Lecture 6

LEDs, Multiplexing, Building an LED Display

Roadmap

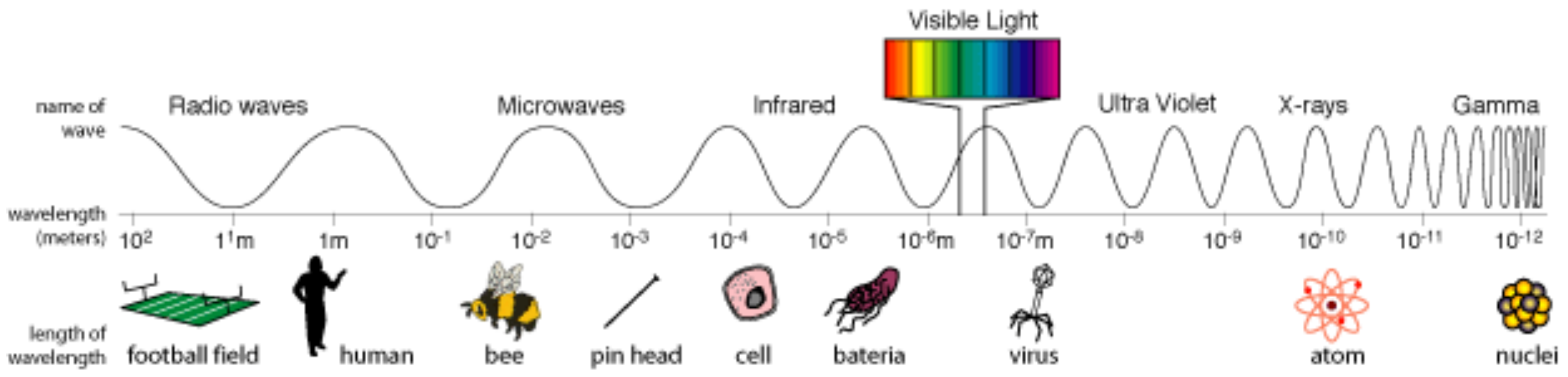
Today we start describing the techniques that you will use to create your LED cube, or LED array. We will review how LEDs work, how to use them in simple circuits and how to use **time division multiplexing** to control many LEDs.

Then we will look at how planning will make this construction project easier. Since you have so many LEDs to connect up, building a jig to make that job easier is a very good idea.

If there's time, we'll also talk a bit about driver circuits for your LED array.

Key Ideas From The Last Lecture
What is Light?

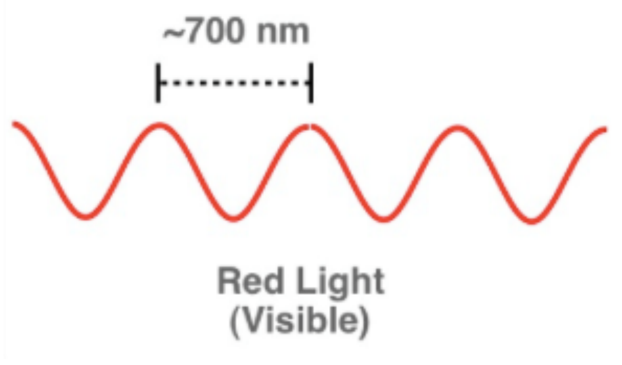
- It is an electromagnetic wave
  - Speed of light, c = 3E8 m/s
  - Frequency = c/λ
- Part of electromagnetic spectrum:



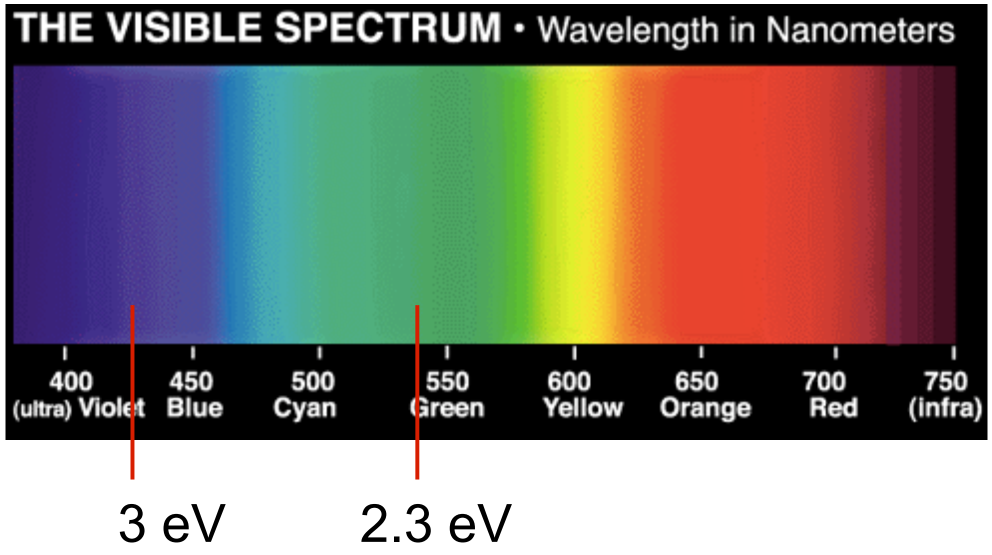
- All waves transport power

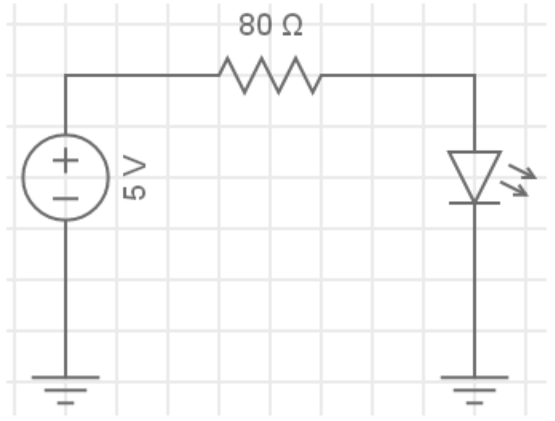
http://science.hq.nasa.gov/kids/imagers//ems/index.html

Key Ideas From The Last Lecture
Photons and Energy



$$f = \frac{c}{\lambda} \qquad E = \frac{hc}{\lambda} = \frac{1.24\text{eV}}{\lambda(\mu\text{m})}$$

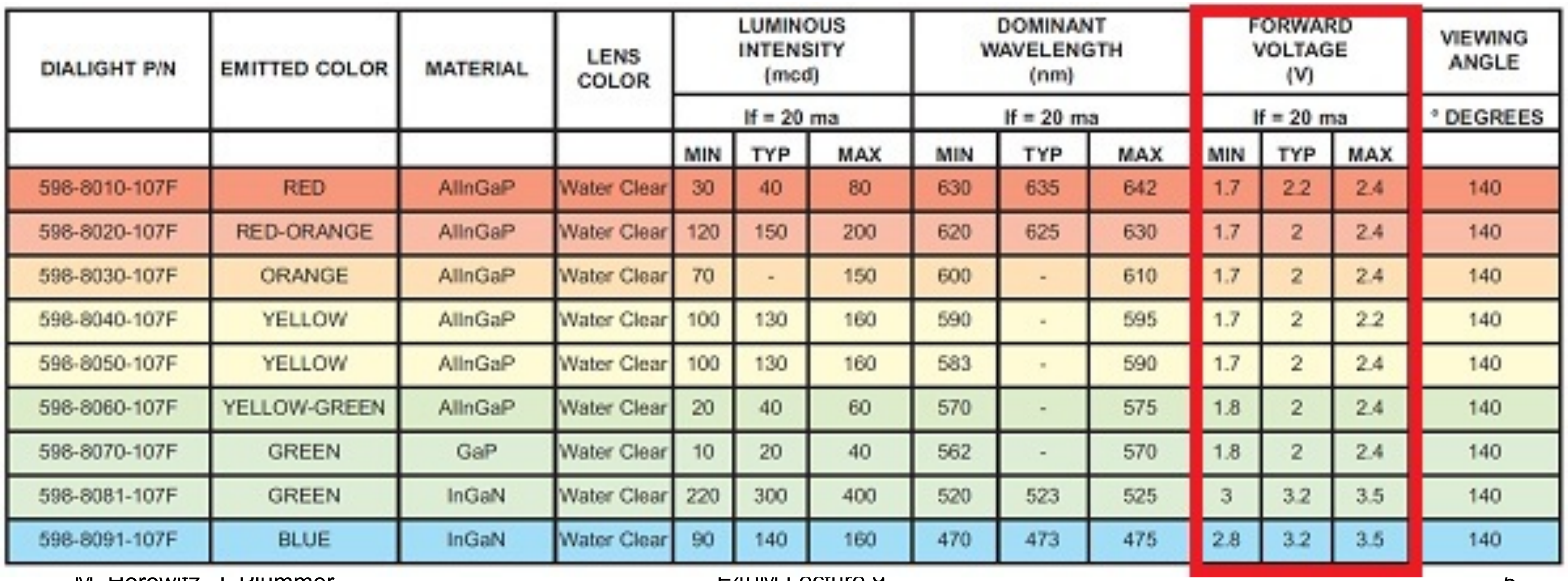




- Current drops 2.3 volts across diode and green photons are emitted.
- Green photons strike a diode, current and **up to** 2.3 volts can be generated.

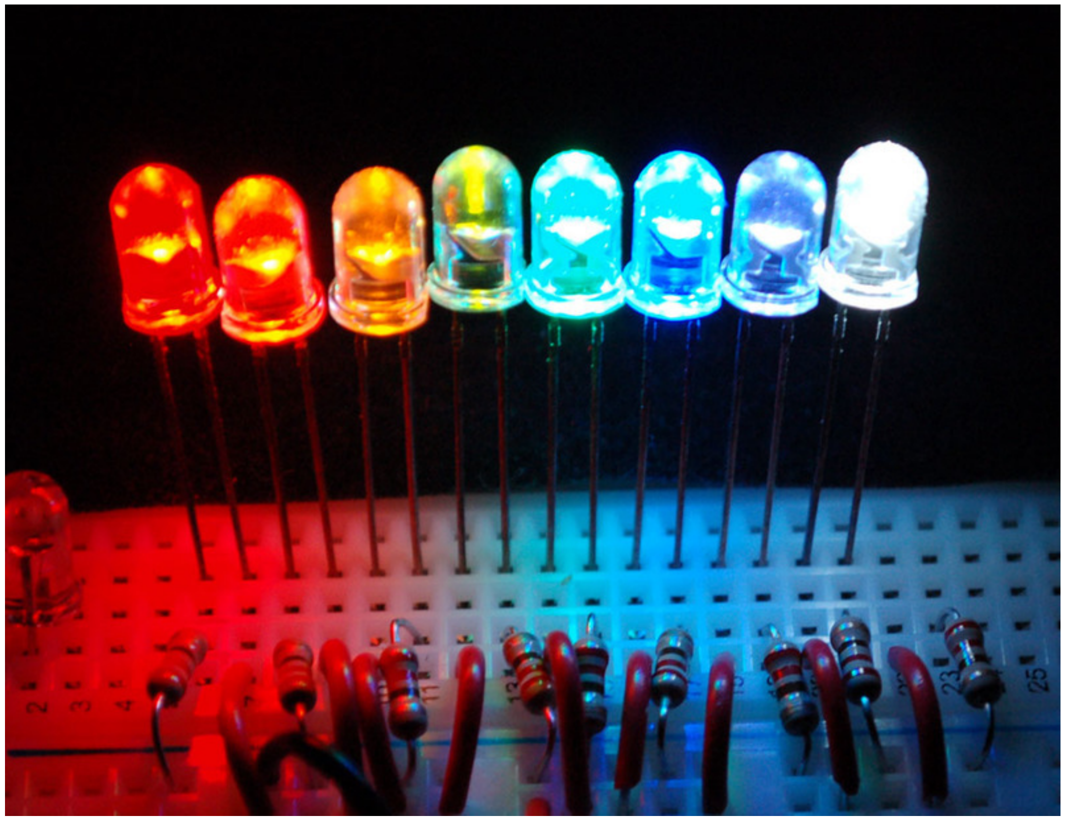
LED Voltage Drop and Color

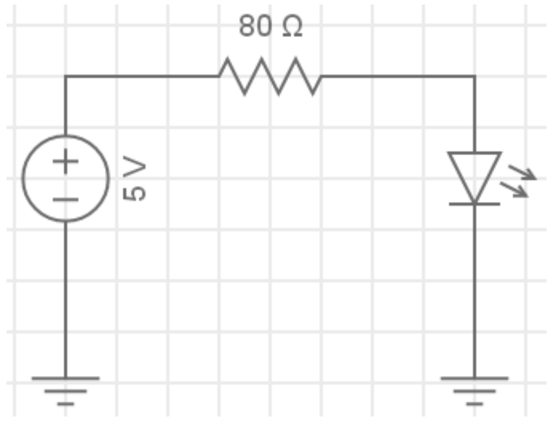
- The color of the photon depends on energy
- The energy available depends on the voltage
  - Each electron that flows can create one photon
    - If it takes two, the two have to happen at the same time
      - This is very unlikely
  - V_f for a blue LED is larger than for a Red LED



DIALIGHT P/N	EMITTED COLOR	MATERIAL	LENS COLOR	LUMINOUS INTENSITY (mcd)			DOMINANT WAVELENGTH (nm)			FORWARD VOLTAGE (V)			VIEWING ANGLE
				If = 20 ma			If = 20 ma			If = 20 ma			° DEGREES
				MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
598-8010-107F	RED	AlInGaP	Water Clear	30	40	80	630	635	642	1.7	2.2	2.4	140
598-8020-107F	RED-ORANGE	AlInGaP	Water Clear	120	150	200	620	625	630	1.7	2	2.4	140
598-8030-107F	ORANGE	AlInGaP	Water Clear	70	-	150	600	-	610	1.7	2	2.4	140
598-8040-107F	YELLOW	AlInGaP	Water Clear	100	130	160	590	-	595	1.7	2	2.2	140
598-8050-107F	YELLOW	AlInGaP	Water Clear	100	130	160	583	-	590	1.7	2	2.4	140
598-8060-107F	YELLOW-GREEN	AlInGaP	Water Clear	20	40	60	570	-	575	1.8	2	2.4	140
598-8070-107F	GREEN	GaP	Water Clear	10	20	40	562	-	570	1.8	2	2.4	140
598-8081-107F	GREEN	InGaN	Water Clear	220	300	400	520	523	525	3	3.2	3.5	140
598-8091-107F	BLUE	InGaN	Water Clear	90	140	160	470	473	475	2.8	3.2	3.5	140

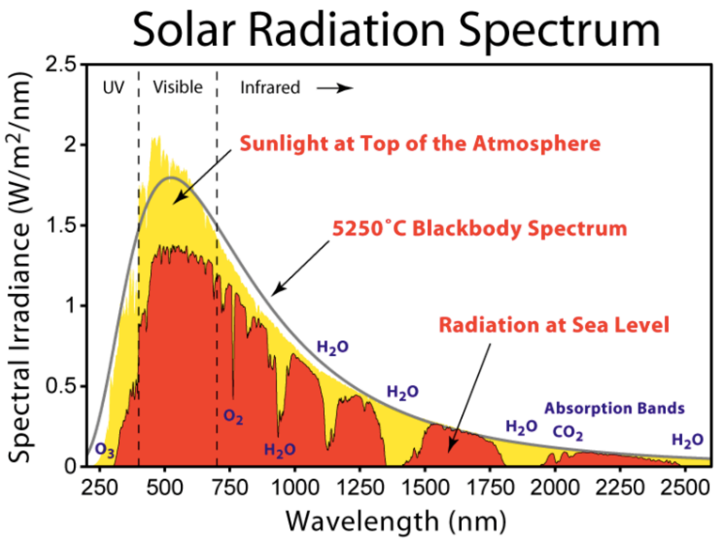
Key Ideas From The Last Lecture Light Emitting Diodes (LEDs)

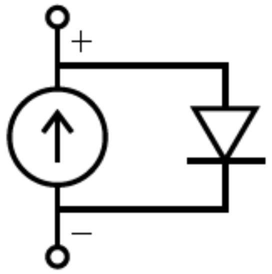


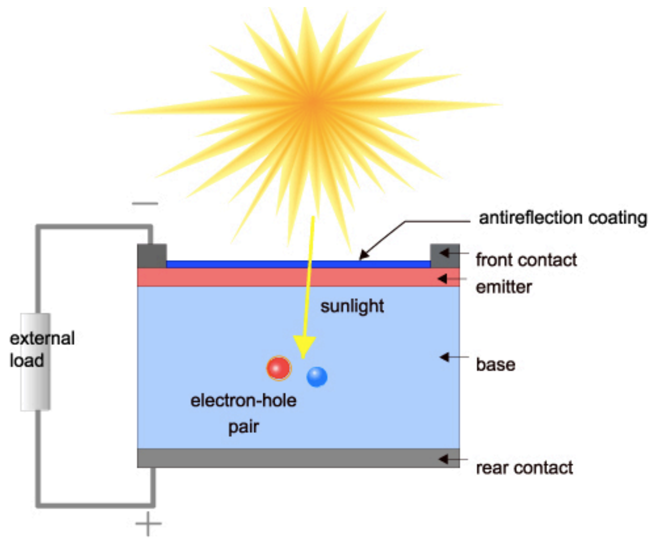


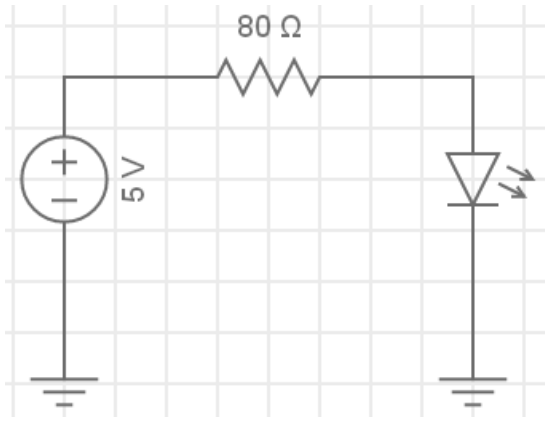
- How do we get different colors?
- How does this relate to a solar cell which operates in reverse?

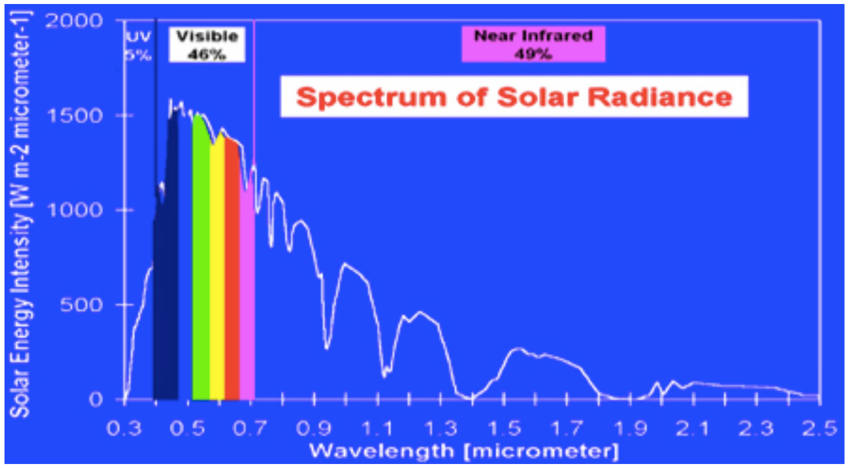
FYI – How Do Light Emitting Diodes and Solar Cells Actually Work?





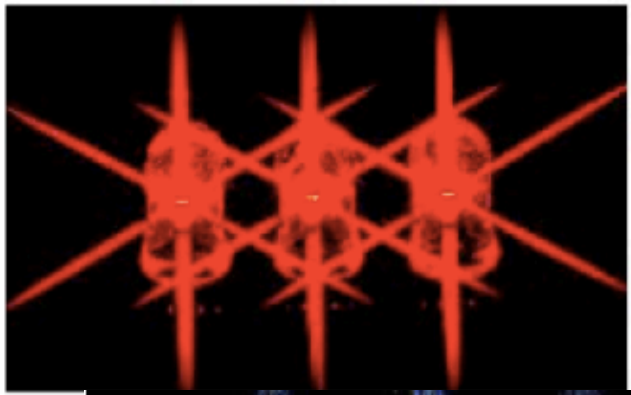
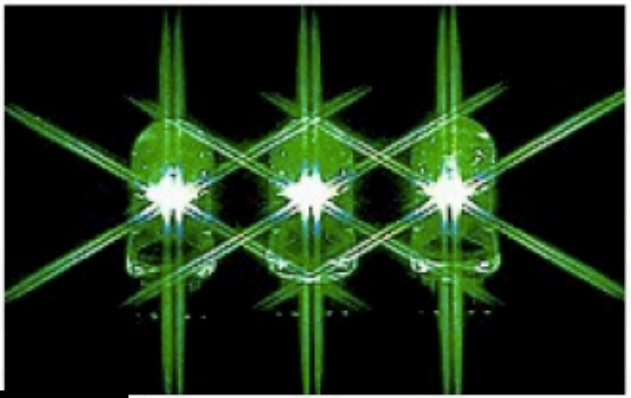
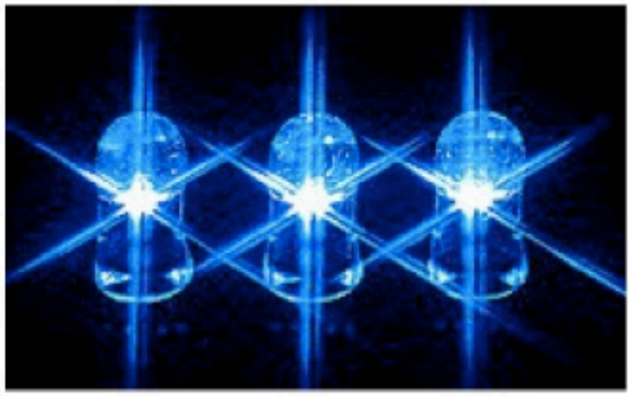
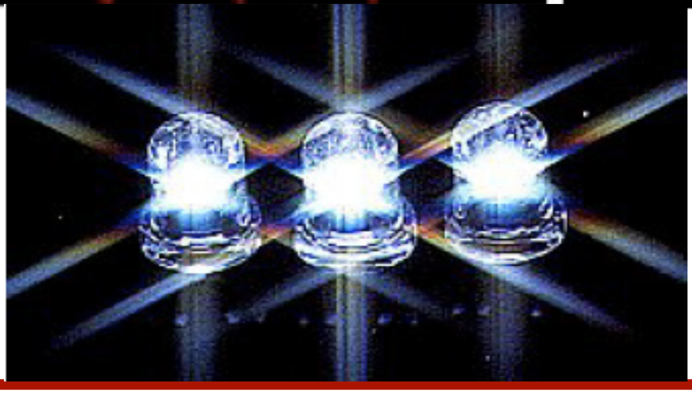






FYI – Full Color LED Displays and Solid State Lighting (https://en.wikipedia.org/wiki/Light-emitting_diode)

- Red/orange/green LEDs have been used in small displays for 30 years. Nakamura's invention of GaN LEDs has dramatically changed the lighting world – not only creating blue LEDs for full color displays, but creating the possibility of solid state lighting.

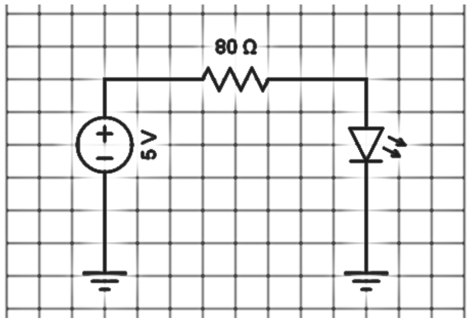





- White LEDs utilize UV emission of GaN to excite fluorescence in phosphors which emit white light. GaN based LEDs are now beginning to displace fluorescent lighting.

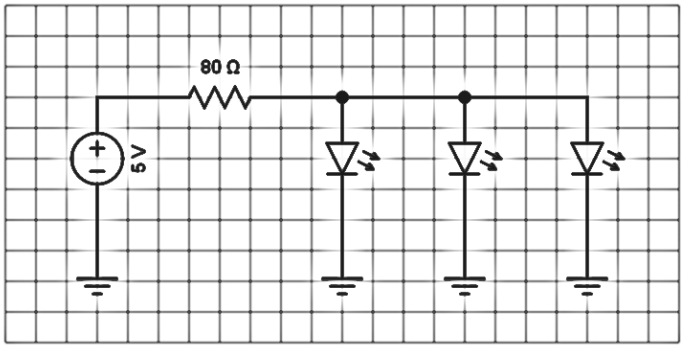
Learning Objectives for Today

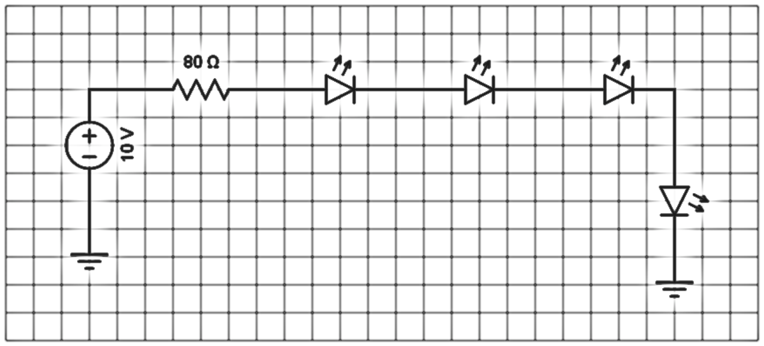
- Understand how to use LEDs in simple circuits
- Be able to use **time division multiplexing** to control LEDs
  - Control n^2 lights using only 2n wires
  - By turning n lights on in n different time slices
- Be able to create a 2-D plane of LEDs that can be multiplexed
  - And be able to stack planes to create a cube
- Be able to create the drive circuit for your LED array

Using LEDs in Simple Circuits



- Always use a series R with an LED





- Do not wire LEDs in parallel

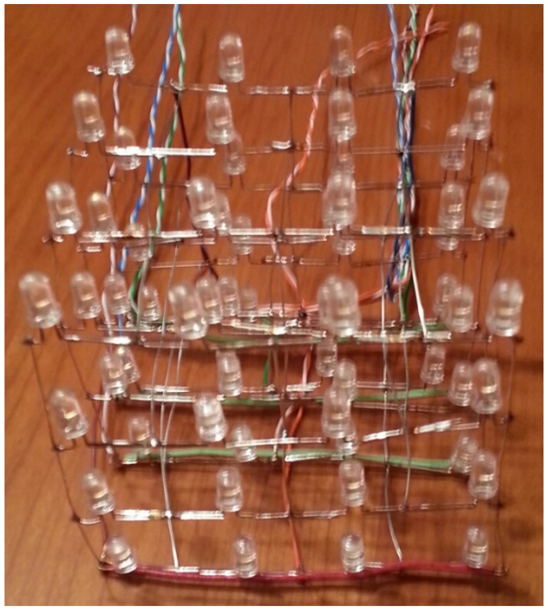
- Series connection is fine with a higher V

LED Plane

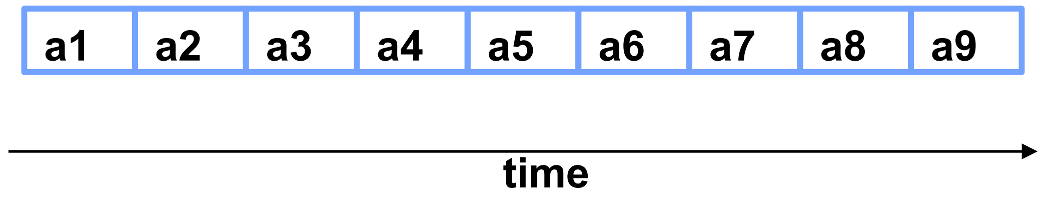
- You will build a 6 x 6 array of LEDs

- You will get to choose
  - Red, Green, Blue, White
  - Or can mix it up

- Two challenges
  - How to control 36 lights?
  - How to build something
    - With 36 elements
      - That is a lot of soldering
      - A little planning will go a long way



Serial Communication

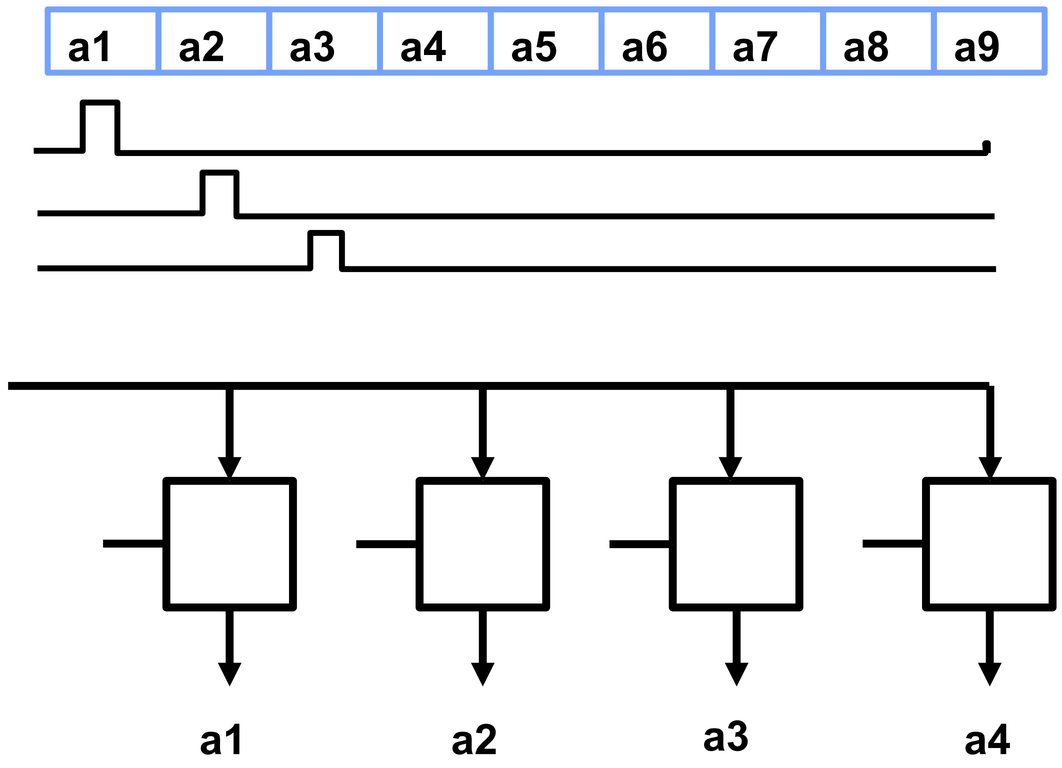


- Also called
  - Time division multiplexing
  - Or just multiplexing

- Heavily used
  - Ethernet
  - Serial ports
  - USB (universal serial bus)
  - I^{2}C, SPI, HDMI, JTAG, etc.

Serial to Parallel Converters

- If you use a string of our memory cells can get all the bits
  - Load each memory cell at the "right" time



Optical Persistence

- Ever notice that all your electronics seems to blink at you
  - Take any of it and shake it in front of your eyes
    - Generally the lights will form dashes
    - That is because it is not on all the time
  - Your eyes are slow
    - The flicker fusion rate is around 30Hz
    - Your eye averages the signal

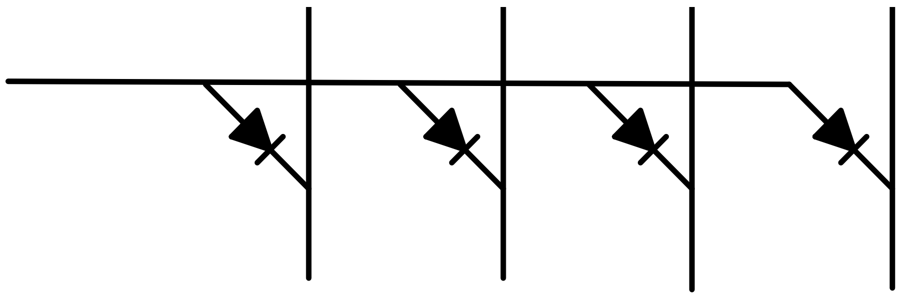
- Electronics takes advantage of the fact that your eyes are slow
  - In two ways
    - Creates more outputs than wires
    - Creates analog light output values on digital pins

Basic Approach

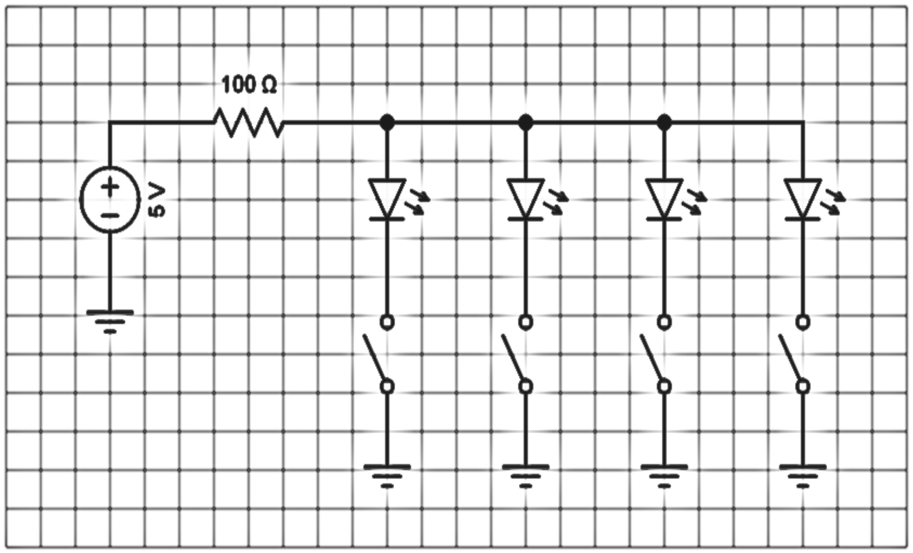
- If I have many lights, I don’t need to turn them all on at once
  - I can create different slots in each time period
    - Say I created 6 slots
  - Then I only need to light 36 / 6 lights in each slot

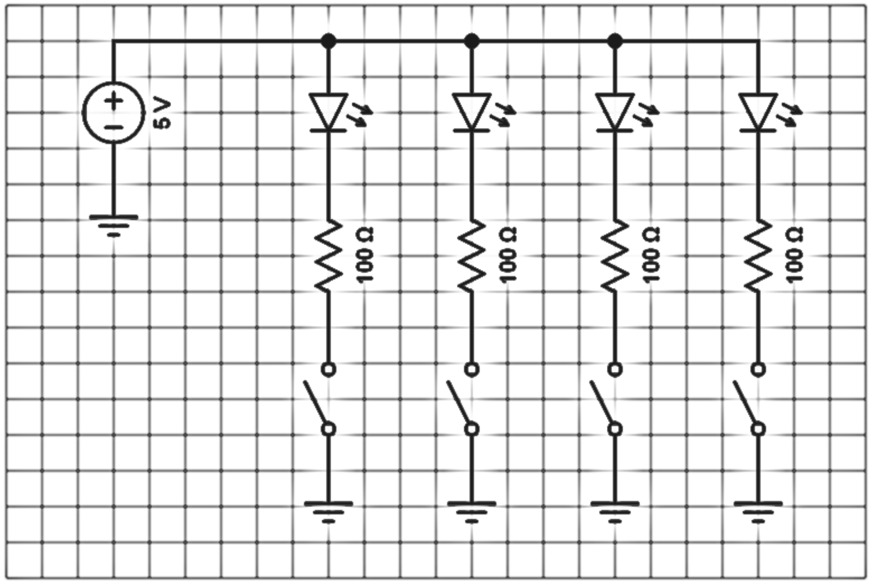
- But how do I get the right lights to light up at the right time?
  - Leverage the diode nature of the LED

LED Wiring Diagram

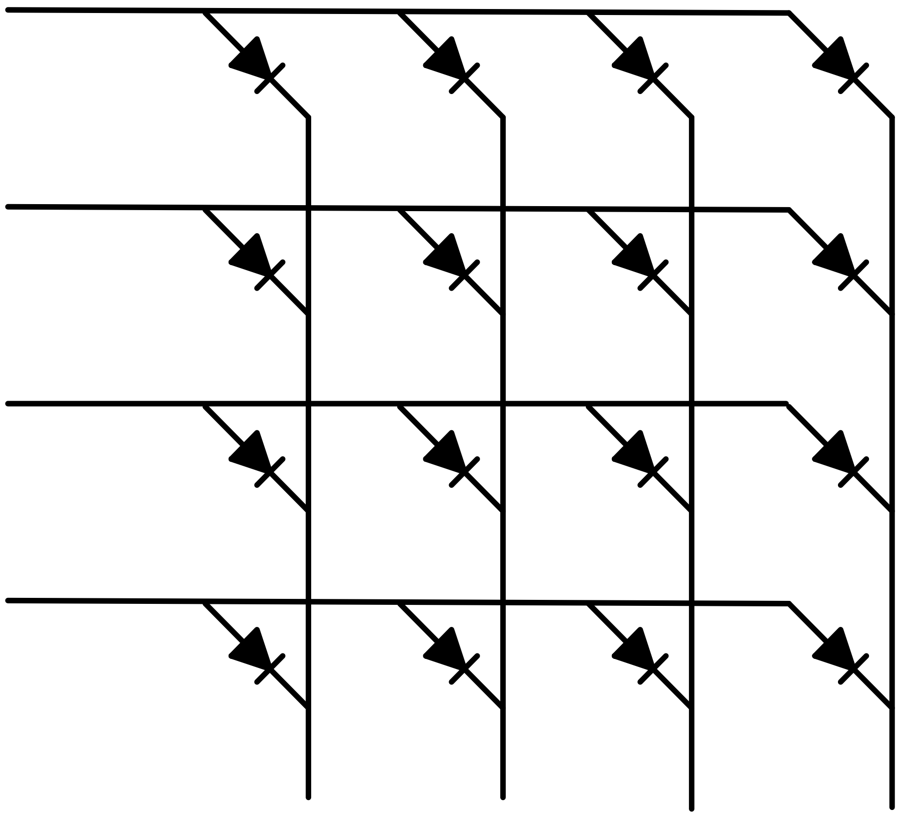


LED Wiring Diagram - EveryCircuit



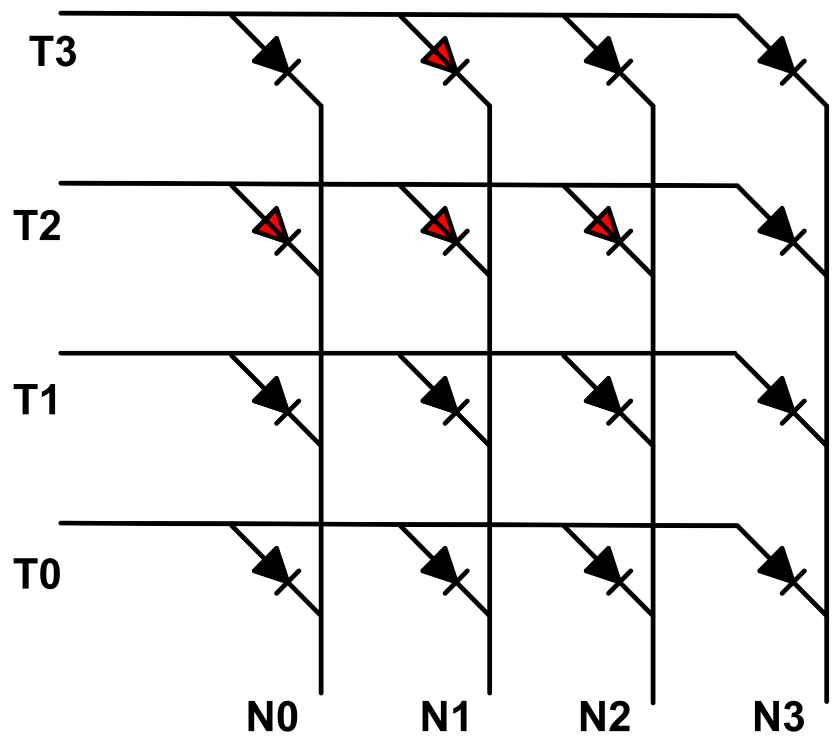


LED Array Wiring Diagram



Testing Our Understanding

- If we use time division multiplexing to drive the LED array
  - How do you light up the red LEDs?
  - How many time slots?



BUILDING THE LED CUBE

Watch Mark Horowitz build an LED cube: Tutorial 4 on class website

https://www.youtube.com/watch?v=4u4eAnd1yEk&feature=youtu.be

The Numbers

- You have 36 LEDs
  - Each has 2 wires, generally are soldering two wires together
  - End up with 36+ solder joints

- This is not a huge amount
  - But it is larger than a few
  - And there are not to a well defined structure
    - They are not going to be fixed on a printed wiring board

- You should be thinking about
  - How to minimize the work you need to do
  - How to catch your mistakes early
    - When they are easier to fix, and so you don’t repeat them

Making the Soldering Easier

- Since we need to do the same thing multiple times
  - Need to build each plane
    - And each plane consists of a set of rows

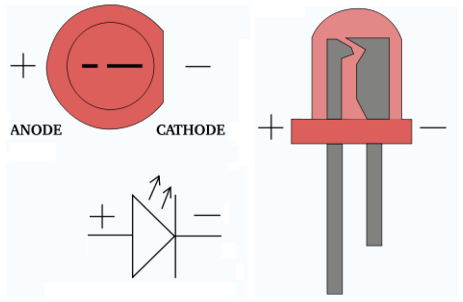
- What can we do to make the task easier?
  - Think about how the lights will connect to each other
    - Is there a way to make the connections easier to solder
  - Before building the cube, test out your ideas
    - Try before committing to a method
    - Optimizing your technique might save you time in the long run
    - If things are not working, think about what is going wrong
      - Why isn't it working, and what can you do to fix that problem

Repetition Is Good

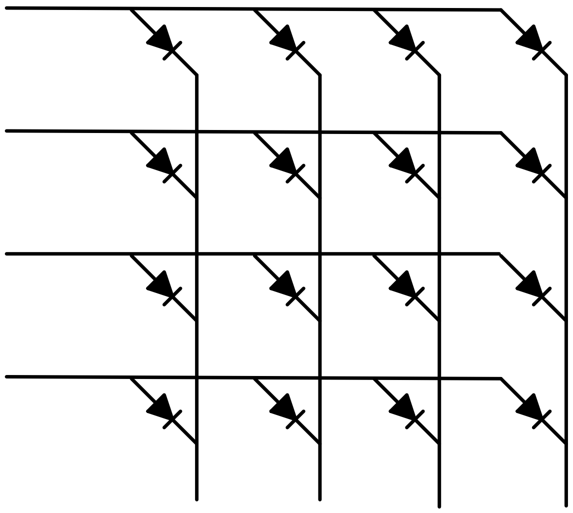
- While doing the same thing multiple times gets boring
  - That is not all bad
- Boring means you don’t need to think very much to do the job
  - At Stanford, that can be relaxing ;-)
- It also means that your design will be modular
  - You are building the same part multiple times
    - So you can use one jig to test all the modules
  - Can even build a jig to help you do the soldering
    - Soldering things hanging in space is hard
      - You don’t have enough hands

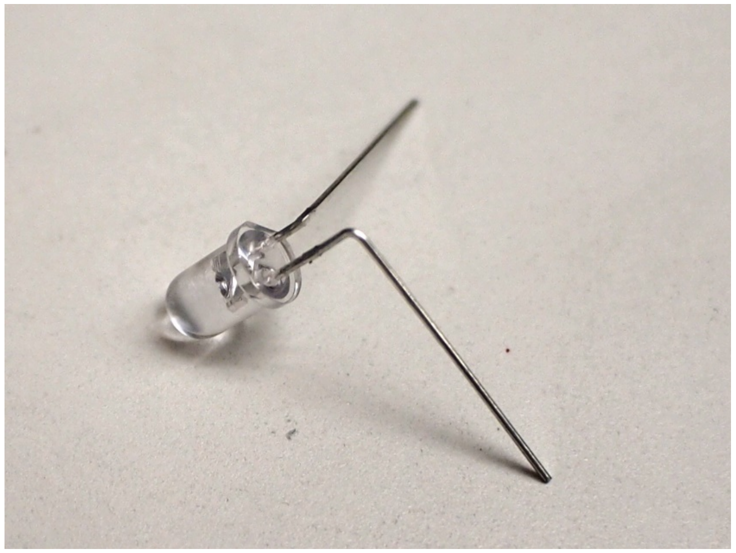
For the LED Array

- Want the '+' to run horizontally
  - And the '-' to run vertically

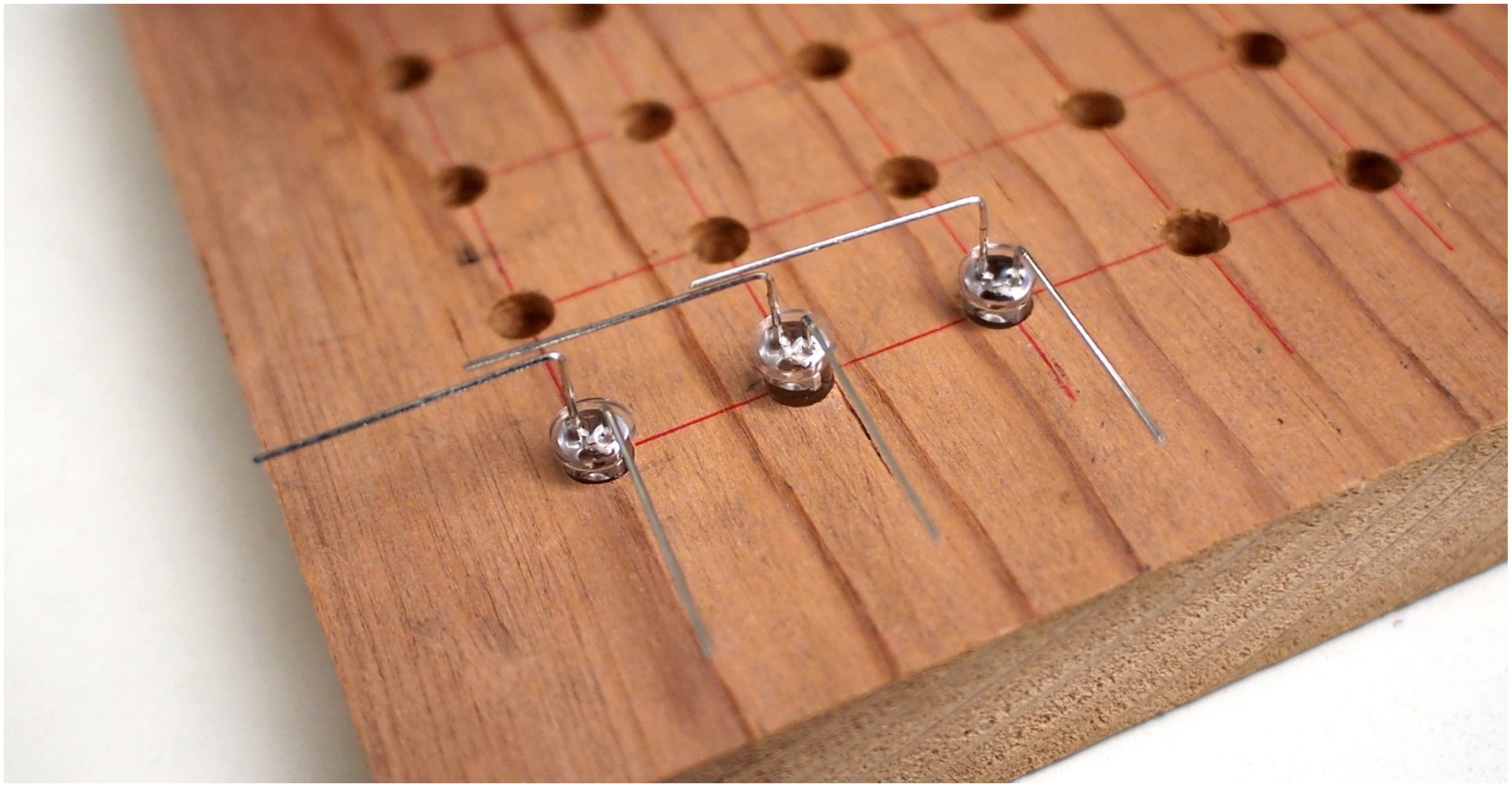


- Bend the leads of each LED 90^o to each other
  - And make them different heights!

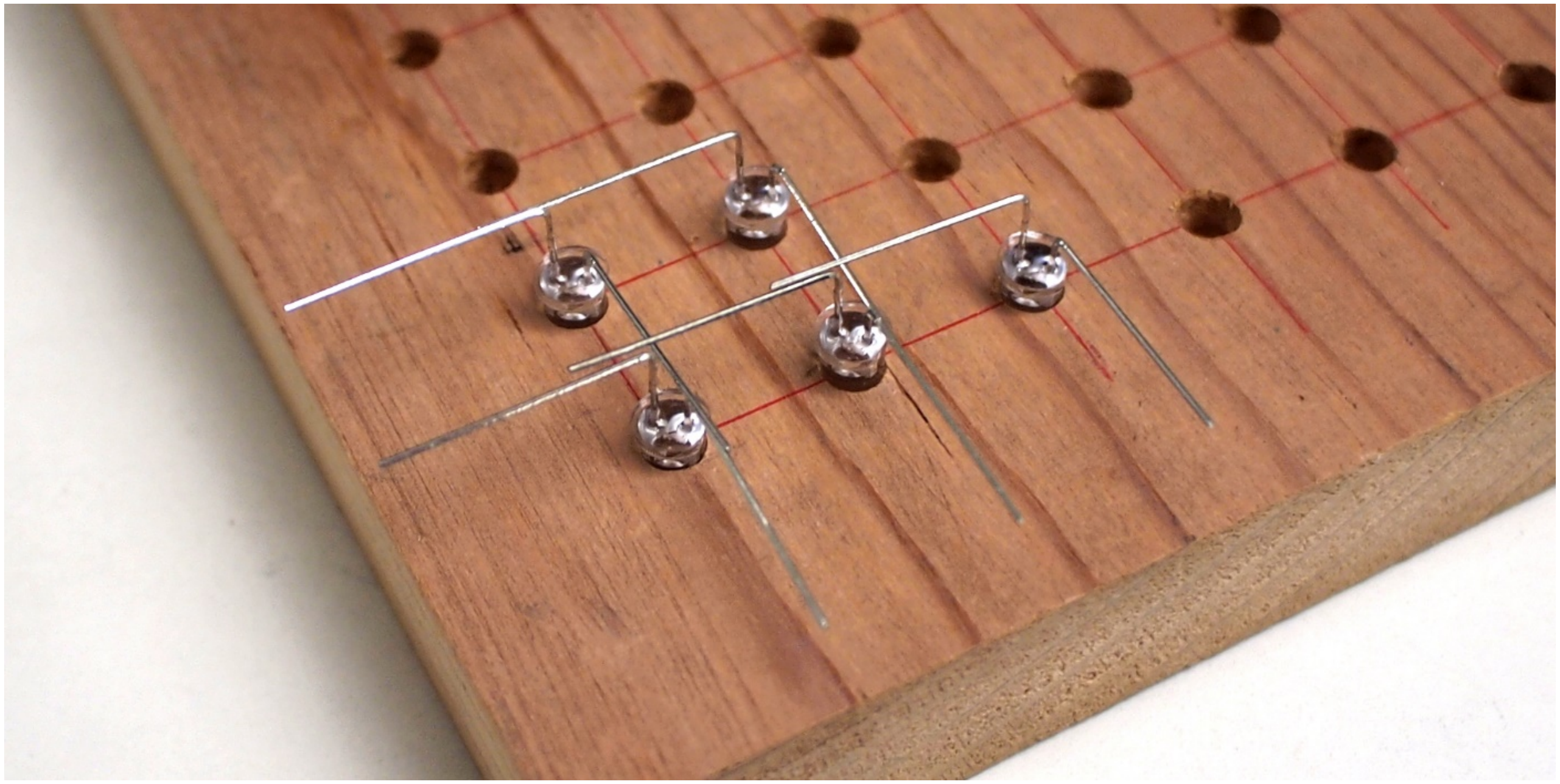




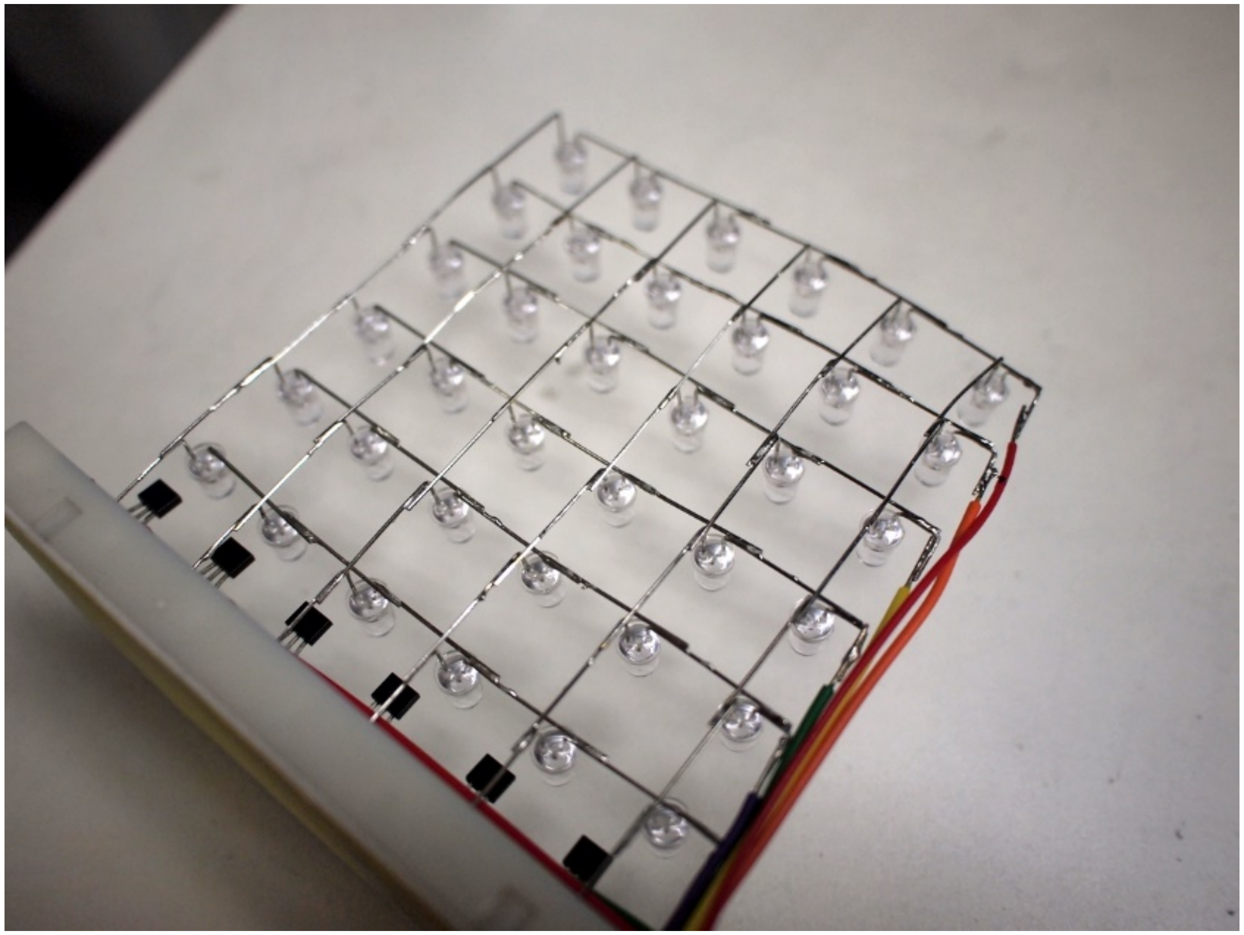
Building a Row



Adding Next Row



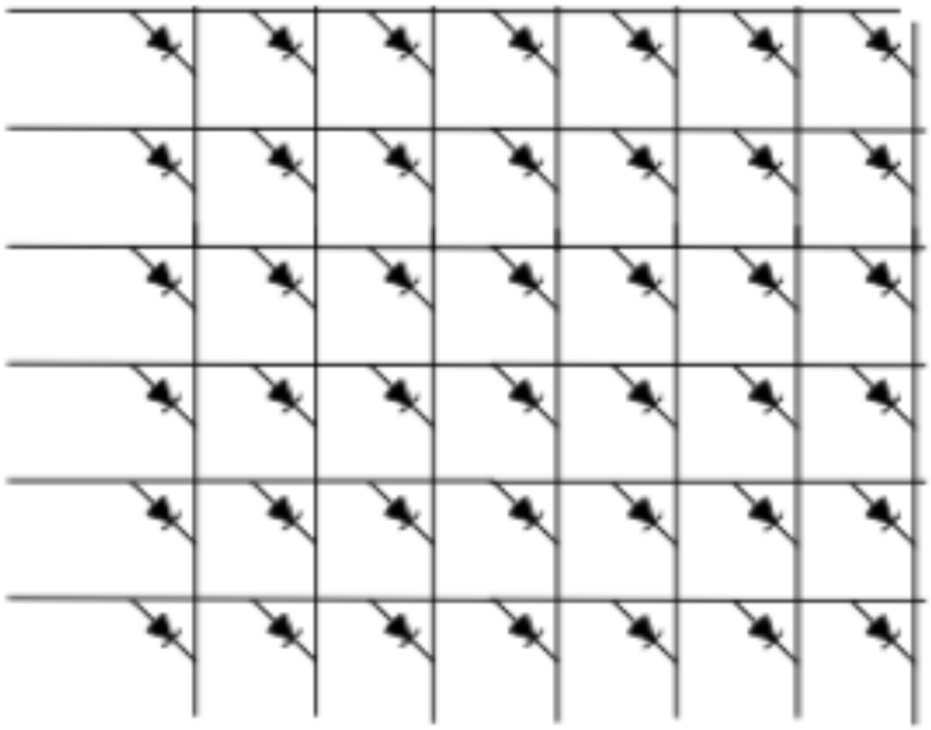
Final Array



LED Plane Driver

So We Soldered Our Plane…

- We have an 6 x 6 array
  - Logically looks like:
  - Physically it is different



- Need to drive it
  - To light up the lights
- For independent light control
  - Either only one + wire higl
  - Or only one - wire low
  - Max of 6 LEDs on at once

The LED Driver Current Requirements

- Look at current requirements
  - - wires drive 6 LEDs; + wires drive 6 LEDs
  - But we decided to drive only + wire high at a time

- So each - wire can drive only one LED that is on at any time
  - But each + wire can drive 6 LEDs that are on at on time

- How much current will each LED take?
  - Series resistance is about 100Ω total
    - 82Ω from explicit resistor, 20-30Ω from pin
  - Voltage drop across this resistance is 2-3V
  - Current is approximately 20-30mA (for green/blue or red)

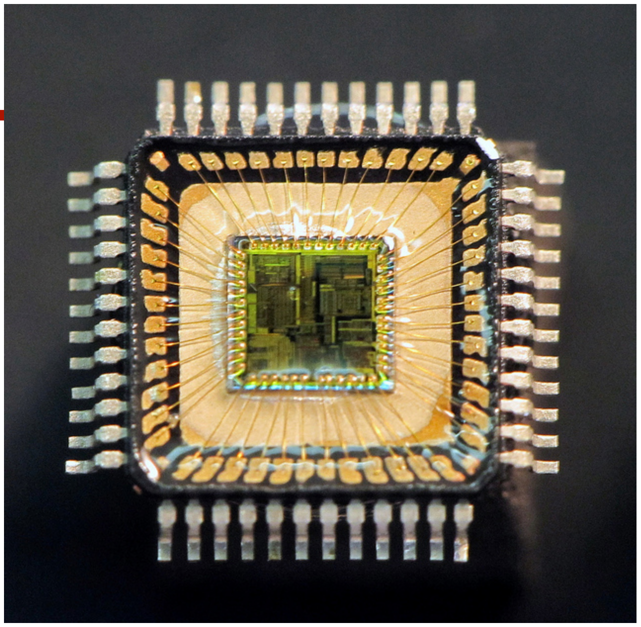
Arduino Current Limitations

- If you read Arduino specs
  - Each pin can drive 40mA
  - Whole chip should drive less than 200mA

- Pin current is limited by MOS devices on the chip
  - Have measured 20 ohm
    - If you measure the current it is less ~60mA
    - This makes sense. If the spec is 40mA
      - That is the guaranteed value. Nominally it will be higher

https://www.flickr.com/photos/john_gammell/4562350638/

Max Chip Current



- Why is there a max total chip current?
  - When current flows out of pin
    - It must flow in from somewhere
      - Σ i = 0
    - Somewhere is the Gnd pin
      - Or Vdd
- The current must flow through some small wires
  - Called bonding wires, which connect chip to package
  - Too much current and these wires become fuses
    - Poof, and it is gone
  - Very conservatively spec’d. Probably 400mA is ok
    - But remember Poof is possible and permanent

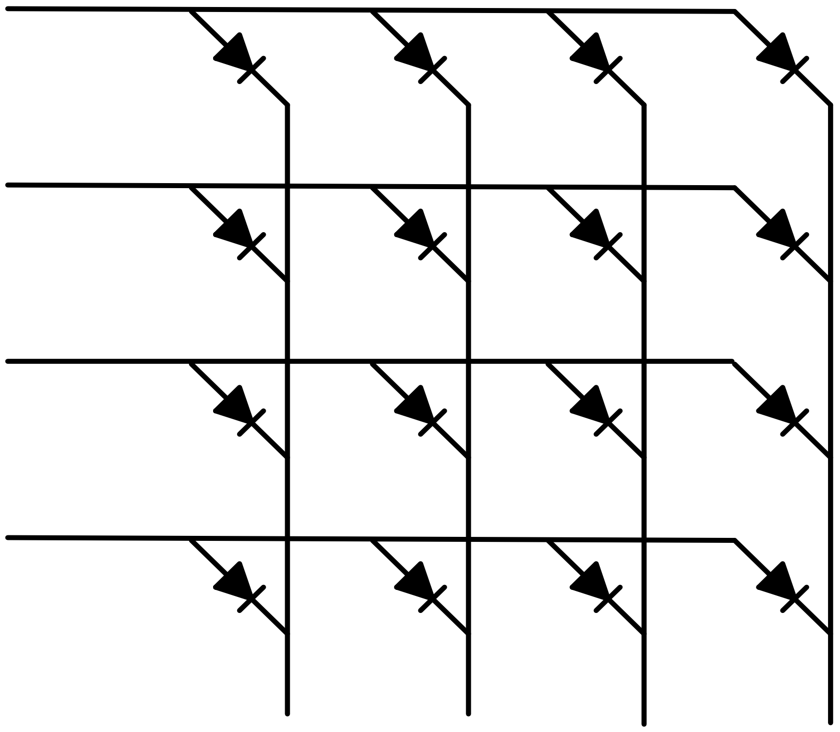
How to Drive Your LEDs

- Arduino can drive the - wires directly
  - Use 82Ω series resistor to limit current
  - Current will be 20mA for blue/green LEDs, 30mA for red
- Total Arduino current will be around 240mA max
  - Above spec, but will be ok
- + wire driver will need to supply 160-240mA
  - Arduino can't drive that current
  - Will need an external driver for that
  - What type or driver do we need?

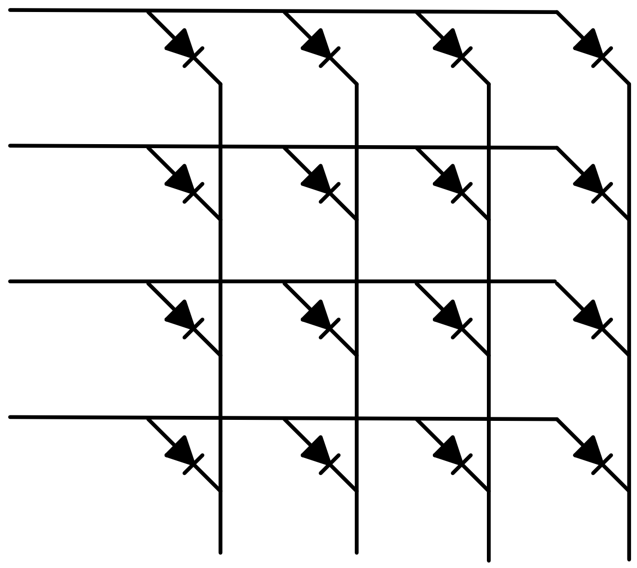
Driving the + Wires of the Cube

- Want to build as little as possible
  - Need to build 8 of them

- What do we need?
  - Need to connect to Vdd
  - Need to disconnect from Vdd
  - Don't need to drive it to Gnd



Where To Put The Resistor and Transistor?

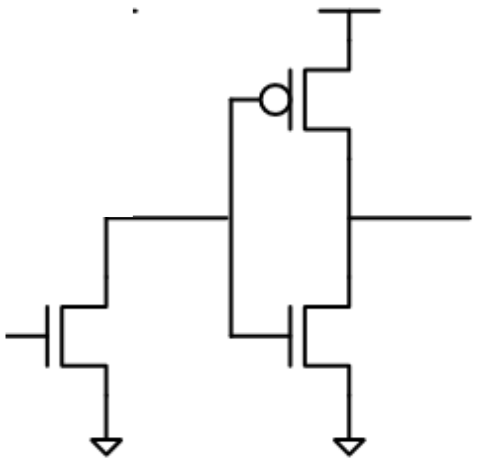


Remember, No Current Doesn't Mean 0V

- When the left nMOS is off
  - What is the input voltage of the inverter?
    - There is no current from node A
    - So any voltage is a solution to KCL!
  - It could be anything

- If you a voltage signal
  - You need to drive it to Vdd and to Gnd

- So why can we use only a pMOS transistor?



A Disconnect Switch Means No Current

- In a digital system, the pMOS transistor can:
  - Connect the source and drain (Vg = Gnd; Vs=Vdd)
  - Leave the source and drain unconnected (Vg=Vs)

- If the quantity you are interested in is current
  - Then you know that i = 0, when Vg=Vs
  - Since current can’t flow through a disconnected switch
    - Called an open circuit (no current path)

- So no LED can turn on if the pMOS transistor is off
  - Since no current can flow through them

Learning Objectives

- Understand how to use LEDs in simple circuits
- Be able to use time division multiplexing to control LEDs
  - Control n^2 lights using only 2n wires
  - By turning n lights on in n different time slices
- Be able to create a 2-D plane of LEDs that can be multiplexed
  - And be able to stack planes to create a cube
- Be able to create the drive circuit for your LED array (if time)

Reading

- Failing Fast
  - http://www.thedailybeast.com/articles/2014/01/05/fail-fast-fail-often-how-losing-can-help-you-win.html

GOOD DESIGN PRACTICES

Debugging

- Debugging complex systems is very hard
  - It is too hard to figure out what should be happening
  - This is why we try to create clean software interfaces

- Debugging hardware has similar issues
  - Want to check well defined interfaces
    - First check your tools/inputs
      - Measure inputs (including Vdd)
      - And check that the scope/meter are working
    - Then check the smallest subcircuit you can
  - Unfortunately hardware can break
    - Could work yesterday and not work today
      - Good to record what a working solutions looks like
      - Check these voltages when it stops working

Debugging Hardware / Software Systems

- Often project has hardware and software
  - What happens when this system breaks?

- Again isolation is the key
  - In addition to having the main software
  - Always have test software
    - This should just double check that the hardware is working
    - E.g. checking that all the LEDs light in the right order
  - Test jigs are great
    - Either software only (output on a screen/ computer)
    - Or other hardware/software that is known to work
      - Can test/debug your software/hardware

Debugging the Smart Useless Box

- Isolation is key:
  - In prelab
    - Tested power inverters in simulation
    - Wrote test programs to read switches and drive the motor
  - In lab
    - Measure the output of the Arduino directly
    - Test power inverters directly
    - Use test programs to debug your wiring
- Use std simple debugging
  - Use Serial.println(keyVariable)
    - To help figure out where the code is going wrong
  - Check for the std stupid "C" errors
    - Like using "=" in an if statement

Archive Your Test Programs

- Remember that hardware sometimes stops working
  - Either because you hit a bug you missed before
  - Or a component that used to work isn't anymore
  - Need to figure out which is which

- Having code that can measure/test hardware is crucial
  - Then when system stops working can check hardware is ok
  - Or narrow down to which component is no longer working

Minimizing Work Required

- When you start a project you have a choice
  - What will you design
  - What will you borrow from others
- Design only the critical stuff you need
  - Borrow the rest from others
  - Don’t invent stuff you don’t need
    - Good enough is really good enough
- Think about the minimal system that is ok
  - Think also about the things you might like to add later
  - But build this minimal system first
    - When it works, you can then start adding stuff

Hacking is Rarely the Best Approach

- If you are going to do this only once
  - And will never use it again
  - You can hack your way through it, just to get it done
- If you are going to do this more than once
  - Creating an efficient method will save you time in the end
    - Building jigs is often a time saver
- And remember you rarely do things only once
  - Because you make mistakes
  - Or performance / aesthetics of unit is important
  - Remember if you are doing something for the 2nd time
    - There will probably be a 3rd, so think about your methodology

Failing Fast

- People don’t like to fail
  - So they tend to avoid areas that they are worried about
  - This is exactly backward to being successful

- You actually want to fail as fast as you can
  - If something is not going to work
    - Why waste time working on the other stuff?
    - Don’t you want to know ASAP so you can’t stop working on it?
  - Should always work on the most risky part of the project first

- Mantra in entrepreneurship
  - Often knowing what the customer wants is the hardest point
    - So try many products/variations until you find one that “wins”

Failing Fast

- It is easier to modify designs early in the process
  - If something is not going to work
    - Find out early when you can change other components/project
  - Have less sunk investment to lose

- Failing is learning
  - You fail because of an conceptual error
    - The world is different than what you thought
  - Learning the way the world really is, is important

- Debugging is easier in smaller chunks
  - Want to test the smallest chunk possible
  - If something is risky, would like to isolate it to debug it

Learning Objectives

- Hacking your way to a solution is rarely the optimal approach
  - Rarely do anything once, so script stuff, and save it
  - Clean interfaces are key for isolation
- Understand why you should try to “fail fast” in your projects
  - Do risky stuff first, not last, to minimize cost of failure
  - Faster learning cycles
- Be able to create the drive circuit for you LED cube
  - And map control signals to x, y, z position