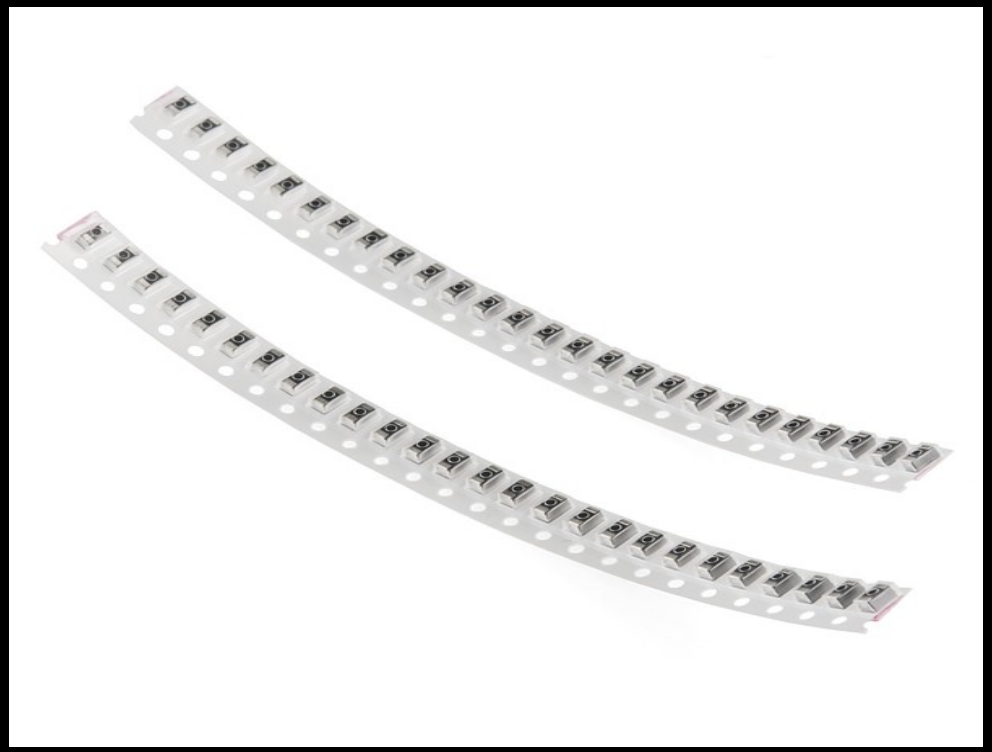
Pick and Place Robotics in ROS

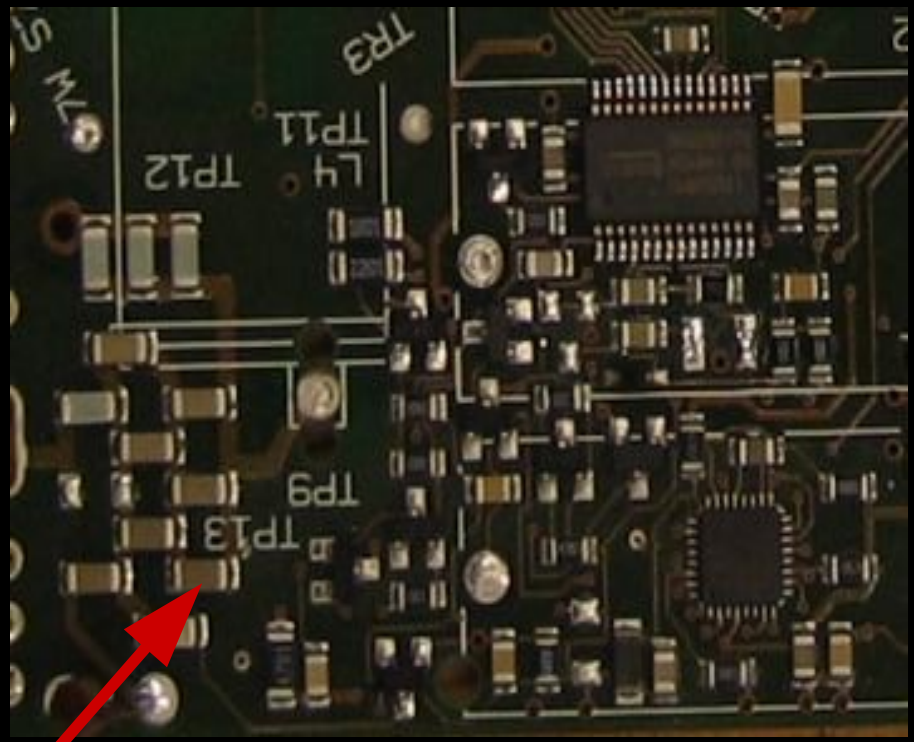
Katherine Scott
Co-Founder Tempo Automation
katherine@tempoautomation.com
@kscottz

Overview

- What's the problem
- What we're doing
- How it works
- Lessons from manufacturing world
- Pain points

Surface Mount Device Printed Circuit Board (SMD PCB)

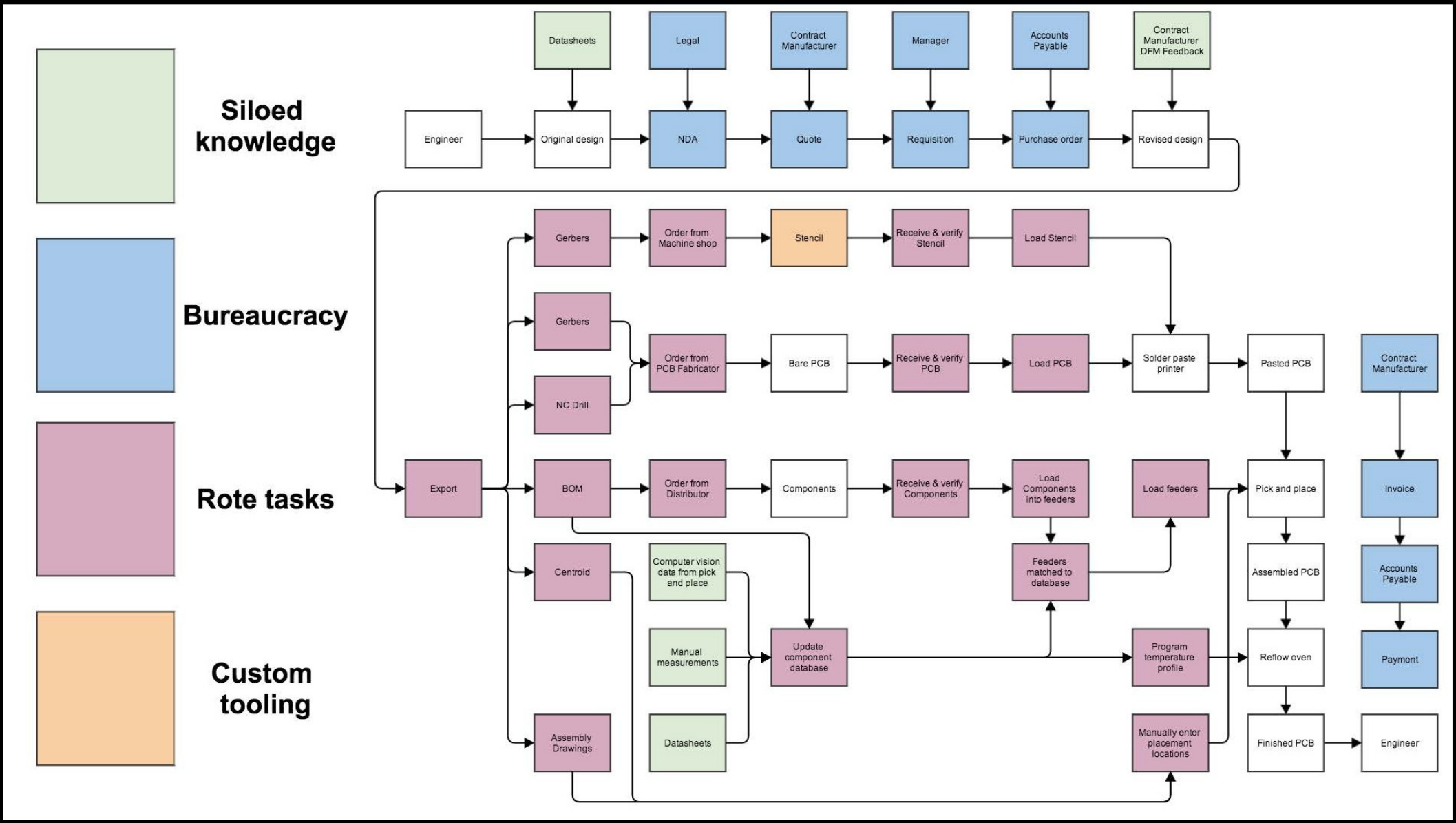




About the size of a grain of rice

How people think these things are made...

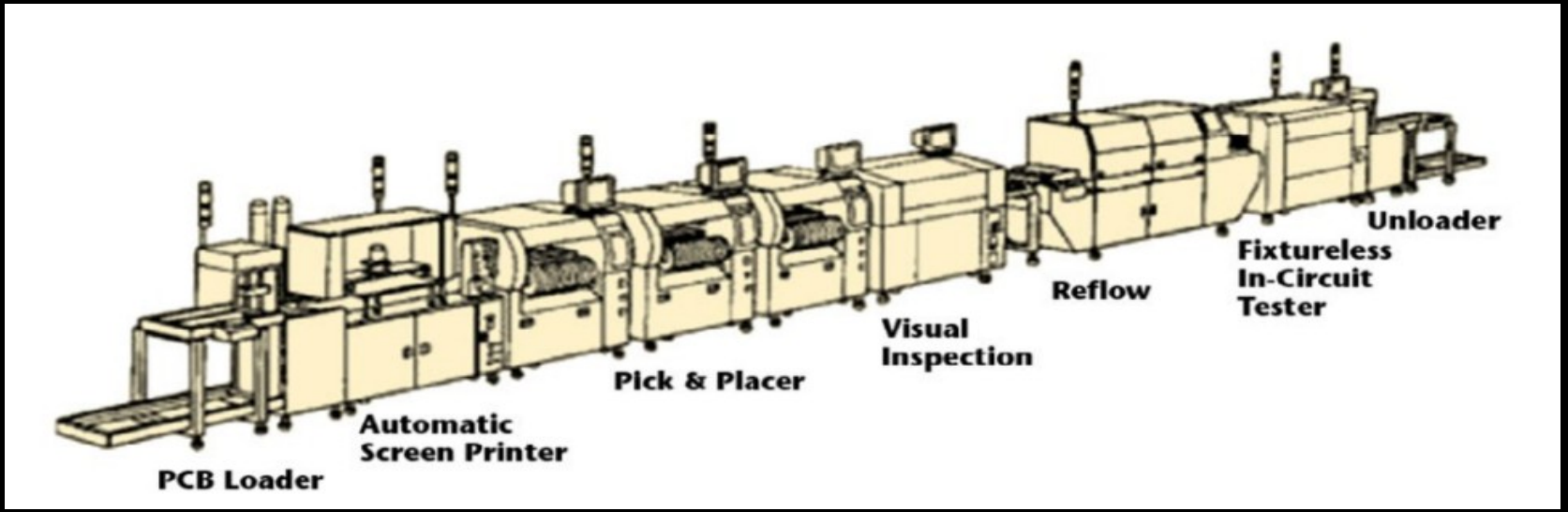
How it really happens



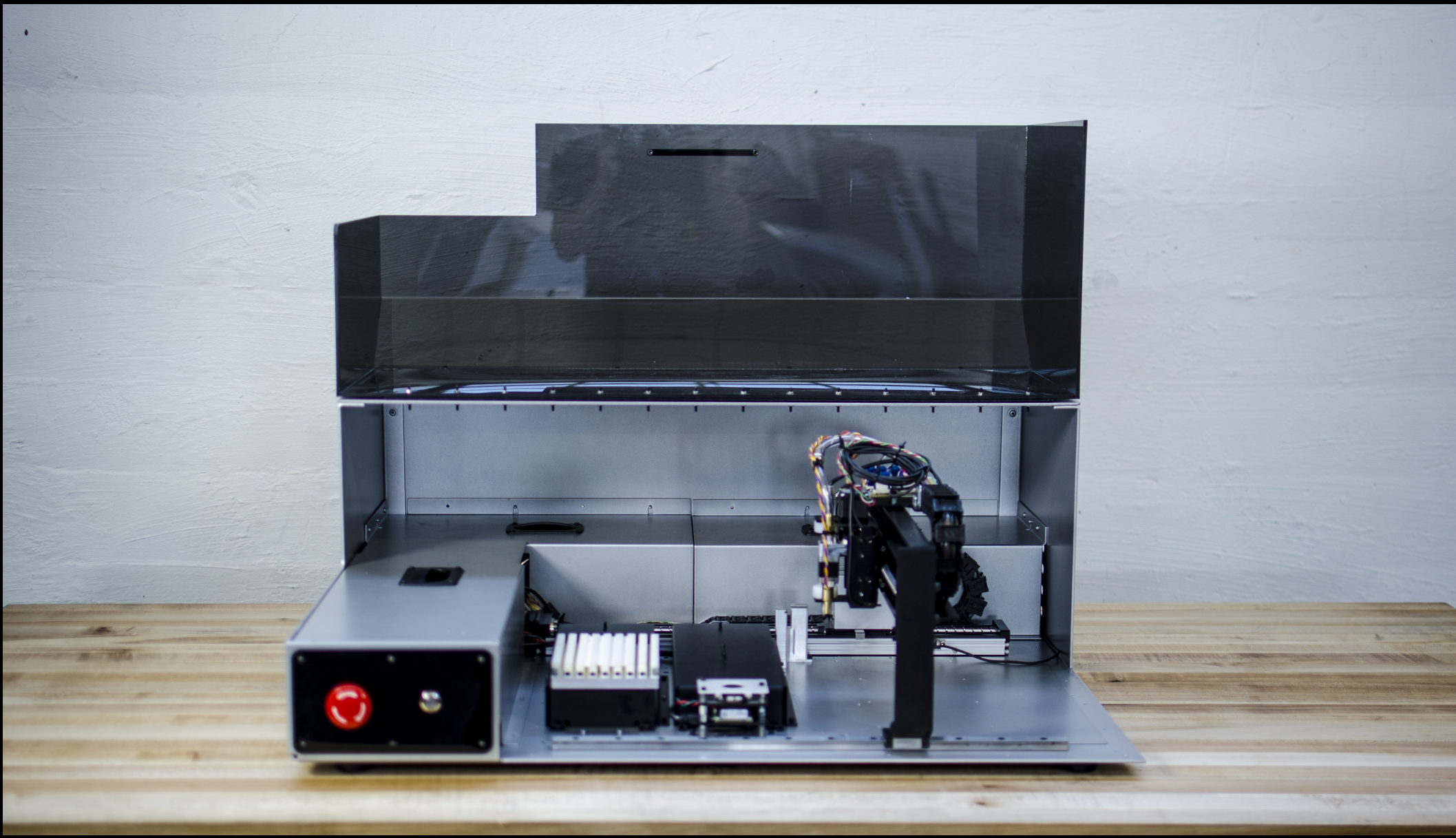
Hand Prototyping

- Error prone
- “Like driving the 101 at rush hour”
- Time consuming and expensive

Contract Manufacturer



- Slow and/or expensive
- Research and contracting overhead
- Design file massaging by engineers



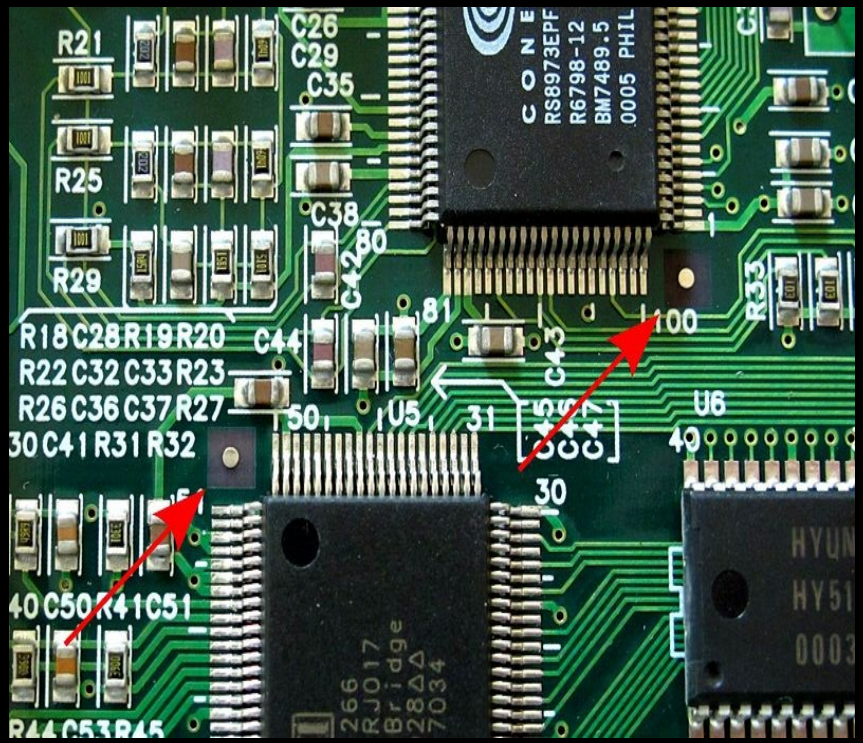
What it is?

- XYZ Gantry with dedicated motion controller
- Headless uITX PC running ROS/Ubuntu
- End Effectors: dual pick tips
- Two cameras for feedback
- No feeder mechanisms – cut tape

A bit of background

- Working on this a little over a year.
- First three iterations all in Meteor/JS/Python
- In June we said, this robot needs to be smarter.
- Now pure python and C++ in ROS

How does it work?

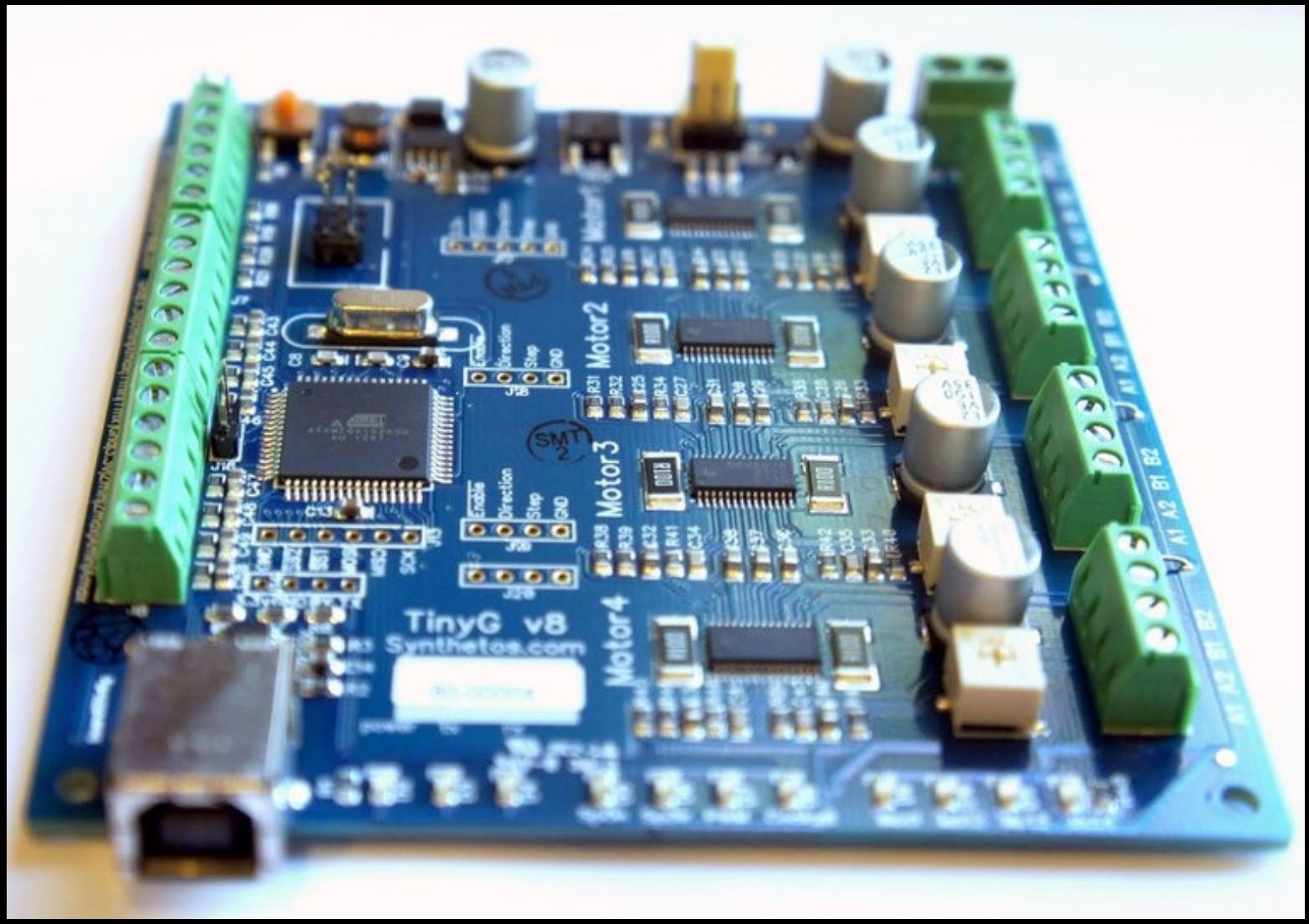


- Export your design file.
- Robot looks for parts and board fiducials
- Validate the design.
- Picks up each part.
- Inspects it to correct position orientation
- Place it.

Movie Time


ON
OFF
START
STOP
OK

TinyG – Motion Controller



OPEN HARDWARE AND SOFTWARE!!!

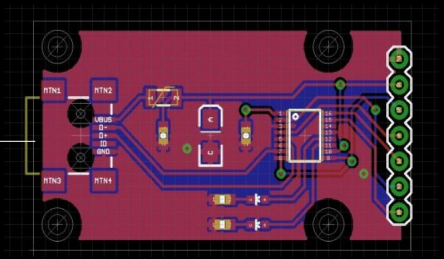
TinyG

- Talks over serial over USB
- Speaks gCode and mCode wrapped in json
- Does real time motion control on the fly
- Allows for control of velocity, jerk, etc.

How we use TinyG

- Implement a subset of gcode and mcodes in a python wrapper
- All gcode commands are synchronous, KISS
- Robot runs g codes as ROS service.
- Robot publishes state updates a 50Hz

How it works



Front End Control

Build Utilities

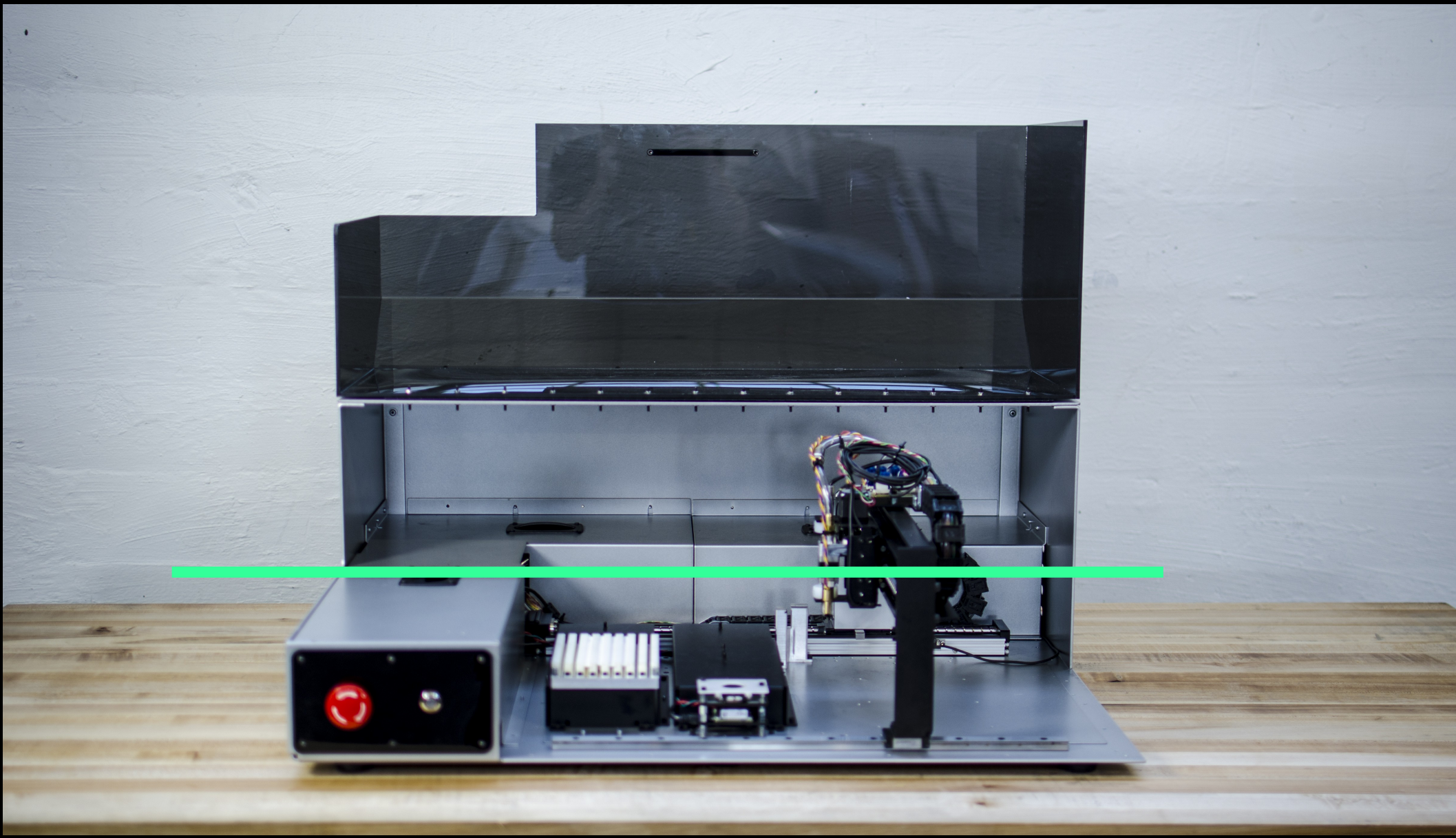
Boss Node

- Manager
- Action Server
  - Scan PCB
  - Scan Parts
  - Pick Component
  - Inspect Component
  - Place Component
  - AOI

Bottom Camera Node

Top Camera None

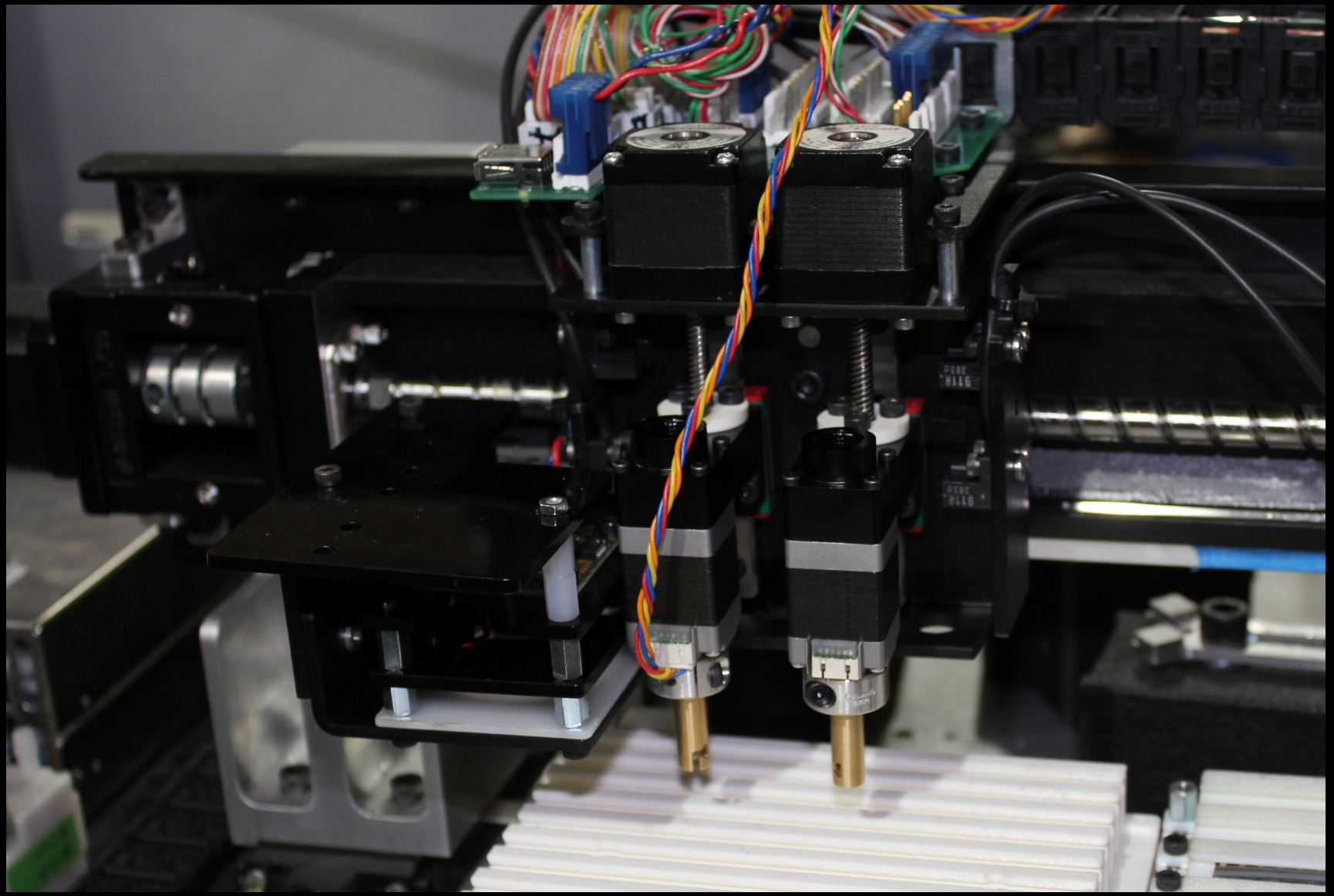
TinyG Node



Computer Vision

- Custom message with part/tape description
- Includes tape type, part dimensions, package, etc
- Lots of multi-level detection. E.g. find tape “wells” then part.
- All CV is keyed on part data for top and bottom with catch all generics.
- Mostly written in pure python so we can pop data into and out of ipython
- Strategy allows us to create a database of custom part data.

End Effectors



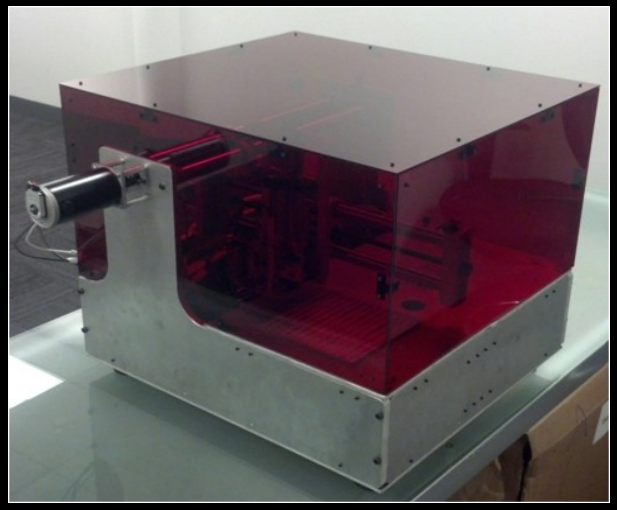
User Interface

- Original robot was all web based and headless
- THIS IS THE WAY FORWARD
- Linux, OSX, windows, whatever...
- Looking forward to really working out robot web tools.

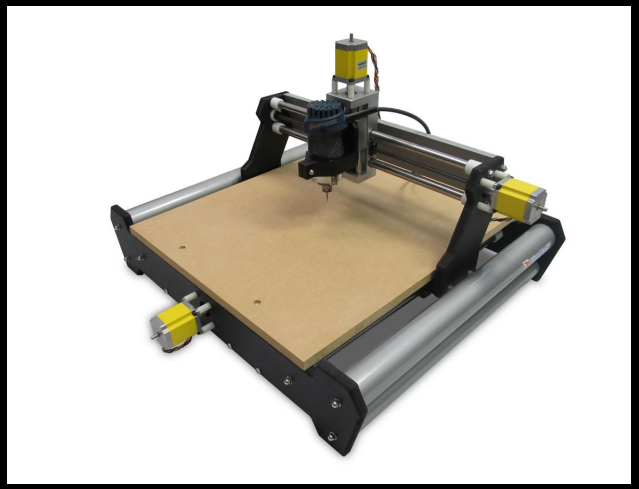
This is a process – lots of lessons

Mark I

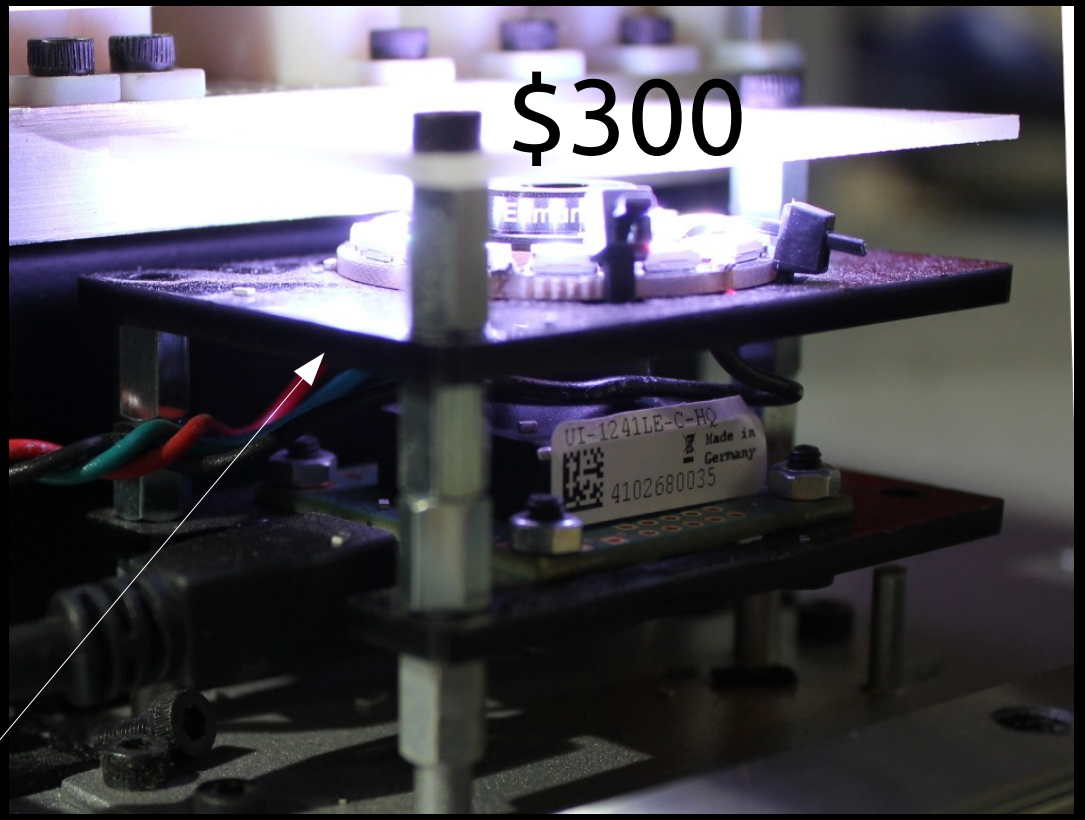
Mark II



Mark III



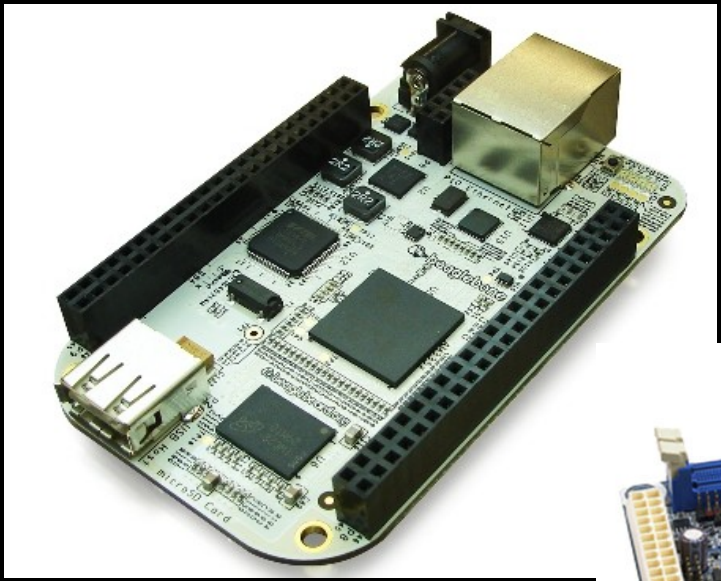
Engineer Cost Last: Cameras



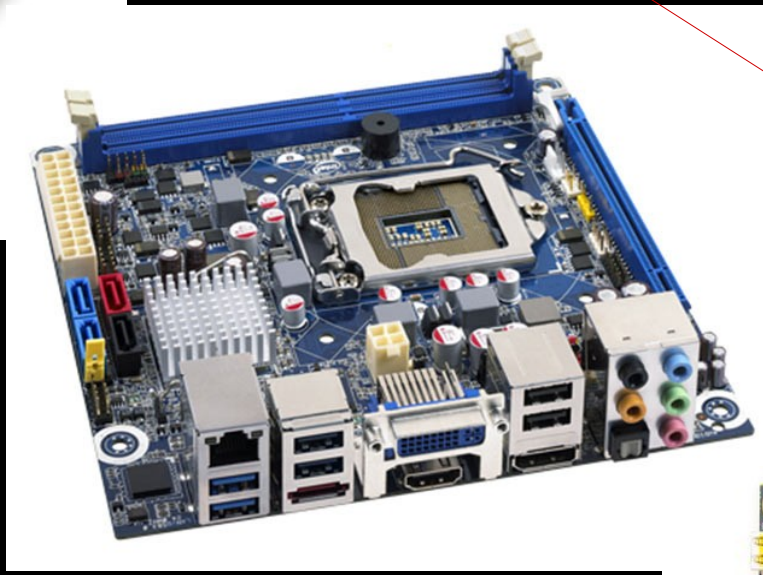
Cheap DIY ring light.

Engineer Cost Last: Computing

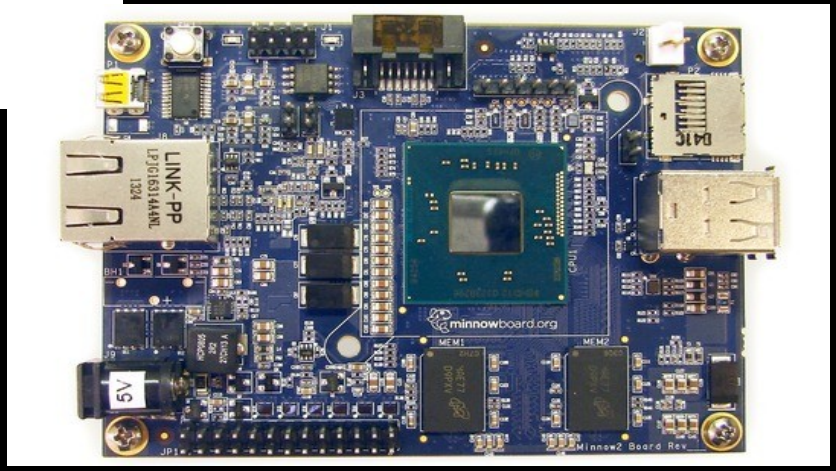
BeagleBone Black



MiniITX

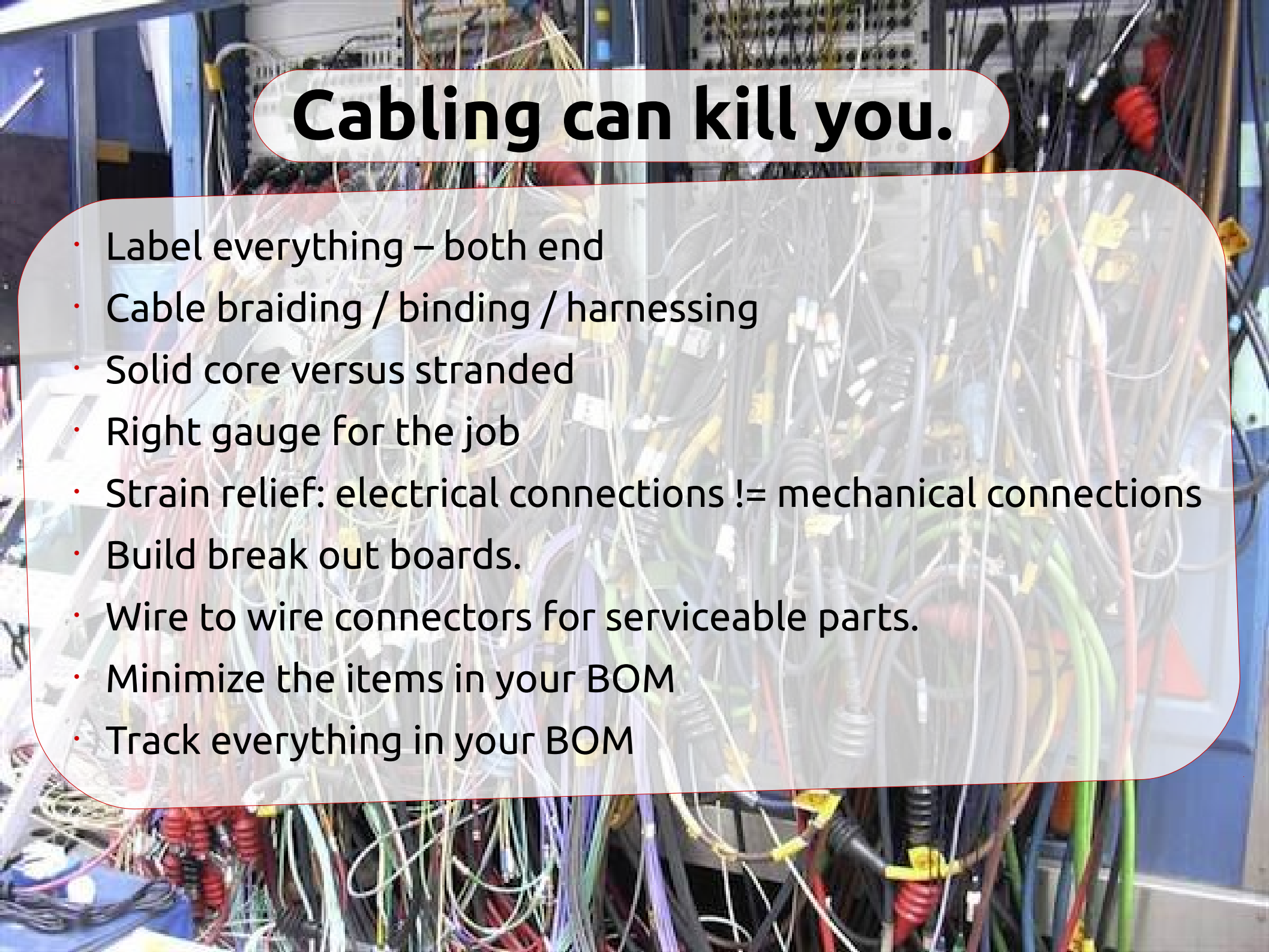


Minnow Board Max



Lesson: we have a shop for that

- Network to find people and places for every task
  - Sheet Metal
  - 3D Printing
  - Powder Coat
  - Machining
  - Acrylic / Plexi
  - Same day parts
  - General Labor
  - PCB / CM
- Track lead time, suggestions, contact, price.



Cabling can kill you.

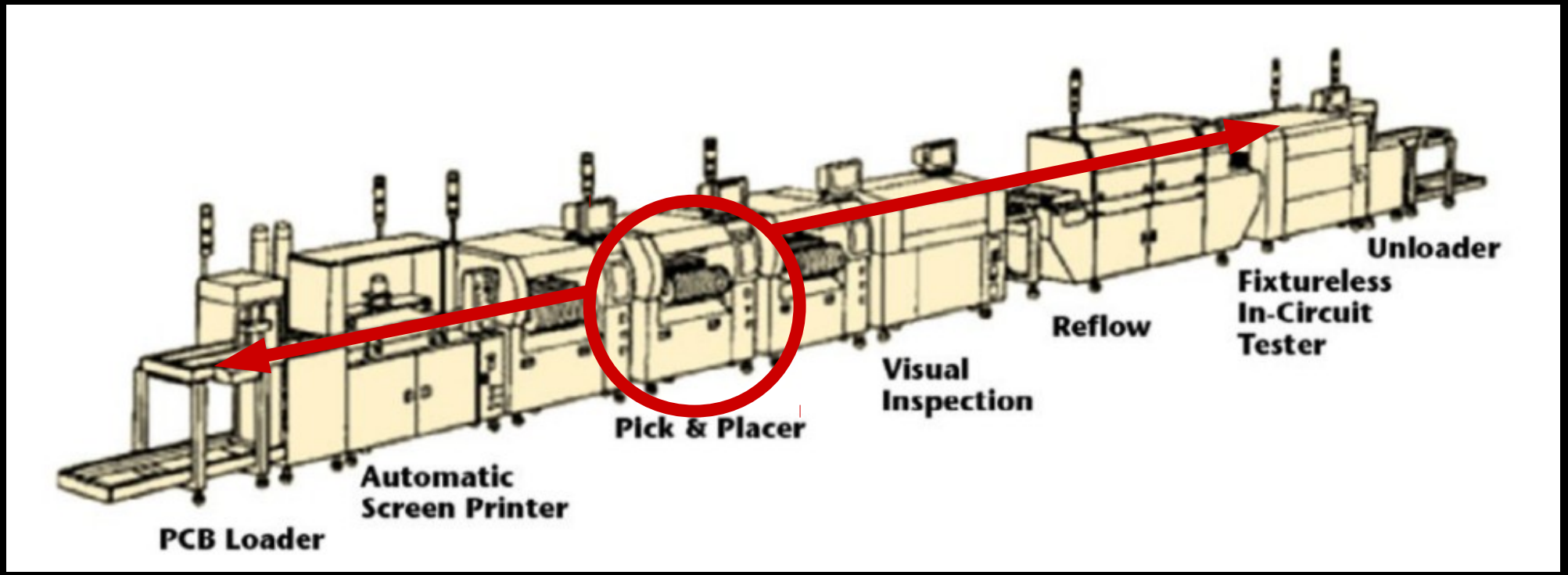
- Label everything – both end
- Cable braiding / binding / harnessing
- Solid core versus stranded
- Right gauge for the job
- Strain relief: electrical connections != mechanical connections
- Build break out boards.
- Wire to wire connectors for serviceable parts.
- Minimize the items in your BOM
- Track everything in your BOM

Lesson: Play Well with Others

Altium

- Building the robot is easy
- Getting data into the robot is hard.
  - CAD standardization is a mess

Lesson: Modularity Reduces Scope



Current Pain Points for Manufacturing

- Getting data out of CAD
  - Yes there is open source...
  - But industry doesn't use open source.
- This is not just EECAD
- Similarly industry PLC buses are lacking (modbus,profibus)
- Minimal Standardization
- Huge supply of useful data.

Current Pain Point – Data

- Bagging in ROS is incredible.
- But...
- Bagging is not a database for web persistence, QC, or learning over LONG times.
- Database would be swell
  - Archive information, particularly binary
  - Search through the binary and data
  - Play with data in iPython
  - Spin up roscore squirt data to a node / simulator

Industrial 3D at Micro Scale

- Depth sensors are great, and our application could benefit from them.
- Nothing works at the sub mm level.
- Need high accuracy stereo
- Structured light / laser scanning.

Plugs

- Come help us
- Prototyping ROS hardware?