cost to manufacture the boards, but the additional components (only a few cents), and assemble the board, doing it yourself will cost around $2.50-$3.50 total per board.

Having gone through this pricing exercise, I think I’ll stick with the manufacturer’s board. If I trust the battery, I should be able to trust the board, and ideally, the manufacturer should know best which chip to use with their particular batteries.

I’ve received another battery quote with good details (from Zhejiang Nuociss New Energy Technology Co.,Ltd.). The cost per battery is about a dollar more than the quote from Shenzhen Puchuangyuan Technology Co. Ltd, but the cycle life is significantly higher, which I think should be well worth the cost. I’m building a simple spreadsheet as quotes come in, so that I can see more easily which offer is the best overall. I think it’s important to consider price, but to remember that price alone should not be of higher importance than overall quality.

I sometimes wonder if others develop things the same way as I do. I generally think up the project and begin to build it in my mind, using scratch paper to work out more complex problems and design the high-level schematics using black-boxes when I don’t know all the details yet). Once I’ve got all of the major components of the project figured out, I begin sourcing them. Then, once I’m confident there’s enough of each part on the market to manufacture effectively, I input the schematics in KiCad (or sometimes Cadsoft Eagle). From there, I begin to fill in the basic parts like resistors, determining the best part and values. And once that’s all don, it’s on to the PCB. I’ll back-edit as needed if there are changes to the design, but this can be exceedingly difficult if it’s a complex board, or ig there are specific routing requirements, like in this project. The traces to the capacitive sensors must follow very specific rules in order to obtain the best sensitivity and noise immunity. You can read more about these requirements in Atmel’s [QTAN0079: Buttons, Sliders and Wheels Touch Sensor Design Guide](http://www.atmel.com/products/touchsolutions/bsw/buttons_10.aspx?tab=documents), which covers all sorts of information about PCB and sensor layouts for Atmel’s QTouch and QMatrix technologies.

# 2012/05/01

Today I decided to switch LDO voltage regulators, from the Microchip MCP1826 to the SPX29150T-L-5-0/TR made by Exar Corporation for the reasons below.

**Simplicity** – The SPX29150T-L-5-0/TR is a simple fixed-voltage regulator. No need to mess with adjustments, calibrations, or lots of external components

**Flexibility** – The MCP1826 can only handle an input of up to 6V, but I may be using batteries at 7.2V. Making the decision to use a regulator that can handle an input of up to 16V gives me a lot more flexibility with which battery solution to use.

**Designed for devices like mine** – "These regulators are specifically designed for... fast transient response"

This new choice of regulator also has a relatively low drop out voltage (0.23V @ 750mA)

The price is a few cents higher, but I think the benefits of using this component will outweigh the difference in cost.

Sadly, this is about all I was able to accomplish today. Long day at my real job.

# 2012/05/02

Now that I’ve selected most of the main components, have written on paper several rough high-level drafts of the design, and have read enough to understand the requirements for each of the parts I’ve selected, I’m moving on to starting the schematic drawings in KiCad.

For non-standard parts (those not included in KiCad by default), I’m drawing them in KiCad’s Library Editor, based on the datasheet for the part. For those parts that are included in KiCad’s default libraries, I’ll open the part drawing in KiCad, and open the PDF Datasheet for the part, and compare pin-by-pin that the drawing is accurate. It’s tedious, but it takes less time than drawing the part myself from scratch (don’t reinvent the wheel), and it’s important to take the time now to be sure you have the right part and that it’s drawn correctly, so that you don’t waste valuable time later.

I tend not to use KiCad’s libraries for the actual pad drawings, because dimensions are more difficult to verify, but for schematic symbols it’s not really an issue.

I sort of work through this process the same way that I paint. I have an overall layout, and now I start working in the details here and there. Determining which additional components each major part requires for proper functionality, and adding them to the BOM. As I do so, I look for commonalities. If one part requires a .1uF capacitor on its output, and another part needs a decoupling capacitor of around .1uF to .22uF, for instance, I’ll select the same part for both situations. This accomplishes two important things: it reduces price, because items are cheaper in larger quantities, and it simplifies design and manufacture of the project by reducing the overall number of different parts.

I also try to stick with common part sizes. For instance, in this project, most of the discreet components (resistors, capacitors, etc) are sized 1206 in American (0.126" x 0.063"), which is 3216 in Metric (3.20mm x 1.60mm). Keeping things the same size saves time on the PCB, because it’s easier to work with several components all of the same pad size.

Today I really got to work on the schematics. I managed to get the display and sensor schematic sheets mostly completed. The main sheet is still in progress. I’ll have to figure out which microcontroller pins are required for the small QMatrix setup I’m working on (a 4-by-1 array of sensors - the final project will be an 8-by-3 matrix), and which will be optimal for the display driver (the final project will have 3 AS1107 ICs, but the test board only needs 1).

Overall it has been a pretty productive evening.

(Note to self: I think something’s wrong with the display matrix, but it’s too late now to think. Check in the morning, and report on findings)

Late Entry: Found the problem in the Matrix and corrected it. Just had things linked up incorrectly.

# 2012/05/04

Worked on fleshing out the schematics this evening. Some of the specifics:

* Added the ISP headers for programming the AVR
* Added input filtering for VCC and AVCC on the AVR (AS1107 input filtering was completed previously)
* Added reset functionality for the AVR
* Began labeling Nets for clarity

The one factor that slows down this portion of the project is making sure to understand the datasheets. Every datasheet has very important information that you cannot just skip over. Determining which info is important in the schematic step, which info is important for programming the project, and which info isn’t relevant to the project is something that takes time. If you’re just beginning, or are unfamiliar with a brand/family of devices, it’s important to read through the whole datasheet. It’ll take time, but it’s so easy to make mistakes, and mistakes are costly. Better to take the time to do it right the first time.

All that being said, from the list of tasks completed tonight (over about 3 hours), it doesn’t look like much, but I’ve been poring over datasheets, Atmel’s Application Notes (particularly AVR042 which has a bunch of useful little tips about setting up your inputs, reset, ISP, etc).

The Application Note above mentions using series resistors on lines shared by the ISP (used to program the microcontroller) and SPI (Serial Peripheral Interface – used by the microcontroller to control and communicate with other devices). Instead of using these serial resistors on the lines that the AS1107 shares with the microcontroller, I opted instead to place an external pull-up resistor on the Slave Select (or Chip-Select Input) of the AS1107. What this means is that unless I specifically set the Slave Select (SS) pin on the AVR to low, the Slave Select pin of the AS1107 will be held high, which tells the AS1107 that it’s not currently selected. This effectively disables the chip.

I’m going to start including a PDF file with the schematic images (and eventually the PCB layout) with each push, because that will make it much easier to see changed visually. I probably should have done this from the beginning, and I will do so on future projects.

One thing to note for this test board is that because we’re using only one AS1107, the colors for the LEDs are not going to be tuned correctly. With the final board there will be one AS1107 per LED color, allowing good fine-tuning of the colors. The main point here is to practice using the chip and to see how the capacitors on the sink lines (which should improve the stability of the capacitive sensors) affect operation of the AS1107.

Note to self: Perhaps I should go ahead and spend the extra $10 on a couple more AS1107’s for this test board to get the most out of the testing. I’ll document tomorrow which decision was made.

# 2012/05/05

I decided that yes, spending the money now on three AS1107s, vice one, will be beneficial to the success of this project. It will give me a better idea how well the final colors of the LEDs will look and it will help me work out the best way to deal with the complexity of multiple daisy-chained SPI chips.

I am working on figuring out the best value for the bypass capacitors for the LED matrix. Atmel’s Touch Sensors Design Guide recommends a typical value of 1nF for bypassing the LEDs, but then notes that this is a "Non critical value. The idea is to simply provide a constant low-impedance path as seen by the sensor, on both ends of the LED. Low means less than 1 kΩ at 100 kHz."

But if we run some simple calculations, a 1nF capacitor at 100kHz has an impedance of about 1.5kΩ.

Since we want low impedance, and we also want to keep parts as standard as possible throughout the project, we’ll go with the 0.1uF capacitors used elsewhere, which result in the following calculation:

Hopefully I’m not misunderstanding something here, but my calculations seem to make sense in this context. Could Atmel’s sensor guide be incorrect on this matter? Maybe they meant 1uF, which would result in an impedance of about 1.5 Ω.

One concern I have about my current design is that LED brightness may not be high enough, due to the limited source current provided by the AS1107. I may need to pick a different chip or use external sources (FETs/Transistors) driven by the AS1107, which can sink a maximum of 500mA each. I won’t really know for sure until I build and test the prototype. I guess that’s half the fun. What surprises, good and bad, will I find? Even the ‘bad’ ones can turn out to be an amazing learning opportunity.

I continued making minor changes and improvements this evening. I found the best resistor for the pull-ups in the project. Basically, I knew I needed a resistor between 4.7k and 10k, so I went on DigiKey, selected all the surface mount resistors with those values able to handle 1/8W or above. Since I need to per board, and I’m currently aiming to produce 500 boards, I sorted by price for quantity 1,000. That showed me a 10k resistor with over 1 million in stock for about 4/10 of a cent each.

I also added a couple more LEDs to the matrix to make the prototype more realistic (well, it tiny bit). 14 RGB LEDs in all for the prototype, plus a power LED and an indicator LED on the bottom side of the board.

Last, I found a nice little tool today for formatting a Bill of Materials (BOM) generated from KiCad into a more useable format. [KiCAD Excel Add-In](http://sourceforge.net/projects/kicadexceladdin/).

# 2012/05/06

I’m pretty confident about which battery supplier I’ll be using (Shenzhen Puchuangyuan Technology Co. Ltd). They’ve done an excellent job of customer support, and have answered a myriad of questions. I’m waiting right now for a reply to my latest batch of questions. In particular, I need to know a little more about the circuit they’re supplying with the batteries for charge management. The datasheet they sent was a bit ambiguous about what the optimal and minimum charge voltages are. It included the maximum, but I need these other two values for determining the best methods of charging. For instance, if it requires at least 7 volts, maybe I can use a boost regulator with USB input to charge the battery (though this will probably be a slow method of charging such a big battery and may require more current than USB can provide. I’ll have the check on this). The other option, pretty much by default, is a standard DC jack. Ideally, I’d like to include both options. Flexibility is always a good thing.

# 2012/05/19

After completing the hardware design for the Chameleon Project, I’m moving back to working on this while waiting on the prototype PCBs.

I’ve renamed this project from Beer Pong Sense, which had become sort of a placeholder, to its new name, Pong.0. Because of this change, I’ve also renamed the repository and several files.

I installed Atmel Studio 6 and used the QTouch Project Builder to generate the skeleton for this project. It almost feels like cheating. You input how many sensors, which microcontroller you’ll be using, and which pins your using for each channel, and it creates a ready-to-go project. This is very convenient, and will cut probably cut a full day or two of development time, since I don’t have to manually fill all this out in the source files. Wizards can be nice sometimes.