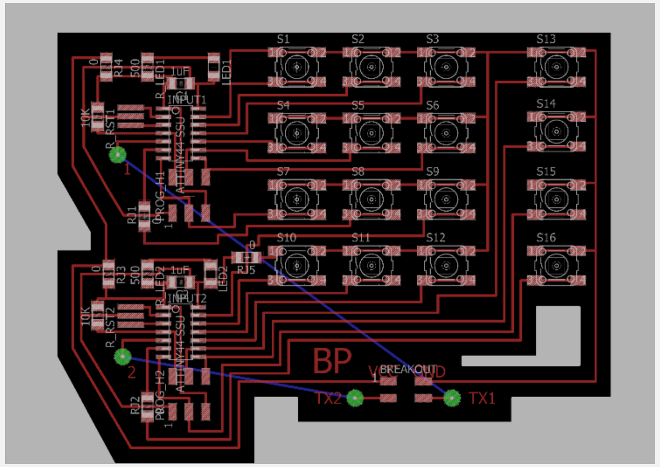
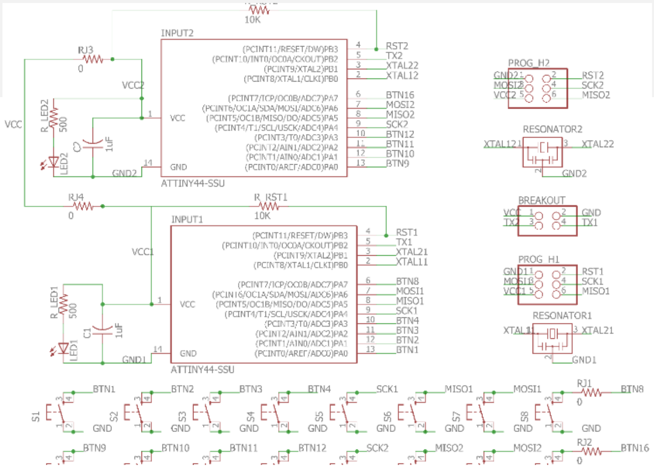




Electronics Production and Design

brian_plancher@g.harvard.edu
Fall 2021

The road ahead

This week

Electronics Production

- Mill and stuff a circuit board

The road ahead

This week

Electronics Production

- Mill and stuff a circuit board

2 weeks from now

Electronics Design

- Design your own circuit board
- Mill and stuff it

The road ahead

This week	2 weeks from now	4 weeks from now
Electronics Production	**Electronics Design**	**Embedded Programming**
• Mill and stuff a circuit board	• Design your own circuit board • Mill and stuff it	• Design your own circuit board • Mill and stuff it • Program it

The next few weeks have a simple task!

1. Already know basic electrical engineering

The next few weeks have a simple task!

1. Already know basic electrical engineering
2. Use it to design a custom circuit board in a new software program

The next few weeks have a simple task!

1. Already know basic electrical engineering
2. Use it to design a custom circuit board in a new software program
3. Mill it and solder on all the parts right

The next few weeks have a simple task!

1. Already know basic electrical engineering
2. Use it to design a custom circuit board in a new software program
3. Mill it and solder on all the parts right
4. Already know basic programming

The next few weeks have a simple task!

1. Already know basic electrical engineering
2. Use it to design a custom circuit board in a new software program
3. Mill it and solder on all the parts right
4. Already know basic programming
5. Write a custom program to test your board

The next few weeks have a simple task!

1. Already know basic electrical engineering
2. Use it to a new software
3. Mill it an
4. Already know basic programming
5. Write a custom program to test your board

So if you are now feeling like…..


I HAVE NO IDEA
WHAT I'M DOING


RELAX
WE GOT THIS

The road ahead

Today's Focus

This week	2 weeks from now	4 weeks from now
Electronics Production	Electronics Design	Embedded Programming
• Mill and stuff a circuit board	• Design your own circuit board • Mill and stuff it	• Design your own circuit board • Mill and stuff it • Program it

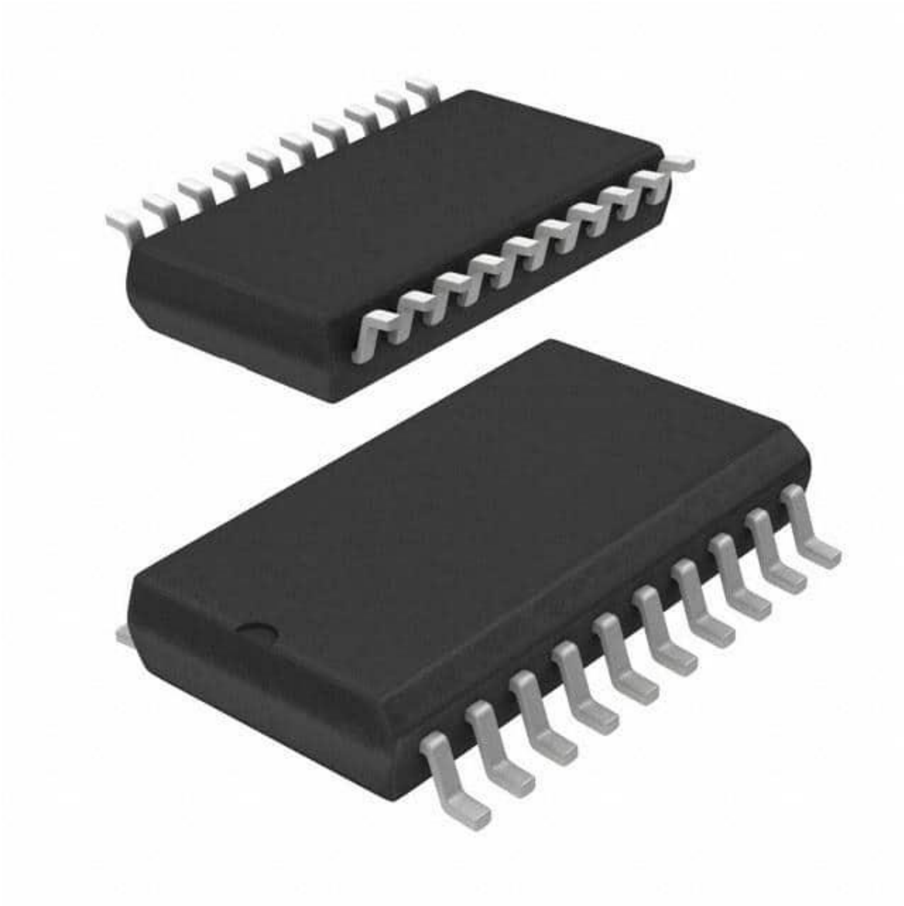
A short outline for today

1. **Almost all you need to know about Electrical Engineering**
2. Almost all the tips you need to design custom boards
3. Almost all the steps it will take to produce a custom board

Why do I even need to know anything about electrical engineering?

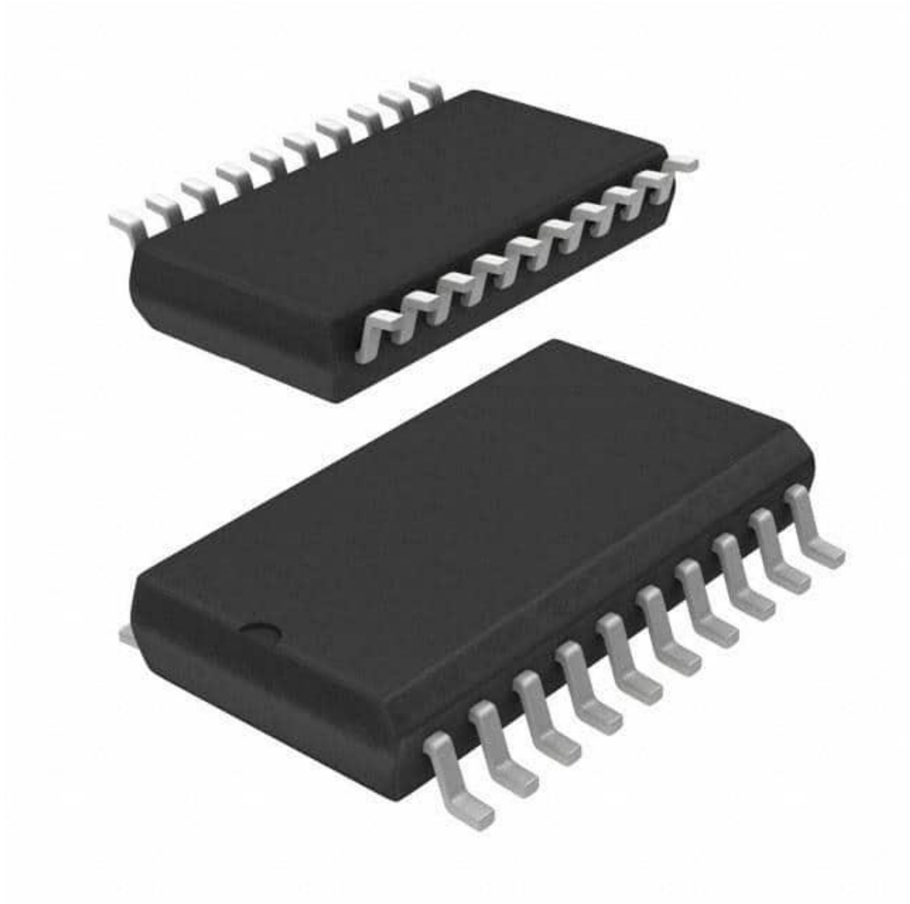
Why do I even need to know anything about electrical engineering?

Why do I even need to know anything about electrical engineering?



Why do I even need to know anything about electrical engineering?

$\neq$



Why do I even need to know anything about electrical engineering?

We are going to learn how to make our own custom circuit boards that connect inputs and outputs to our own microcontrollers!

Ohm's Law:

$$V = I * R$$

Voltage (measure in volts)

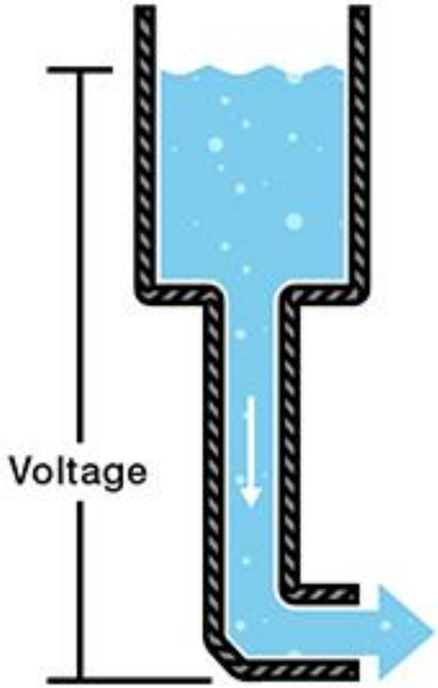
I: Current (measure in amps)

Resistance (measured in ohms)

https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law/all

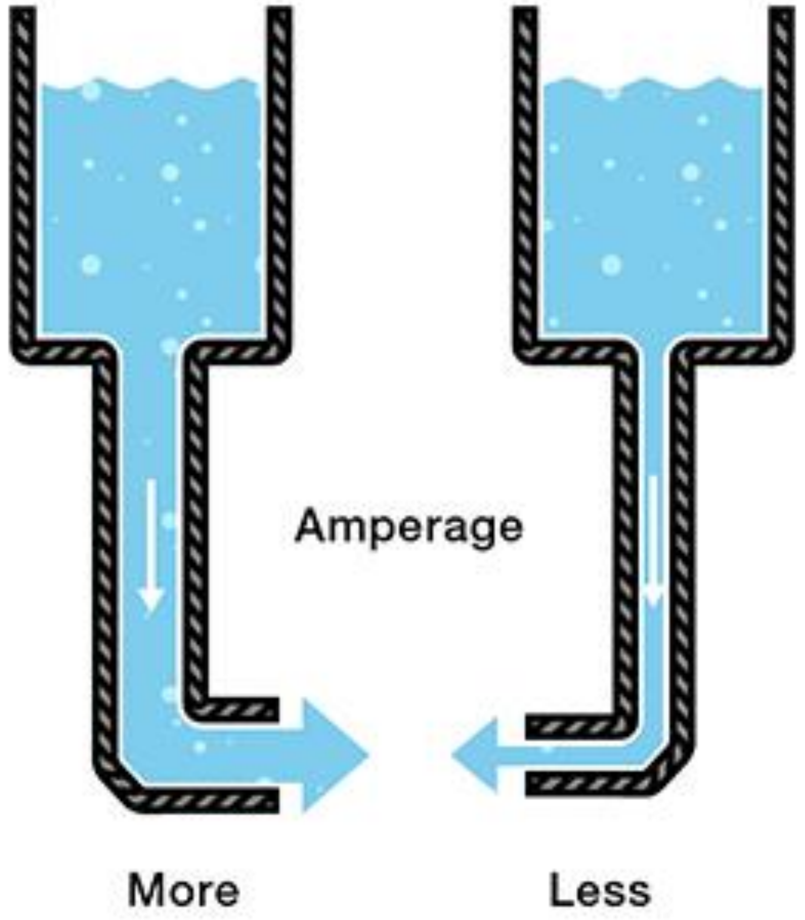
Voltage

Measures the **difference in electrical potential** between two points – often an input voltage (vcc) and ground (gnd)



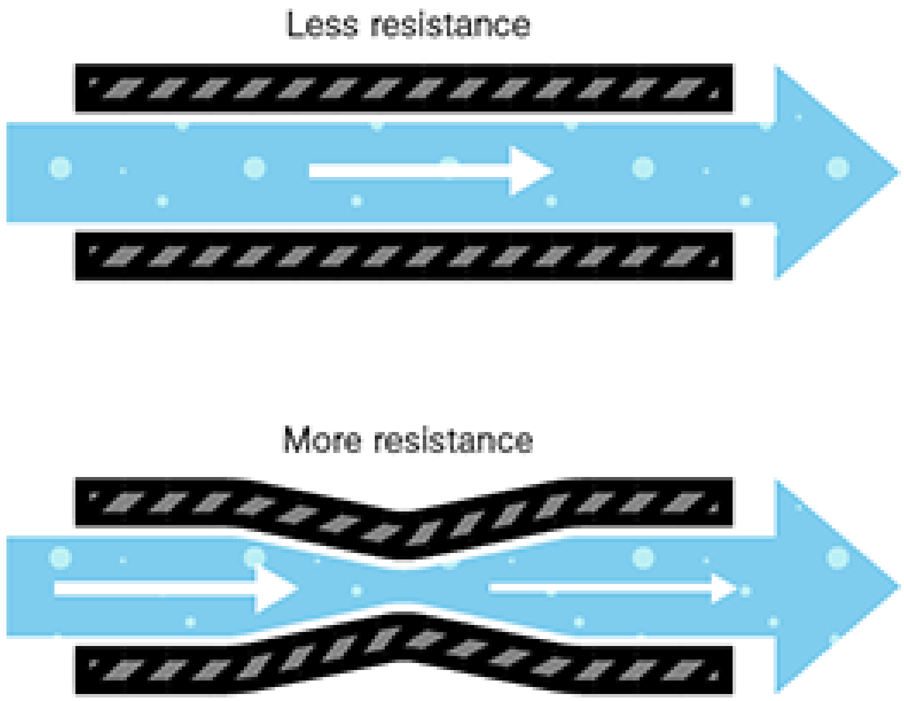
Current

Measures the **rate of flow of electrons** in a circuit



Resistance

Measures **how hard it is for electrons to move** through a circuit



Ohm's Law:

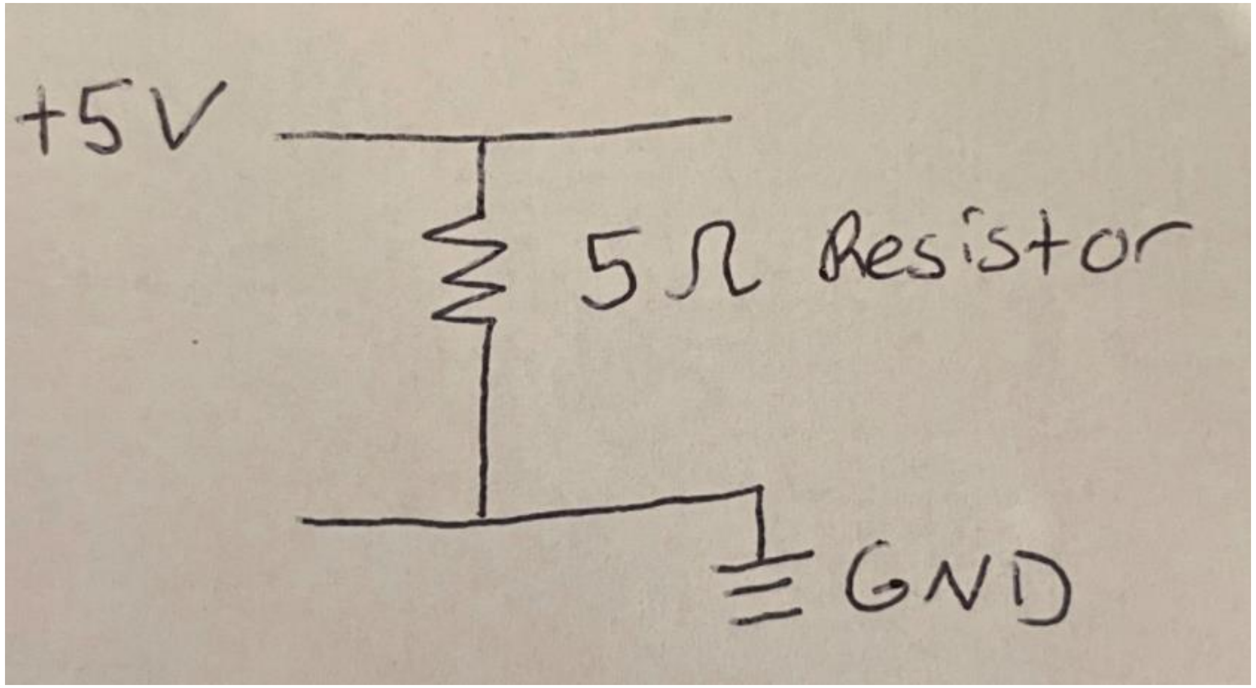
$$V = I * R$$

Voltage (measure in volts)

I: Current (measure in amps)

Resistance (measured in ohms)

How much current goes through this resistor?



Ohm's Law:

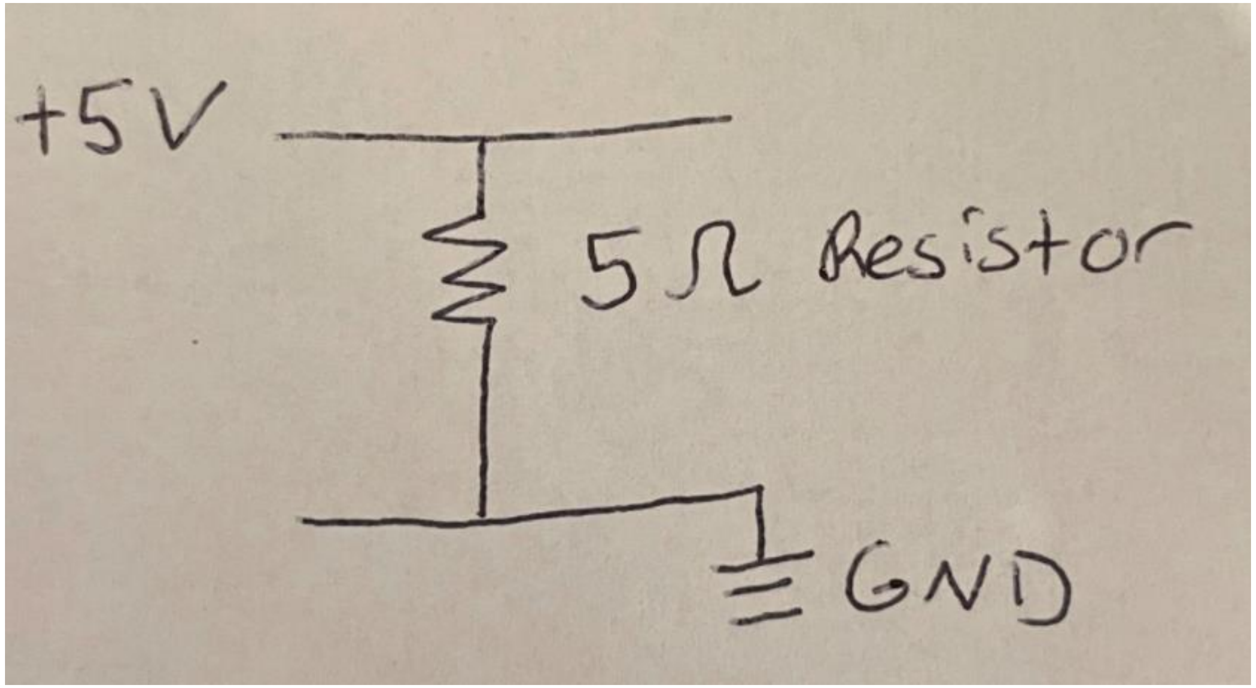
$$V = I * R$$

Voltage (measure in volts)

I: Current (measure in amps)

Resistance (measured in ohms)

How much current goes through this resistor?



1A

Ohm's Law:

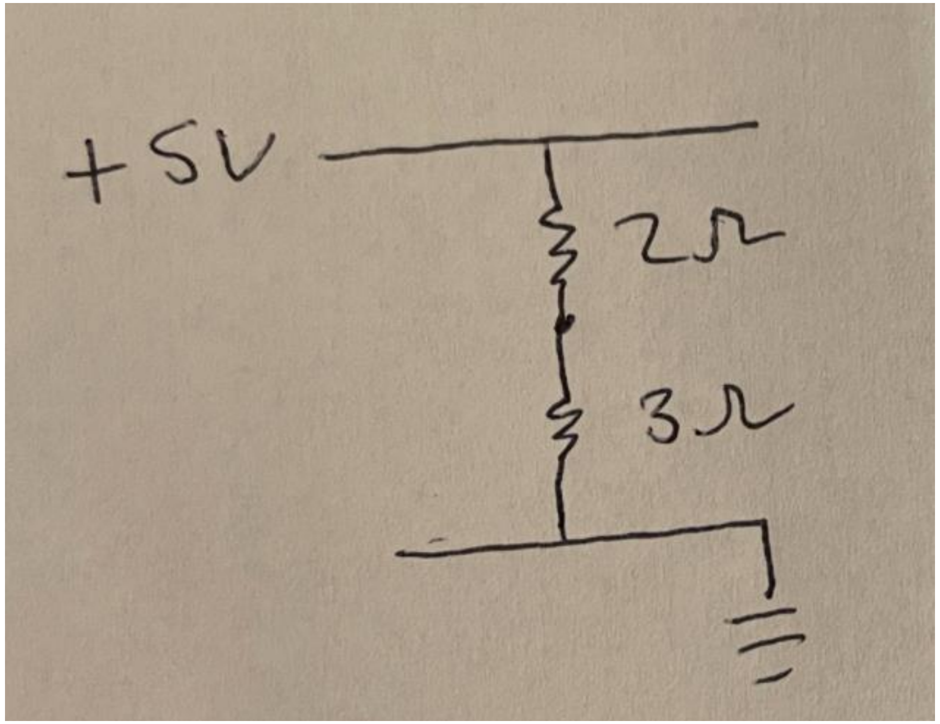
$$V = I * R$$

Voltage (measure in volts)

I: Current (measure in amps)

Resistance (measured in ohms)

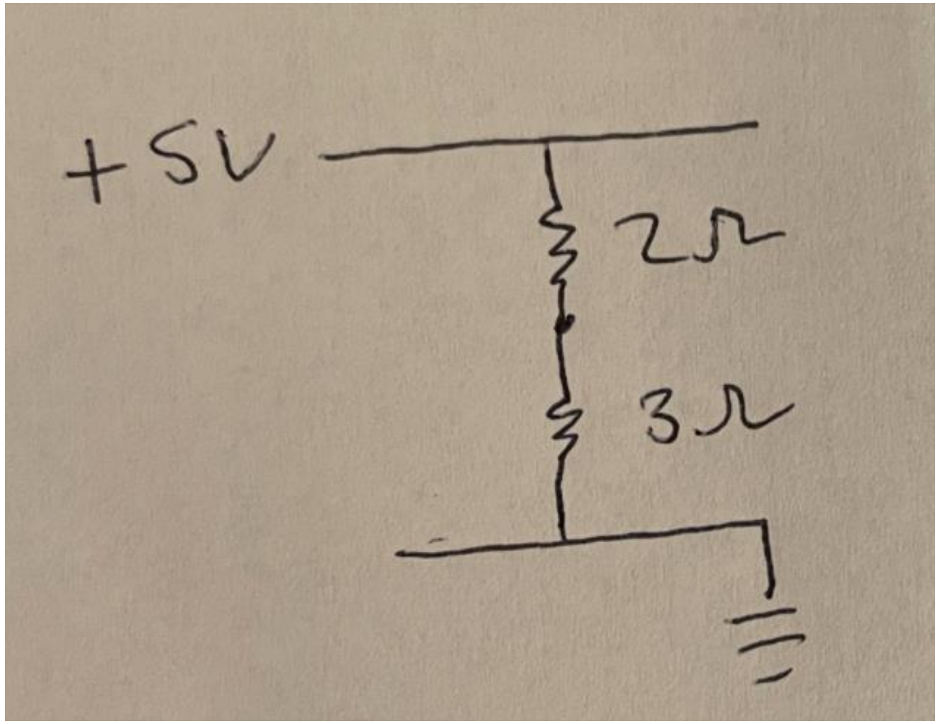
How about these resistors?



Ohm’s Law:

- Resistance in series adds
- To learn more about series and parallel check out this link: https://en.wikipedia.org/wiki/Series_and_parallel_circuits

How about these resistors?



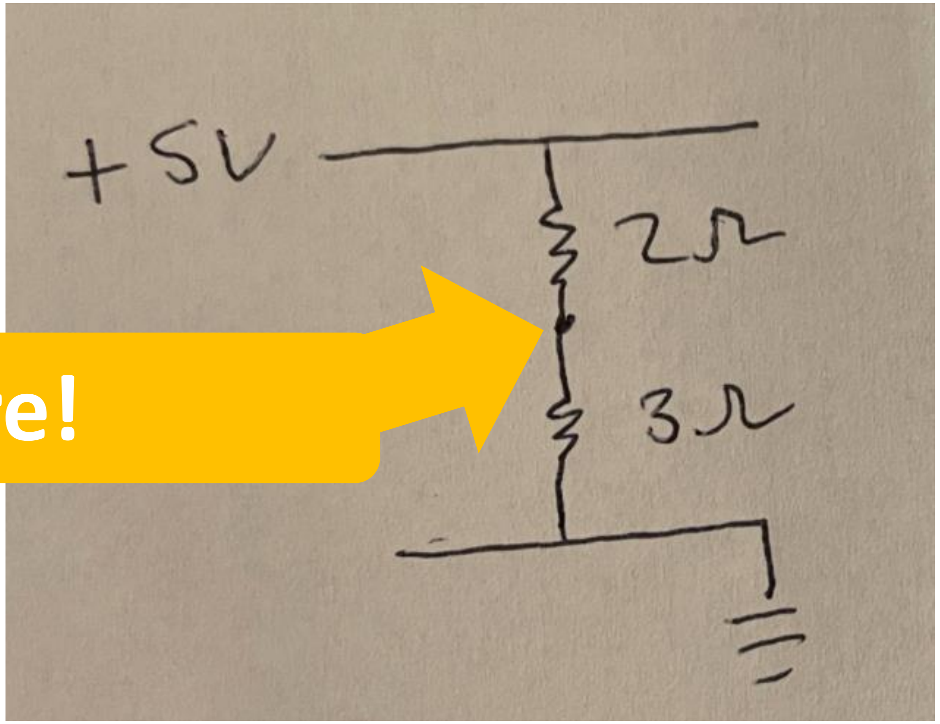
1A

Ohm's Law:

How about these resistors?

- Resistance in series adds
- To learn more about series and parallel check out t
https://en.wikipedia.org/wiki/Series_and_parallel_circuits

+3V here!



1A

Ohm's Law:

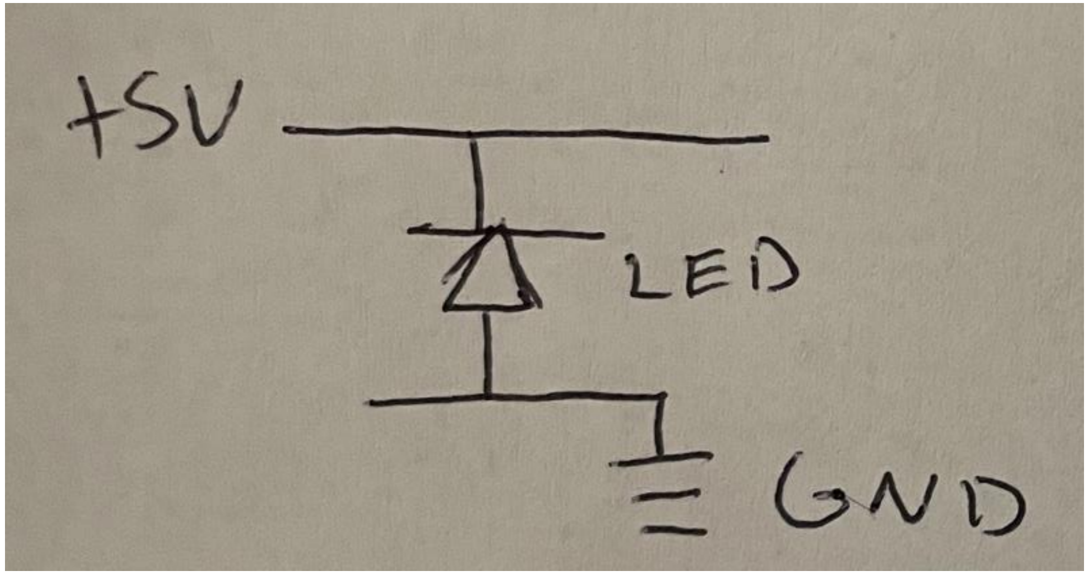
$$V = I * R$$

Voltage (measure in volts)

I: Current (measure in amps)

Resistance (measured in ohms)

What about this LED?



Ohm’s Law:

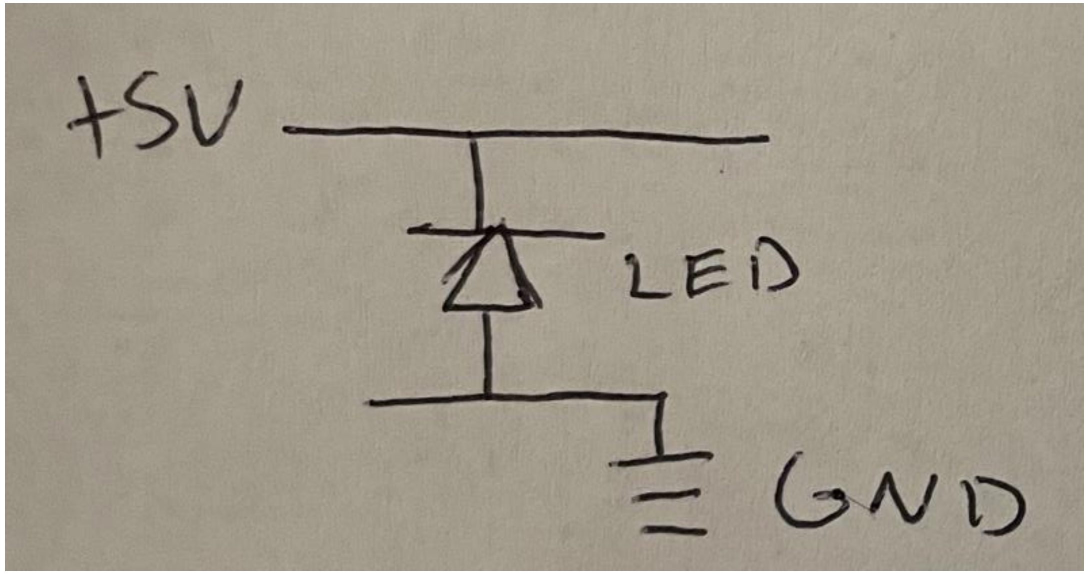
$$V = I * R$$

Voltage (measure in volts)

I: Current (measure in amps)

Resistance (measured in ohms)

What about this LED?



Trick Question – 0A
All diodes are one way!

Ohm's Law:

$$V = I * R$$

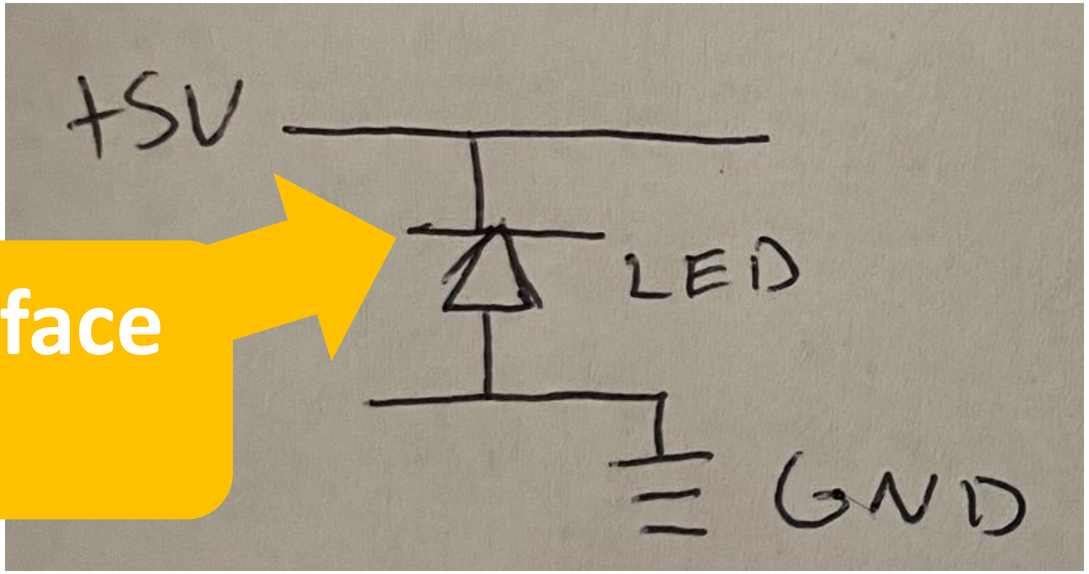
Voltag

I: Curre

Resistance (measured in ohms)

Green line on surface mount parts!

What about this LED?



Trick Question – 0A
All diodes are one way!

Ohm's Law:

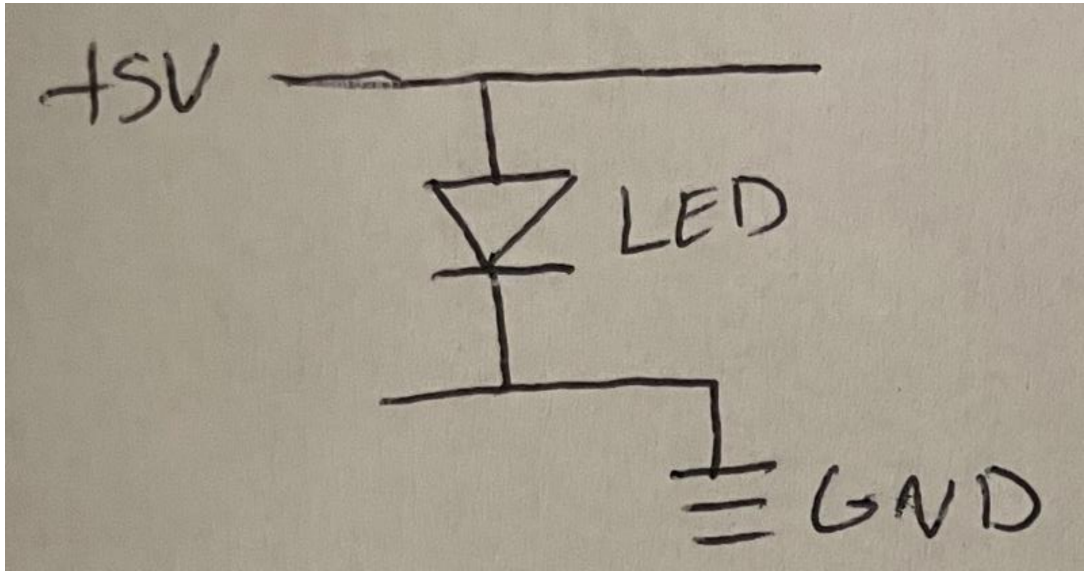
$$V = I * R$$

Voltage (measure in volts)

I: Current (measure in amps)

Resistance (measured in ohms)

Ok so what about this (correct direction) LED?



Ohm's Law:

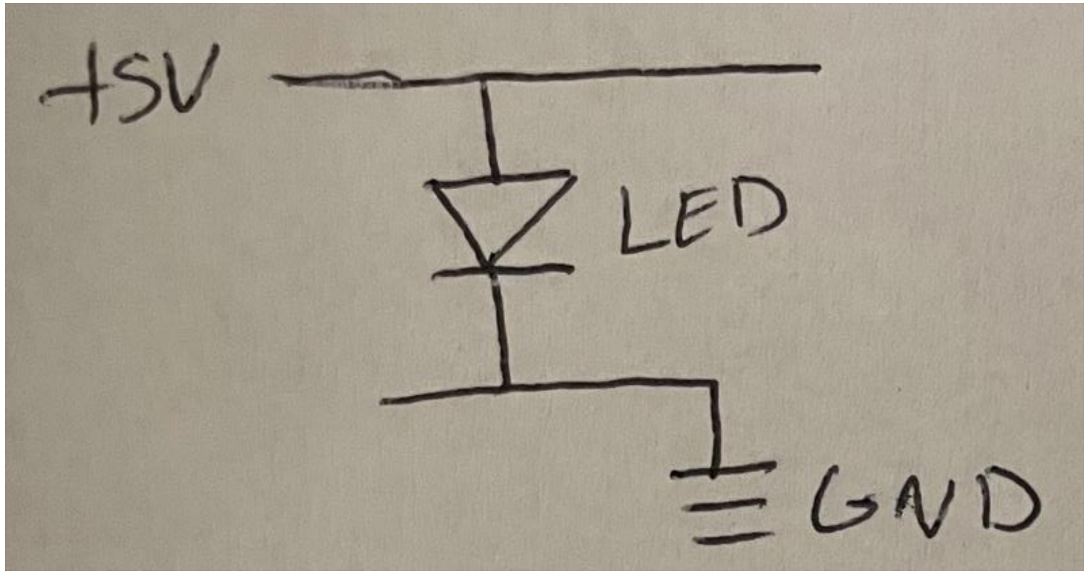
$$V = I * R$$

Voltage (measure in volts)

I: Current (measure in amps)

Resistance (measured in ohms)

Ok so what about this (correct direction) LED?



Trick Question Again – ∞A
Diodes have 0 resistance!

Ohm's Law:

$$V =$$

Voltage (r

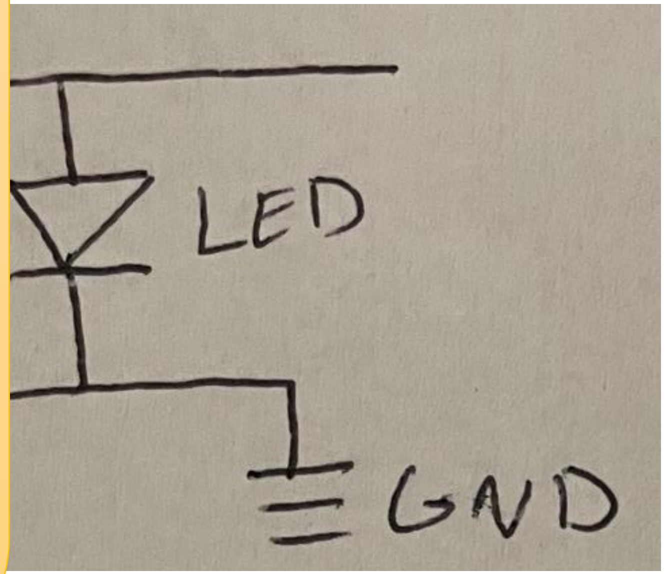
I: Current

Resistance

Infinite current will go BOOM!

Ok so what about this

ct direction) LED?



Trick Question Again – ∞A

Diodes have 0 resistance!

Ohm's Law:

Ok so what about this
t direction) LED?

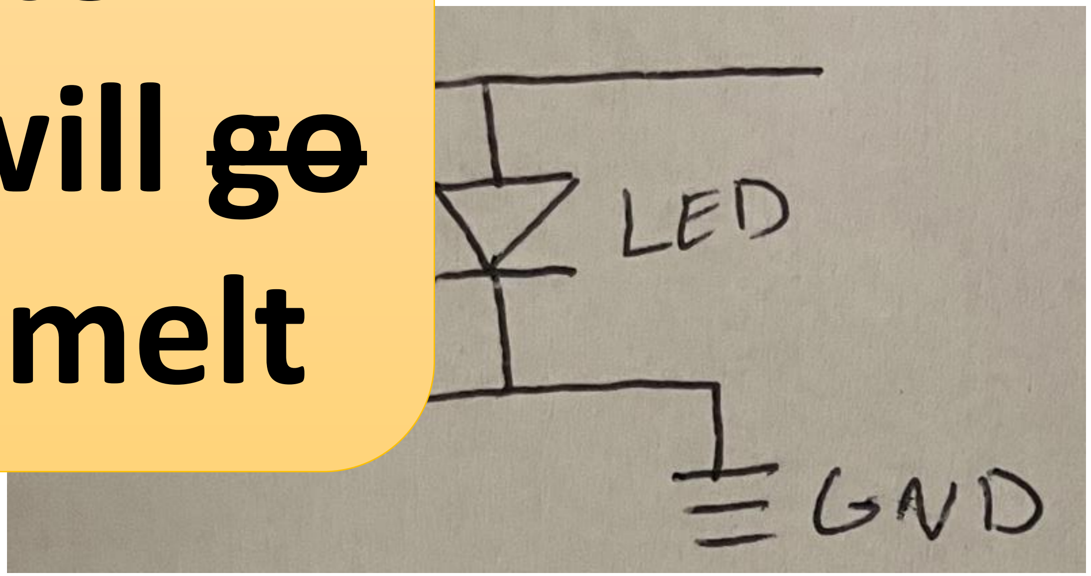
$V =$

Voltage (r

I: Current (me

Resistance (measured in ohms)

Infinite current will ~~go BOOM!~~ melt



Trick Question Again – ∞A
Diodes have 0 resistance!

Ohm's Law:

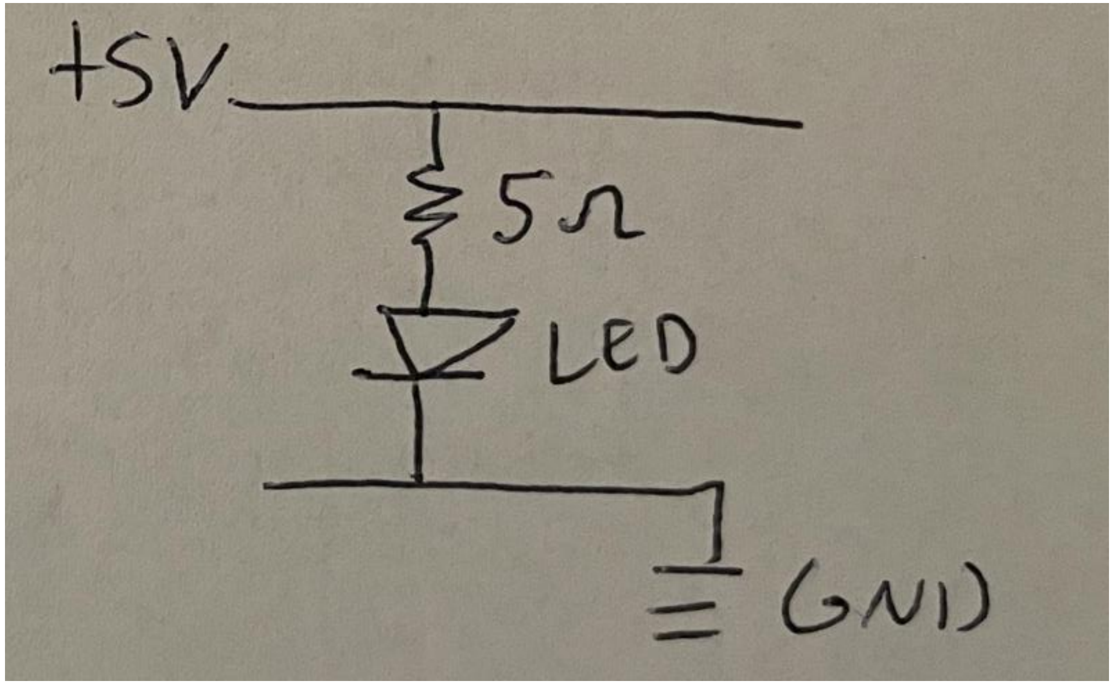
$$V = I * R$$

Voltage (measure in volts)

I: Current (measure in amps)

Resistance (measured in ohms)

Ok so what about this (correct direction) LED with a **current limiting resistor!**



Ohm's Law:

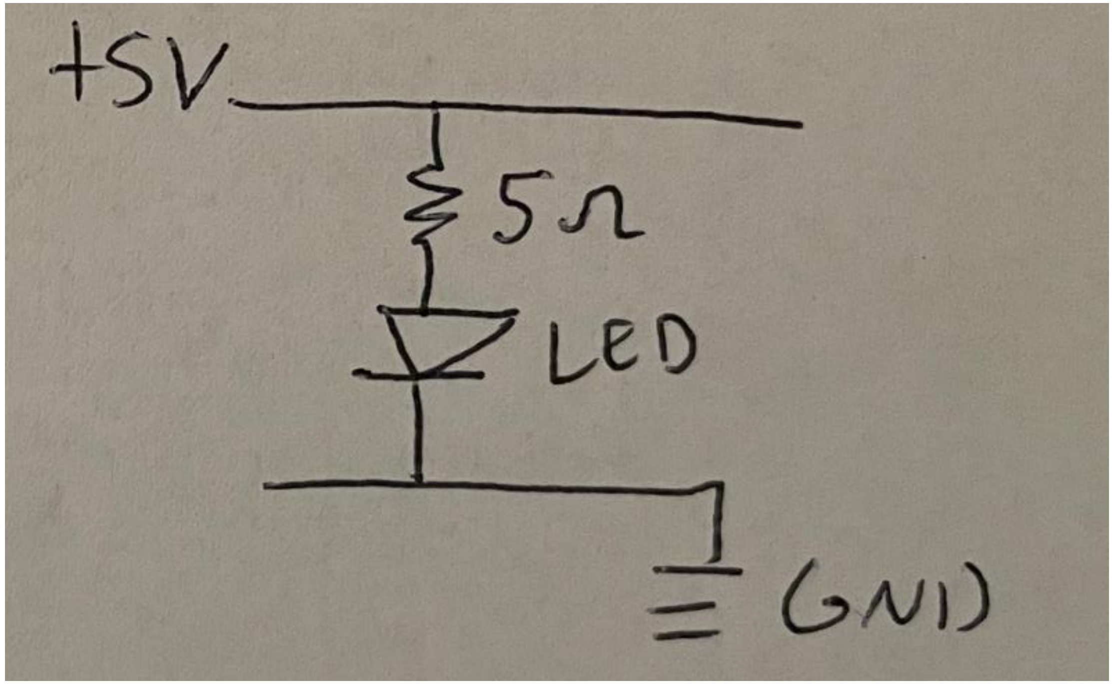
$$V = I * R$$

Voltage (measure in volts)

I: Current (measure in amps)

Resistance (measured in ohms)

Ok so what about this (correct direction) LED with a **current limiting resistor!**

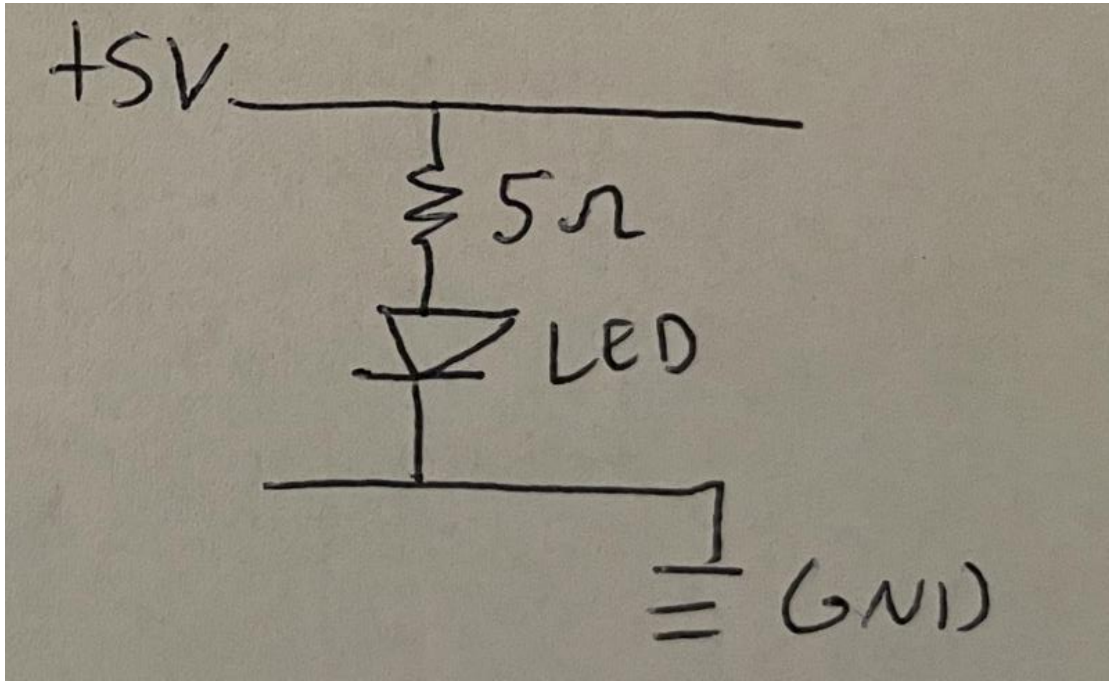


1A

Ohm's Law:

- 500 to 1K ohm resistors work well (for me)
- The order of the resistor and LED does NOT matter

Ok so what about this (correct direction) LED with a **current limiting resistor!**



1A

Our second and final equation - Capacitance

$$C = I * dv/dt$$

Capacitance (measured in farads)

I: Current (measure in amps)

dV/dt: Change in Voltage over time (measure in volts/second)

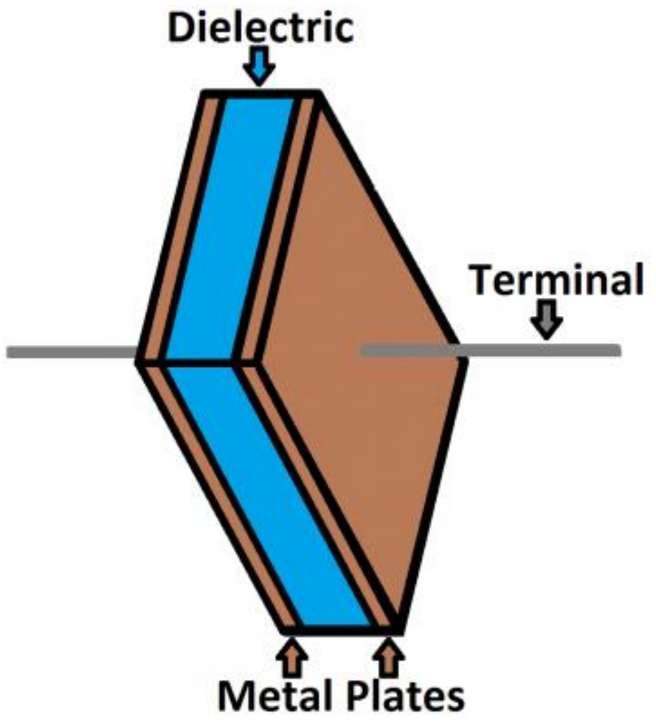
https://learn.sparkfun.com/tutorials/capacitors/

Capacitance

$$C = I * dv/dt$$

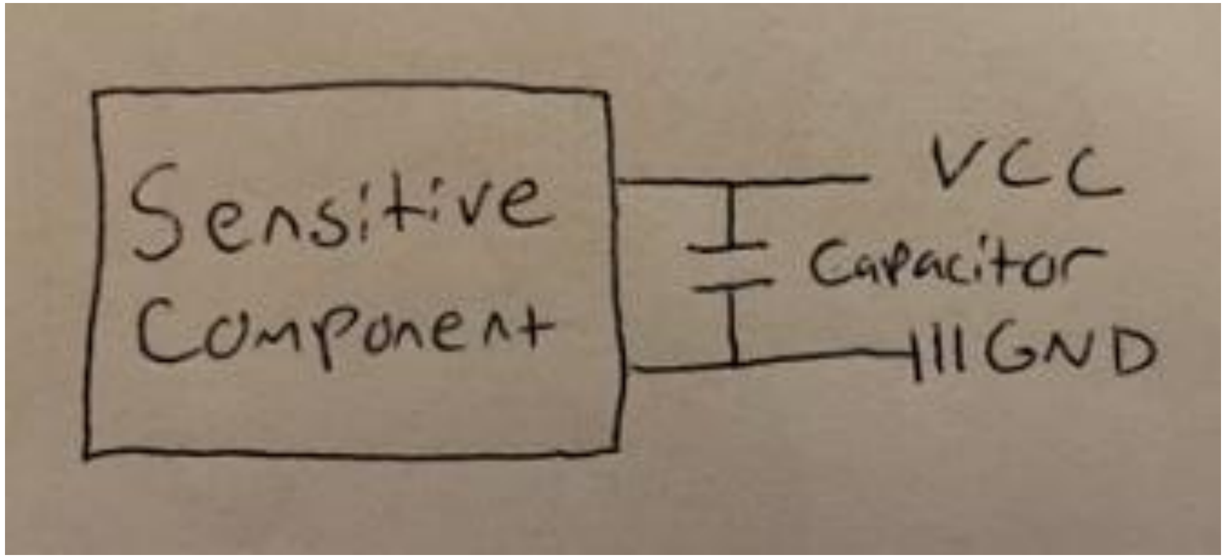
Capacitance (measured in farads)
I: Current (measure in amps)
dV/dt: Change in Voltage over time
(measure in volts/second)

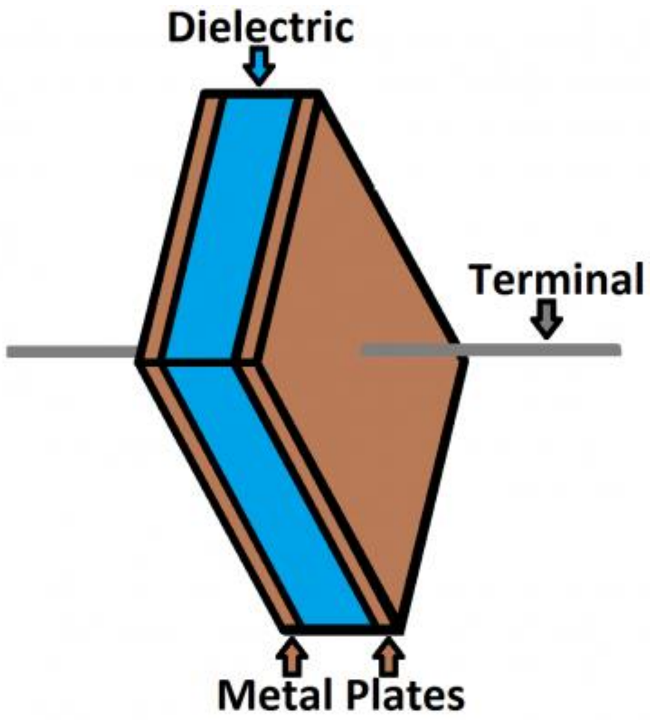
The science here can get a little complicated but/and I like to think of a capacitor as a **filter** for changes in voltage



Capacitance

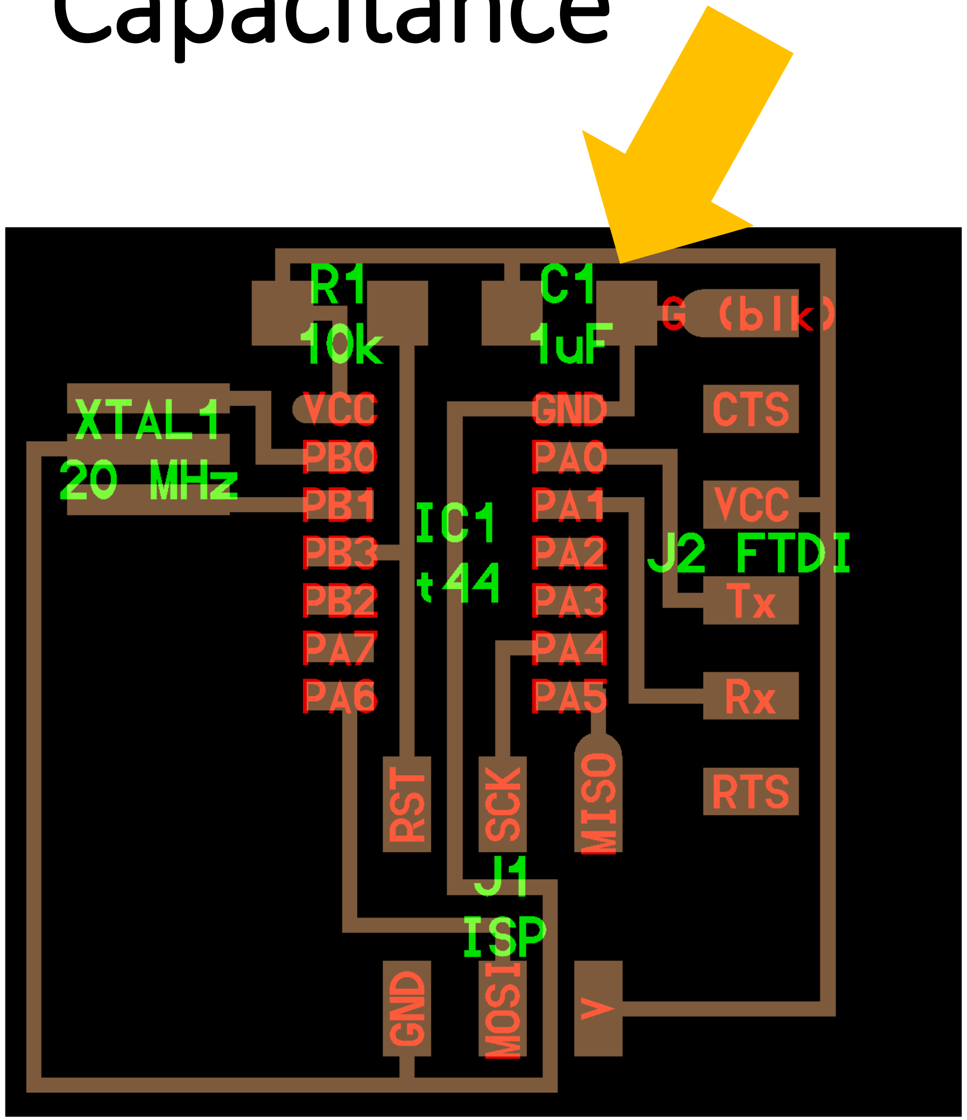
The science here can get a little complicated but/and I like to think of a capacitor as a **filter** for changes in voltage



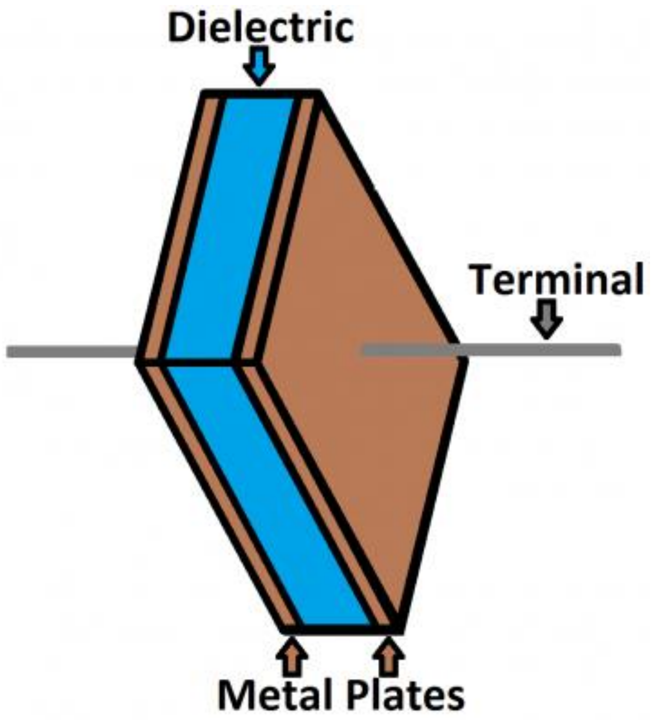


https://learn.sparkfun.com/tutorials/capacitors/application-examples

Capacitance

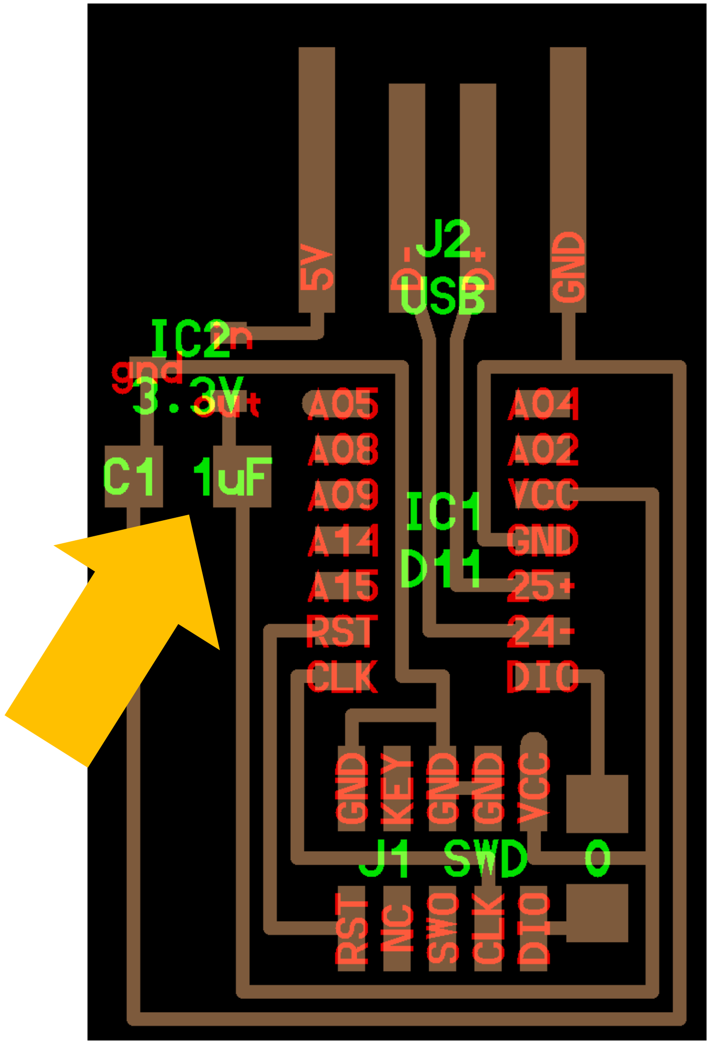


The science here can get a little complicated but/and I like to think of a capacitor as a **filter** for changes in voltage

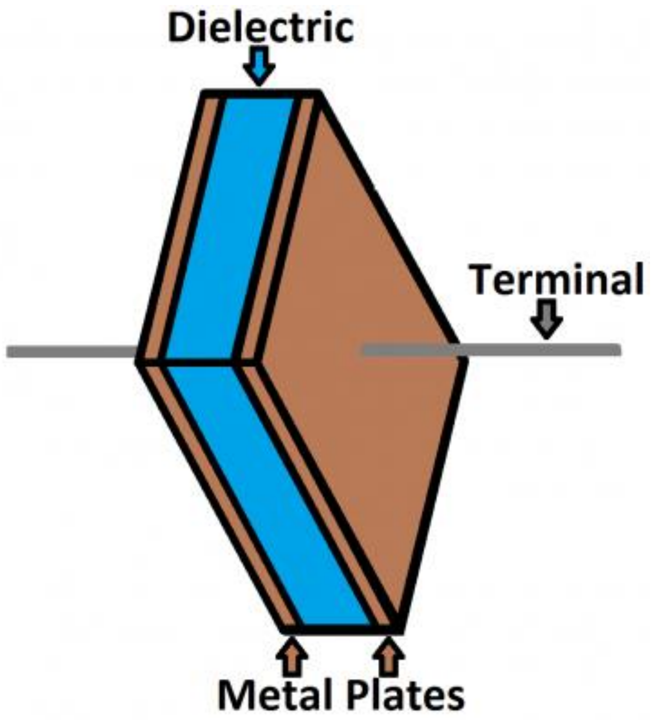


https://learn.sparkfun.com/tutorials/capacitors/application-examples

Capacitance

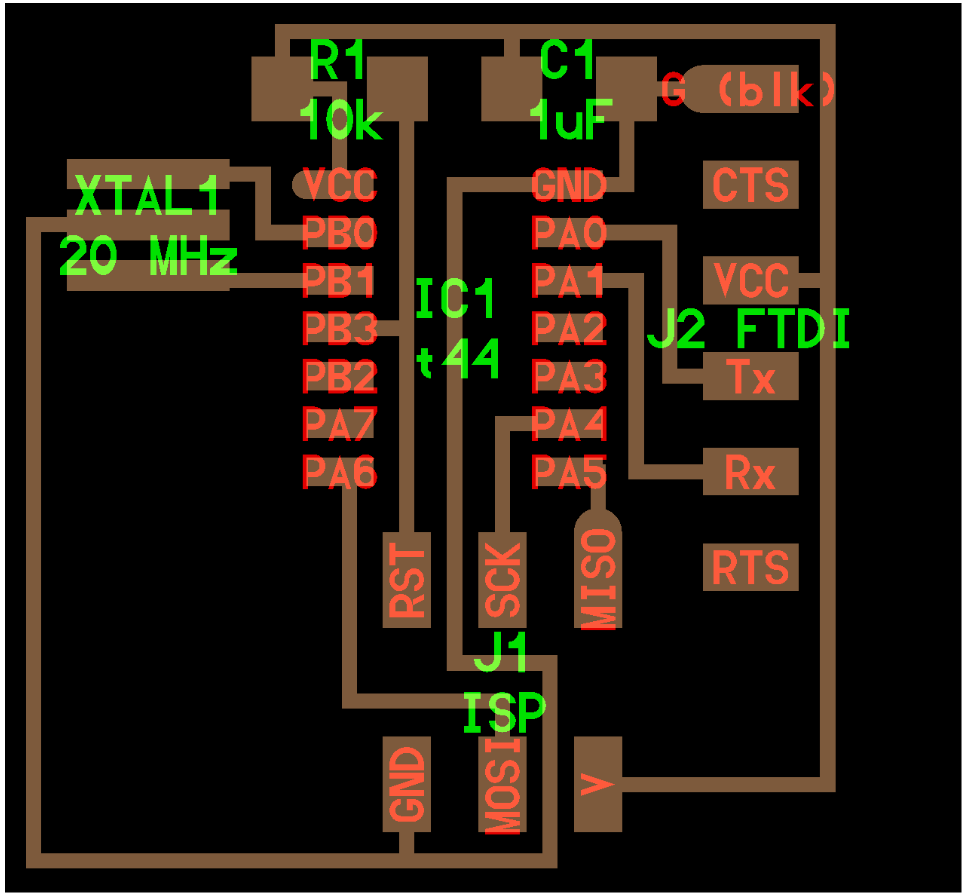


The science here can get a little complicated but/and I like to think of a capacitor as a **filter** for changes in voltage



https://learn.sparkfun.com/tutorials/capacitors/application-examples

But how will I know if my component needs a capacitor? And how big of a capacitor will I need?



But how will I know if my component needs a capacitor? And how big of a capacitor will I need? **(and what are all of those labels?)**

http://academy.cba.mit.edu/classes/embedded_programming/doc8183.pdf

Features

- High Performance, Low Power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
- High Endurance, Non-volatile Memory Segments
  - 2K/4K/8K Bytes of In-System, Self-programmable Flash Program Memory
    - Endurance: 10,000 Write/Erase Cycles
  - 128/256/512 Bytes of In-System Programmable EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 128/256/512 Bytes of Internal SRAM
  - Data Retention: 20 years at 85°C / 100 years at 25°C
  - Programming Lock for Self-programming Flash & EEPROM Data Security
- Peripheral Features
  - One 8-bit and One 16-bit Timer/Counter with Two PWM Channels, Each
  - 10-bit ADC
    - 8 Single-ended Channels
    - 12 Differential ADC Channel Pairs with Programmable Gain (1x / 20x)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Universal Serial Interface
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable via SPI Port
  - Internal and External Interrupt Sources
    - Pin Change Interrupt on 12 Pins
  - Low Power Idle, ADC Noise Reduction, Standby and Power-down Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit with Software Disable Function
  - Internal Calibrated Oscillator
  - On-chip Temperature Sensor
- I/O and Packages
  - Available in 20-pin QFN/MLF/VQFN, 14-pin SOIC, 14-pin PDIP and 15-ball UFBGA
  - Twelve Programmable I/O Lines
- Operating Voltage:
  - 1.8 – 5.5V
- Speed Grade:
  - 0 – 4 MHz @ 1.8 – 5.5V
  - 0 – 10 MHz @ 2.7 – 5.5V
  - 0 – 20 MHz @ 4.5 – 5.5V
- Industrial Temperature Range: -40°C to +85°C
- Low Power Consumption
  - Active Mode:
    - 210 µA at 1.8V and 1 MHz
  - Idle Mode:
    - 33 µA at 1.8V and 1 MHz
  - Power-down Mode:
    - 0.1 µA at 1.8V and 25°C

8-bit AVR® Microcontroller with 2K/4K/8K Bytes In-System Programmable Flash

ATtiny24A
ATtiny44A
ATtiny84A

Rev. 8183F–AVR–06/12

Time to read the data sheet!

http://academy.cba.mit.edu/classes/embedded_programming/doc8183.pdf

Features

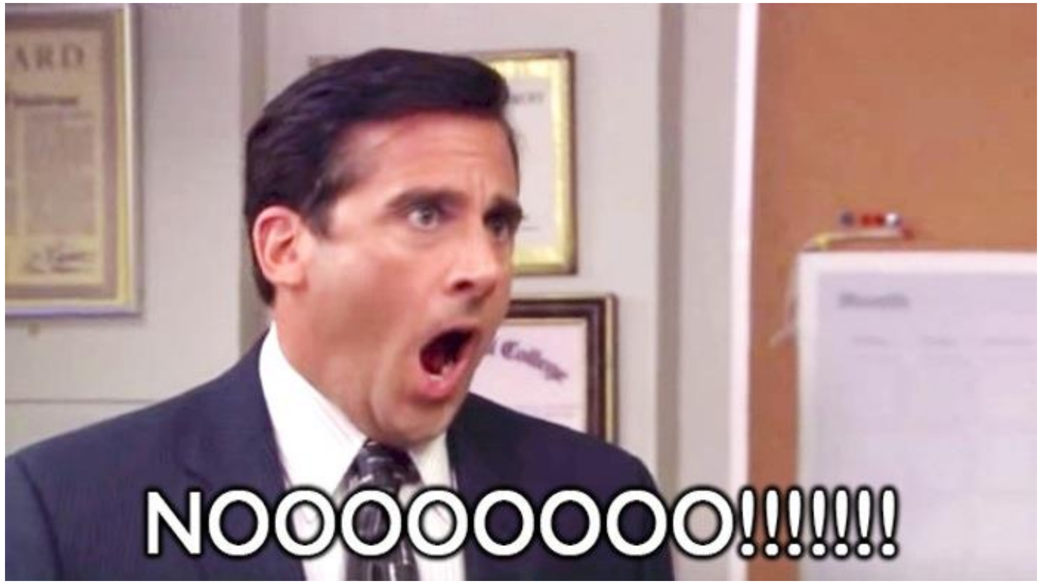
- High Performance, Low Power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
- High Endurance, Non-volatile Memory Segments
  - 2K/4K/8K Bytes of In-System, Self-programmable Flash Program Memory
    - Endurance: 10,000 Write/Erase Cycles
  - 128/256/512 Bytes of In-System Programmable EEPROM
    - Endurance: 100,000 Write/Erase Cycles
  - 128/256/512 Bytes of Internal SRAM
  - Data Retention: 20 years at 85°C / 100 years at 25°C
  - Programming Lock for Self-programming Flash & EEPROM Data Security
- Peripheral Features
  - One 8-bit and One 16-bit Timer/Counter with Two PWM Channels, Each
  - 10-bit ADC
    - 8 Single-ended Channels
    - 12 Differential ADC Channel Pairs with Programmable Gain (1x / 20x)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Universal Serial Interface
- Special Microcontroller Features
  - debugWIRE On-chip Debug System
  - In-System Programmable via SPI Port
  - Internal and External Interrupt Sources
    - Pin Change Interrupt on 12 Pins
  - Low Power Idle, ADC Noise Reduction, Standby and Power-down Modes
  - Enhanced Power-on Reset Circuit
  - Programmable Brown-out Detection Circuit with Software Disable Function
  - Internal Calibrated Oscillator
  - On-chip Temperature Sensor
- I/O and Packages
  - Available in 20-pin QFN/MLF/VQFN, 14-pin SOIC, 14-pin PDIP and 15-ball UFBGA
  - Twelve Programmable I/O Lines
- Operating Voltage:
  - 1.8 – 5.5V
- Speed Grade:
  - 0 – 4 MHz @ 1.8 – 5.5V
  - 0 – 10 MHz @ 2.7 – 5.5V
  - 0 – 20 MHz @ 4.5 – 5.5V
- Industrial Temperature Range: -40°C to +85°C
- Low Power Consumption
  - Active Mode:
    - 210 µA at 1.8V and 1 MHz
  - Idle Mode:
    - 33 µA at 1.8V and 1 MHz
  - Power-down Mode:
    - 0.1 µA at 1.8V and 25°C

8-bit AVR® Microcontroller with 2K/4K/8K Bytes In-System Programmable Flash

ATtiny24A
ATtiny44A
ATtiny84A

Rev. 8183F–AVR–06/12

286 PAGES !!!!!!!



22. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	Page 14
0x3E (0x5E)	SPH	–	–	–	–	–	–	SP9	SP8	Page 13
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	Page 13
0x3C (0x5C)	OCR0B	Timer/Counter0 – Output Compare Register B								Page 83
0x3B (0x5B)	GIMSK	–	INT0	PCIE1	PCIE0	–	–	–	–	Page 50
0x3A (0x5A)	GIFR	–	INTF0	PCIF1	PCIF0	–	–	–	–	Page 51
0x39 (0x59)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	Page 83
0x38 (0x58)	TIFR0	–	–	–	–	–	OCF0B	OCF0A	TOV0	Page 84
0x37 (0x57)	SPMCSR	–	–	RSIG	CTPB	RFLB	PGWRT	PGERS	SPMEN	Page 156
0x36 (0x56)	OCR0A	Timer/Counter0 – Output Compare Register A								Page 83
0x35 (0x55)	MCUCR	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	Pages 36, 50, 66
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	Page 44
0x33 (0x53)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	Page 82
0x32 (0x52)	TCNT0	Timer/Counter0								Page 83
0x31 (0x51)	OSCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	Page 31
0x30 (0x50)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	Page 79
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	Page 106
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	Page 108
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								Page 110
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								Page 110
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Compare Register A High Byte								Page 110
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Compare Register A Low Byte								Page 110
0x29 (0x49)	OCR1BH	Timer/Counter1 – Compare Register B High Byte								Page 110
0x28 (0x48)	OCR1BL	Timer/Counter1 – Compare Register B Low Byte								Page 110
0x27 (0x47)	DWDR	DWDR[7:0]								Page 151
0x26 (0x46)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	Page 31
0x25 (0x45)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								Page 111
0x24 (0x44)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								Page 111
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	–	PSR10	Page 114
0x22 (0x42)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	Page 109
0x21 (0x41)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	Page 44
0x20 (0x40)	PCMSK1	–	–	–	–	PCINT11	PCINT10	PCINT9	PCINT8	Page 51
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	EEAR8	Page 20
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	Page 21
0x1D (0x3D)	EEDR	EEPROM Data Register								Page 21
0x1C (0x3C)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	Page 23
0x1B (0x3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	Page 66
0x1A (0x3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	Page 66
0x19 (0x39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	Page 67
0x18 (0x38)	PORTB	–	–	–	–	PORTB3	PORTB2	PORTB1	PORTB0	Page 67
0x17 (0x37)	DDRB	–	–	–	–	DDB3	DDB2	DDB1	DDB0	Page 67
0x16 (0x36)	PINB	–	–	–	–	PINB3	PINB2	PINB1	PINB0	Page 67
0x15 (0x35)	GPIOR2	General Purpose I/O Register 2								Page 22
0x14 (0x34)	GPIOR1	General Purpose I/O Register 1								Page 23
0x13 (0x33)	GPIOR0	General Purpose I/O Register 0								Page 23
0x12 (0x32)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	Page 52
0x11 (0x31))	Reserved	–								
0x10 (0x30)	USIBR	USI Buffer Register								Page 127
0x0F (0x2F)	USIDR	USI Data Register								Page 126
0x0E (0x2E)	USISR	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	Page 125
0x0D (0x2D)	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	Page 123
0x0C (0x2C)	TIMSK1	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	Page 111
0x0B (0x2B)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	Page 112
0x0A (0x2A)	Reserved	–								
0x09 (0x29)	Reserved	–								
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	Page 129
0x07 (0x27)	ADMUX	REFS1	REFS0	MUX5	MUX4	MUX3	MUX2	MUX1	MUX0	Page 144
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	Page 146
0x05 (0x25)	ADCH	ADC Data Register High Byte								Page 148
0x04 (0x24)	ADCL	ADC Data Register Low Byte								Page 148
0x03 (0x23)	ADCSRB	BIN	ACME	–	ADLAR	–	ADTS2	ADTS1	ADTS0	Pages 130, 148
0x02 (0x22)	Reserved	–								
0x01 (0x21)	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	Pages 131, 149
0x00 (0x20)	PRR	–	–	–	–	PRTIM1	PRTIM0	PRUSI	PRADC	Page 37

Figure 2-1. Block Diagram

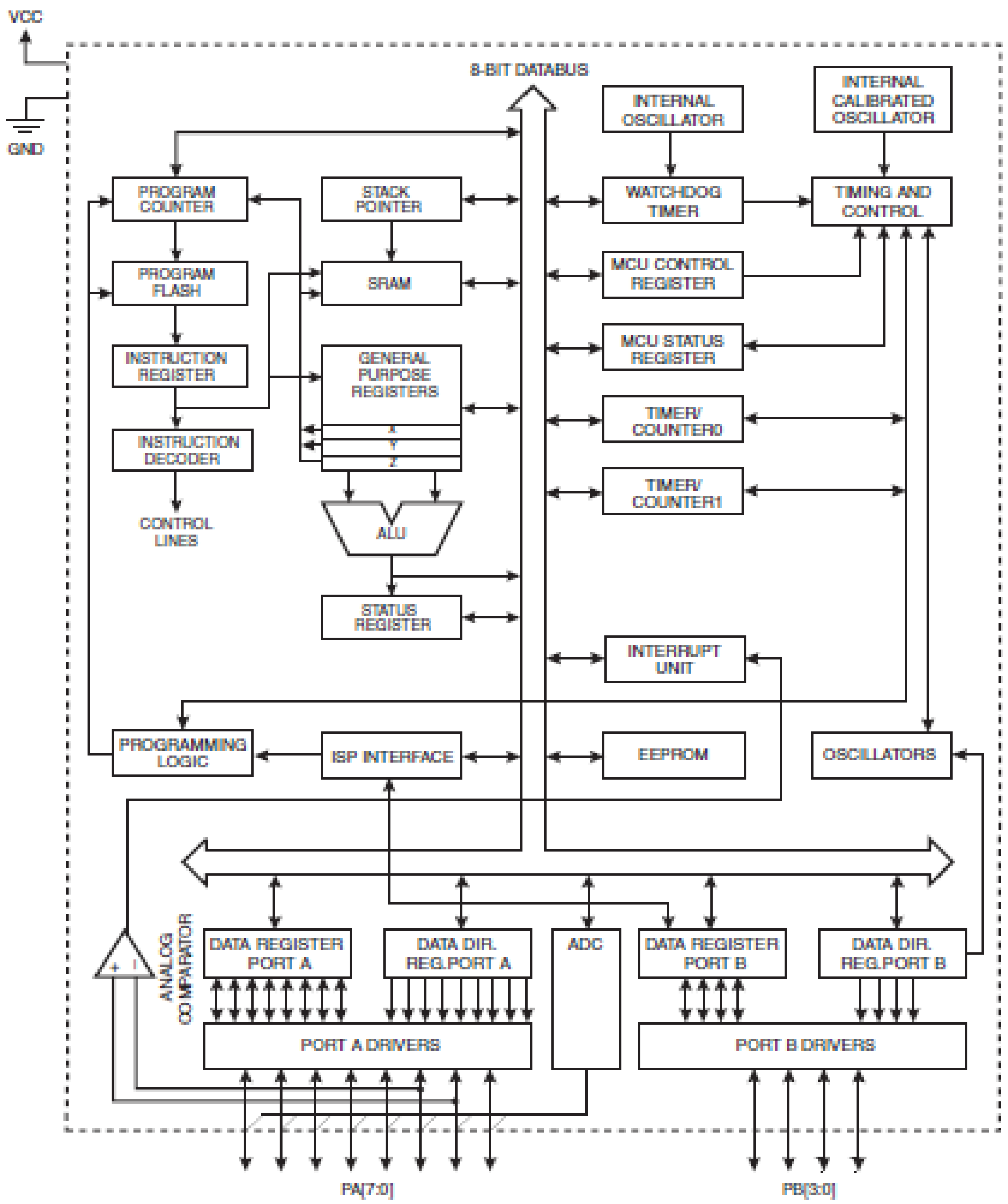
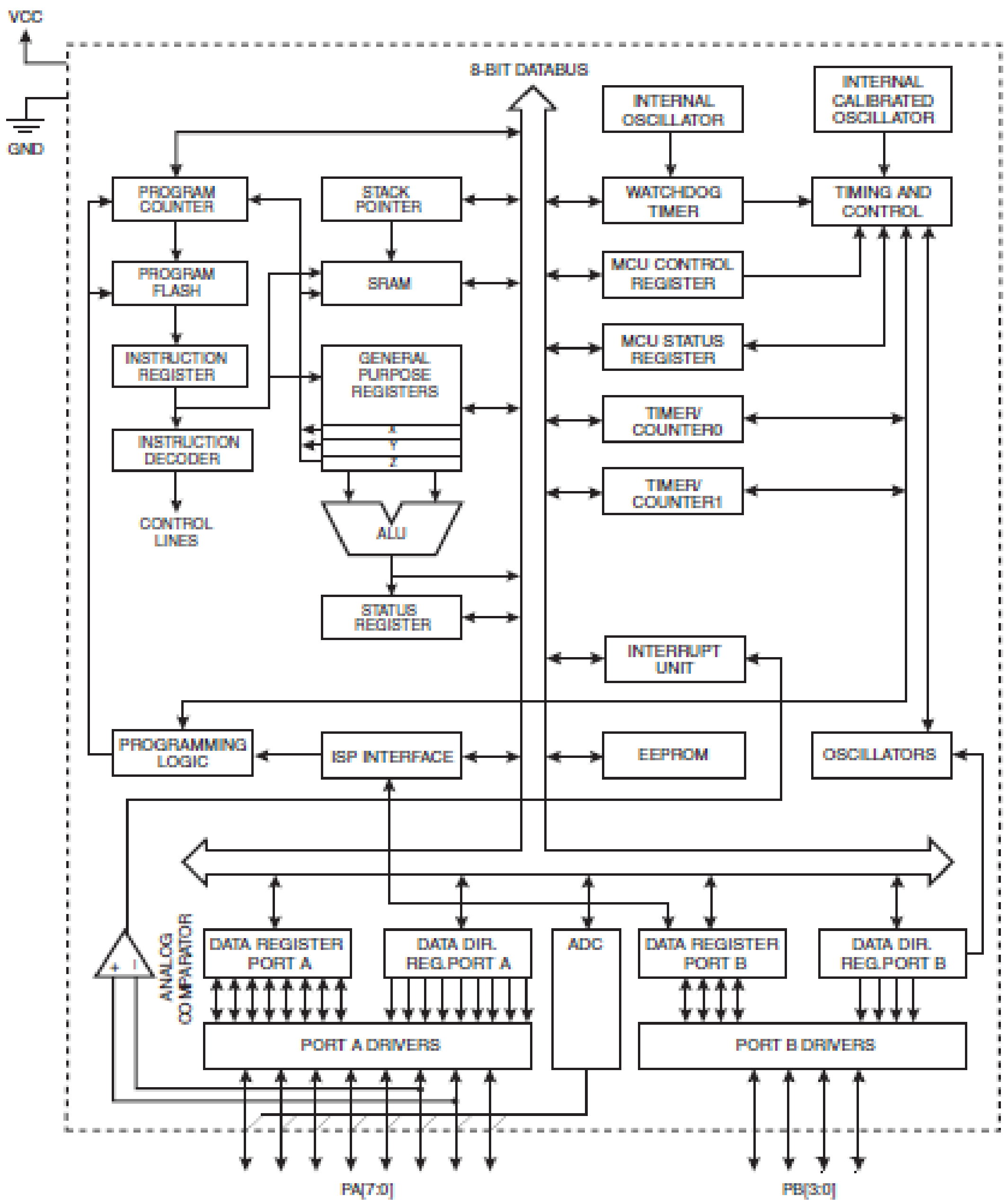
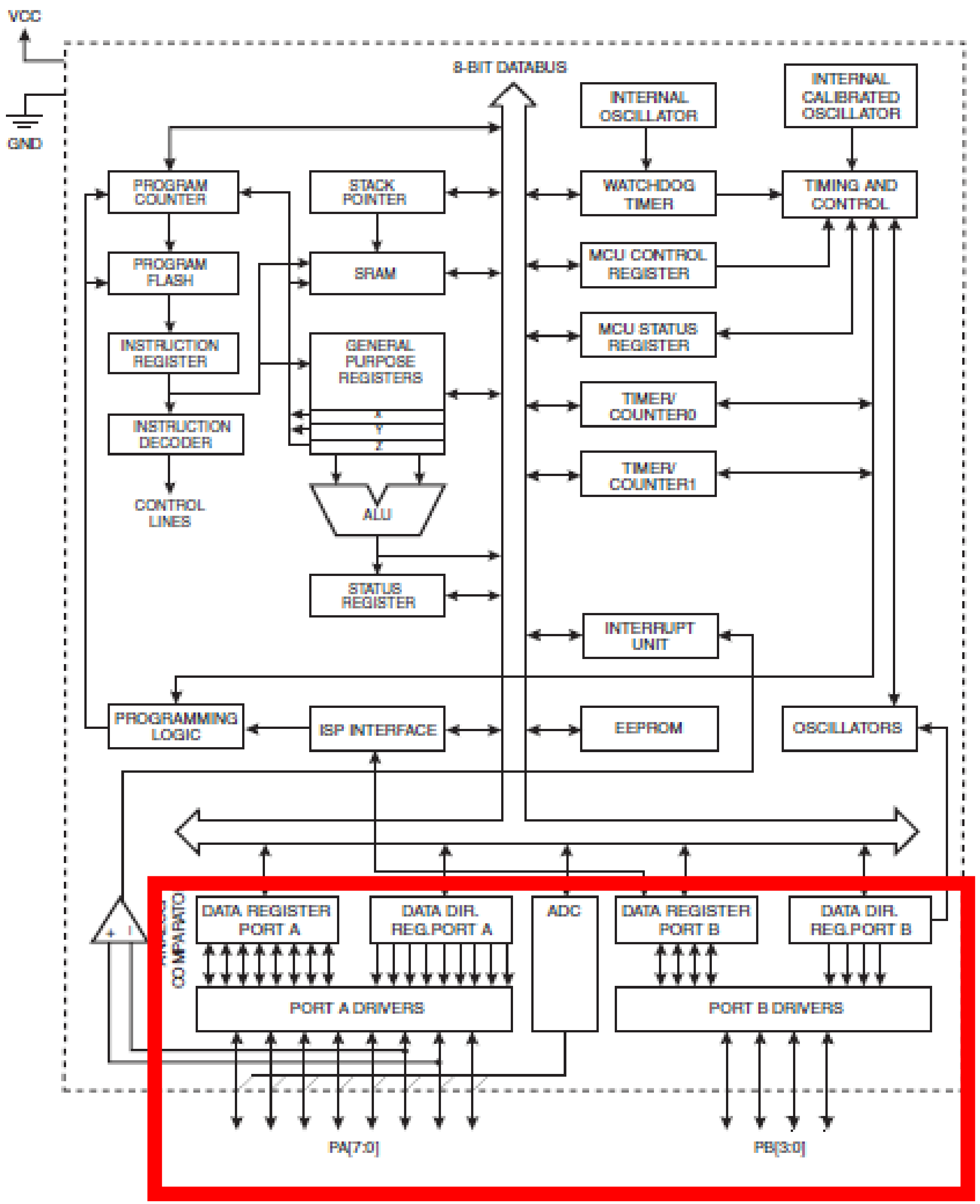


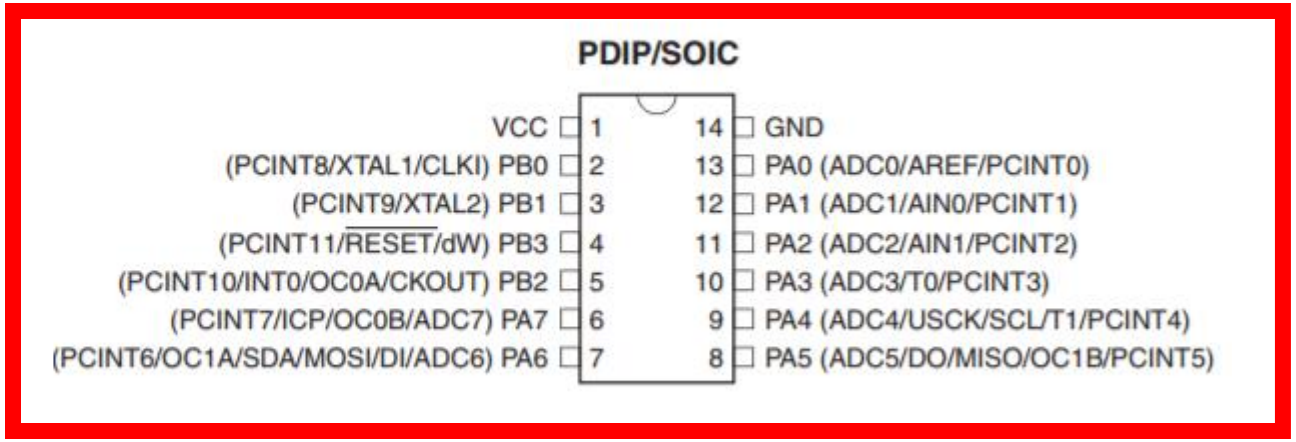
Figure 2-1. Block Diagram



Its not actually that scary I promise --- also we don't need to memorize all of it! In fact most of the TAs don't know all of it!

Figure 2-1. Block Diagram

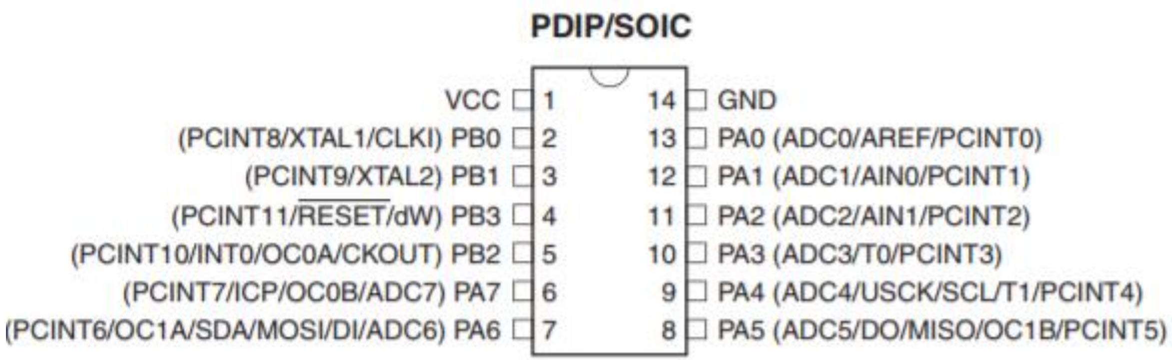




Hey look here's some port stuff seems like it has something to do with the inputs!

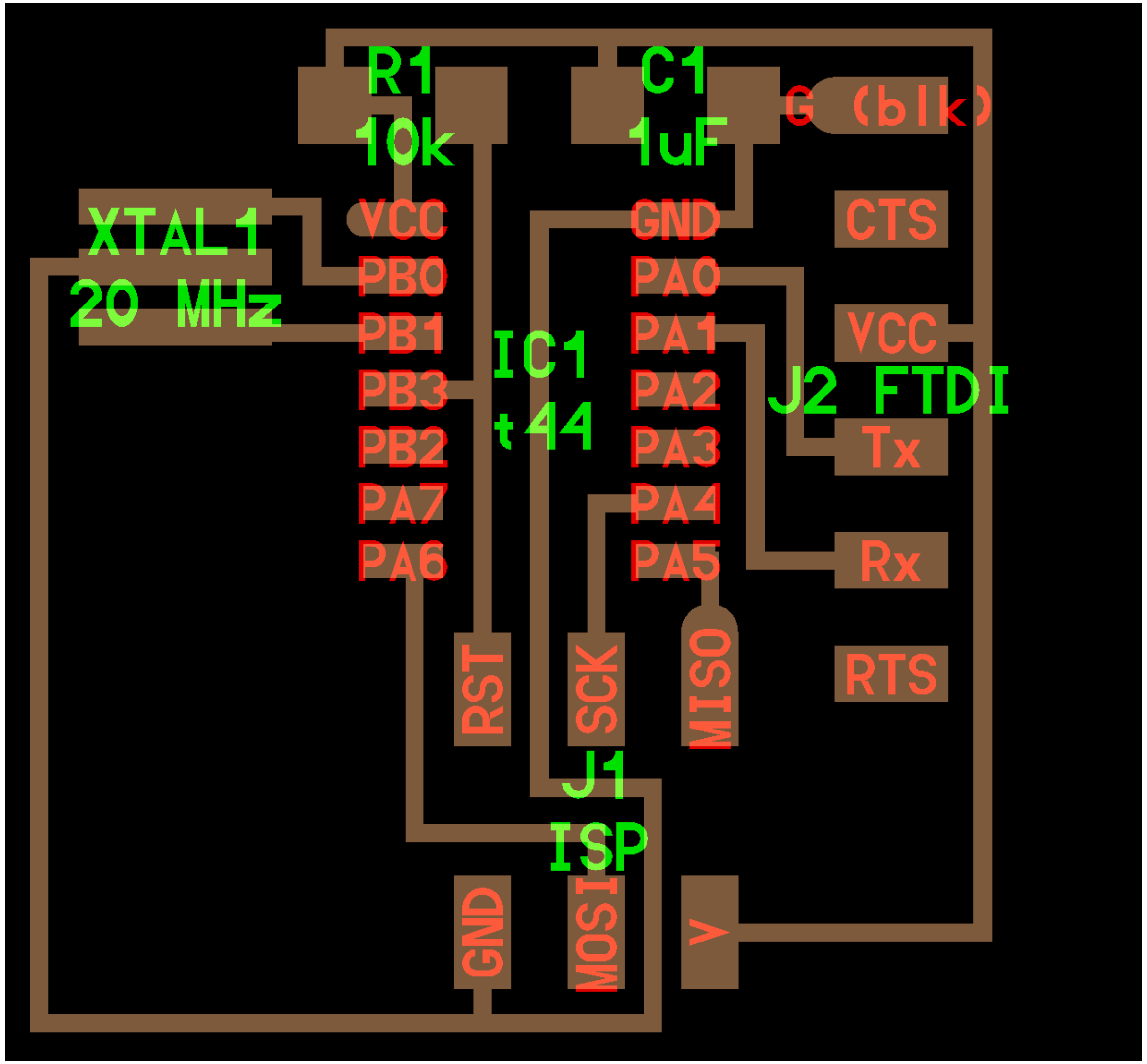
Table 10-3. Port A Pins Alternate Functions

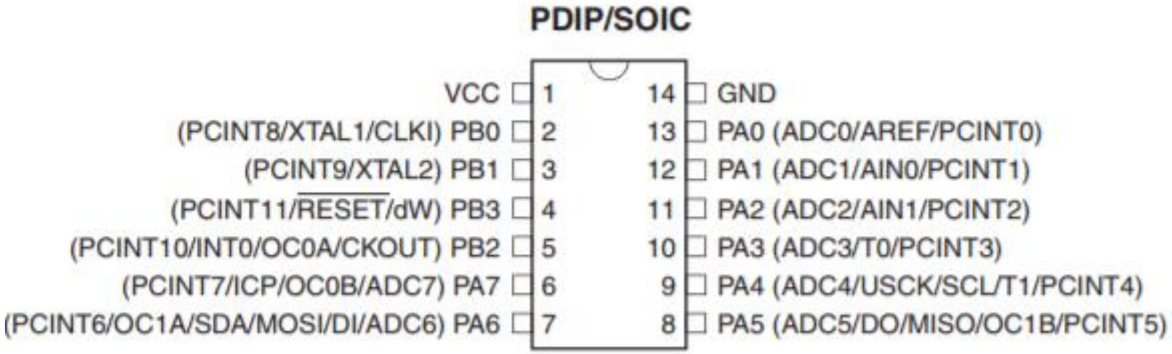
Port Pin	Alternate Function
PA0	ADC0: ADC Input Channel 0 AREF: External Analog Reference PCINT0: Pin Change Interrupt 0, Source 0
PA1	ADC1: ADC Input Channel 1 AIN0: Analog Comparator, Positive Input PCINT1:Pin Change Interrupt 0, Source 1
PA2	ADC2: ADC Input Channel 2 AIN1: Analog Comparator, Negative Input PCINT2: Pin Change Interrupt 0, Source 2
PA3	ADC3: ADC Input Channel 3 T0: Timer/Counter0 Clock Source. PCINT3: Pin Change Interrupt 0, Source 3
PA4	ADC4: ADC Input Channel 4 USCK: USI Clock (Three Wire Mode) SCL : USI Clock (Two Wire Mode) T1: Timer/Counter1 Clock Source PCINT4: Pin Change Interrupt 0, Source 4
PA5	ADC5: ADC Input Channel 5 DO: USI Data Output (Three Wire Mode) MISO: SPI Master Data Input / Slave Data Output OC1B: Timer/Counter1 Compare Match B Output PCINT5: Pin Change Interrupt 0, Source 5
PA6	ADC6: ADC Input Channel 6 DI: USI Data Input (Three Wire Mode) SDA: USI Data Input (Two Wire Mode) MOSI: SPI Master Data Output / Slave Data Input OC1A: Timer/Counter1 Compare Match A Output PCINT6: Pin Change Interrupt 0, Source 6
PA7	ADC7: ADC Input Channel 7 OC0B:: Timer/Counter0 Compare Match B Output ICP1: Timer/Counter1 Input Capture Pin PCINT7: Pin Change Interrupt 0, Source 7



Ok so on the Attiny44 we have two ports one with 8 pins and one with 4 pins that logically are connected to different internal things so they can have different roles.

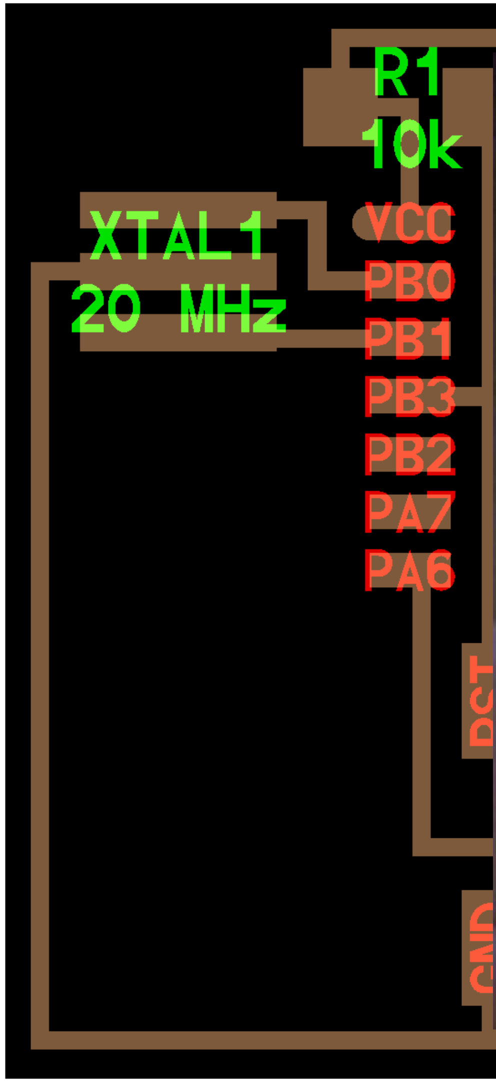
That wasn't so scary!

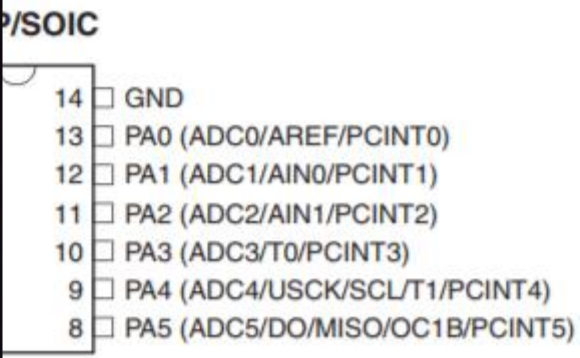




Oh hey look at Neil's hello world board – it looks like the programming 6 pin header has all of it's named things connected to the ports on the Attiny with those names!

Oh and the clock too (XTAL)!





Neil's hello world
ke the programming
s all of it's named
to the ports on the
those names!

ock too (XTAL)!

We got smarter! We know the basics now!

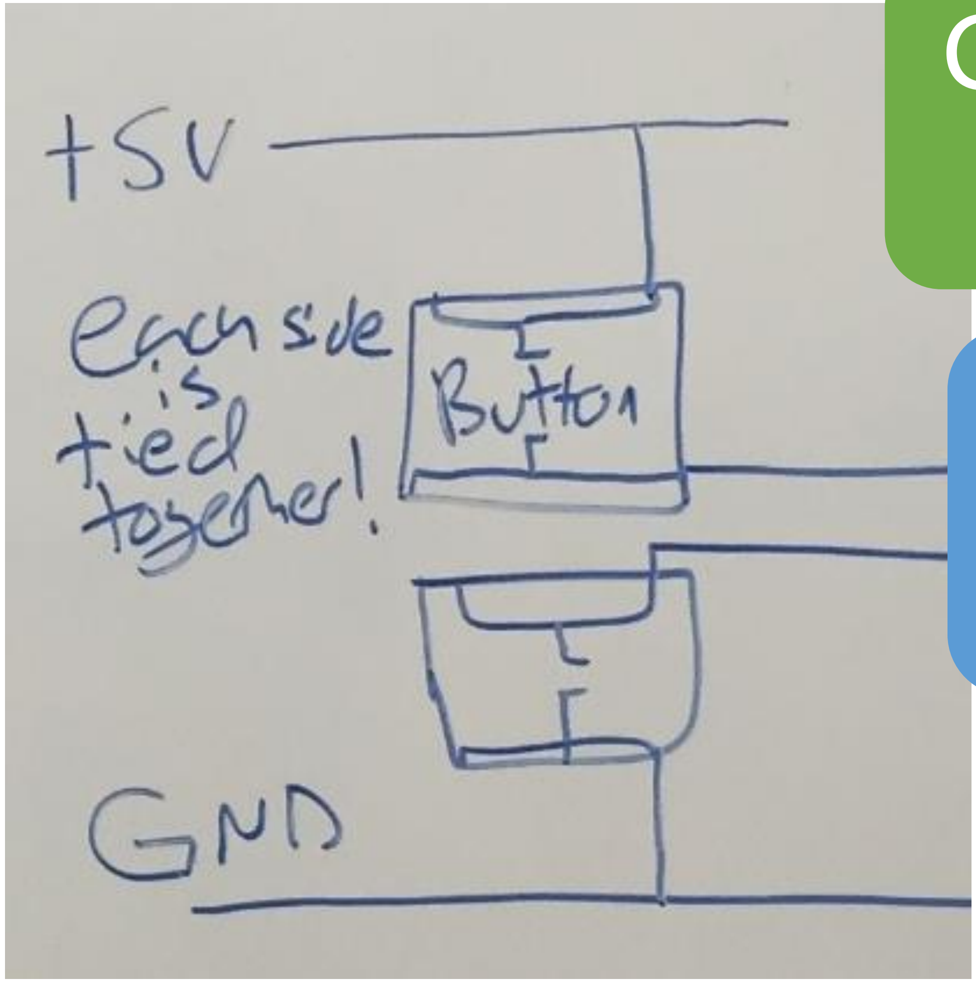
A short outline for today

1. Almost all you need to know about Electrical Engineering
2. **Almost all the tips you need to design custom boards**
3. Almost all the steps it will take to produce a custom board

Board design quick tips and tricks

1. How to wire up a button (and other inputs)

How do I wire up a button? What do you think?

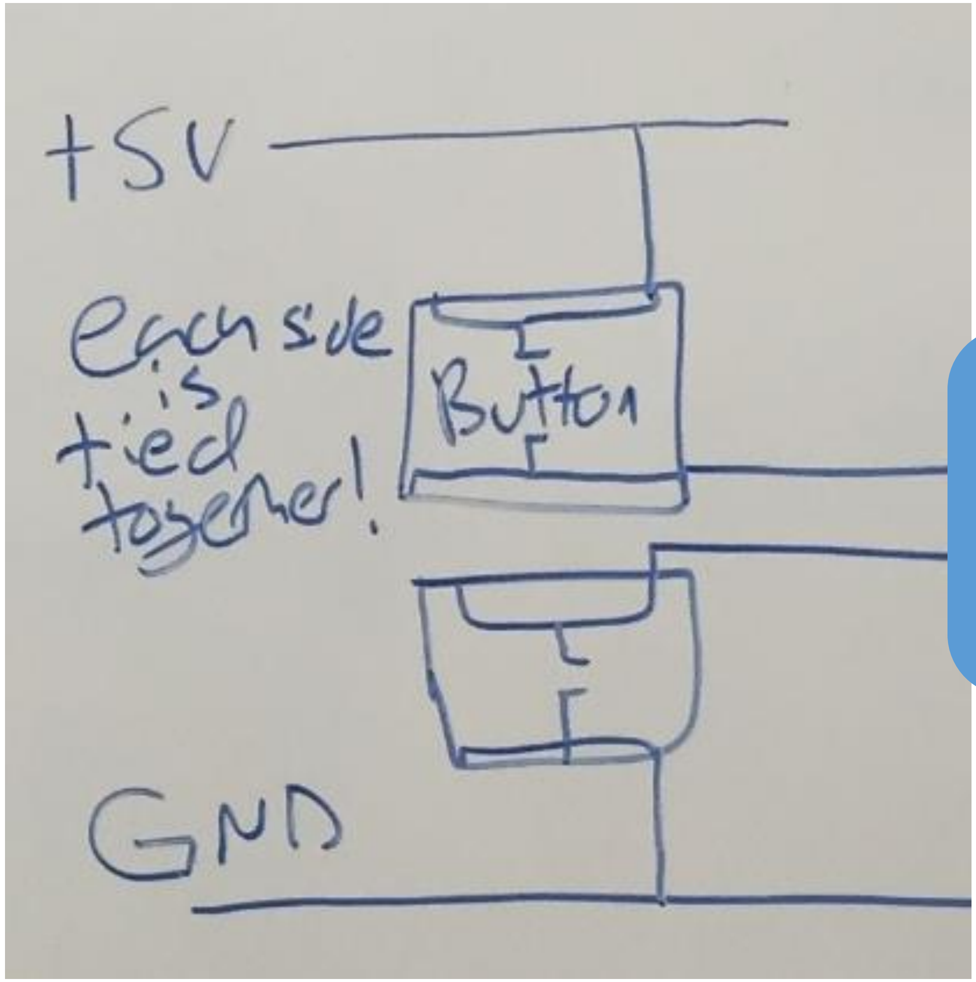


Green Check for Pin 1

Red X for Pin 2

Pin 1
Pin 2
Microcontroller

How do I wire up a button?

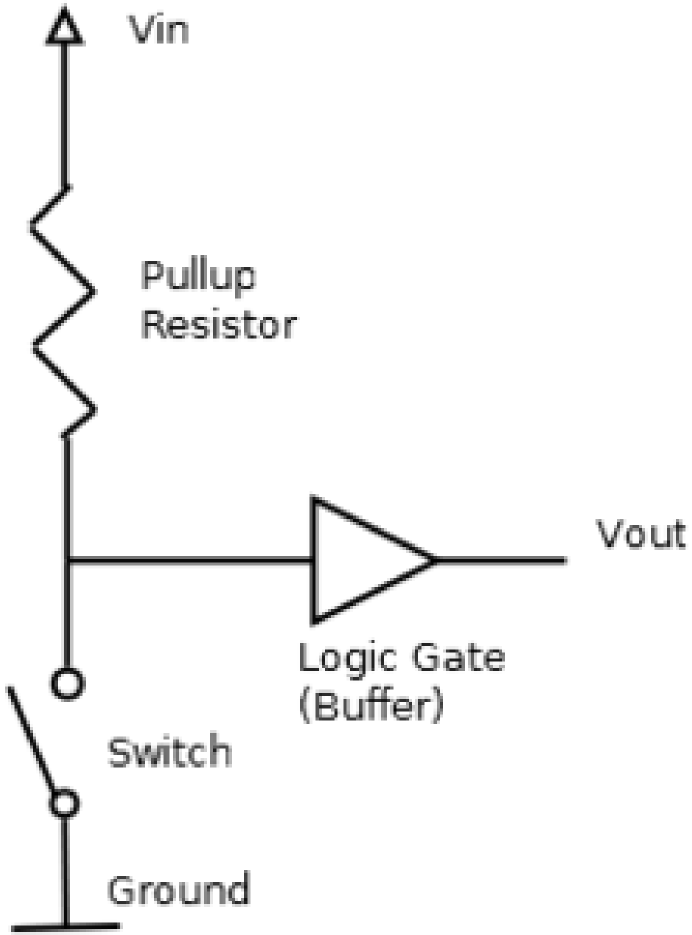


Red X for Pin 2

Pin 1
Pin 2
Microcontroller

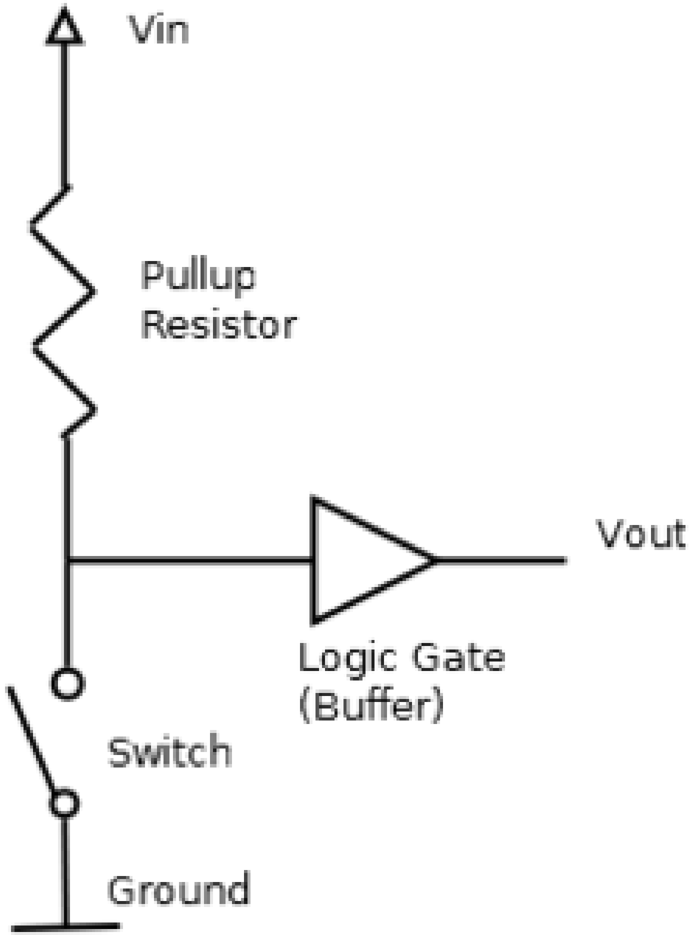
Connecting to GND is more power efficient

How do I wire up a button?



You need a **pullup resistor** – but this is so common there are **built-in pullups** in most microcontrollers you can **turn on in software**!

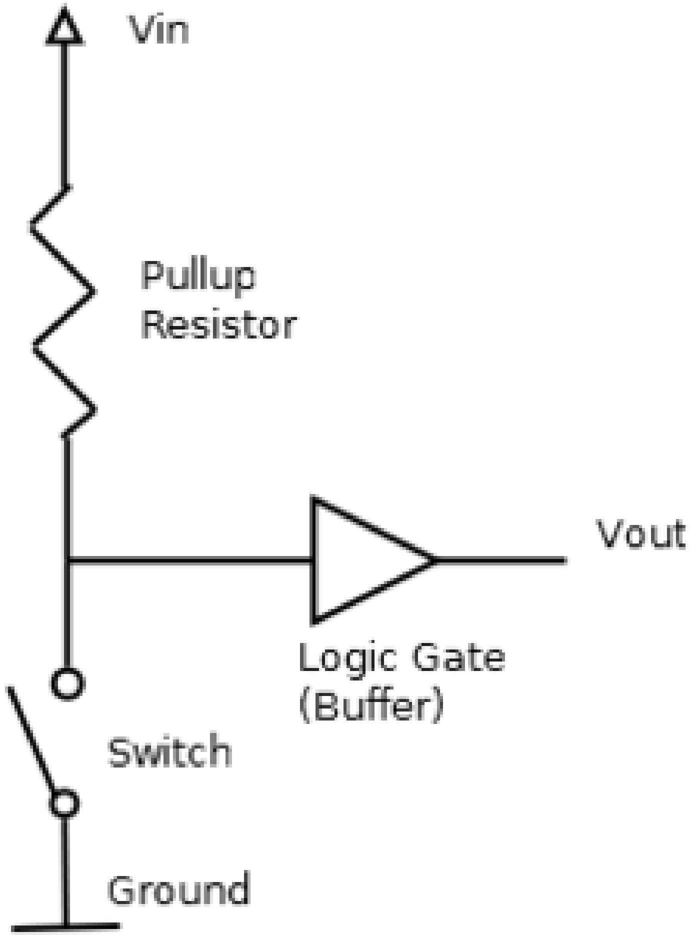
A pull what?



Long story short here is that about 0 current goes through a gate (transistor) so we need the resistor to "force" the value to 5v or 0v

https://www.electronics-tutorials.ws/logic/pull-up-resistor.html

How do I wire up a button?



Be careful though if you are connecting to a device that gives a HIGH (+5v) signal you will want the pullup turned off!

Board design quick tips and tricks

1. Connect buttons to ground (and turn on the pullup but no pullup for many other inputs)
2. Always place a filter capacitor AS CLOSE AS POSSIBLE to the chip it is protecting

Board design quick tips and tricks

1. Connect buttons to ground (and turn on the pullup but no pullup for many other inputs)
2. Always place a filter capacitor AS CLOSE AS POSSIBLE to the chip it is protecting
3. Place an LED between power and ground near each microcontroller for testing

Board design quick tips and tricks

1. Connect buttons to ground (and turn on the pullup but no pullup for many other inputs)
2. Always place a filter capacitor AS CLOSE AS POSSIBLE to the chip it is protecting
3. Place an LED between power and ground near each microcontroller for testing
4. **READ THE DATA SHEETS FOR EVERYTHING YOU USE!**

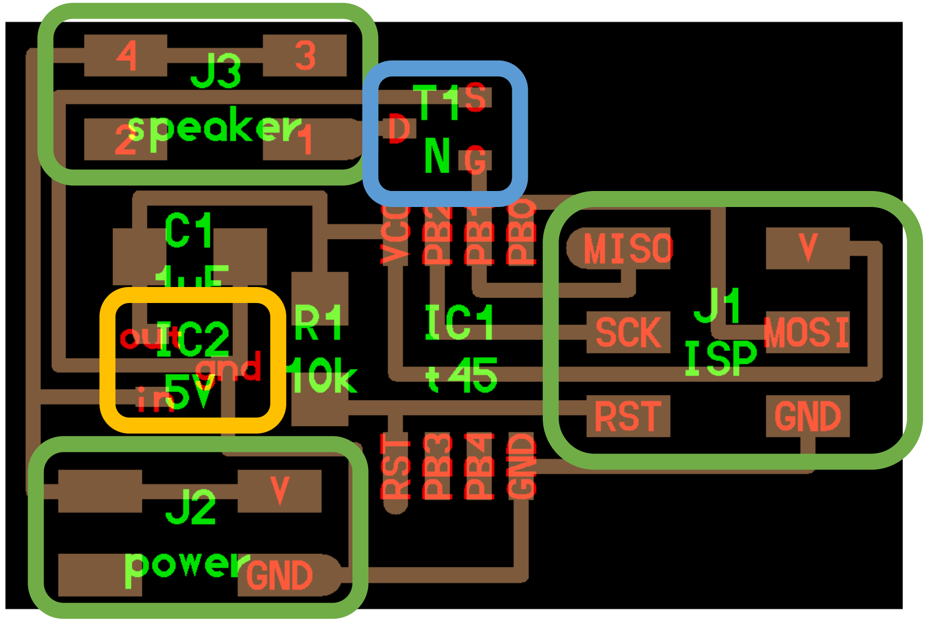
Quick aside: other common components

MOSFET = code controlled switch
(useful for output)

Voltage Regulator = allows for diff-
erent voltages

Header = make it easy to attach
other stuff

Neil's Speaker Hello World



OK FINE BUT HOW DO WE ACTUALLY MAKE A BOARD?!?!?

A short outline for today

1. Almost all you need to know about Electrical Engineering
2. Almost all the tips you need to design custom boards
3. **Almost all the steps it will take to produce a custom board**

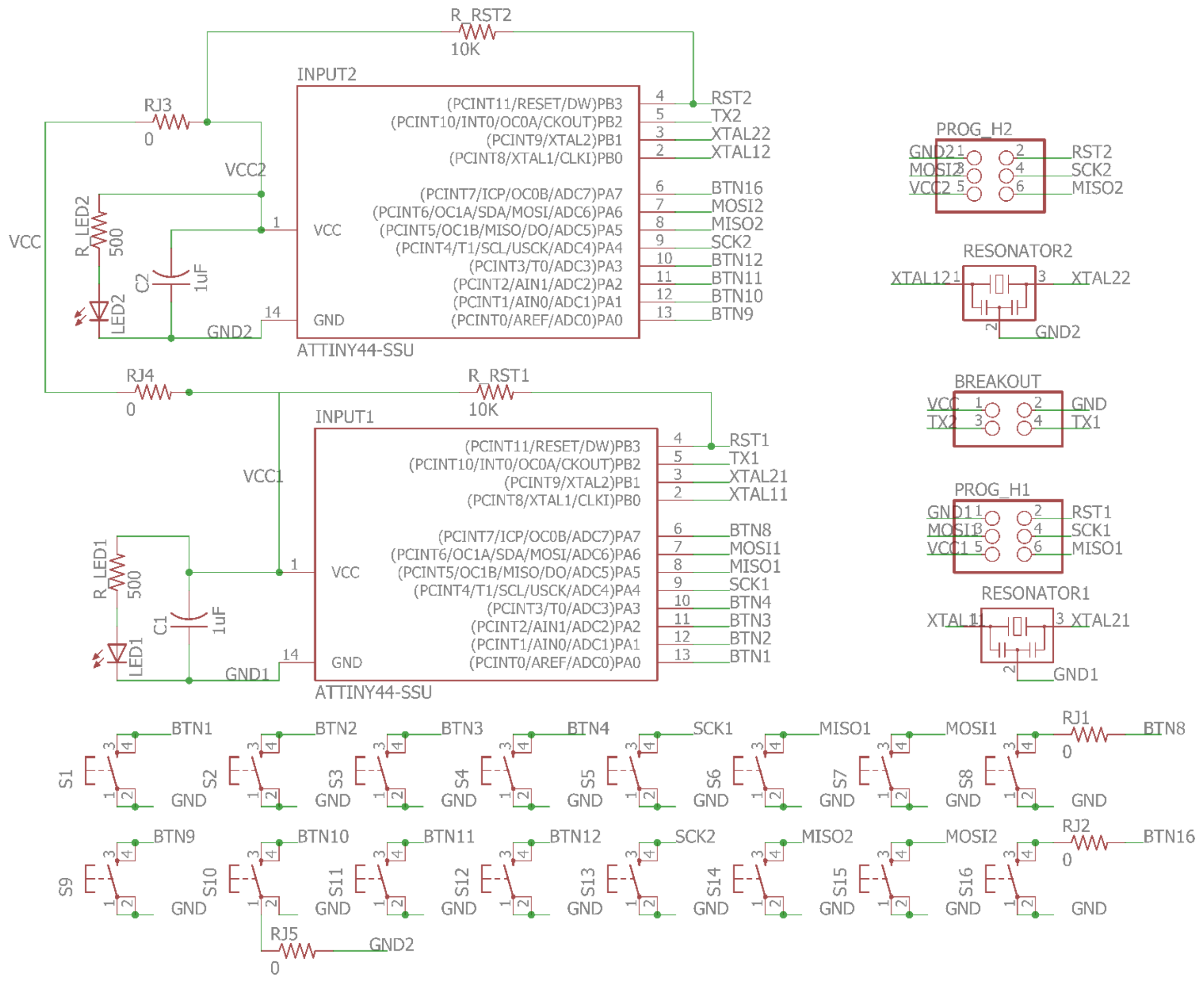

Eagle to the rescue

Sorry wrong rescue eagle… but also I’ve heard good things about KiCad

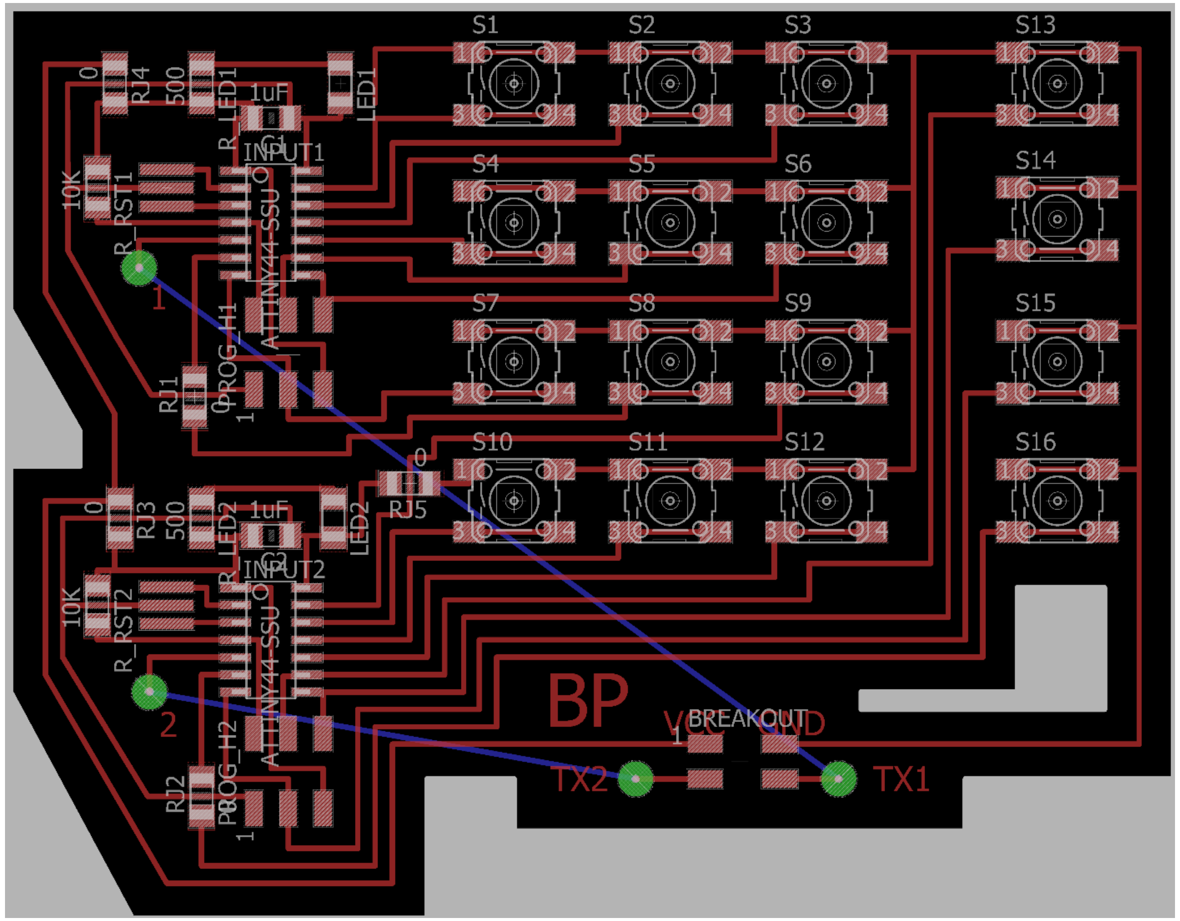
Sorry wrong rescue eagle… but also I’ve heard good things about KiCad

Details about the software will be given in a later recitation but I figure I can give you my “quick tips and tricks”

Schematic

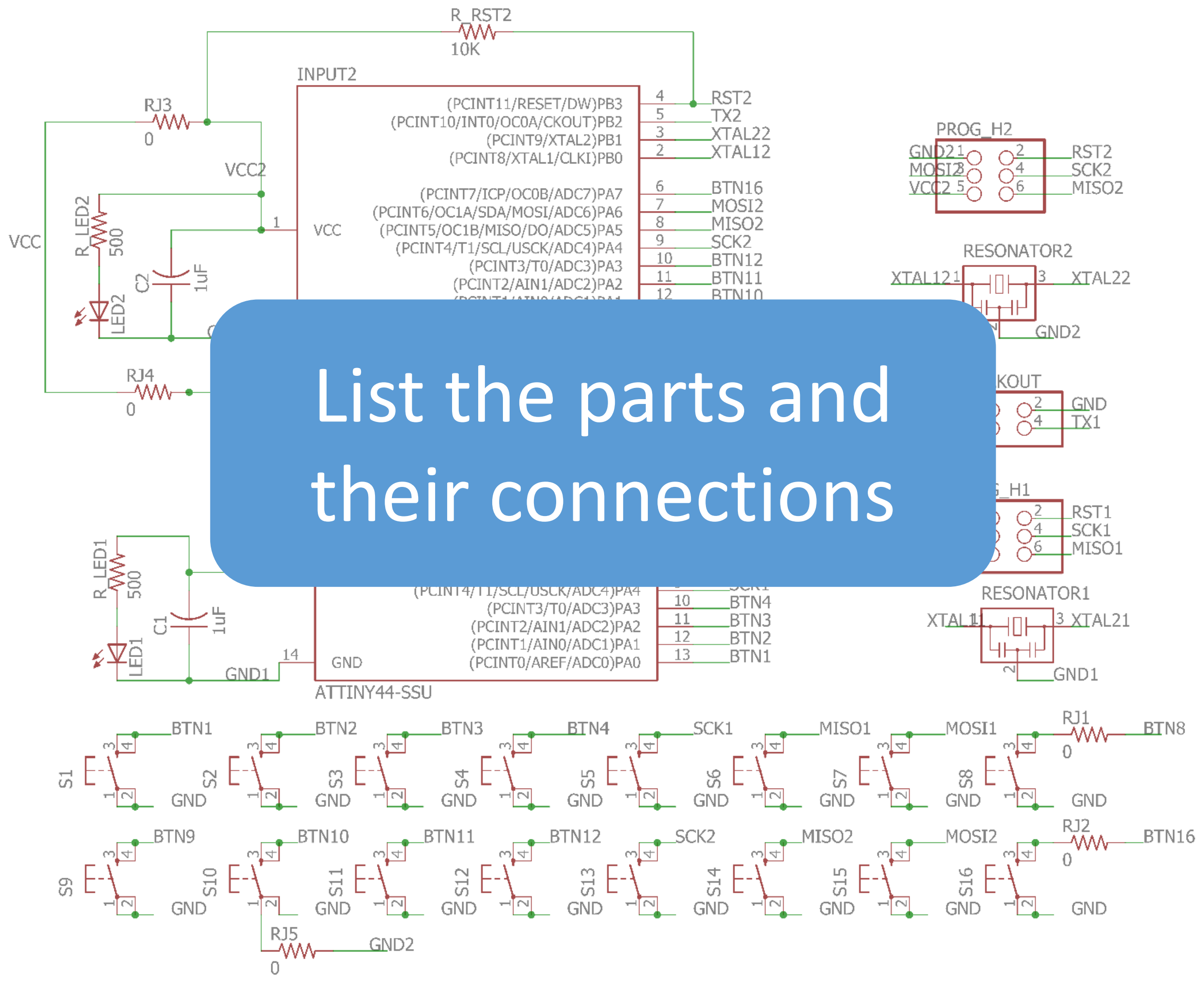


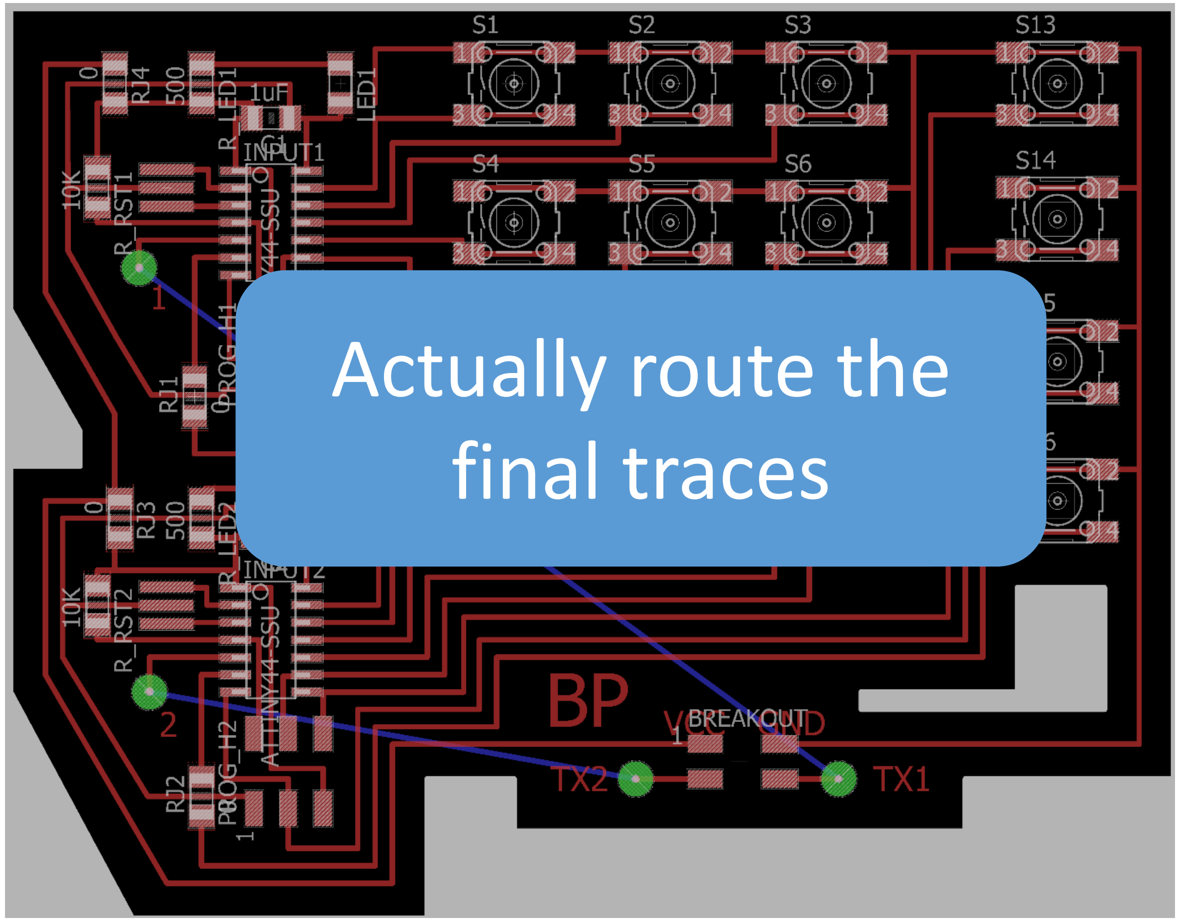
Board File



Schematic

Board File





Tips for board schematics and routing:

1. **Do the schematic first** (and finish it before moving on to routing)

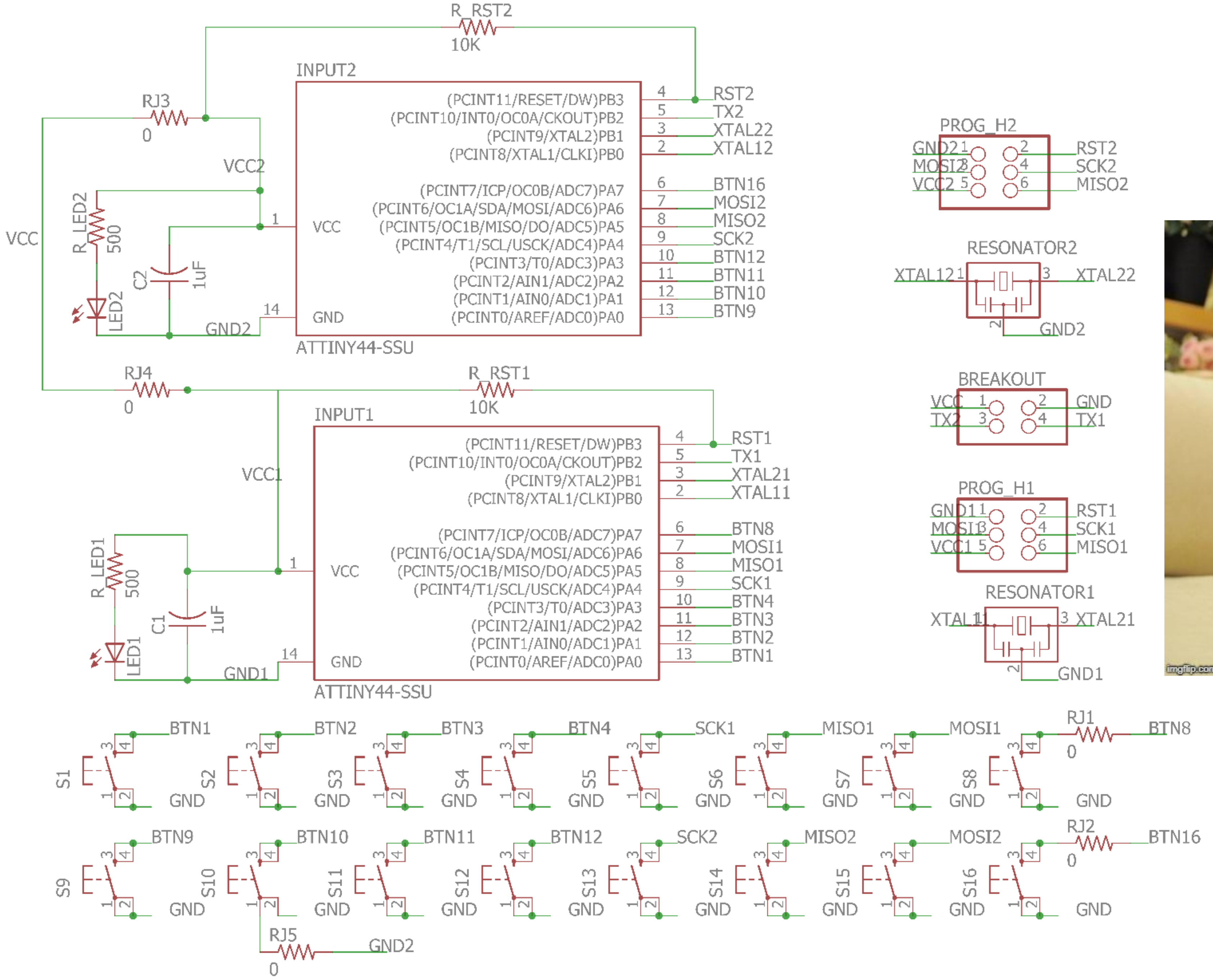
Tips for board schematics and routing:

1. **Do the schematic first** (and finish it before moving on to routing)

All of the parts can be found at
https://gitlab.fabcloud.org/pub/libraries/index

Tips for board schematics and routing:

1. **Do the schematic first** (and finish it before moving on to routing)
2. **Use lots of names** to keep the schematic clean


R_RST2
10K
INPUT2
RJ3
0
VCC2
R_LED2
500
VCC
C2
1uF
LED2
GND2
14
1
VCC
GND
(PCINT11/RESET/DW)PB3
(PCINT10/INT0/OC0A/CKOUT)PB2
(PCINT9/XTAL2)PB1
(PCINT8/XTAL1/CLKI)PB0
(PCINT7/ICP/OC0B/ADC7)PA7
(PCINT6/OC1A/SDA/MOSI/ADC6)PA6
(PCINT5/OC1B/MISO/DO/ADC5)PA5
(PCINT4/T1/SCL/USCK/ADC4)PA4
(PCINT3/T0/ADC3)PA3
(PCINT2/AIN1/ADC2)PA2
(PCINT1/AIN0/ADC1)PA1
(PCINT0/AREF/ADC0)PA0
RST2
TX2
XTAL22
XTAL12
BTN16
MOSI2
MISO2
SCK2
BTN12
BTN11
BTN10
BTN9
ATTINY44-SSU
RJ4
0
R_RST1
10K
INPUT1
VCC1
R_LED1
500
C1
1uF
LED1
GND1
RST1
TX1
XTAL21
XTAL11
BTN8
MOSI1
MISO1
SCK1
BTN4
BTN3
BTN2
BTN1
ATTINY44-SSU
PROG_H2
GND2
MOSI2
VCC2
RST2
SCK2
MISO2
RESONATOR2
XTAL12
XTAL22
GND2
BREAKOUT
VCC
TX2
GND
TX1
PROG_H1
GND1
MOSI1
VCC1
RST1
SCK1
MISO1
RESONATOR1
XTAL11
XTAL21
GND1
S1 S2 S3 S4 S5 S6 S7 S8
BTN1 BTN2 BTN3 BTN4 SCK1 MISO1 MOSI1
RJ1
0
BTN8
GND
S9 S10 S11 S12 S13 S14 S15 S16
BTN9 BTN10 BTN11 BTN12 SCK2 MISO2 MOSI2
RJ2
0
BTN16
RJ5
0
GND2

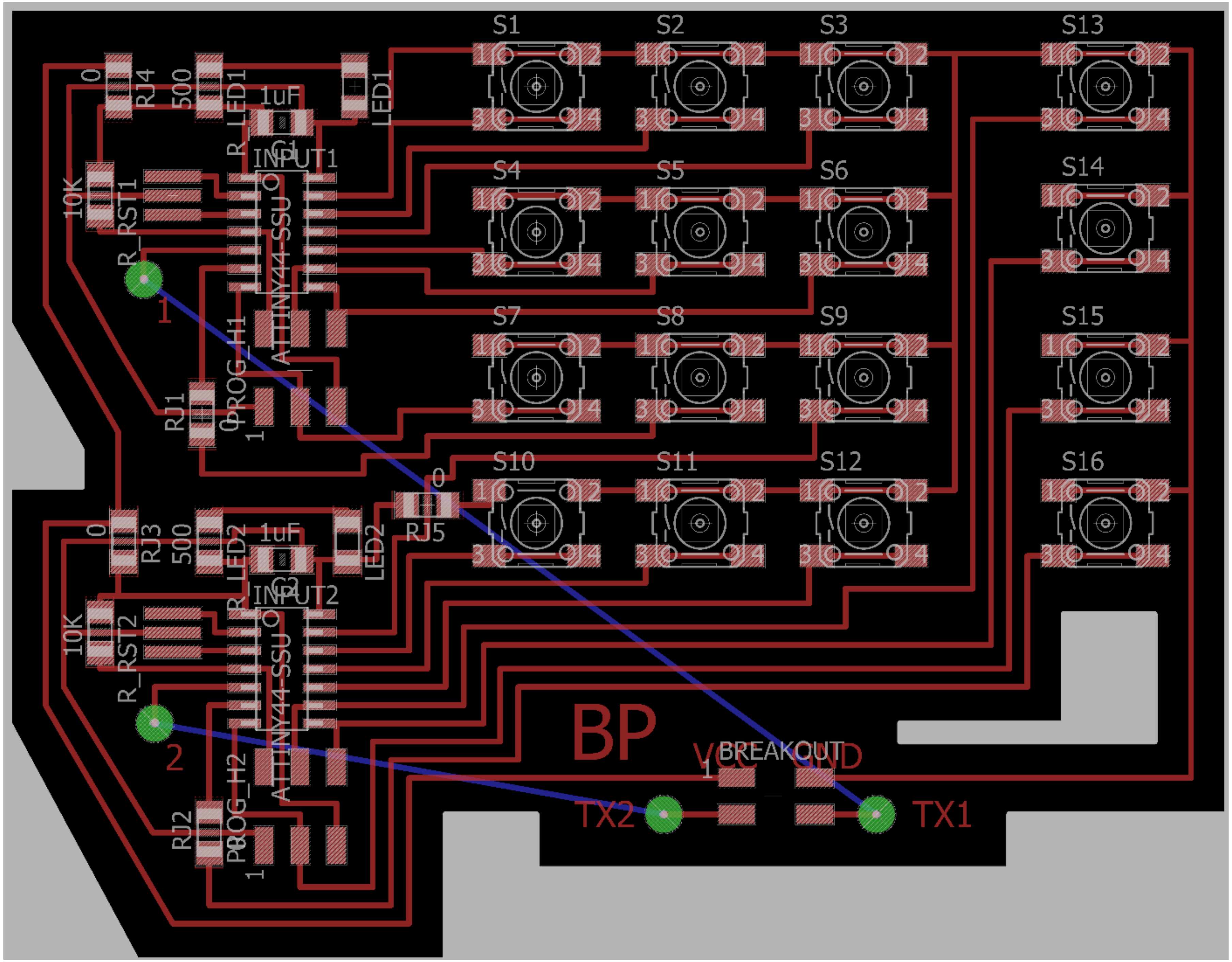

SO CLEAN
MUCH NAMES
imgflip.com

Tips for board schematics and routing:

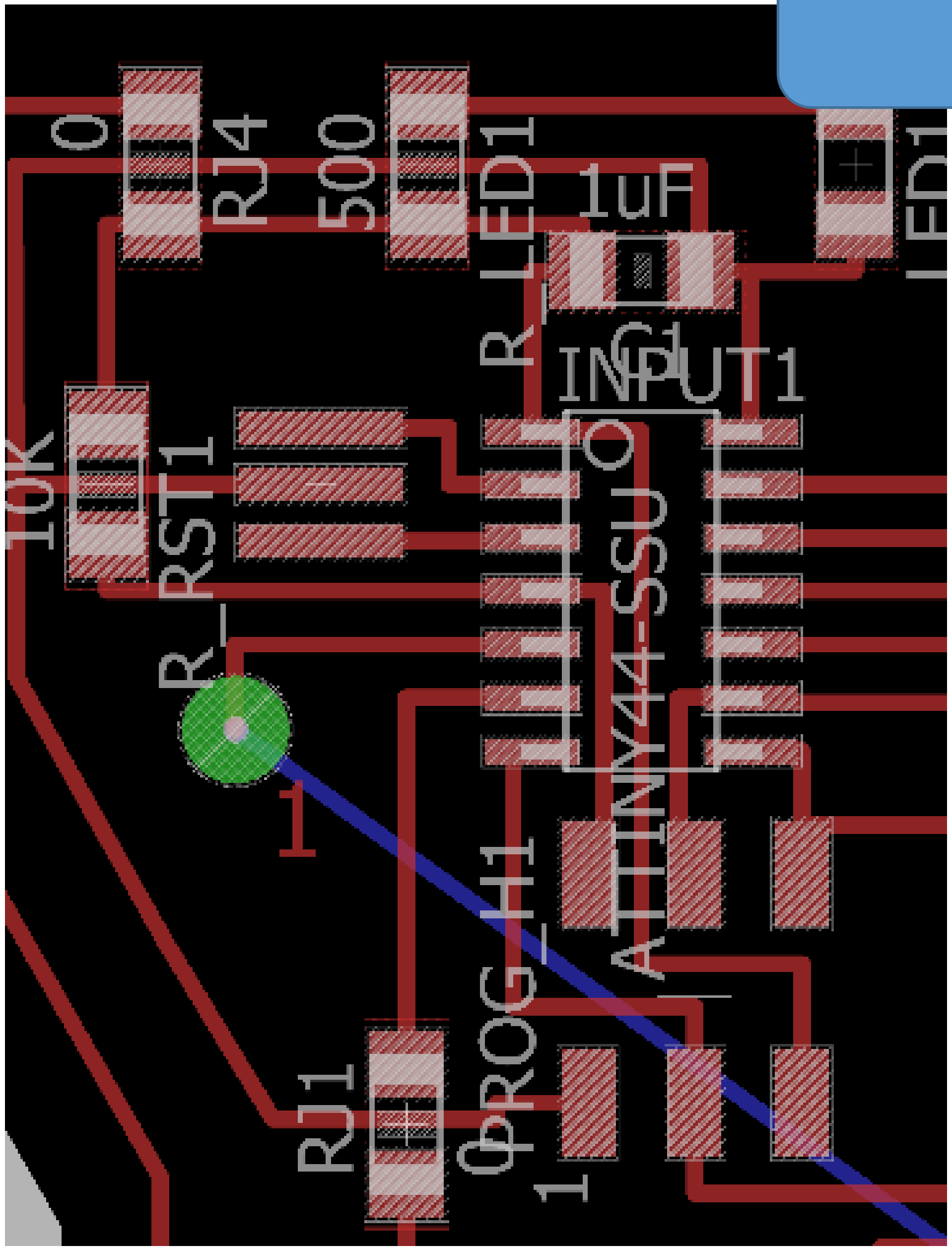
1. **Do the schematic first** (and finish it before moving on to routing)
2. **Use lots of names** to keep the schematic clean
3. **Triple check the schematic** before moving onto routing (and have someone else check it)

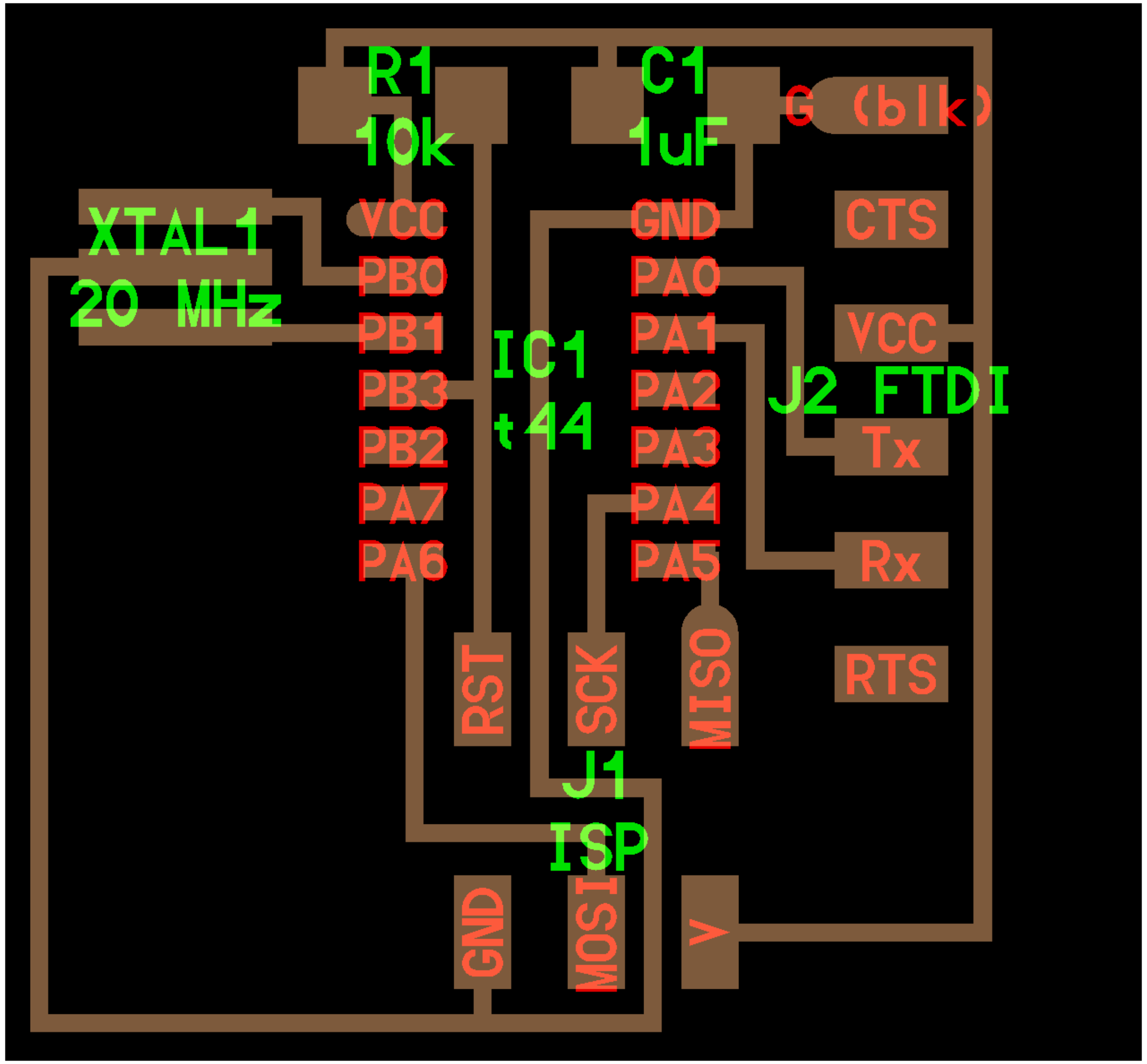
Tips for board schematics and routing:

1. **Do the schematic first** (and finish it before moving on to routing)
2. **Use lots of names** to keep the schematic clean
3. **Triple check the schematic** before moving onto routing (and have someone else check it)
4. **Copy the routing patterns Neil or others use**


S1
S2
S3
S13
S4
S5
S6
S14
S7
S8
S9
S15
S10
S11
S12
S16
RJ4
0
500
R_LED1
1uF
LED1
C1
INPUT1
10K
R_RST1
ATTINY44-SSU
PROG_H1
RJ1
1
RJ5
RJ3
R_LED2
LED2
C3
INPUT2
R_RST2
PROG_H2
RJ2
2
BP
VCC
BREAKOUT
GND
TX2
TX1

Not so different after all…

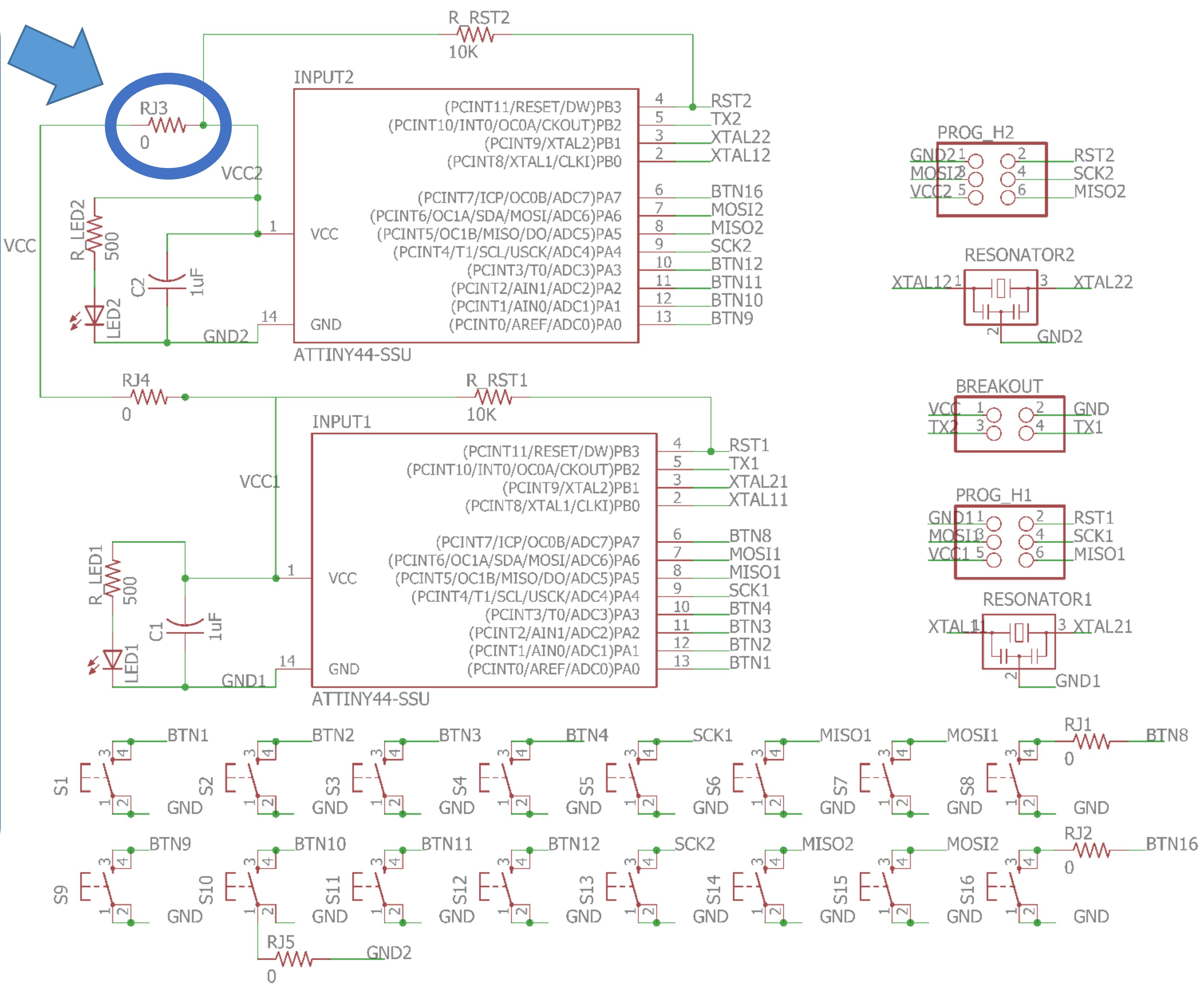


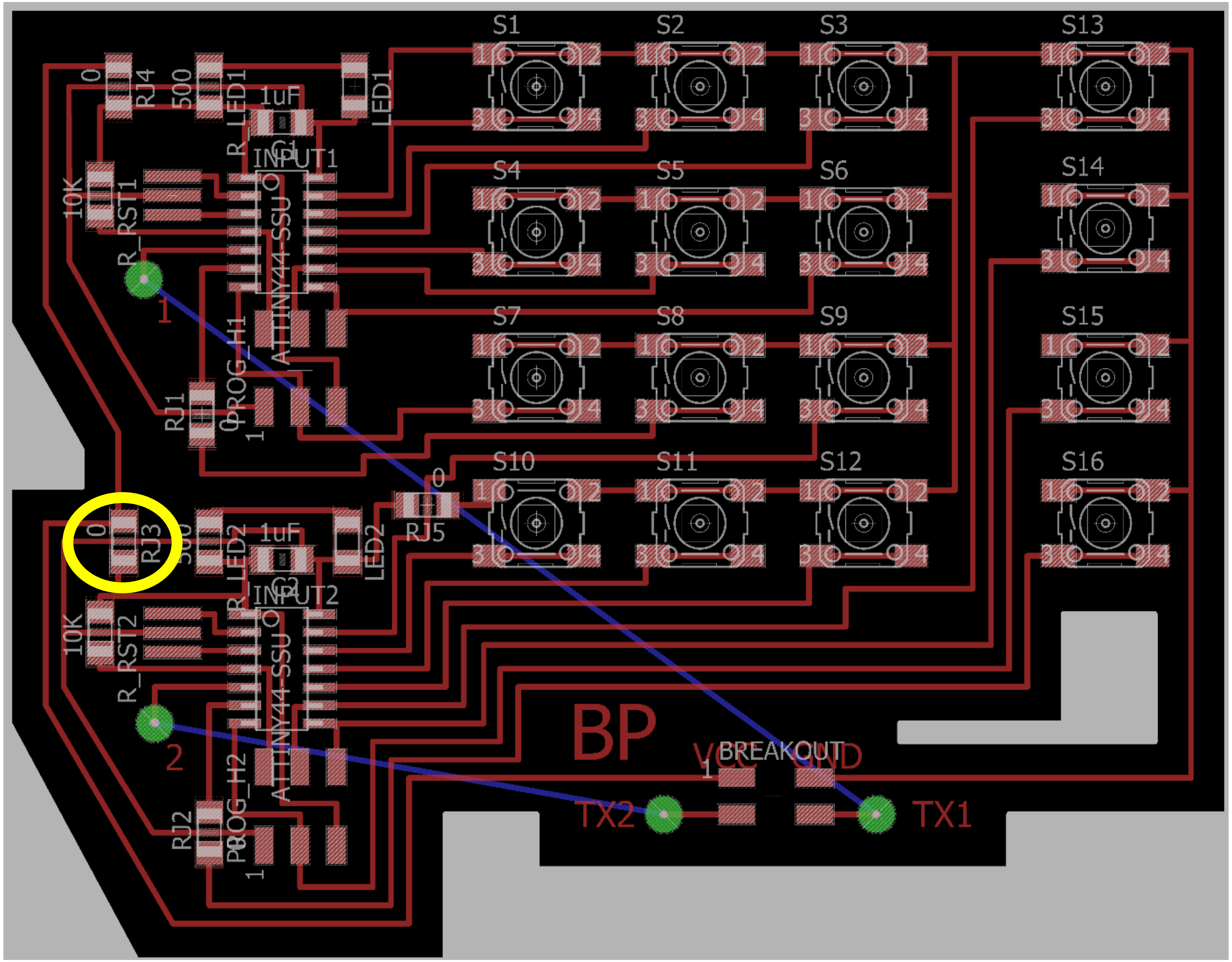


Tips for board schematics and routing:

1. **Do the schematic first** (and finish it before moving on to routing)
2. **Use lots of names** to keep the schematic clean
3. **Triple check the schematic** before moving onto routing (and have someone else check it)
4. **Copy the routing patterns Neil or others use**
5. **Add 0 ohm resistors** if you get stuck routing

Here's a nice example of a 0 ohm resistor that was added later during routing





S1
S2
S3
S13
S4
S5
S6
S14
S7
S8
S9
S15
S10
S11
S12
S16
RJ4
R_LED1
LED1
1uF
INPUT1
10K
R_RST1
ATTINY44-SSU
PROG_H1
RJ1
RJ3
R_LED2
LED2
INPUT2
R_RST2
PROG_H2
RJ2
RJ5
BP
BREAKOUT
VCC
GND
TX1
TX2

Tips for board schematics and routing:

1. **Do the schematic first** (and finish it before moving on t
2. **Use**
3. **Trip**
   rou
4. **Cop**
5. **Add 0 ohm resistors** if you get stuck routing

I promise it gets way way easier after you do this a couple times.

The road ahead

You can save yourself time if you make a board that has an LED and a button so you can program it to turn the LED on and off with the button... (at least that used to be the minimal assignment...)

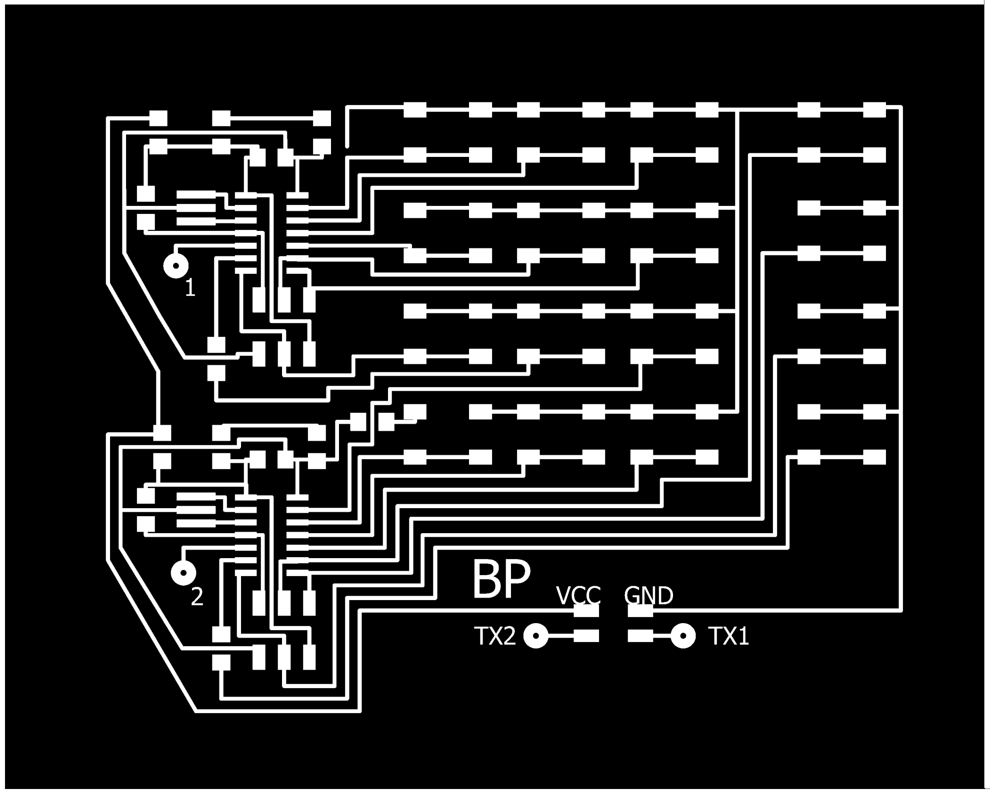
This week	2 weeks from now	4 weeks from now
Electronics Production	Electronics Design	Embedded Programming
• Mill and stuff a circuit board	• Design your own circuit board • Mill and stuff it	• Design your own circuit board • Mill and stuff it • Program it

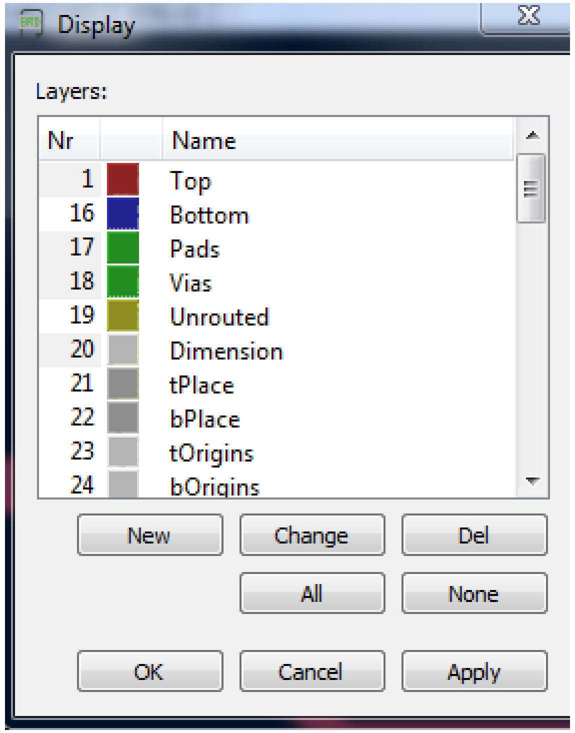
BUT WAIT!

BUT WAIT!

How do we mill a physical board once we have a board file?!?

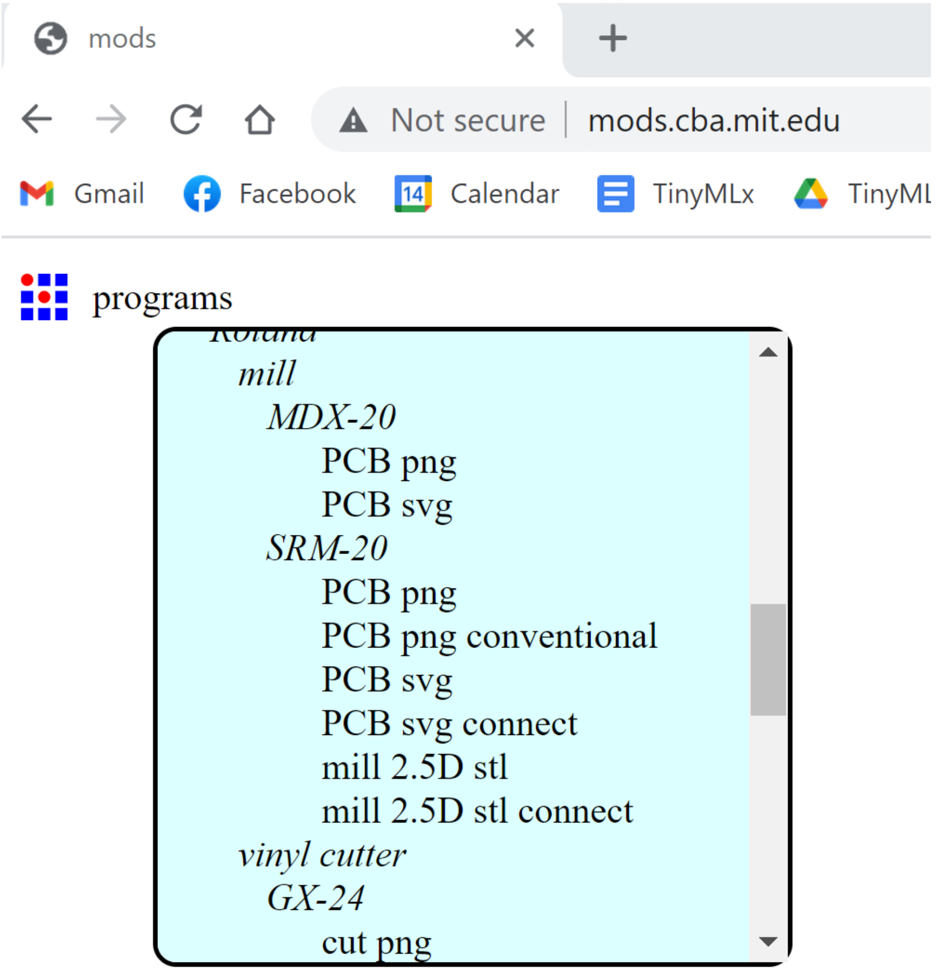
Export the traces and outline





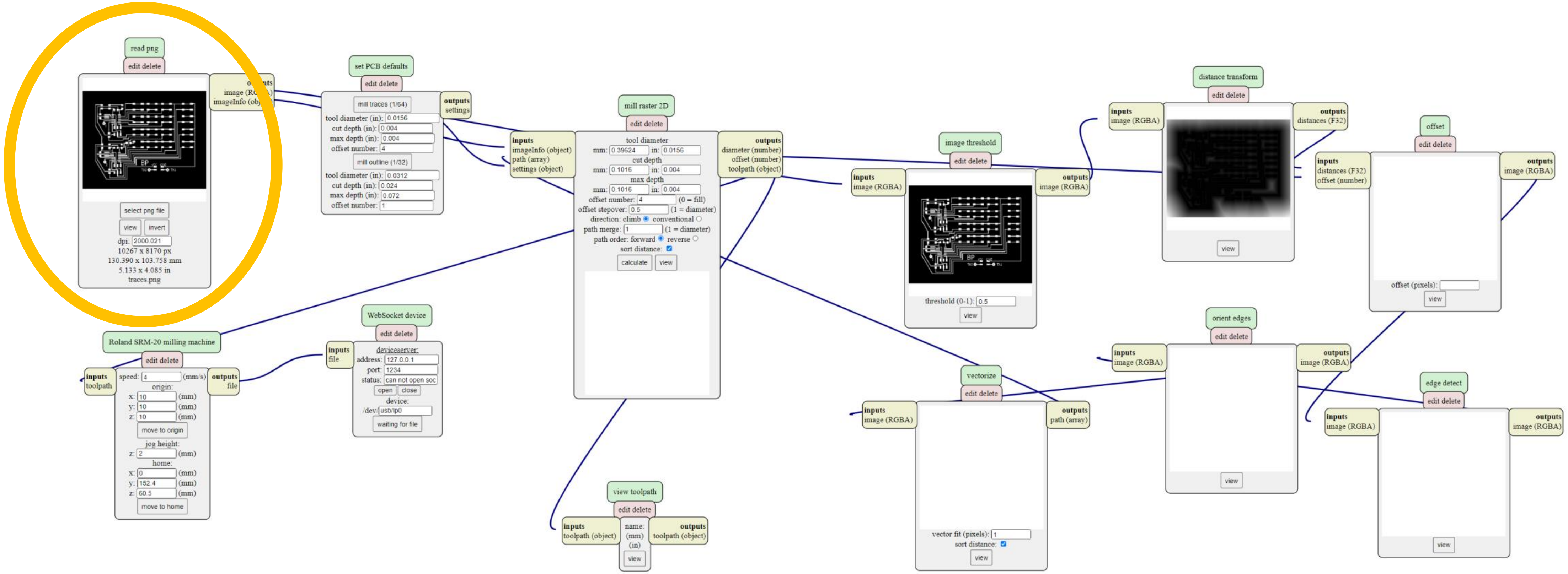
Make sure to export in monochrome and keep track of the DPI

Import it into Neil’s Fab Mods

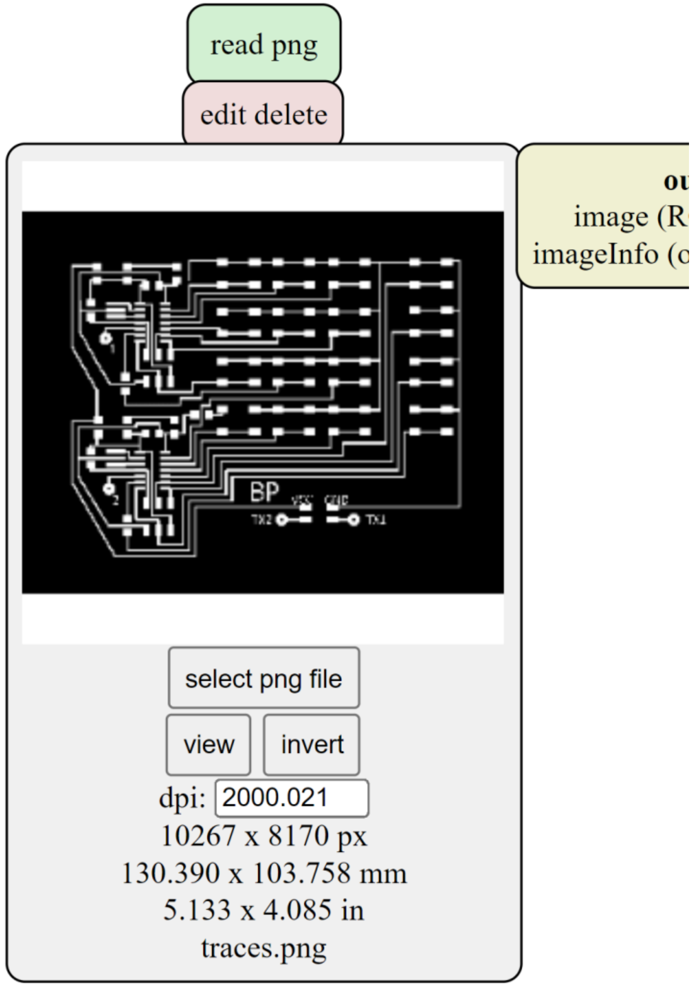


Make sure to pick the right machine and program for your lab – talk to your lab TAs!

Import it into Neil's Fab Mods

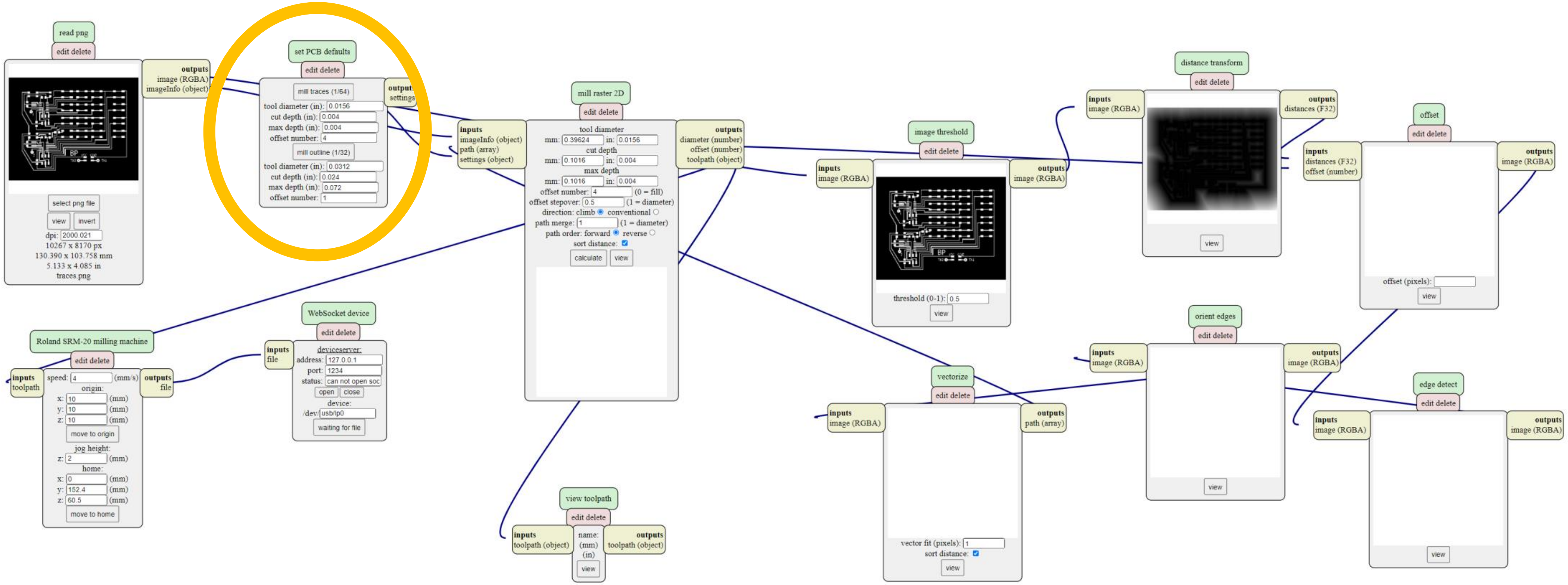


Import it into Neil's Fab Mods

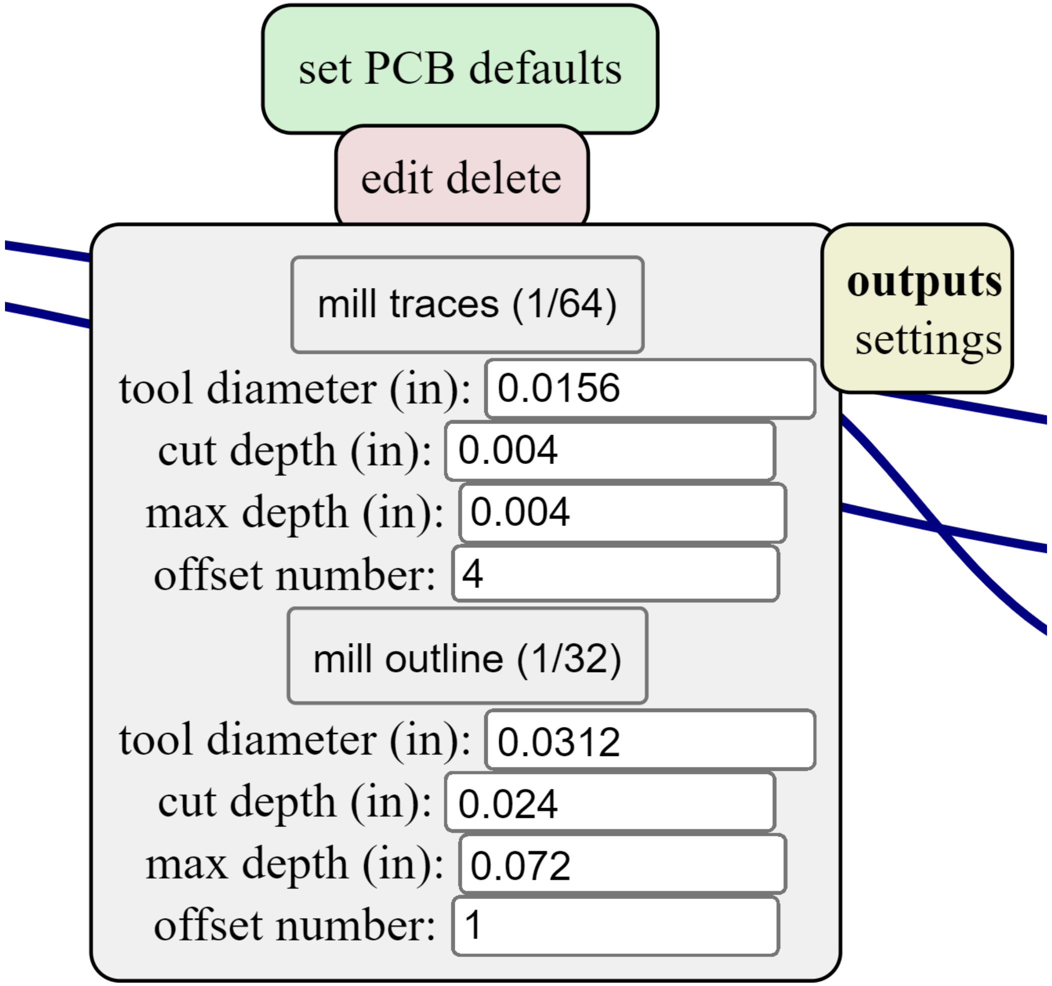


Select your image and **make sure the DPI matches**!

Import it into Neil’s Fab Mods

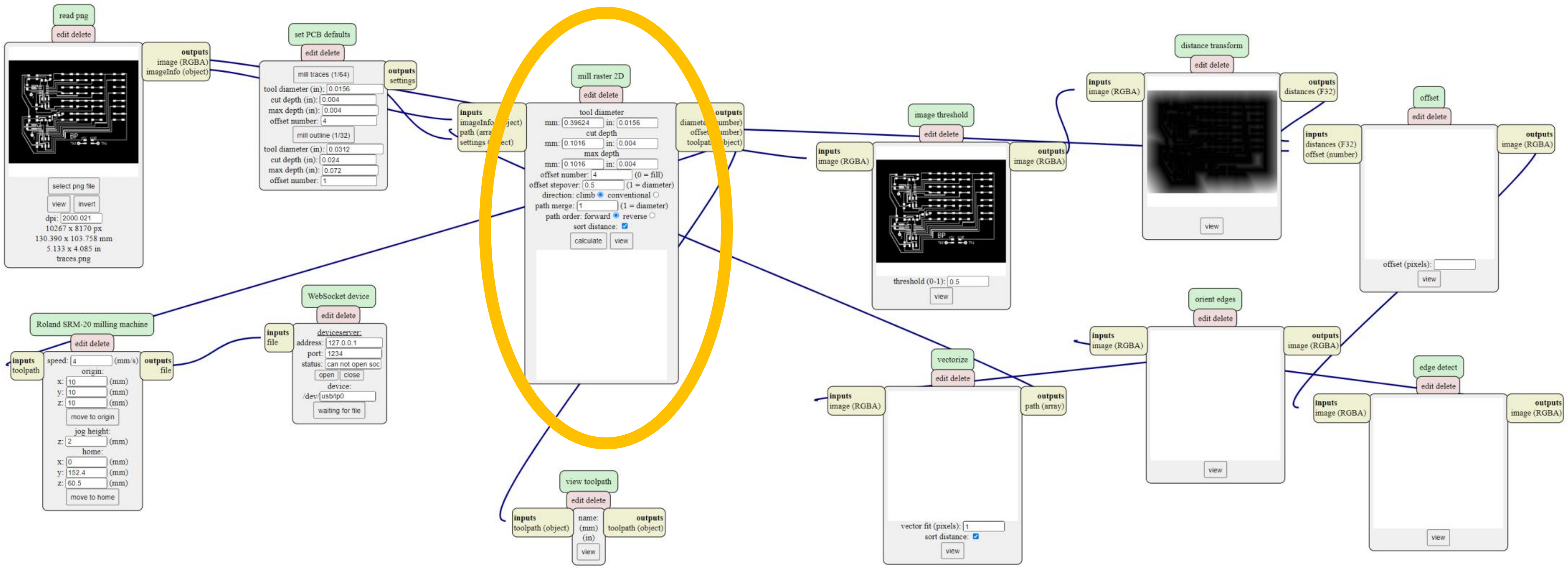


Import it into Neil's Fab Mods

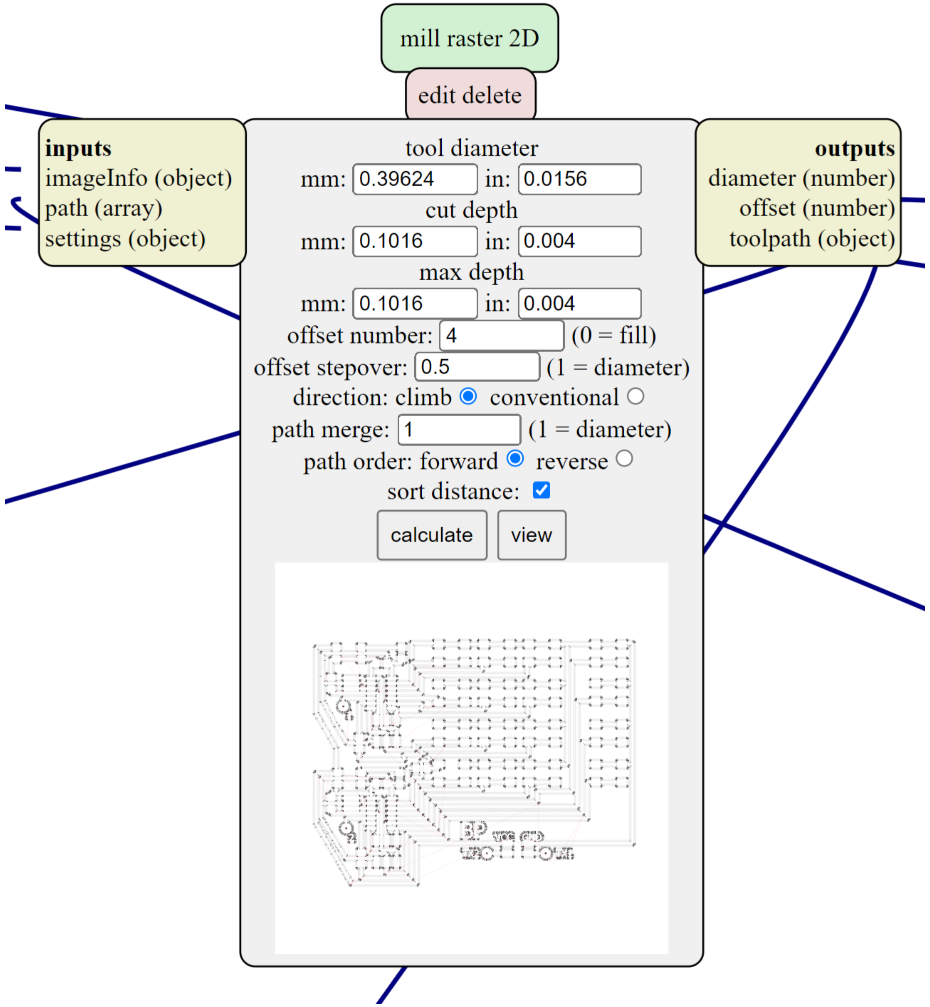


Select traces or outline – **make sure you have the right bit in the machine** – talk to your lab TAs about if you need to adjust this at all

Import it into Neil's Fab Mods

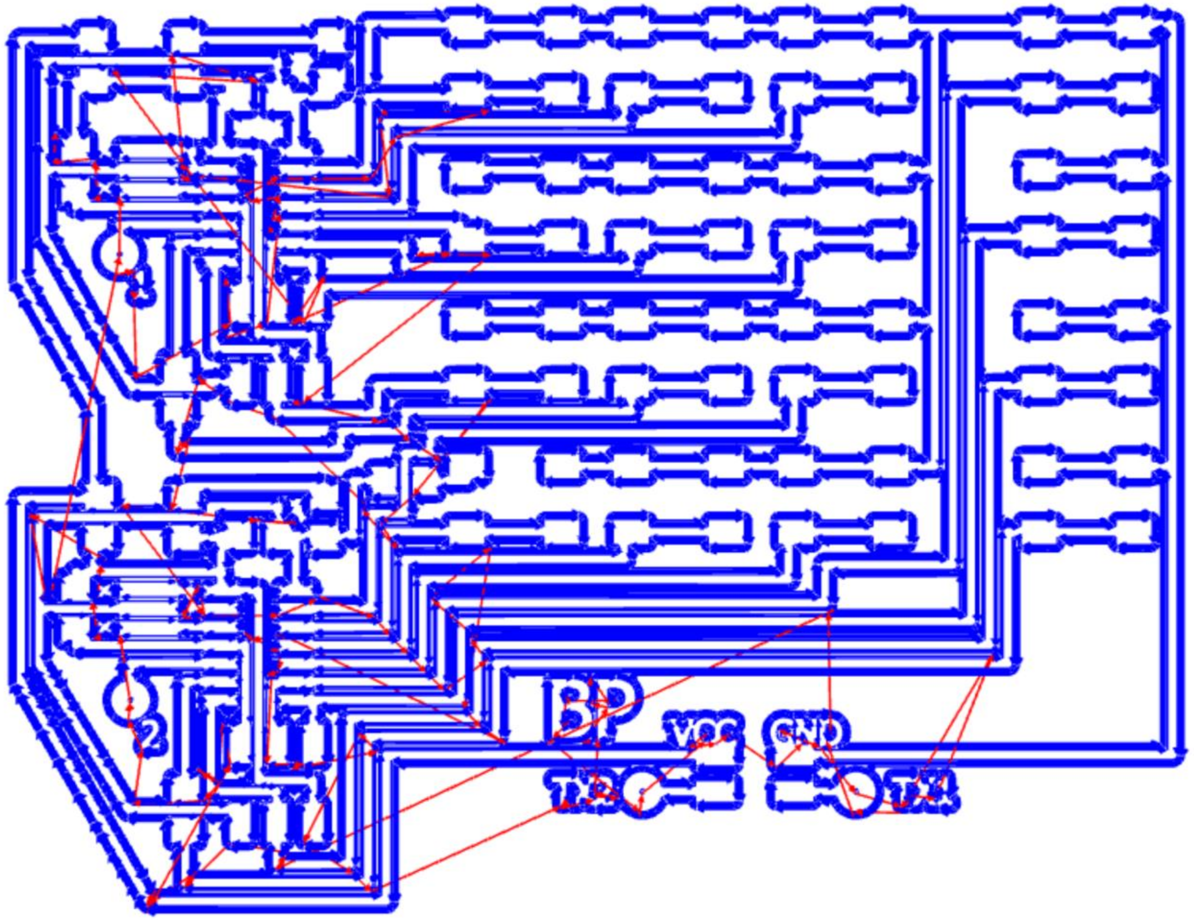


Import it into Neil’s Fab Mods



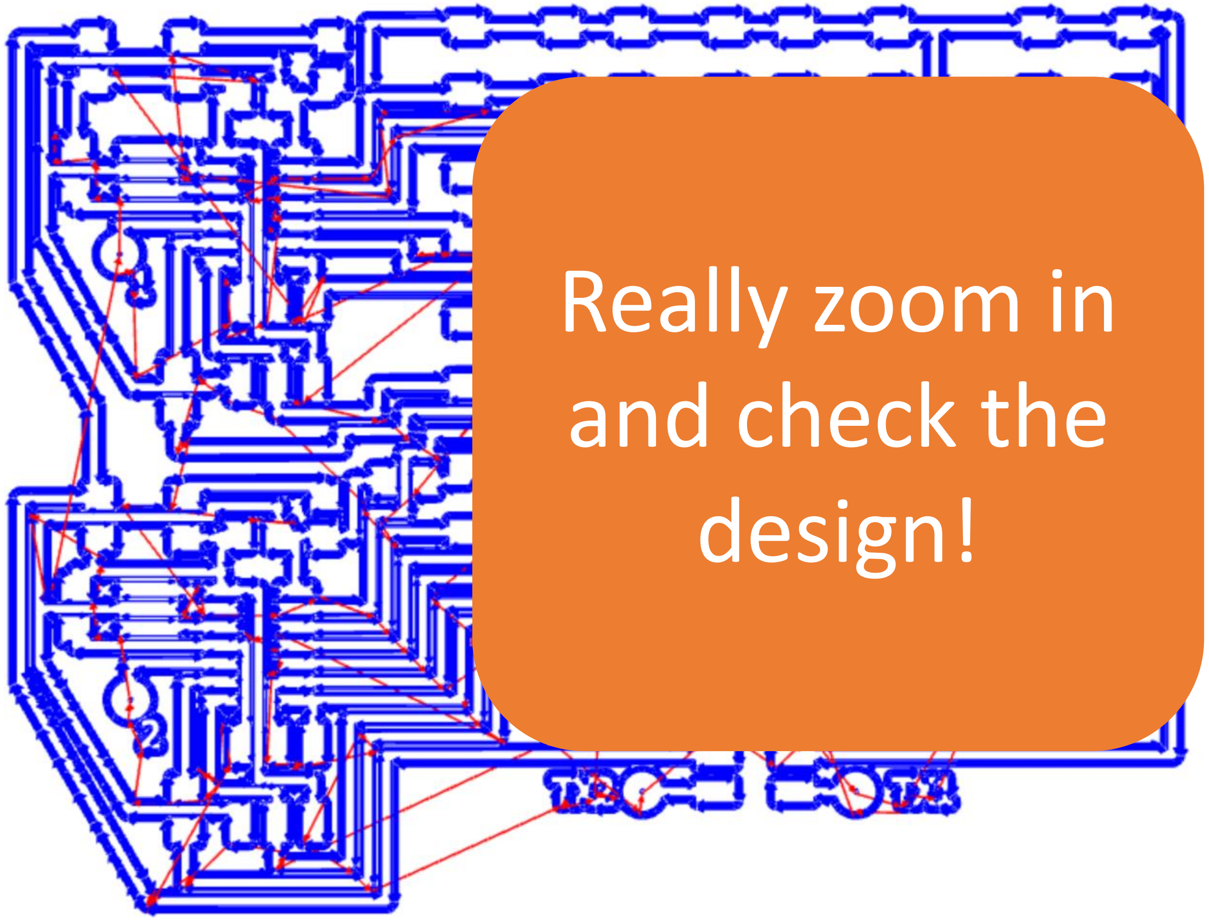
Again check with your TAs about these settings but/and after you press calculate it will compute the toolpath

Import it into Neil’s Fab Mods

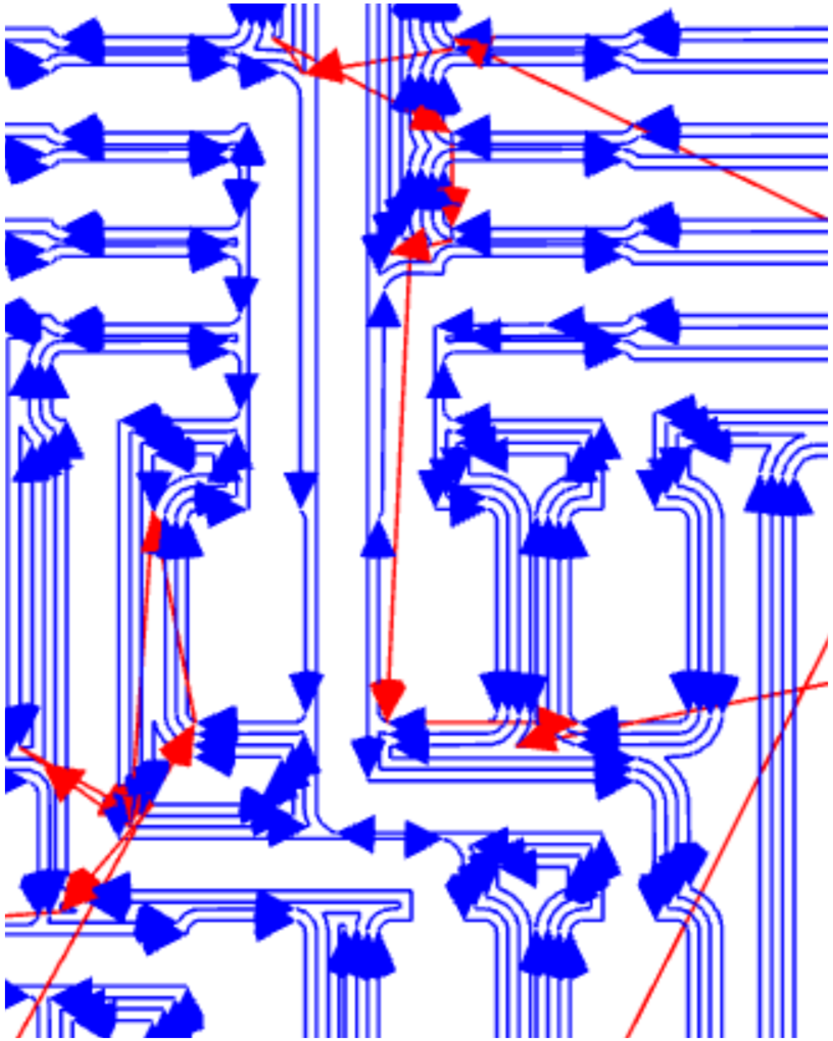


You’ll get a pop-up with the design when its done

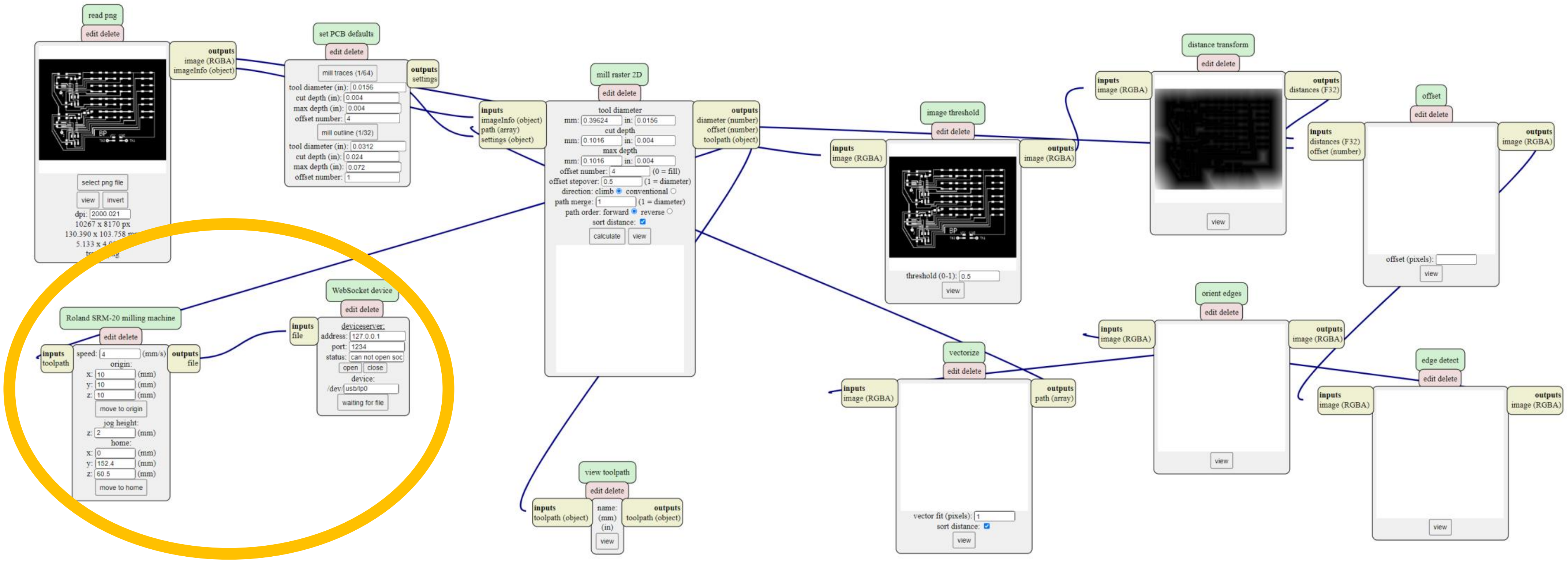
Import it into Neil’s Fab Mods



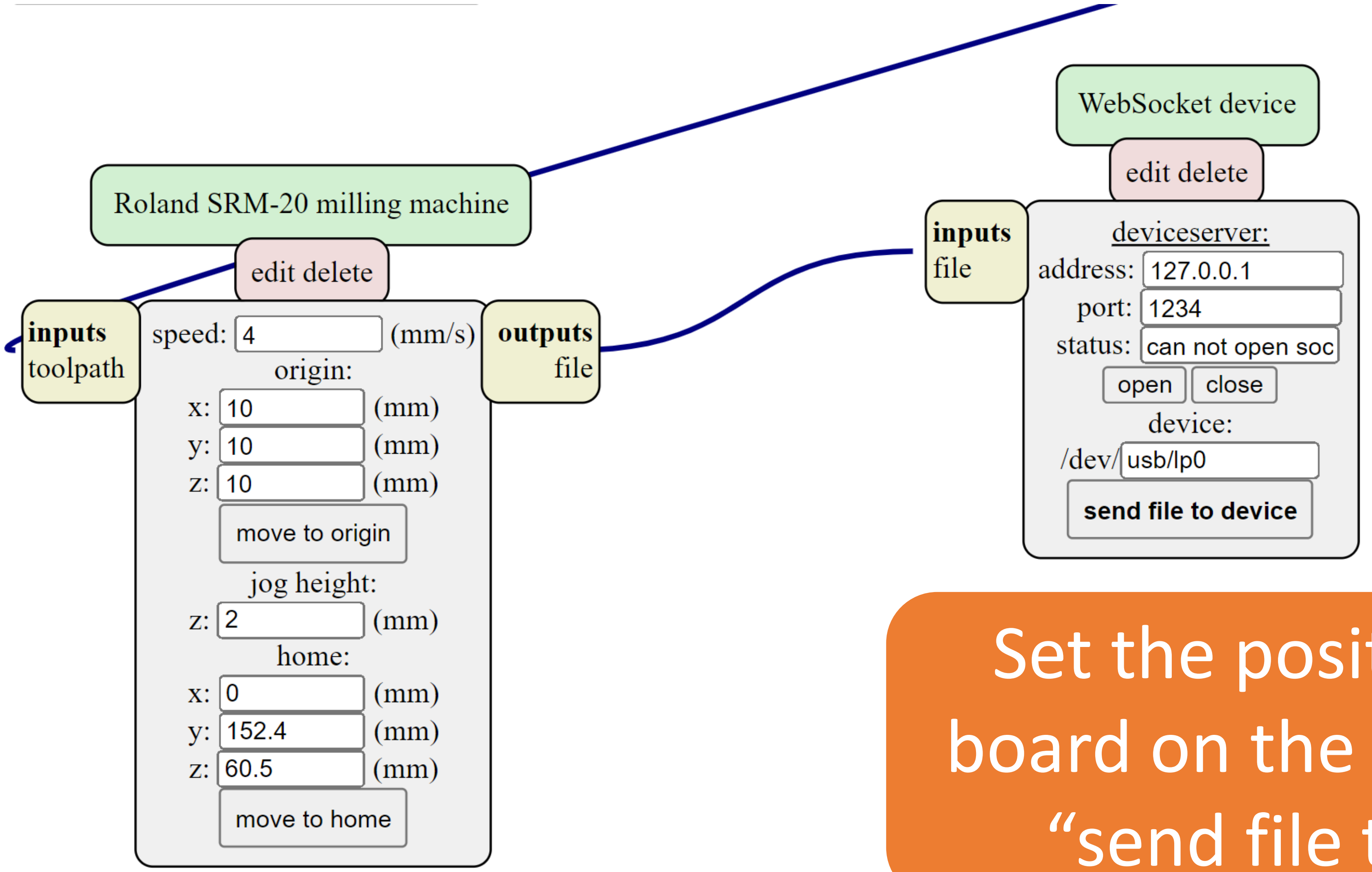
Really zoom in and check the design!



Import it into Neil's Fab Mods

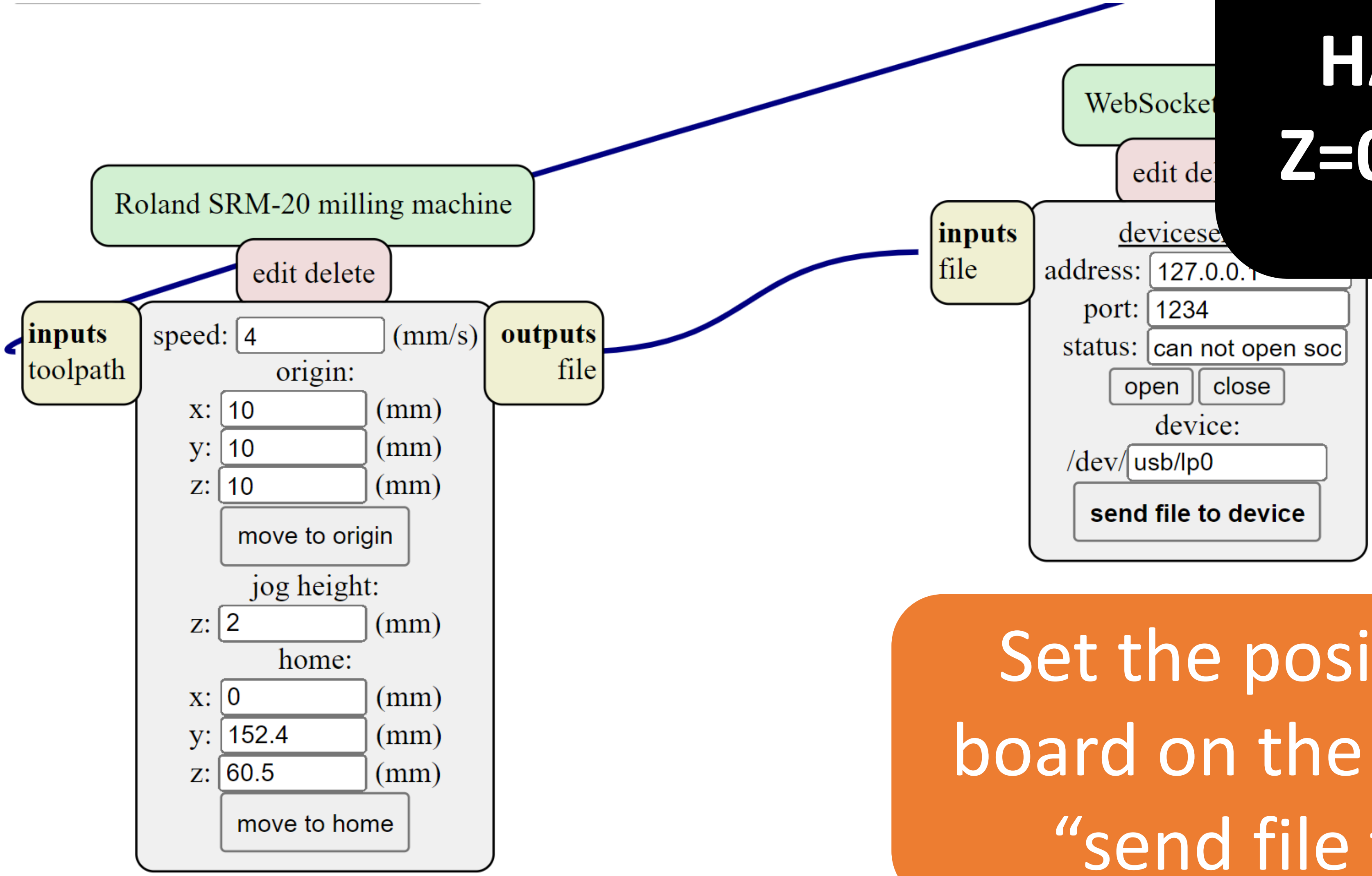


Import it into Neil's Fab Mods



Set the position of your board on the machine and "send file to device"

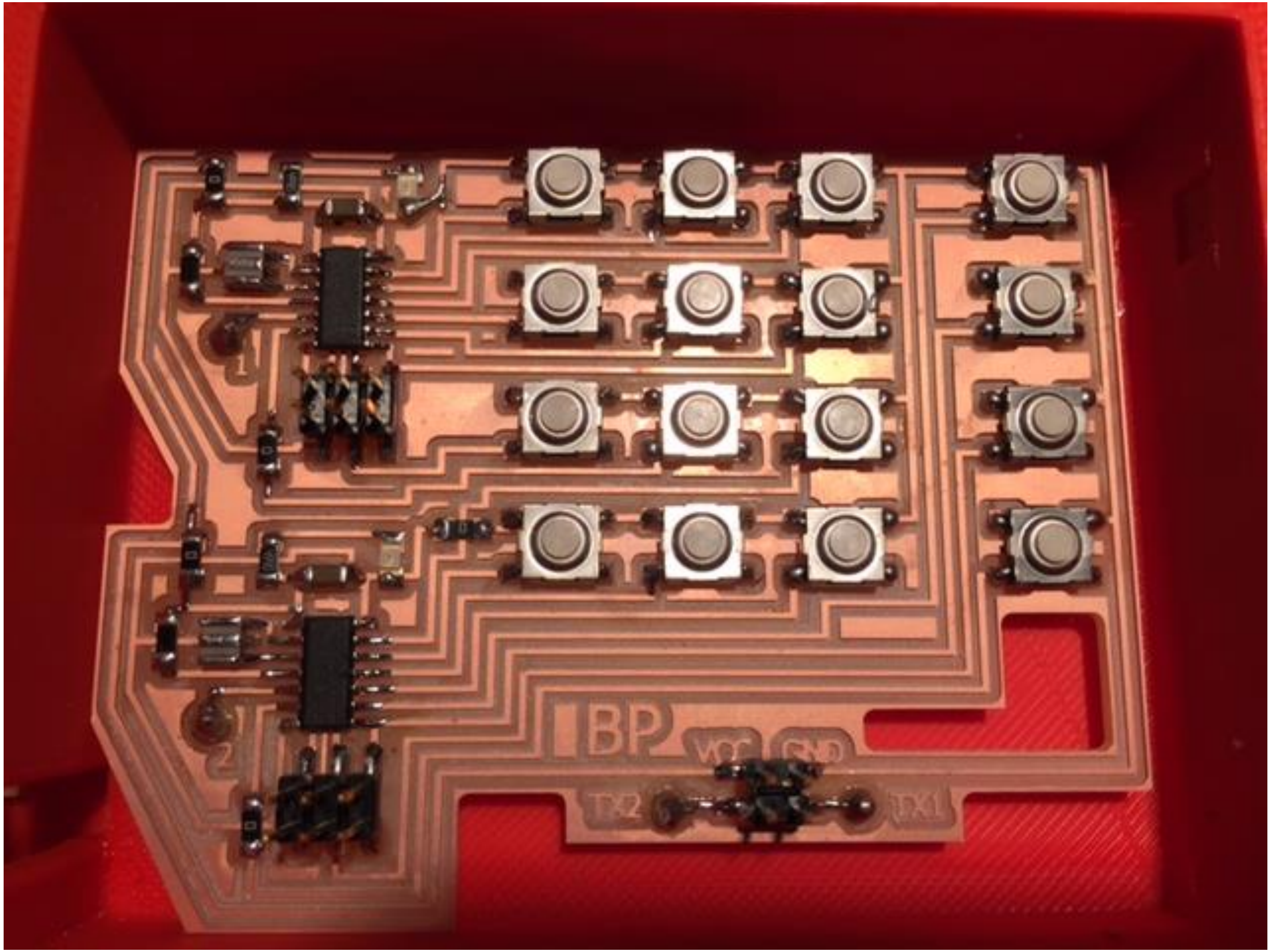
Import it into Neil's Fab Mods



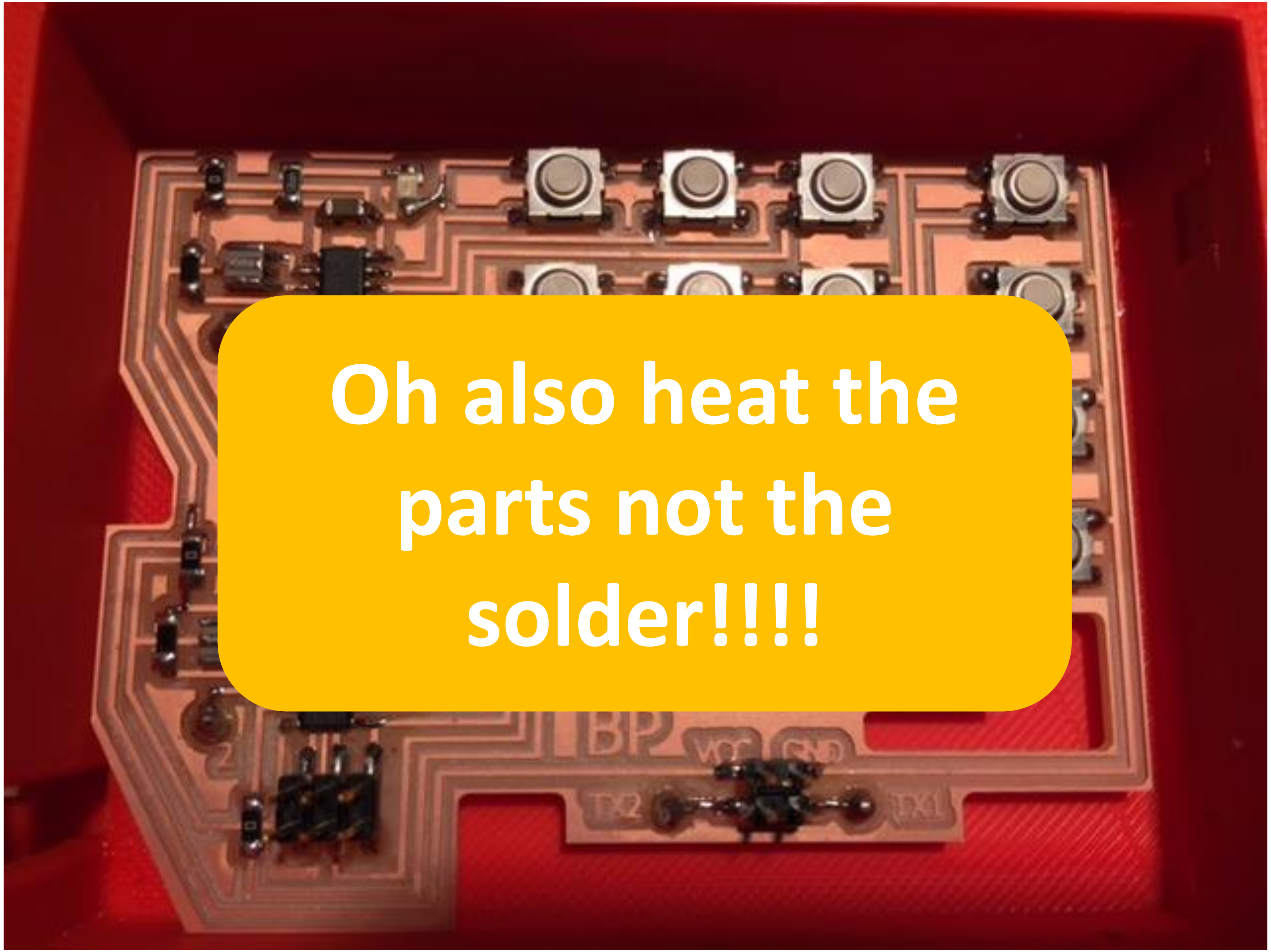
MAKE SURE YOU HAVE SET THE Z=0 Correctly!!!!

Set the position of your board on the machine and "send file to device"

And by your final project you too will be making crazy boards like this one!



And by your final project you too will be making crazy boards like this one!



Oh also heat the parts not the solder!!!!

We now may be too powerful for our own good!

A short outline for today

1. Almost all you need to know about Electrical Engineering

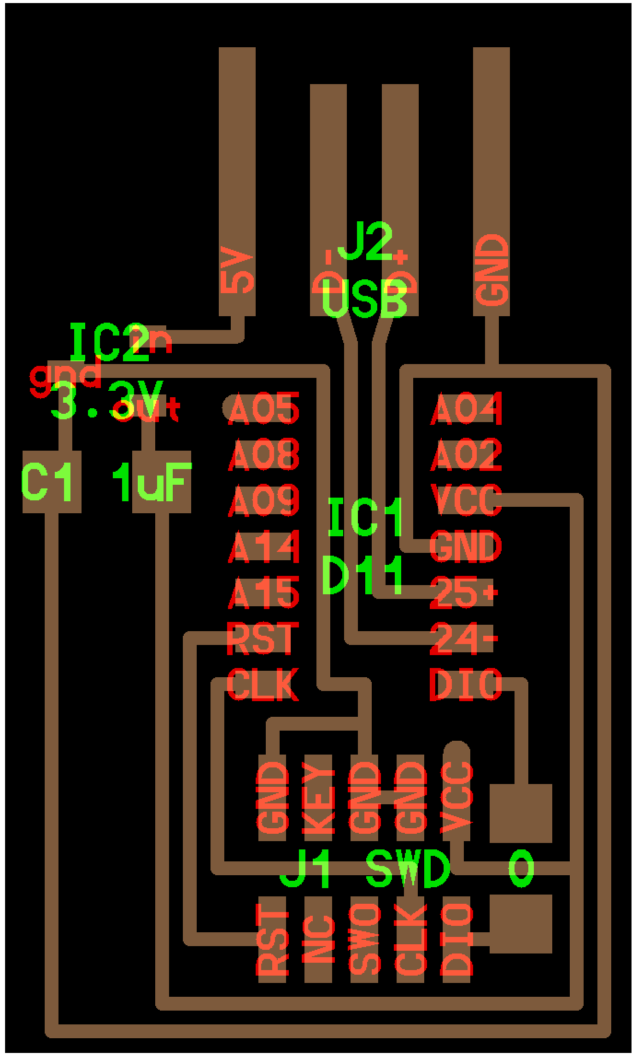
2. Almost all the tips you need to design custom boards

3. Almost all the steps it will take to produce a custom board

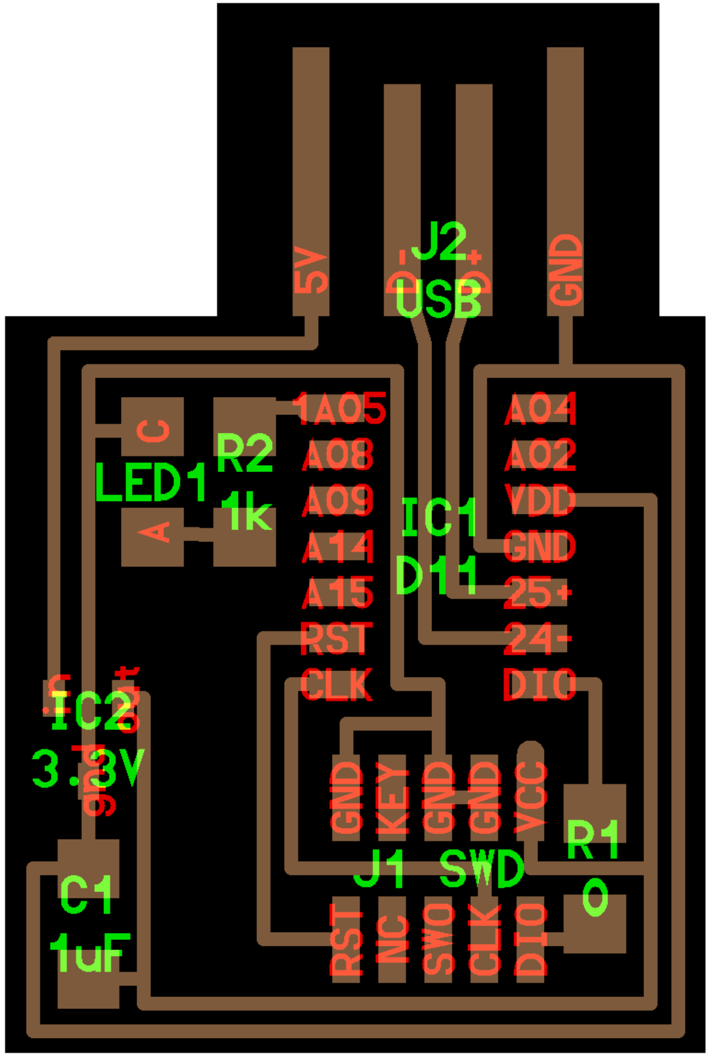
4. **Can you do my homework for me?**

No…. But/and….

No.... But/and....



Plus LED



Plus Button

?

Good Luck!

More demos and details can be found from last year at these links (and more software details to come at a later recitation)

[Electronics 101 Video]

[Eagle and KiCaD Overviews Video]

And I did a version of this two years ago with a little Eagle demo at this Video