Breadboard to Printed Circuit Board

Josh Johnson

10/6/2019

Motivation

- So you have a circuit working on a breadboard, however you need to move it to a PCB.
- You might also have an idea for a design which can't be breadboarded, and want your PCB work the first time.
- You might need it to be battery powered, have a certain form factor, or be lower cost.
- Maybe you are producing them in quantity, or are taking the level of integration up a notch.

How can you achieve these goals?

Project Files: `github.com/joshajohnson/CBRhardware`

Key Considerations

- What quantity is being produced?
- How much time do I have to spend on the project?
- What size / type of components am I willing to work with?
- I have an development / breakout board, how do I replace it with discrete parts?
- How am I going to program it?
- How will I communicate with it?
- How do I want to power it?
- ~~How will Josh debug it when he makes mistakes?~~
- Mechanical design considerations.

Key Considerations

What quantity is being produced?

- Low volume
  - Trade off complexity for cost. Examples are prebuilt modules, commonly used parts, flying wires.
- Medium volume
  - Cost starts to become a consideration. See if a small amount of R&D can lower the bill of materials.
  - Assembly time - small changes (single sided load, surface mount parts) multiply out to save a lot of time.
- Large volume
  - I'm not qualified to talk to this, but more time can be spent in R&D if it means lower BOM and assembly. e.g. Capacitative buttons vs mechanical, code optimisation vs larger microprocessor.

Key Considerations

How much time to I have to spend on the project?

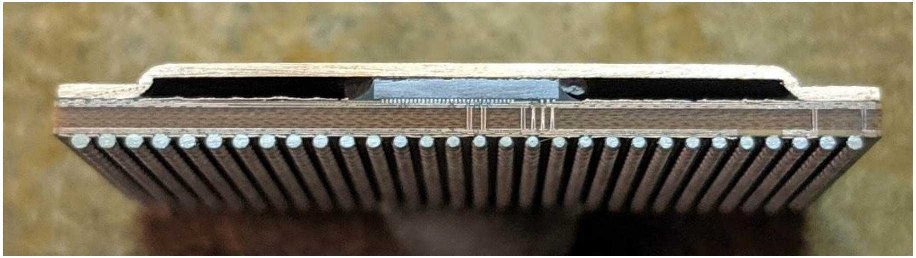
- Similar considerations to quantity, if short on time, look for more integrated solutions.
    - Display on PCB with 0.1" headers vs display + supporting passives.
    - Arduino Nano / Pro Mini vs DIY ATmega 328p.
    - Power supply brick / LIPO charge module vs rolling your own.
    - Steal designs from Adafruit / Sparkfun.

Key Considerations

What size / type of components am I willing to work with?

- Through hole vs surface mount
  - With correct tools, SMT is quicker and easier than THT.
  - THT is easier to resolve bugs, but takes up more space and some parts are SMT only.
- Component size
  - What sizes are parts available in? e.g. DIP vs SOIC vs TQFP vs QFN.
  - Who is assembling them? e.g. if first timers, go with easier parts.
  - With time, you will figure out what size parts you are happy working with. I personally suggest use / suggest 0603 if you have steady hands / good eyesight, 0805/1206 otherwise.
- Space constraints
  - Do you have to use small parts due to form factor / performance reasons? e.g. 0402 result in less impedance mismatch on traces, however you can use 0603 decoupling caps with little downside.

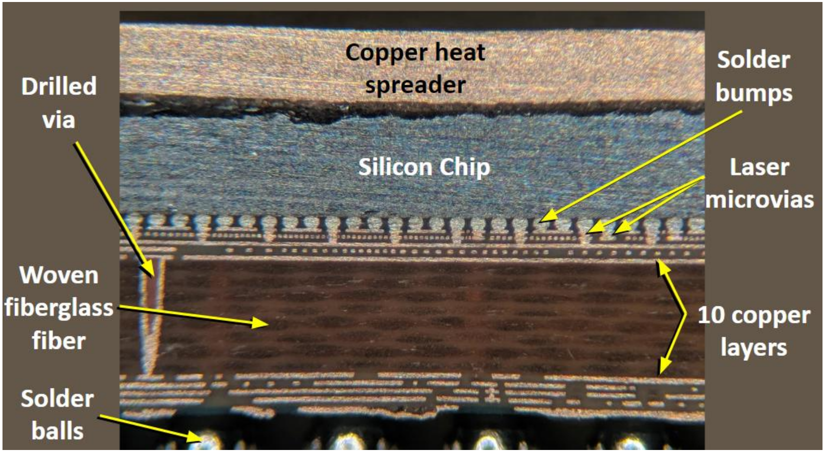
Part size / complexity reduction example



- Flip chip, 0.15mm bump spacing, 1mm BGA, 10 layers, blind and buried microvias, $<$ 2/2 mil trace/space
- AFAIK no one in Australia can manufacture or assemble the board.

From @TubeTimeUS on twitter

Part size / complexity reduction example



Key Considerations

I have an development / breakout board, how do I replace it with discrete parts?

- Reference designs
  - Find who makes the board you are using and look for a schematic.
  - If unmarked, probably a Sparkfun / Adafruit clone - look on their website.
- Datasheet / application notes
  - Google is your friend.
  - Datasheets typically have a recommended schematic 1/3 down.
  - Application notes go into more detail regarding function of the device and different configurations.
  - Manufacturer probably has an evaluation board with schematics available.
- Substitutions
  - Many parts can be substituted for a similar component. e.g. mosfets, transistors, voltage regulators.
  - If looking for a substitute, Texas Instruments, and Analog Devices have great datasheets and app notes.

Key Considerations

How am I going to program it?

- Does my device have a bootloader?
  - 'Arduino compatible' devices all have bootloaders which new ICs don't.
  - Flashing bootloaders may require UART, SPI, SWD, JTAG depending on chip, may also require dedicated programming header.
- UART / Serial
  - Many ICs will have a DFU pin which if held low on startup, will load code from a serial port.
  - STM32, ESP8266 / ESP32, my reflow oven (freescale?), many others are examples of this.
- Serial Peripheral Interface
  - ATmega / ATtiny microprocessors are flashed via SPI.
  - Programmer also required to set fuse bits.
- Single Wire Debug
  - ARM Cortex cores - two pins + power allow programming and debugging.

Key Considerations

How will I communicate with it?

- Serial
  - Serial is the easiest method, but will require USB <-> Serial converter.
  - Can place converter on board with USB connector, or leave header exposed and use dedicated cable.
  - Flashing with a serial bootloader is common, e.g. Arduino series.
  - May require reset circuit for serial programming with bootloader.
- USB
  - USB bootloaders allow for updating over a USB cable, no other parts required.
  - If you device has native USB, setting it up as a virtual COM port makes computer think it is a serial port.
  - Can also do HID emulation to pretend to be a keyboard/mouse.
- Wireless
  - Wireless communications / firmware updating is possible, serial over bluetooth is common for communications.

Key Considerations

How will I power it?

- Power supply methods
  - USB (5V)
  - Battery (Coin Cell, AA, LIPO)
  - 9/12 VDC
- Voltage regulation
  - Low current OR small input -> output difference: linear regulator.
  - High current OR large input -> output difference OR output > input: switchmode supply.
- Rechargeable batteries.
  - Ensure each cell has charge protection built in.
  - Use a charge controller IC.
  - If using LIPO with circuit at 3V3, can get away with a linear regulator.
  - If 5V is required, boost converter will need be used.
  - Can connect directly to battery as well e.g. Neopixel VCC.

Key Considerations

How will I debug it?

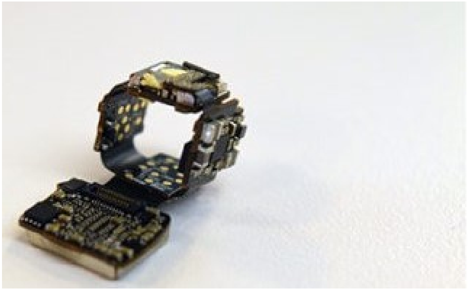
- Power
  - Status LEDs on power rails.
  - Using 1V8? Low Vgs N channel Mosfet on low side of LED.
  - Test points (and ground pads) for every voltage rail.
  - Allow board to be powered from a bench supply, through Vin connection.
- Microcontroller
  - If using a microcontroller / toolchain with debug support, have a debug header on the board.
  - Serial (UART) comms are great to have broken out, even if using USB.
- On Board signals
  - Test points, test points, test points.
  - 1 mm diameter SMD pads don't take up much room, can be easily soldered onto.
  - SMD test points (metal loops) are a great option for ground.
  - Test points are less useful without plenty of available grounds.

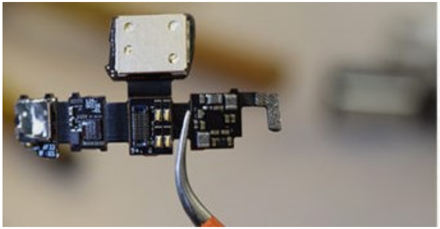
Key Considerations

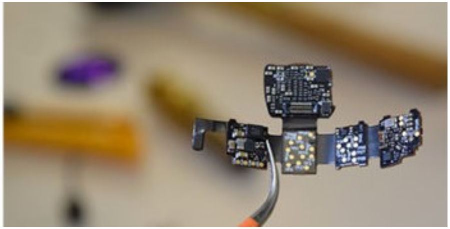
Mechanical Considerations

- Board size / shape
  - Does the board need to fit into a pre defined area?
  - Do components (connectors, displays, buttons, LEDs) need to be located in a certain position or on a certain side?
- Enclosure choice
  - Does my board need an enclosure?
  - If so, do I want to make my own or purchase a prebuilt enclosure?
  - Plastic box vs aluminium extrusion vs acrylic vs 3D printed.
- Mechanical Modelling
  - Import .STEP into ECAD as a 3D model for a footprint.
  - Export .STEP from ECAD into MCAD (I use KiCad and Fusion360).
  - Draw enclosure around board OR check board fits in designed enclosure.
  - Can export .DXF from MCAD into ECAD, and design board around that.

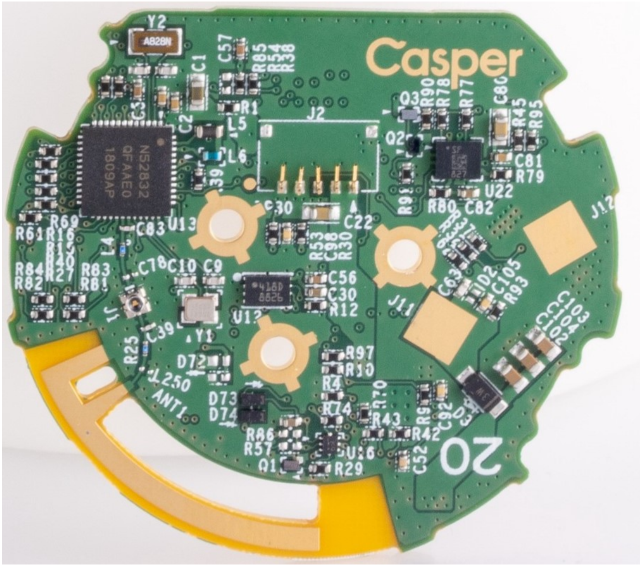
Snapchat Glasses







Casper Night Light



Casper Night Light

</talk>

Say Hello!
BSidesCbr Slack: josh
Twitter: @_joshajohnson
Email: josh@joshajohnson.com

Project Files: `github.com/joshajohnson/CBRhardware`