

Outline of Lecture 4, 28 September 2020

- Recap: you are here (on the SDP schedule)
- Ordering parts instructions
- Reserving lab time instructions
- How to meet and get project help from Dr. Charles Malloch
- How to design and test iteratively. How to debug hardware/software.
- Quick intro to GitHub
- Clarification of the policy on Single Board Computers and Breakout Boards in SDP
- Time for questions

An Important Reminder

PDR: Complete. Congratulations.

50 days until MDR Week

About 200 days until SDP21 demo days

(We are in the process of reworking the Spring schedule)

November 2020						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2 Lecture 6	3	4	5	6	7
8	9 Check-in #4	10 Check-in #4	11 Check-in #4	12 Check-in #4	13	14
15	16 MDR	17 MDR	18 MDR	19 MDR	20 MDR	21
22	23	24	25	26	27 Reading Day	28 Reading Day
29	30 Exams begin					

Ordering parts: at a glance

- Review the list of approved vendors*
- Create your cart on the approved vendor website
- If the website supports sharing the cart, use that feature
Otherwise, create a order using the template order form
- Share the cart or form with **@Keith Shimeld**, in the Slack channel named **“teamxy-purchasing”** that has been created for your team
- Keith Shimeld will share confirmation/tracking numbers and let you know when items are ready for pickup in Marcus 8

*If you need something that is not sold through any pre-approved vendor website, send **@Shira Epstein** a message on the same channel **“teamxy-purchasing”** & explain why this item is unique/cannot be obtained from pre-approved vendors

Ordering parts: pre-approved vendor list

Full instructions and pre-approved vendor list: (updating version)

<https://docs.google.com/document/d/1h-pfmq0214RxX6KacgbE3UFZx9wh3w3lxy>

TinyURL: tinyurl.com/sdp21ven

(same document, shorter URL):

Check out the document!

Digikey

Adafruit

Pololu

...and many more!

Ordering parts: using a vendor w/o cart sharing

Full link: <https://docs.google.com/spreadsheets/d/1t86JKOeYvCBLsqKZzBIVbXVyrhwt4txC9EVsX6DDQxw/>

TinyURL: tinyurl.com/sdp21ord

For pre-approved vendors that DO NOT support cart/wishlist sharing

1. **Make a copy of the template above.**
2. Fill out one sheet per vendor. Consider
 - Copy the Line Total box equations to auto-compute line totals
 - Order total is already automatically computed.
 - Enter any relevant info to help Keith choose the shipping option that works for you.
 - Consider shipping cost and shipping time.
3. **File > Download > Microsoft Excel (.xlsx)**
4. **Post that file to your purchasing channel, “teamxy-purchasing,” with tag @Keith Shimeld**

Logistics: Remote Students

- Teams work to distribute tasks within the team
- You may work on a part of the project that involves needing parts and equipment
- Think about the logistics, what you would need, and then reach out to Shira to come up with a plan



Equipment in SDP Lab

Oscilloscope

Digital Multimeter

Power supply

Function generator

Breadboard wire, hand tools

Picking up the team toolkit

Breadboard, breadboard power supply, handheld digital multimeter, 2 sets wire strippers/cutters, 6 pc tweezer set, 8 piece screwdriver set, 4 pc pliers set.

There is wire in SDP Lab, help yourself.

One toolkit per team. Toolkit lives in your locker (you will get a combination).

A team member will sign out the toolkit and locker access from Fran in Marcus 9
Between 9AM and 3PM, Monday through Friday

Picking up your face shield

Students should go to **Marcus 8** between **9AM and 3PM, Monday through Friday**

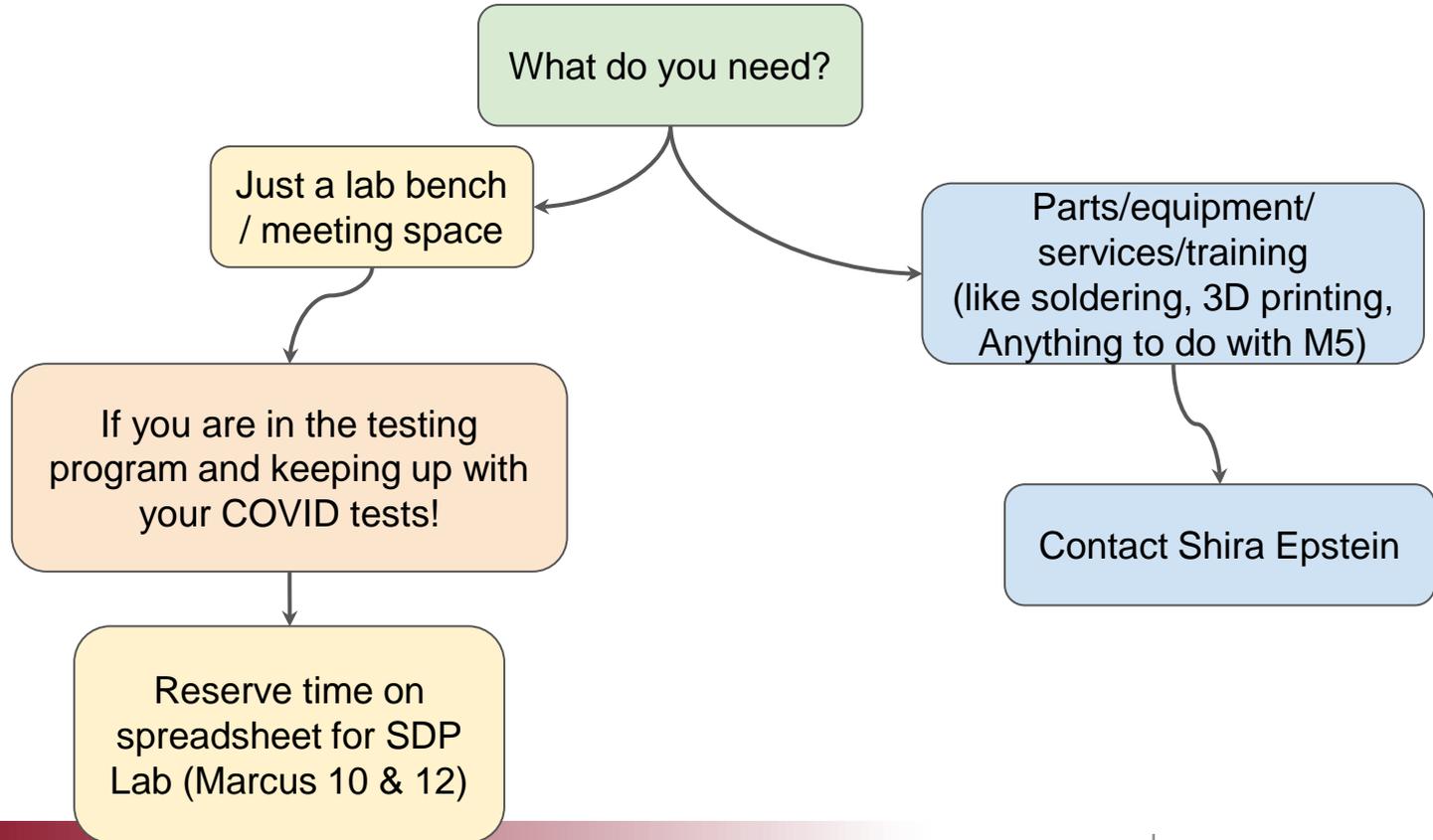
Keith Shimeld's office is in Marcus 8A; students will see Keith to obtain a face shield and use the computer there to enter their info into the face shield checkout spreadsheet.

Students receive one face shield for the semester and must wear it in all laboratory settings along with their face mask.

Signage with instructions is on the door of 8C and on the computer to right of the door.



Reserving lab time: at a glance

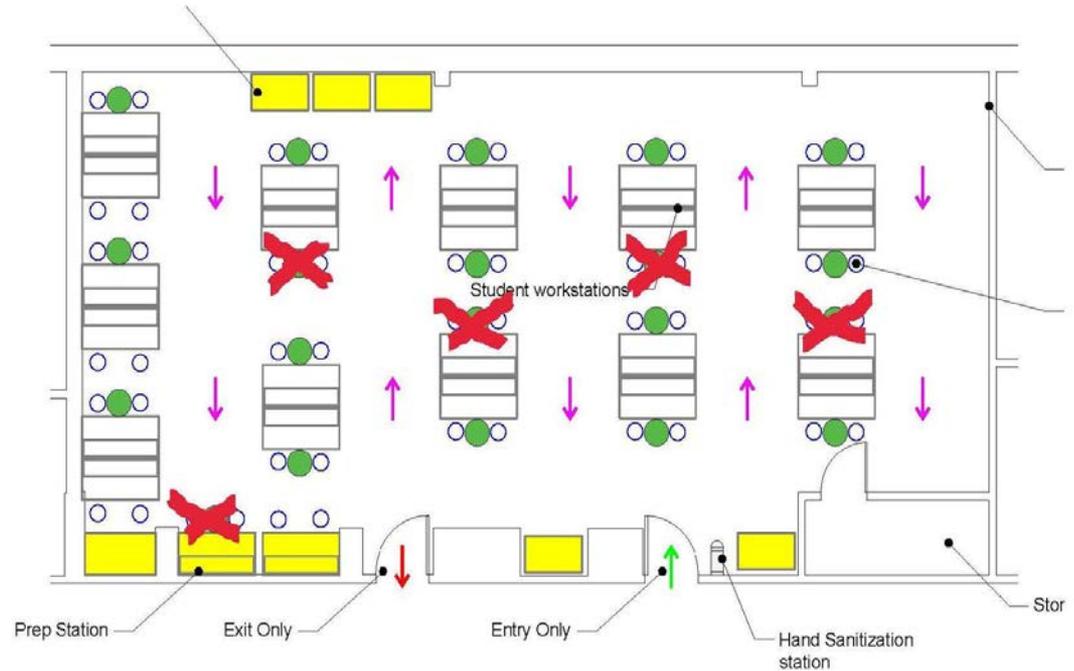


Reserving lab time: more detail

CAPACITY FOR
Marcus 10-12

15

Marcus 10 – 12: COVID Capacity = 15

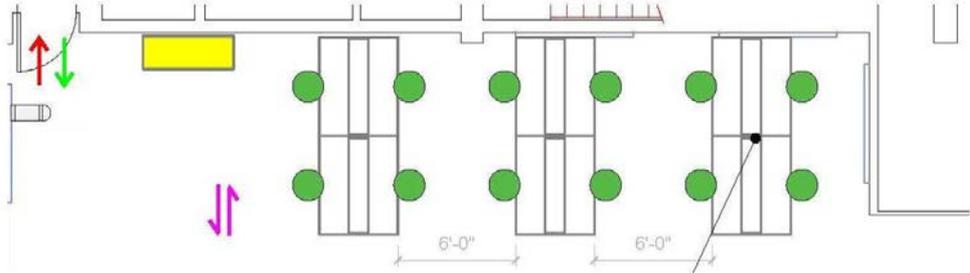


Reserving lab time: more detail

CAPACITY FOR
M5 Great Room

12

Marcus 5 (Great Room): COVID Capacity = 12

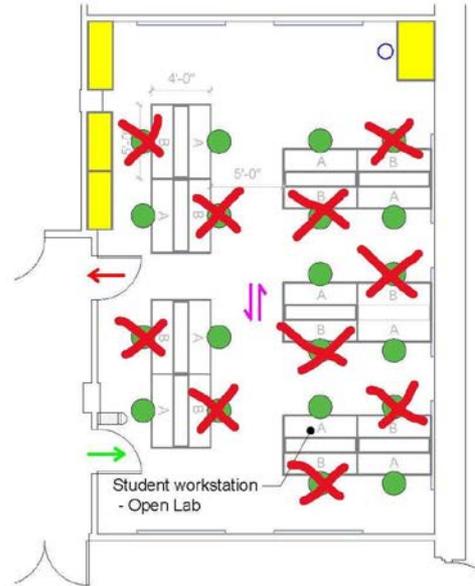


Reserving lab time: more detail

CAPACITY FOR
M5 Good Room

10

Marcus 5 (Good Room): COVID Capacity = 10



Use the Reservation System for SDP Lab.

https://docs.google.com/spreadsheets/d/1c6_w53xWrmEHSxHuIDW8aSjppp5BfxmDklf5PxKRMoE

TinyURL: tinyurl.com/sdp21res

Contact Shira for M5 requests

Why? To help you find things, to get you the training you need.

Let me help you, and we'll keep M5 organized and save you time.

SDP21 Consultant: Dr. Charles Malloch

Tuesdays and Wednesdays, 7-9 PM

(Zoom number to be announced via Slack #main-channel)

Dr. Malloch is a retired Pratt and Whitney engineer. He's programmed in over 50 languages over the past 50 years. There are not very many people on Earth with that kind of experience. On top of all that, add knowledge of electronics, sensors, communication networks (wired and wireless) and you'd be crazy not to get some insights from him. And ask him about his home automation projects. Seriously, SDP21 is lucky that Chuck Malloch is available for consultation. Please take advantage of this opportunity.

Watch this video with Chuck Malloch: tinyurl.com/sdp21dcm

Policy on the Use of Off-The-Shelf PCBAs

All off-the-shelf electronic hardware that your team hopes to use in your April demos must be submitted to the course coordinators for review. Only approved hardware can be used in April. Submit queries ASAP so you can move forward with confidence. Use your **teamxy-and-course-coordinator** channel for your queries.

Examples of hardware likely to receive approval:

- Single-board Linux Computers (Raspberry Pi, Beagle, etc.) (but ONLY if there is a demonstrated need for that level of computing power.)

Examples of hardware unlikely to receive approval:

- Arduinos & Arduino clones, mbed boards
- A breakout board with low level of complexity (count the pins!)

Remember your training & avoid hubris



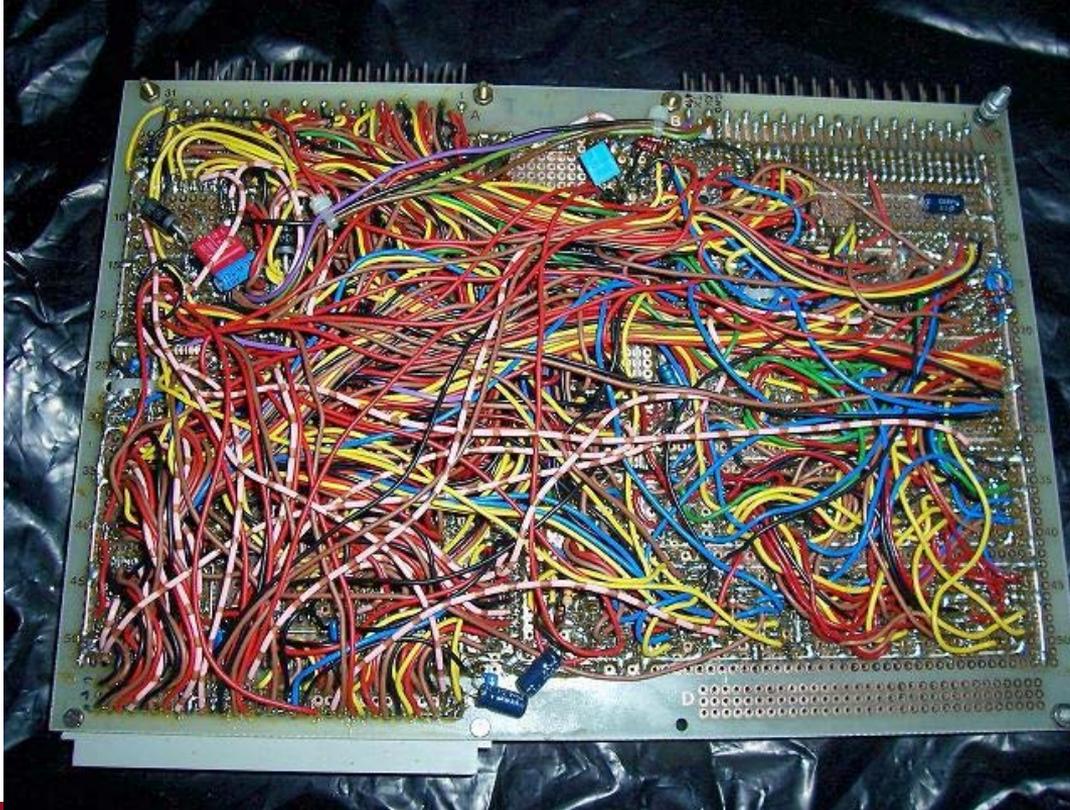
Start early

Question assumptions

Review the basics

Go step by step

Design and test iteratively



If you put it all together and only then test it for the very first time...

Not only will it not work, but also: you will have no hope of figuring out what is wrong...

Compound issues are much harder to debug than isolated issues

“68020 with 32 Bit data bus”

Source:

<http://forum.6502.org/viewtopic.php?f=12&t=4288>

Design and test iteratively



And just because it compiles doesn't mean it works the way you intend....

Remember best practices. ***Do one thing, test one thing***

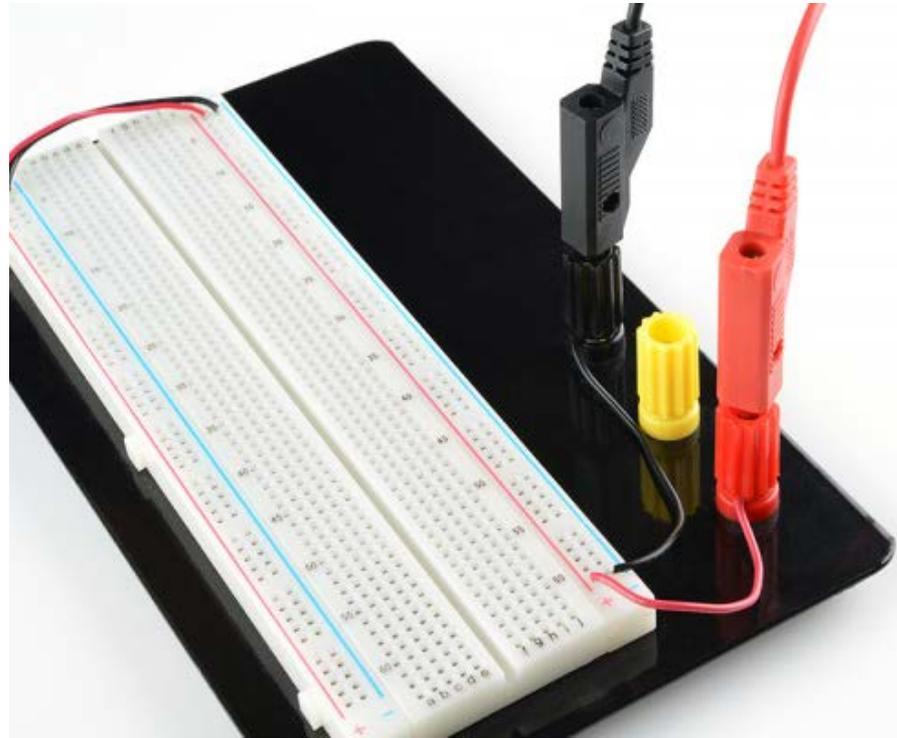
Start at the beginning.

Test the setup first.

For example, using a breadboard?

Plug in the power rails.

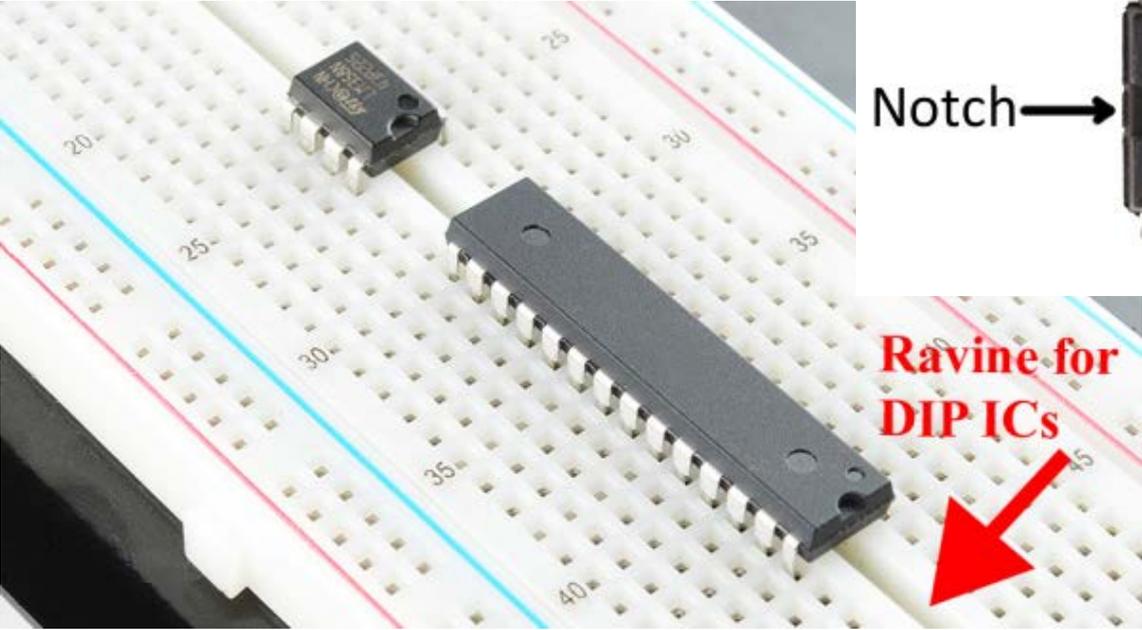
Grab a multimeter. Test it.



<https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard/providing-power-to-a-breadboard>

Get DIP form factor integrated circuits for testing

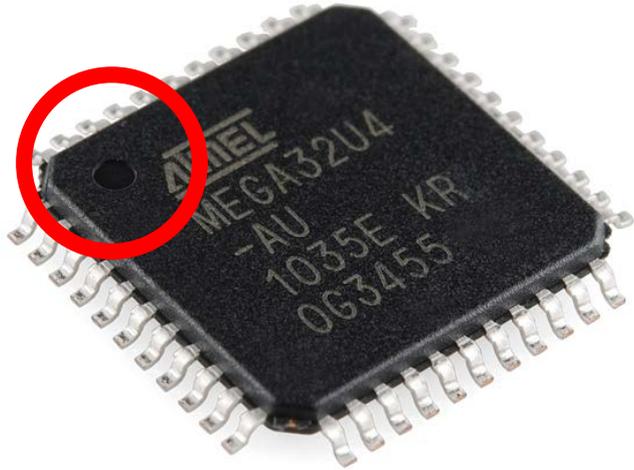
Check that everything is plugged in the right way



Notch →

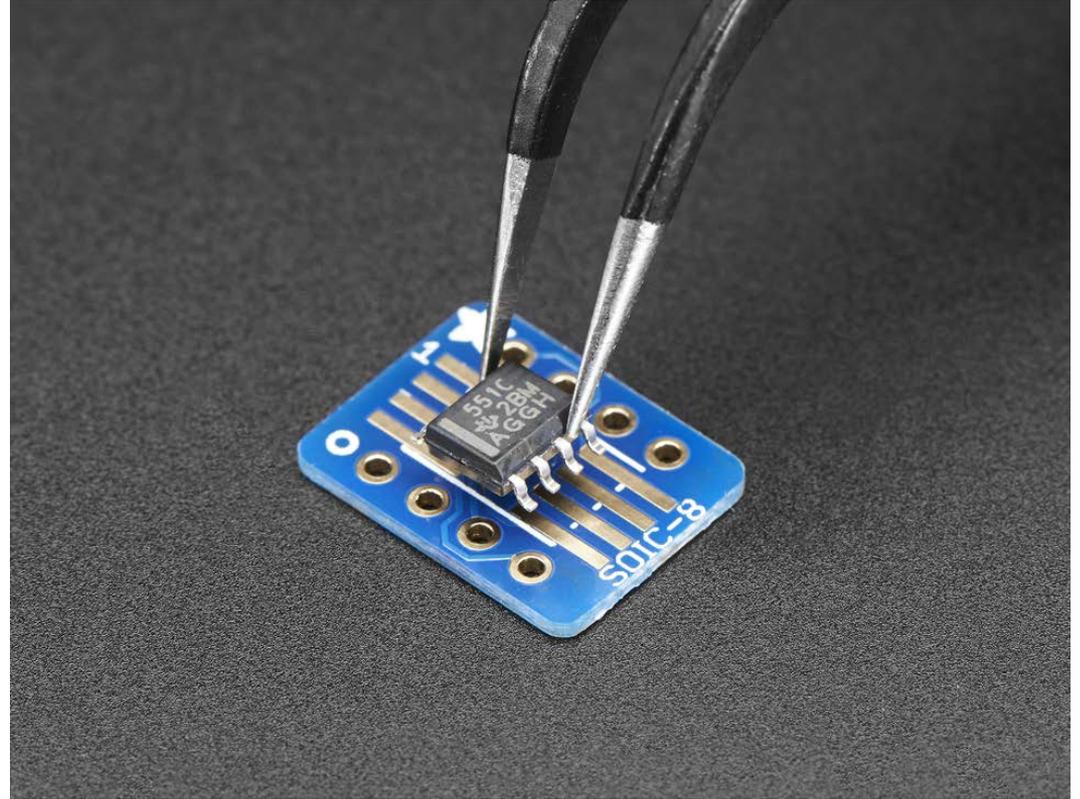


No DIP? Use an adapter. Form factors are standardized.



Keep track of the dot

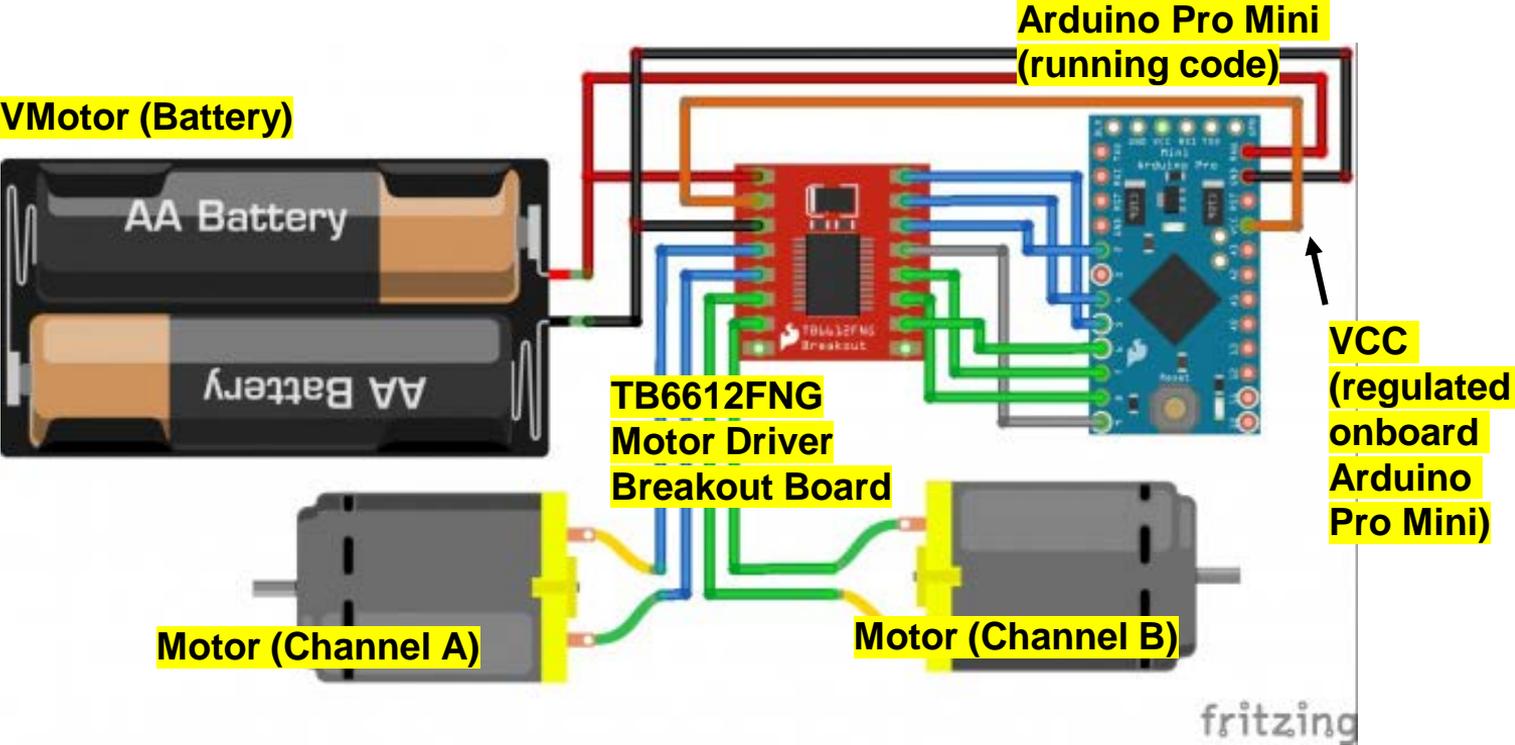
<https://www.adafruit.com/product/1212>



Crawl before you walk before you run. DC motor driver example

Though you can & should start with breakout boards and development boards--

By FPR, you will "ditch the training wheels" entirely



<https://learn.sparkfun.com/tutorials/tb6612fng-hookup-guide#hardware-setup>

Check that your code is actually uploading first

The Blink sketch comes loaded on most Arduino Uno

Confirm that you can make a meaningful change to it, upload it, and see the result.



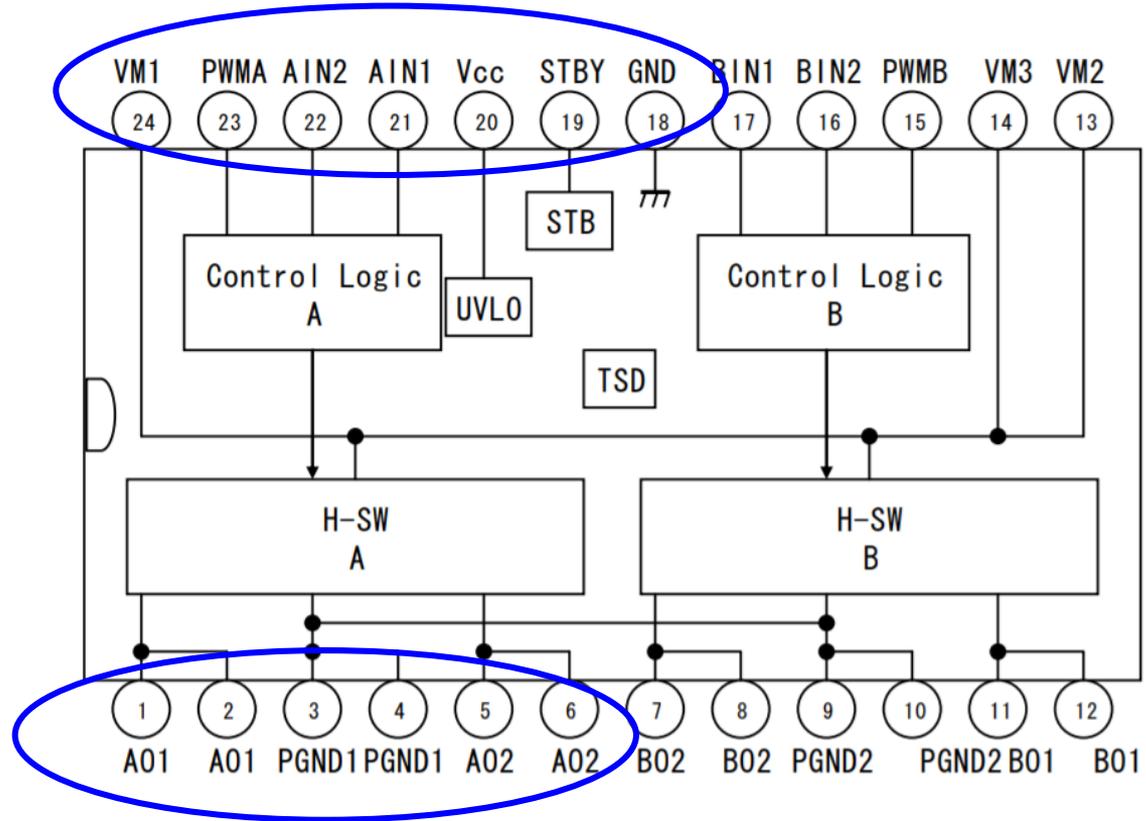
Read the datasheet!

Example: This DC motor driver

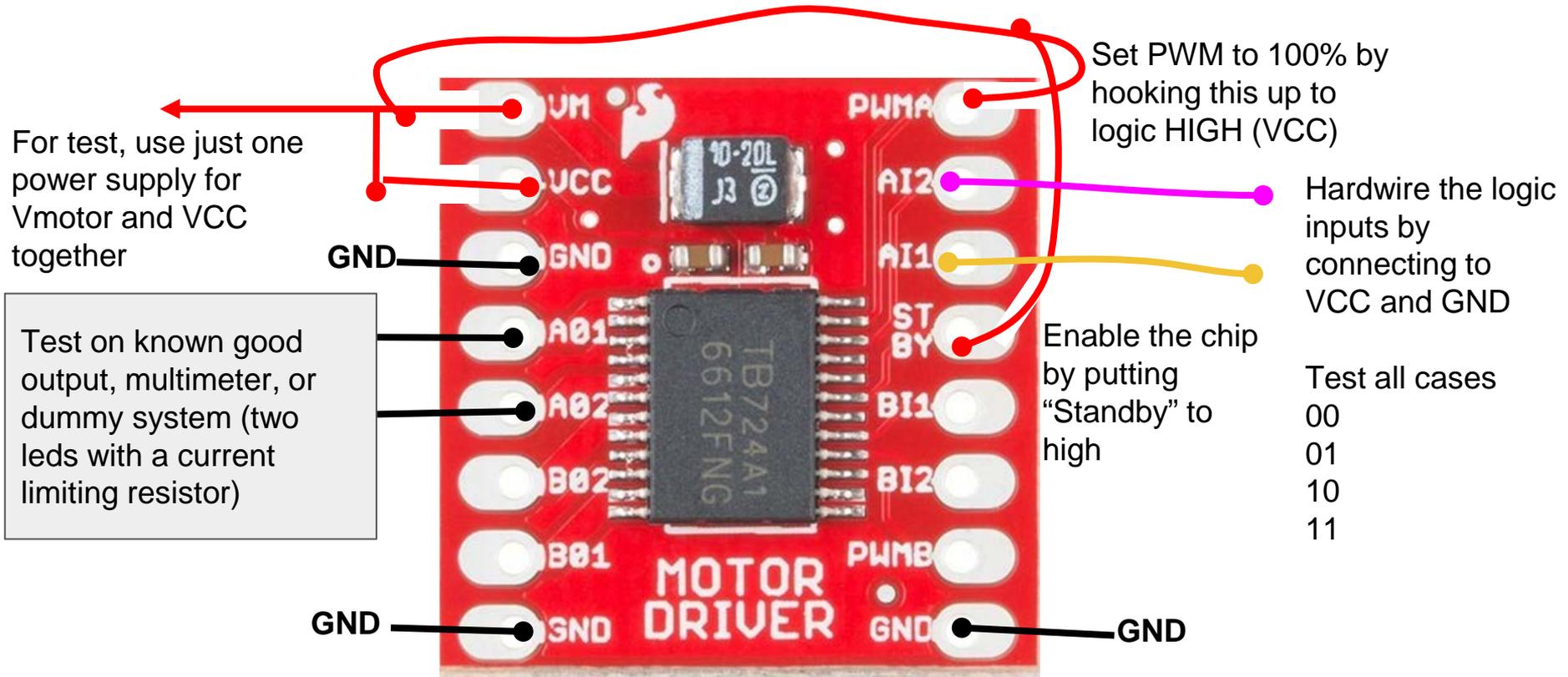
What is the minimum #
of connections you need to make
in order to get ONE channel
working?

Let's hardwire the logic inputs,
and get rid of the microcontroller
and code for the moment.

What if I connect the outputs to
my multimeter, get rid of the
motor for now?



Example: testing just one thing at a time (channel A of driver)

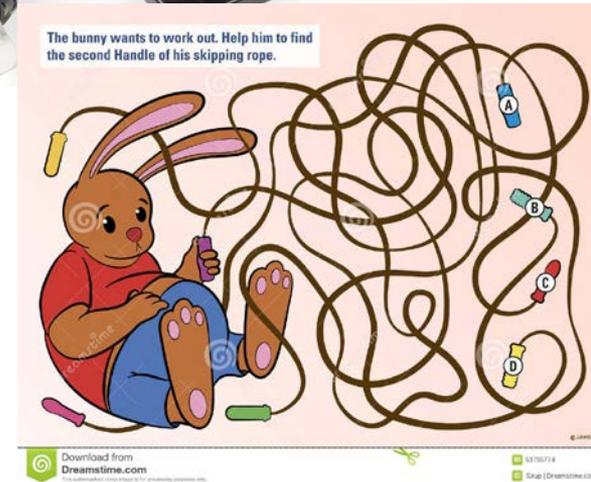
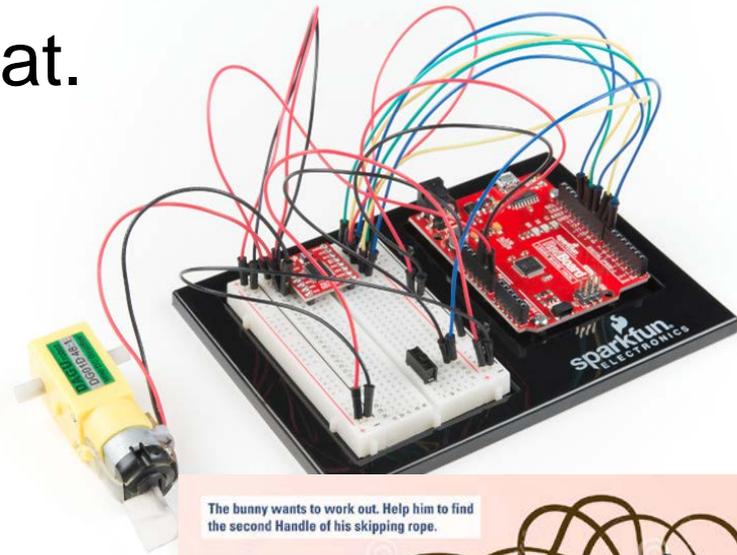


Color code your wires. Keep it neat.

This might work for the first test, with just one motor. But it will get pretty messy when you add your second motor.

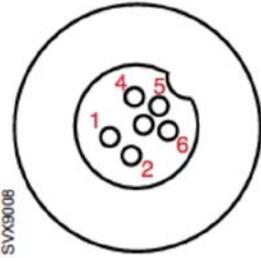
After you test it, go ahead and make it neater! Use color coded wires. Keep as short as possible.

Debugging this is like playing the games on the back of a cereal box. **You should be able to solve this part on your own.**



Pinout troubles: mirror images

Is the diagram the CABLE or SOCKET???"Trust but verify" (left diagram represented the socket, mirror of cable).



N28 / N70 – Control cable

1. +12V_UNREG_PANEL
2. GND
3. LED_CONTROL
4. ON_OFF_CONTROL
5. CAN_PANEL.H
6. CAN_PANEL.L



Debugging



Debugging and analysis tools

Multimeter (examples)

- Continuity Tester
 - Is it connected?
 - Which pin is going where?
 - Is system power off?
- Voltmeter (check voltage)
 - Is it on?
 - Is it logic high or logic low?
- Ammeter (check current)
 - Too much current can be a sign something is broken
- Ohmmeter (check resistance)
 - Check layout of coils on a stepper motor
 - Is system power off?

Portable handheld

Benchtop model



Debug your debugging tools. Sanity checks!

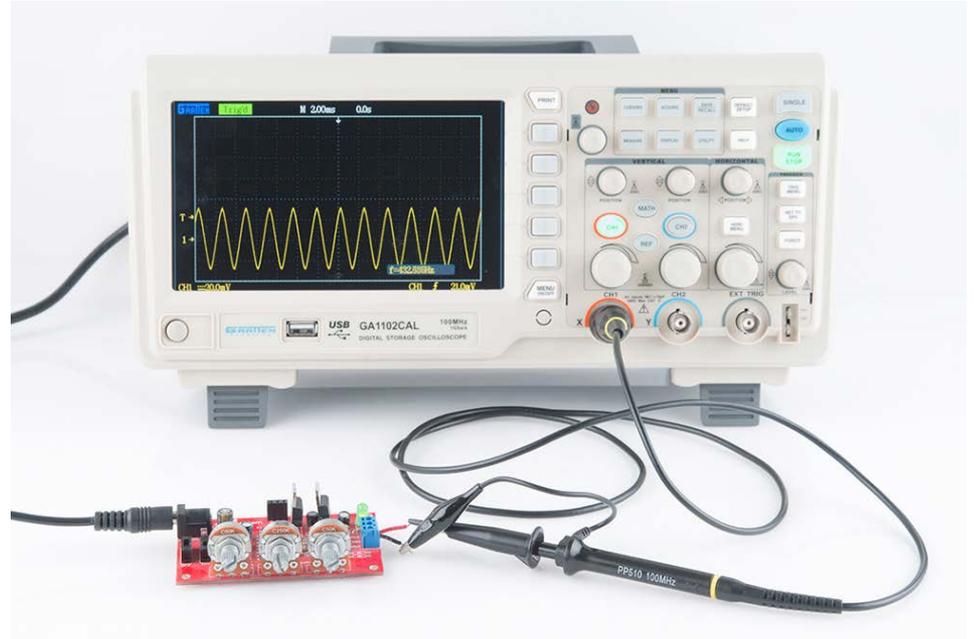
Test something for which you know what result you should get. Are you getting that result? If not, something is wrong with your setup.

- Probes connected to wrong setting for measurement?
- Dial set to wrong setting?
- Probes broken or won't fit ports (mismatched size?)
- Hold mode is on?
- Multimeter fuse needs replacing?
- Measurement is being taken incorrectly?
 - Measuring current in parallel?
 - Measuring voltage in series?
 - Measuring element resistance in parallel with circuit?



Your other debugging & analysis tools

- Oscilloscope
- Logic Analyzer

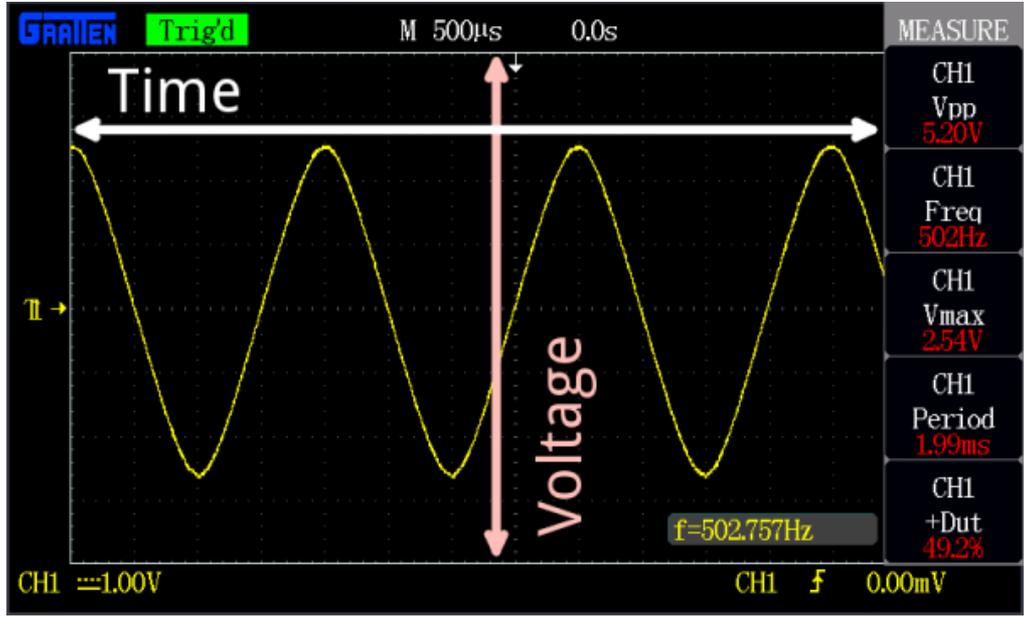


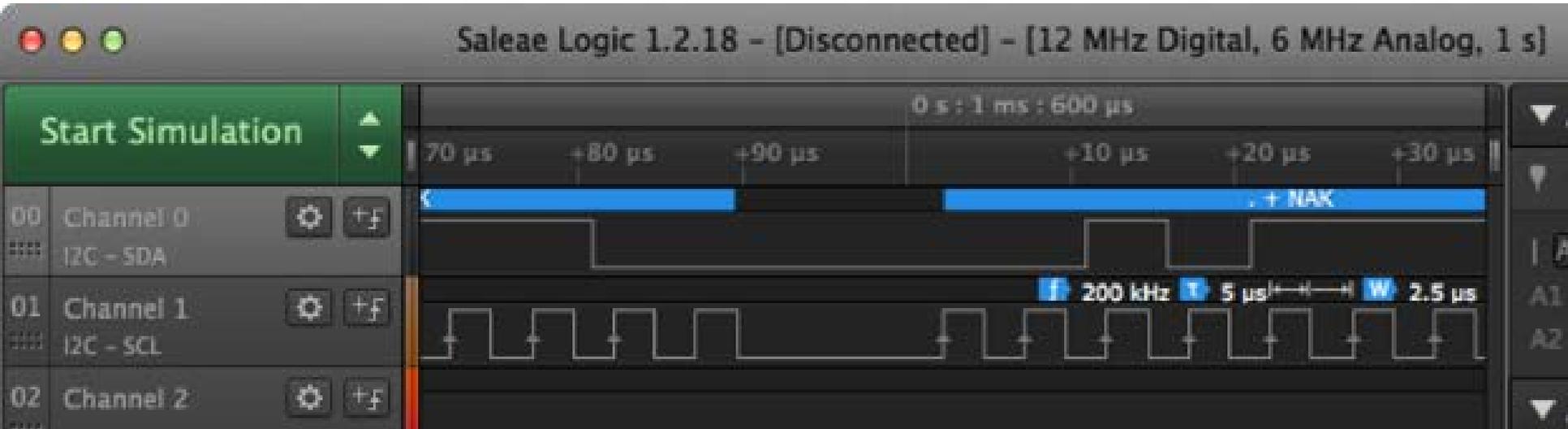
<https://learn.sparkfun.com/tutorials/how-to-use-an-oscilloscope/all>

Oscilloscope

Will go through advanced example next time, but meanwhile refresh your basics with a variety of resources online. Make a reservation in SDP lab or borrow one.

<https://learn.sparkfun.com/tutorials/how-to-use-an-oscilloscope/all>





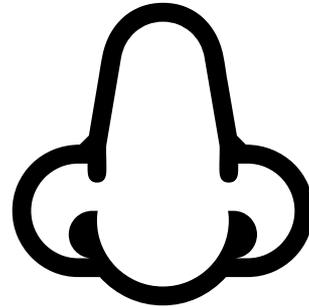
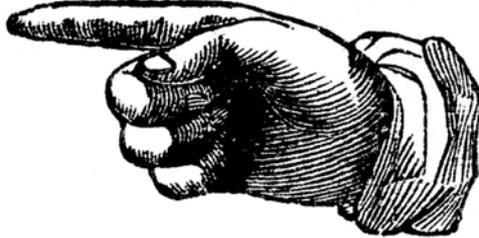
Saleae Logic Analyzers available for the day.

... or purchase \$20 unit at SparkFun with your team funds <https://www.sparkfun.com/products/15033>

Debugging and analysis tools

Finger

- Is it hot?



Nose

- Does it smell like burning plastic?

Don't delay any further. Unplug it and think.

Note: face shield and mask adds some delay already

Using the benchtop power supply features to protect your circuit when you test it

The benchtop power supplies have lots of features. We will discuss, or go online and refresh your memory, visit SDP lab and work with them directly.



Think outside the box. Start with the most obvious but keep looking until you find the culprit

Real problems I have had to debug:

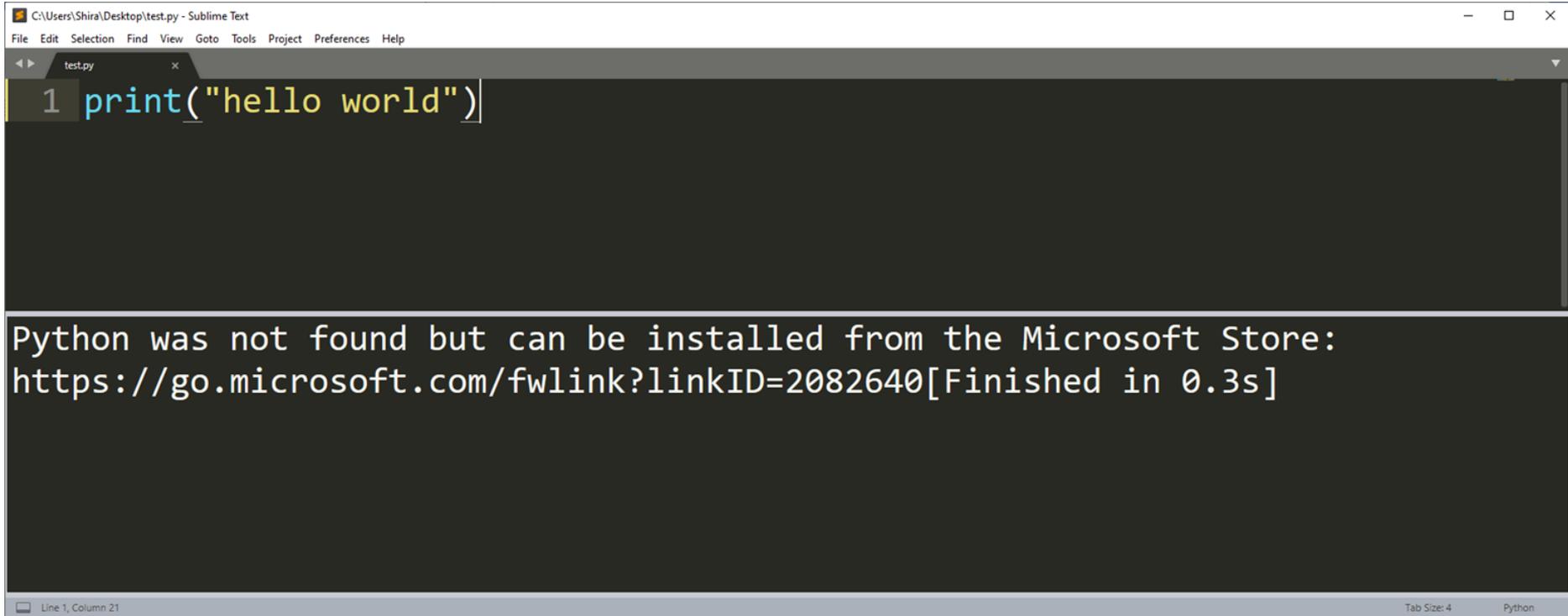
- Wires loose or shorting together
- Voltage drop over severely corroded wires
- Robot only goes straight, can't turn (because battery too low)
- Electromagnetic interference (EMI) from motors resetting microcontroller
- Electromagnetic interference from nearby hobby radio setup caused Raspberry Pi controlling hackerspace door lock to become unlocked...
- Static electricity destroyed part, needed new part
- Cold caused components to stop working properly
- Flickering fluorescent lights in new test environment triggered infrared sensor
- Intermittent issues caused by internally torn wire

If it works, take a picture (abstractly and literally)

- Commit your code (see next section about version control)
- Take a picture or a screenshot or a video
- Take a picture of your setup
- Don't take apart things you don't need to. Breadboards are cheap. If you can, keep what works and start on a new one.
- Take notes all the time

This way when things break you'll have: proof it did work once, and a reference for how things looked when it did work

Do one thing, test one thing



The image shows a screenshot of a Sublime Text editor window. The title bar reads "C:\Users\Shira\Desktop\test.py - Sublime Text". The menu bar includes "File", "Edit", "Selection", "Find", "View", "Goto", "Tools", "Project", "Preferences", and "Help". The editor has a single tab titled "test.py" and contains the following code on line 1:

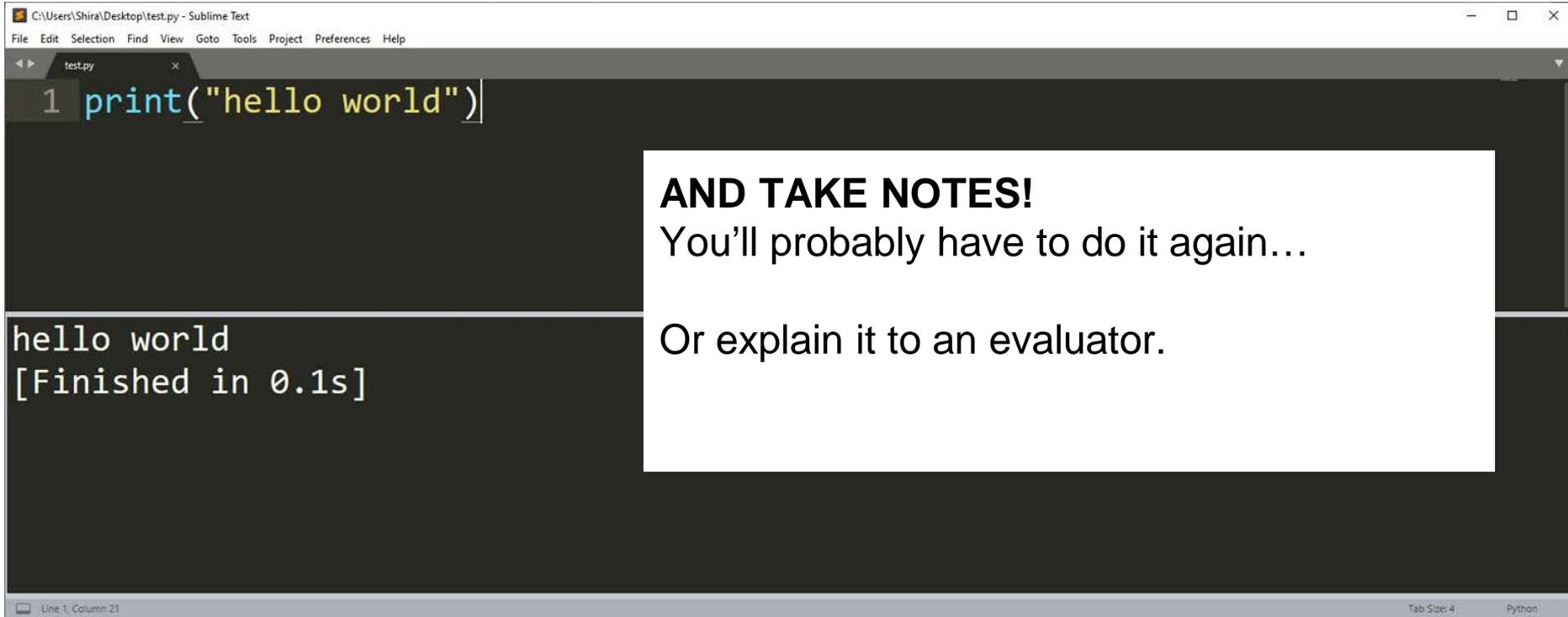
```
1 print("hello world")
```

Below the code editor, a terminal window displays the following message:

```
Python was not found but can be installed from the Microsoft Store:  
https://go.microsoft.com/fwlink?linkID=2082640[Finished in 0.3s]
```

The status bar at the bottom of the editor shows "Line 1, Column 21", "Tab Size: 4", and "Python".

Do one thing, test one thing



The screenshot shows a Sublime Text editor window titled "C:\Users\Shira\Desktop\test.py - Sublime Text". The menu bar includes "File", "Edit", "Selection", "Find", "View", "Goto", "Tools", "Project", "Preferences", and "Help". The editor has a single tab for "test.py" containing the code: `1 print("hello world")`. Below the editor, the terminal output shows "hello world" followed by "[Finished in 0.1s]". The status bar at the bottom indicates "Line 1, Column 21", "Tab Size: 4", and "Python".

```
C:\Users\Shira\Desktop\test.py - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
test.py
1 print("hello world")
hello world
[Finished in 0.1s]
Line 1, Column 21 Tab Size: 4 Python
```

AND TAKE NOTES!

You'll probably have to do it again...

Or explain it to an evaluator.

Make your code modular

- Don't write everything inline in main()
- Write functions
- Functions take in input arguments and result in return values
- You can test functions individually by calling them with test inputs and seeing if the output is as expected

I know you did PDR, but go back and make another (ten) software diagram(s)...

Avoid writing “spaghetti code”

“Plan to throw one away”

Identify the problem

- Describe the current behavior
- Describe the ***desired*** behavior!
- Create a test case that currently fails, but will no longer fail when you fix the bug.
- After you fix the bug, re-run your old test cases. Save the test (design your tests to be functions that call other functions. Don't delete your tests).

You may not end up designing a full framework for [unit testing](#)

But if handpicking test cases, pick carefully. Pick the things that will likely break it.

Think about: Collections with even or odd # of elements. Empty collections.

Negative numbers. Variable overflow...

Tools and techniques for testing -- software

- Roll back to previous version
- Back to basics--does ANYTHING work? [Sanity checks](#).
- Step through line by line of your logic on paper (especially in planning)

Available to you if you are connected to a screen:

- Print statements
- Read the error message! And understand it! Google it!

Use a proper debugger so you can harness these tools

- Break points
- Call stack
- Local/global variable inspector

Atmel-ICE

Atmel-ICE is a tool for debugging and programming Microchip Atmel microcontrollers

- Programming (via TPI) of all Atmel tinyAVR 8-bit microcontrollers with support for this interface
- Programming and on-chip debugging of Atmel AVR microcontrollers (8 bit)
- Programming and debugging of all Atmel SAM ARM Cortex-M based microcontrollers (32 bit)



Debugging and analysis tools

Sounding board/Rubber duck/a friend:

Describe how it is supposed to work aloud...
Something may jump out to you in this process

Assumptions will be challenged



Getting help

If you ask for help, I'll ask what is wrong

If you say “It just doesn't work” I will not have too much useful input for you.

What are the symptoms? When does this happen?

What is your best guess? What is the [minimal working example](#)?

Often, in preparing to get my help, you'll find the answer on your own.



Think outside the box. Start with the most obvious but keep looking until you find the culprit

Real problems I have had to debug:

- File named something that was not allowed (name of existing module)
- Editing wrong file entirely
- Editing correct file, but compiling executable to wrong location
- Compiler optimizer is on and causing strange effects (read the warnings!)
- Input contained unexpected whitespace (carriage returns, '\r')
- Virtual machine ran out of space
- Warning: "Low memory available, stability problems may occur." They did occur.
- Found a bug in the code of the library I was using (that somebody else wrote)
- Ran with wrong interpreter (Python 3 instead of 2.7)

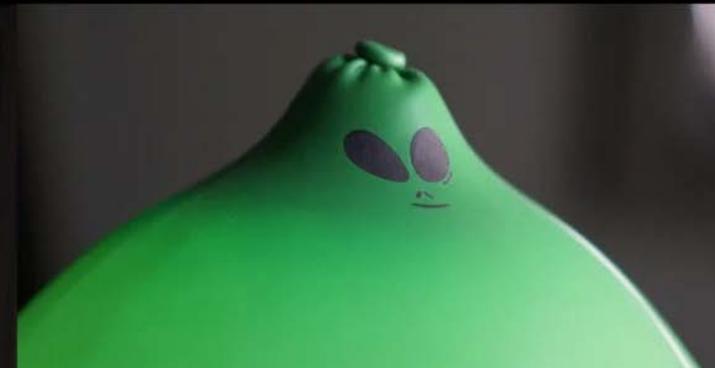
Expectation vs. Reality

Run tests to discover incorrect assumptions about your project earlier rather than later.

Expectation:



Reality:



Distributed Version Control System - What is it?

- Tools for a team of developers to work together
- Track changes, create or merge branches
- Inspect version differences, roll back changes if needed
- Serves as a detailed work log with documentation of who changed what, when, and why
- Typically used for developing software, but can be used for:
 - 3d files
 - Schematics
 - And most other documents!

Getting started with GitHub

Git is the free and open source distributed version control system

GitHub is the popular platform/service built around Git owned by Microsoft

<https://guides.github.com/activities/hello-world/>

Go make an account and follow a tutorial. Get familiar. You can use the command line OR the GUI Desktop client.

Questions?