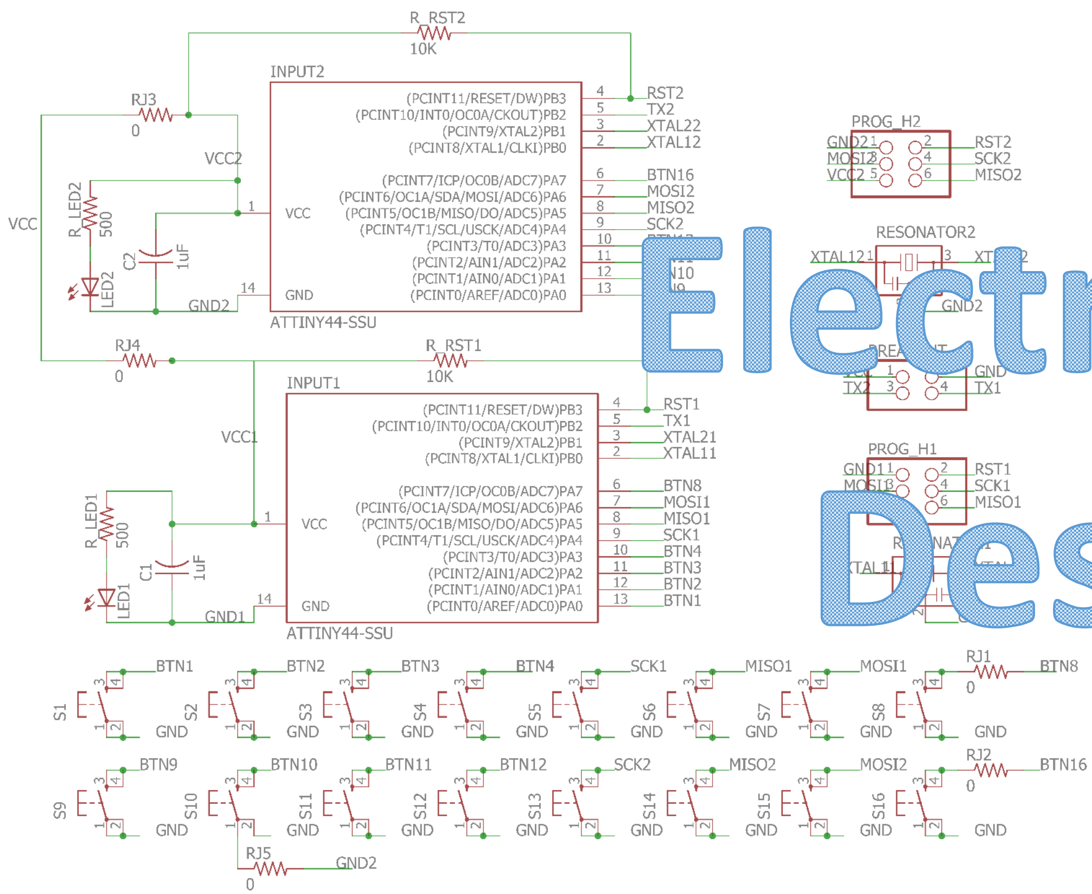
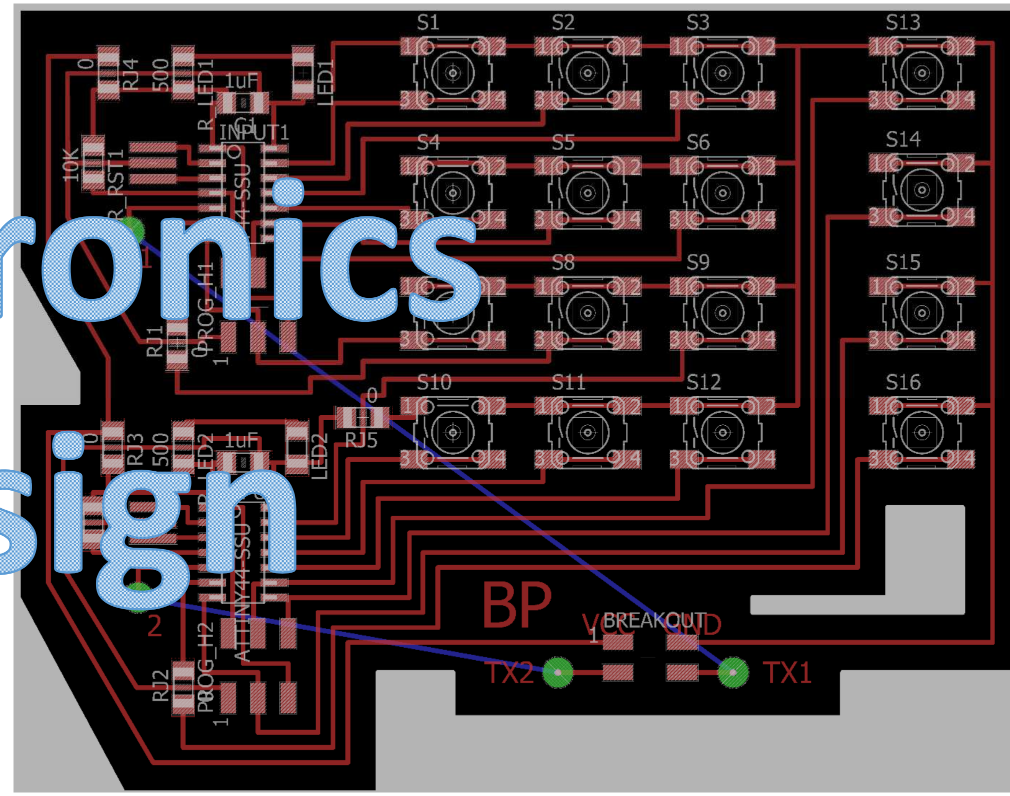
Electronics Design






RELAX
WE GOT THIS

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

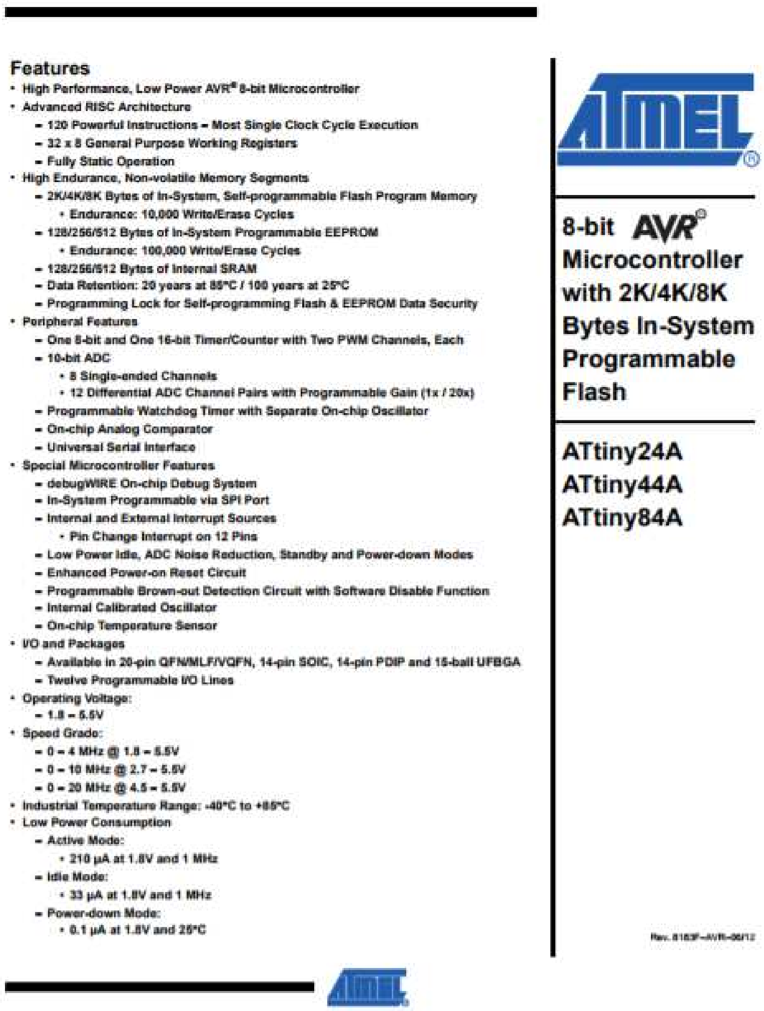
8-bit AVR® Microcontroller with 2K/4K/8K Bytes In-System Programmable Flash

ATtiny24A
ATtiny44A
ATtiny84A

Rev. 8183F–AVR–06/12

We're going to end up basing our circuit around using an ATtiny44 or 45 so lets take a quick look at the specs for the 44 to see what we can have it do….

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



Features

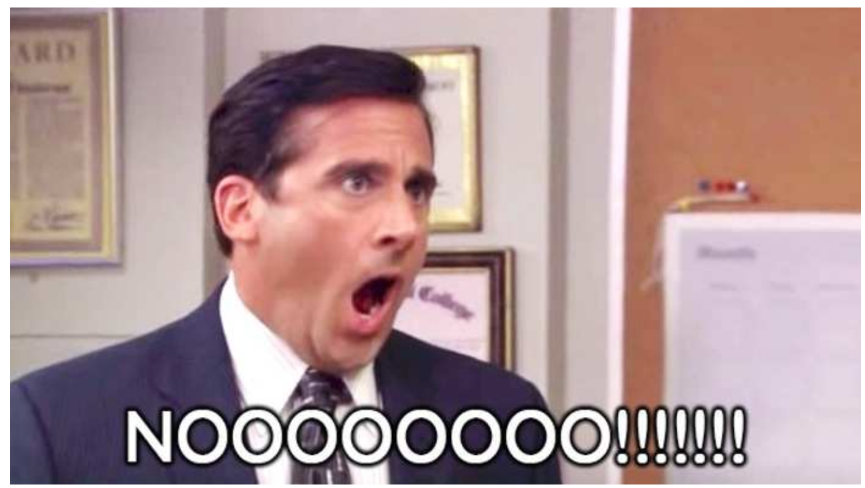
- High Performance, Low Power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 120 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
- High Endurance, Non-volatile Memory Segments
  - 2K/4K/8K Bytes of In-System, 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

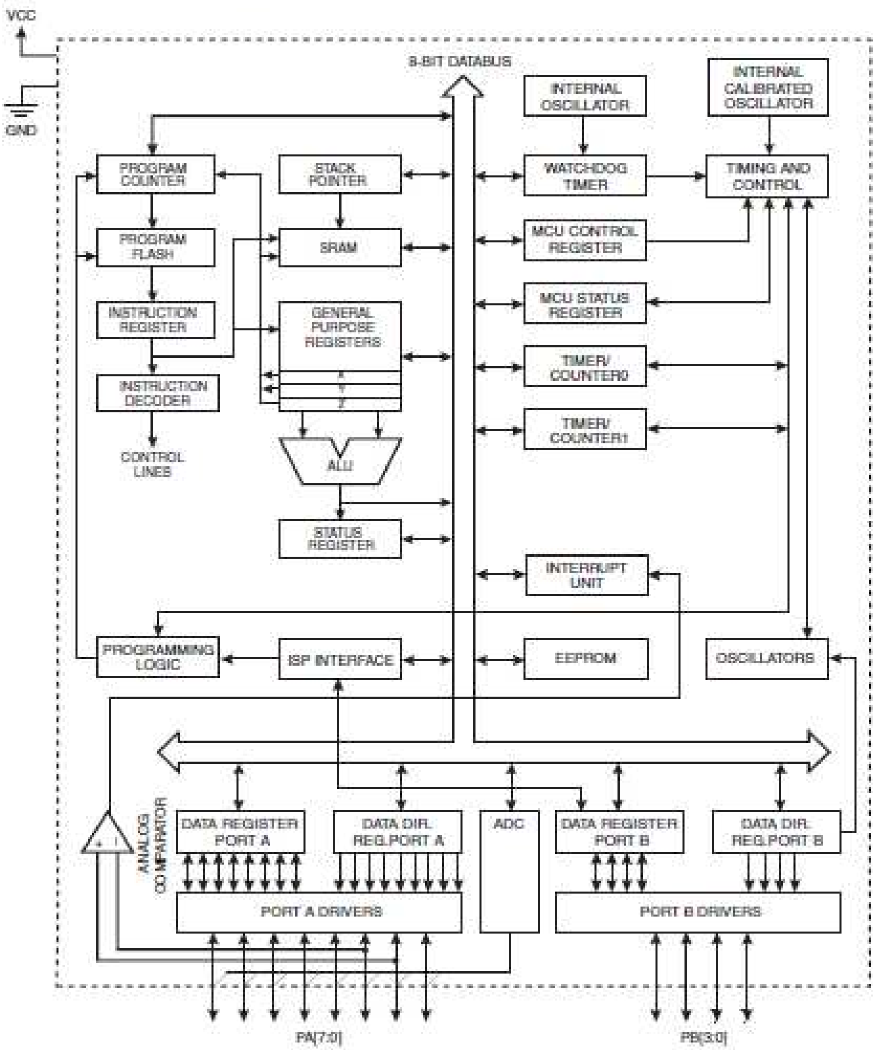
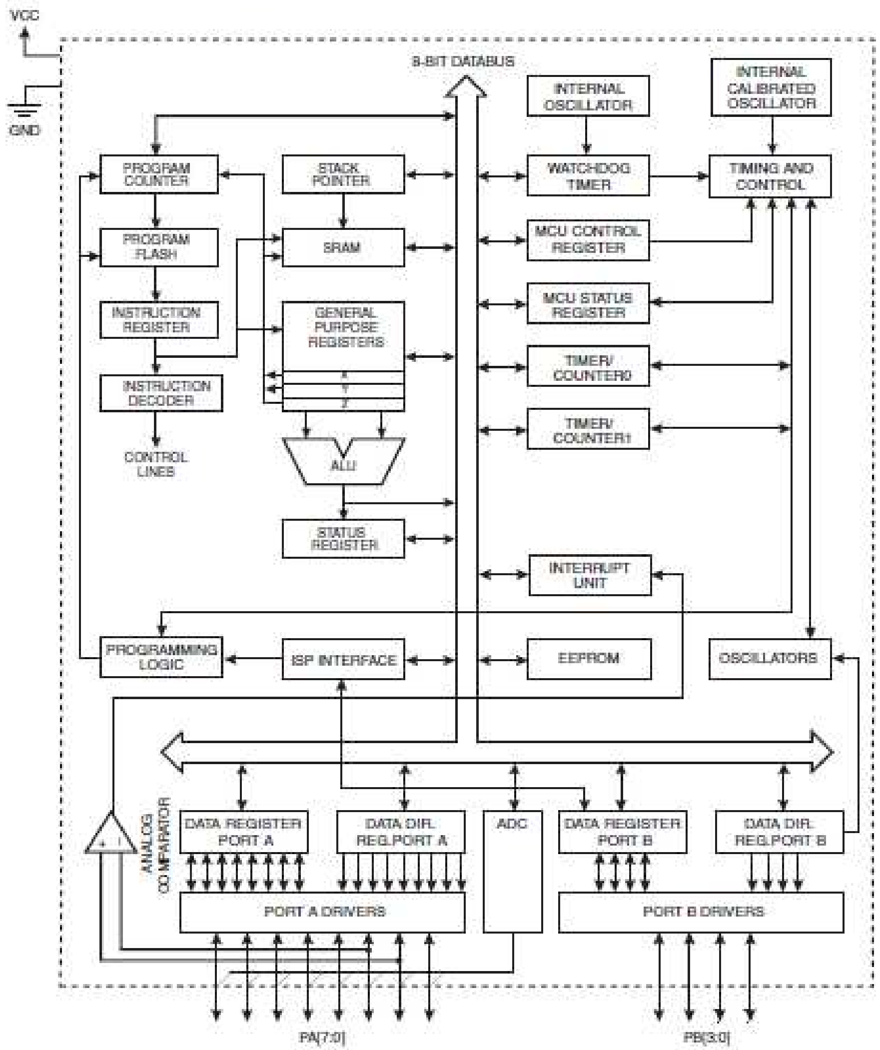
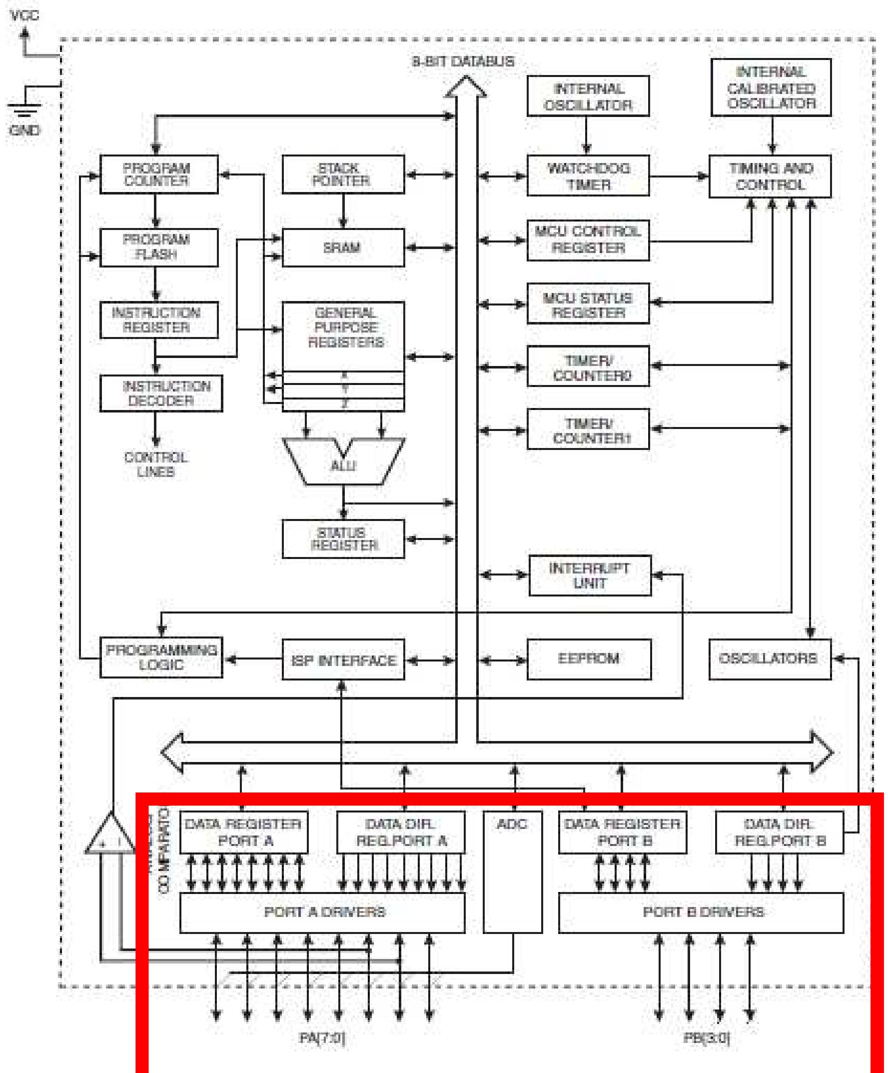


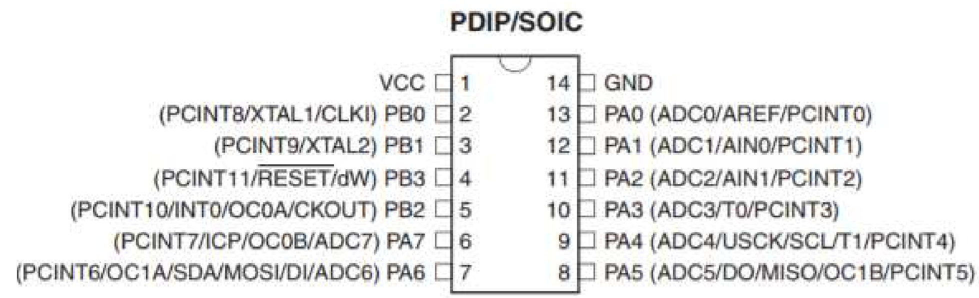
Figure 2-1. Block Diagram



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

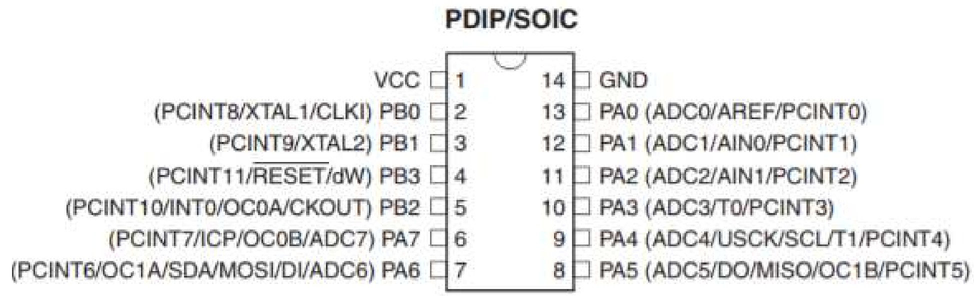




Hey look here's some port stuff seems like it has something to do with the inputs!

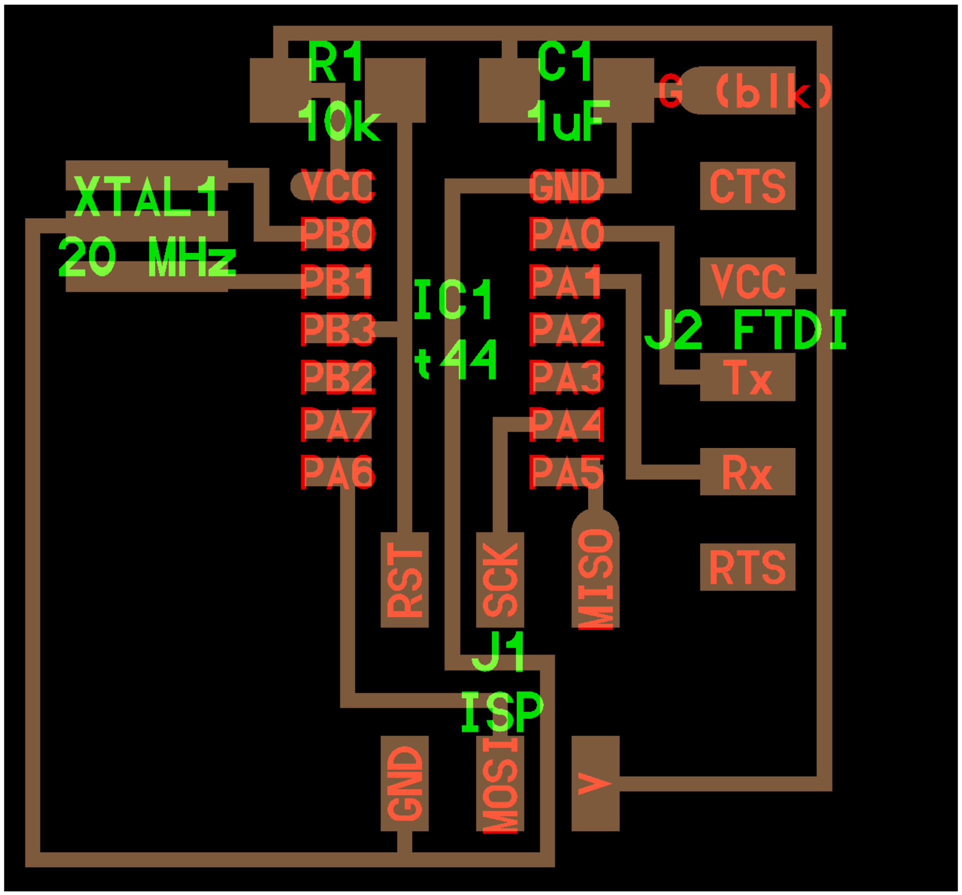
Table 10-3. Port A Pins Alternate Functions

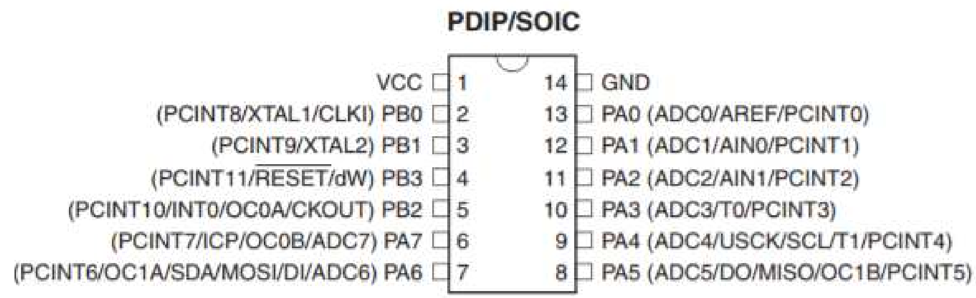
Port Pin	Alternate Function
PA0	ADC0: ADC Input Channel 0 AREF: External Analog Reference PCINT0: Pin Change Interrupt 0, Source 0
PA1	ADC1: ADC Input Channel 1 AIN0: Analog Comparator, Positive Input PCINT1:Pin Change Interrupt 0, Source 1
PA2	ADC2: ADC Input Channel 2 AIN1: Analog Comparator, Negative 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.

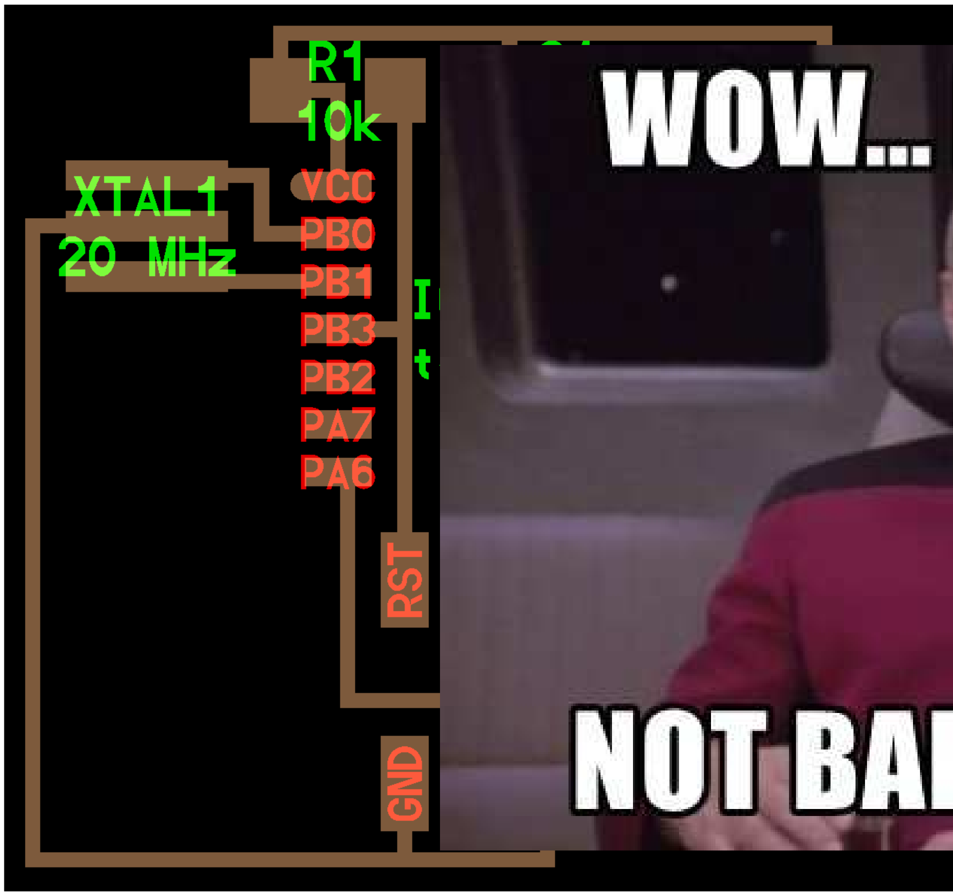
That wasn't so scary!

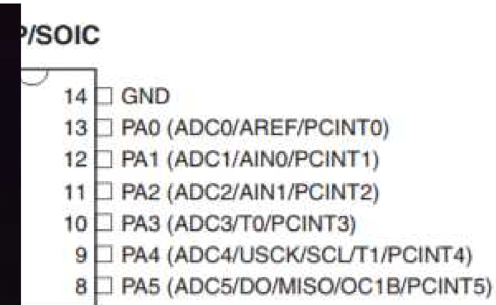




Oh hey look at Neil’s hello world board – it looks like the programming 6 pin header has all of it’s named things connected to the ports on the Attiny with those names!

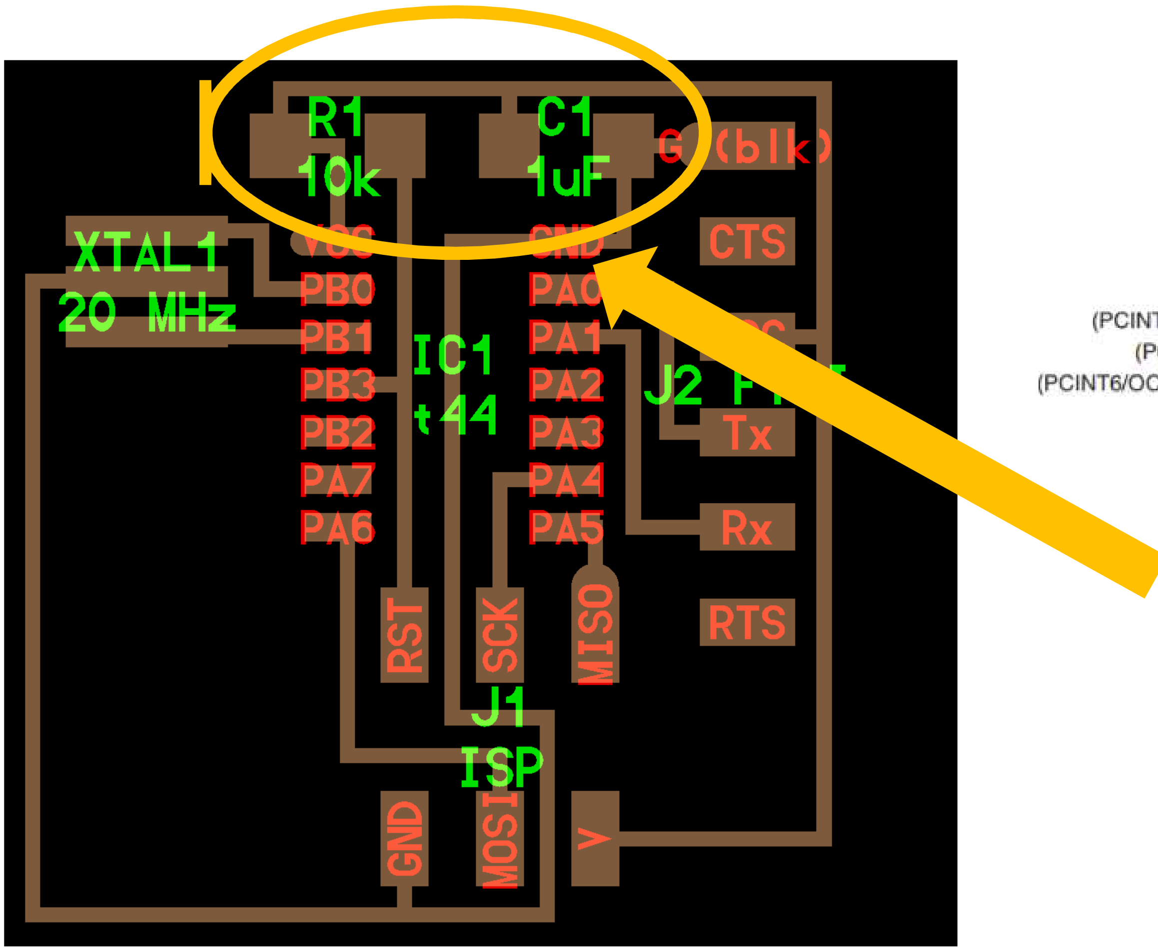
Oh and the clock too (XTAL)!

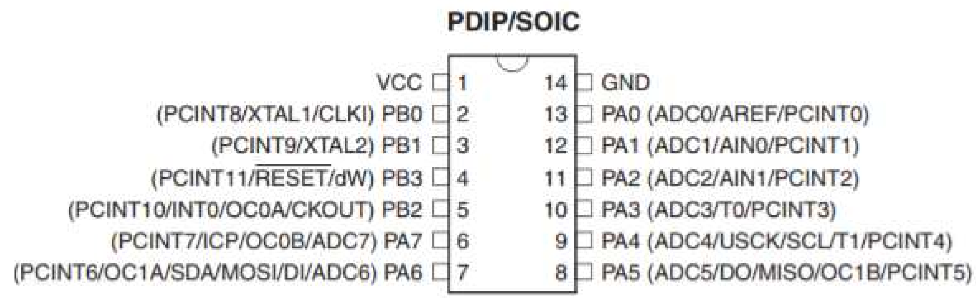




Neil's hello world
ke the programming
s all of it's named
to the ports on the
those names!

ock too (XTAL)!





Yah but what about this stuff? What is it doing there? And what if I want to add things like buttons and LEDs? How do I even think about that?

Lets talk some basic electric engineering…

$$V = I * R$$

Voltage
I: Current
Resistance

Lets talk some basic electric engineering…

$$V = I * R$$

Voltage
I: Current
Resistance

A **L**ight **E**mitting **D**iode has nearly 0 resistance so if we connect it directly from power (+5V) to GND (0V) then it will have nearly infinite current = BOOM!

Lets talk some basic electric engineering…

$$V = I * R$$

Voltage
I: Current
Resistance

A **L**ight **E**mitting **D**iode has nearly 0 resistance so if we connect it directly from power (+5V) to GND (0V) then it will have nearly infinite current = BOOM!

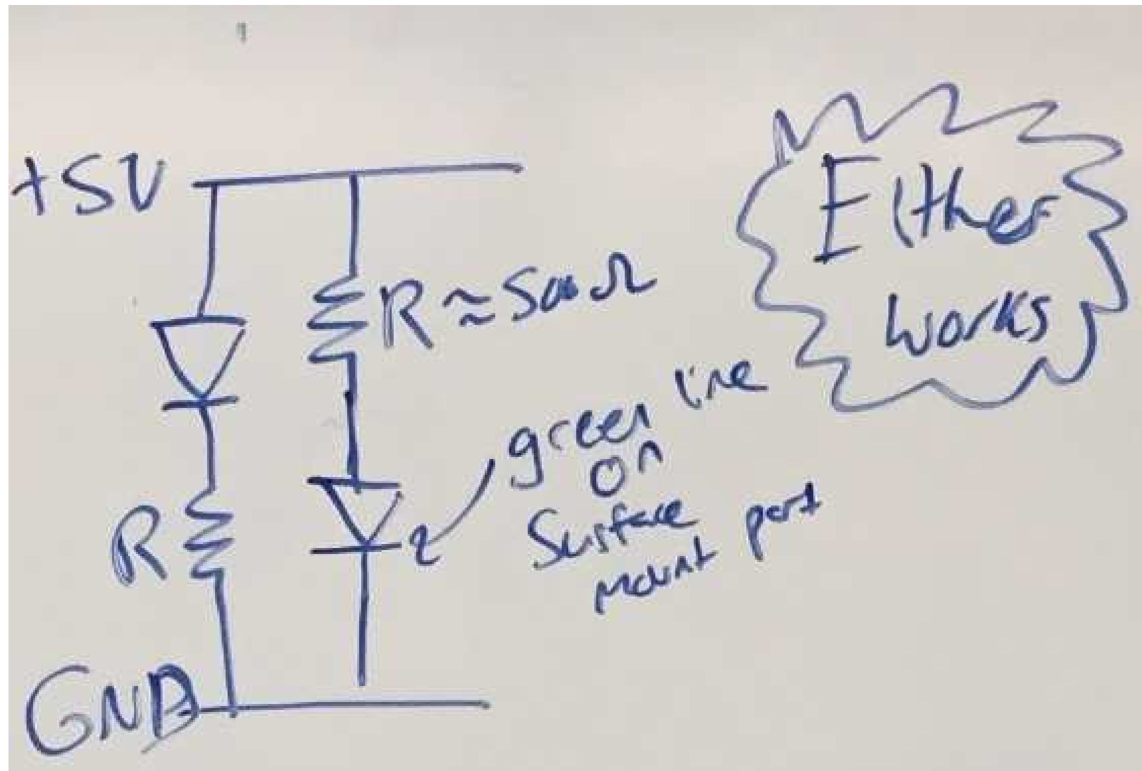
(well really it will just fail and melt but BOOM sounds cooler)

Lets talk some basic electric engineering…

$$V = I * R$$

Voltage
I: Current
Resistance

So we use a ***current limiting resistor*** in series with the LED throughout our designs

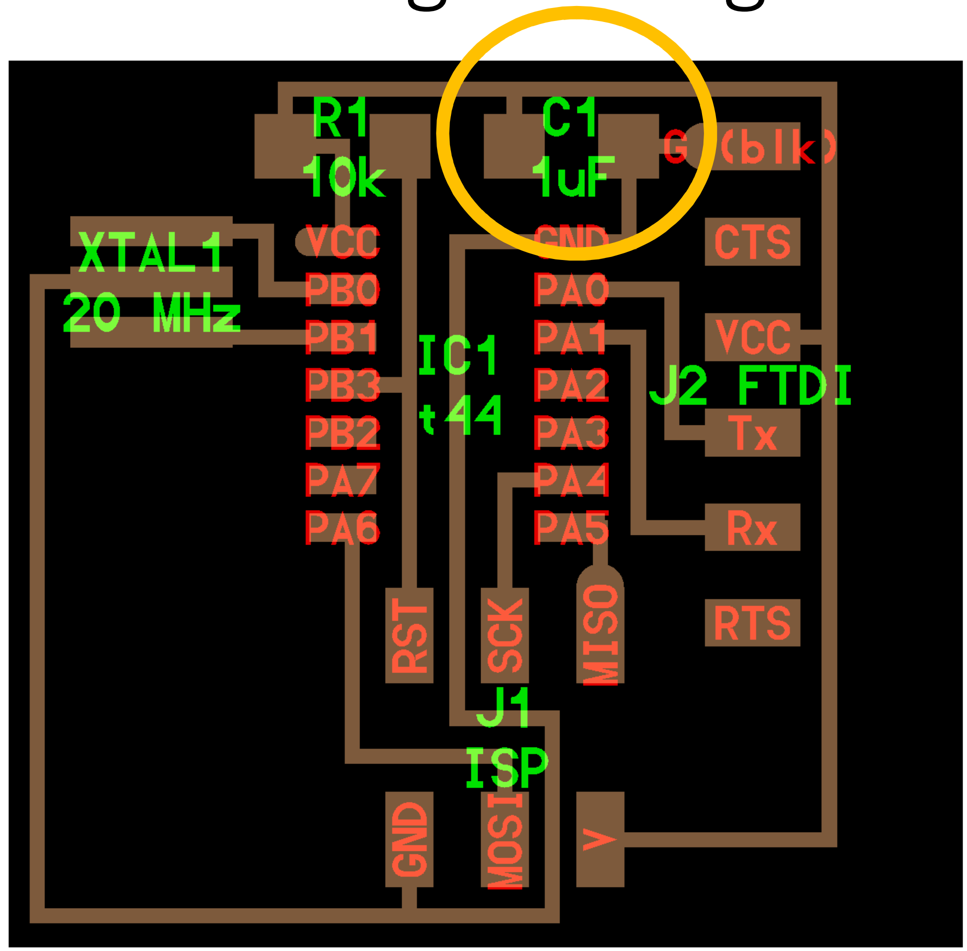


Lets talk some basic electric engineering…

$$I = C * \frac{dv}{dt}$$

I: Current
Capacitance
dv/dt : change in voltage

What is the capacitor doing sitting between GND (0V) and VCC (5V) on Neil's board?

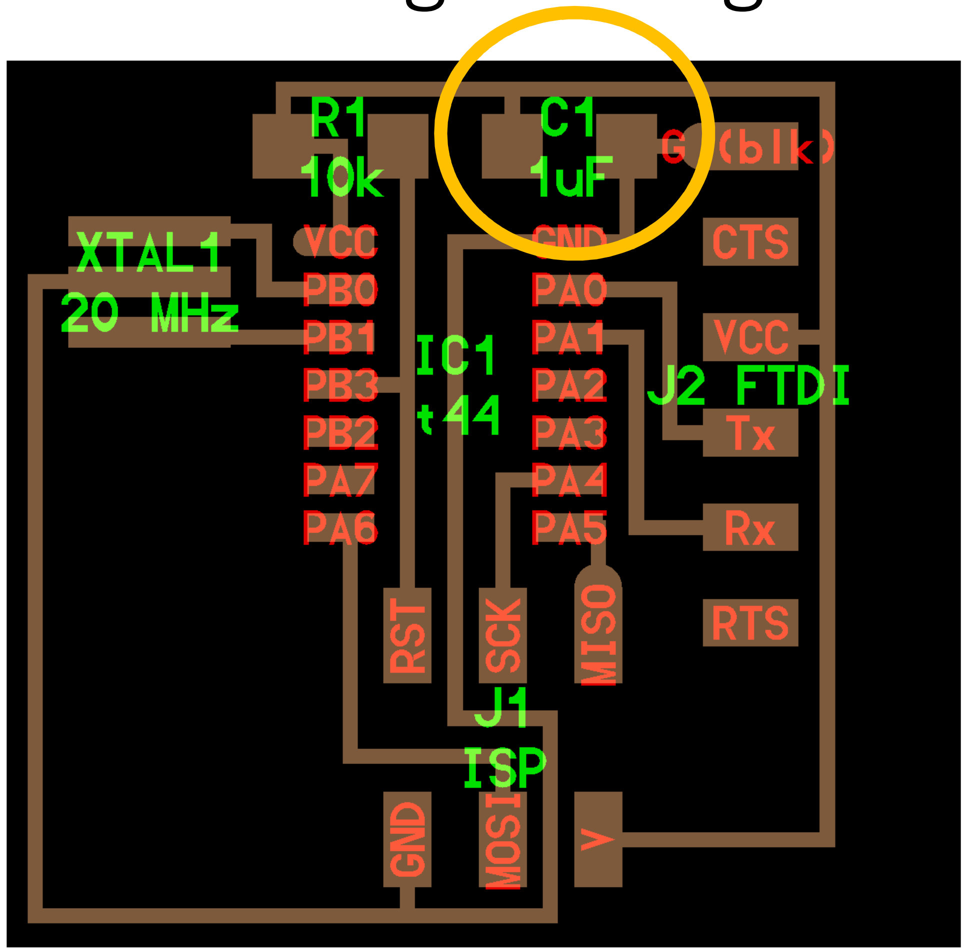


Lets talk some basic electric engineering...

$$I = C * \frac{dv}{dt}$$

I: Current
Capacitance
dv/dt : change in voltage

Think of it as a tiny **backup battery** for changing spikes down in voltage (it ***filters*** them out)!

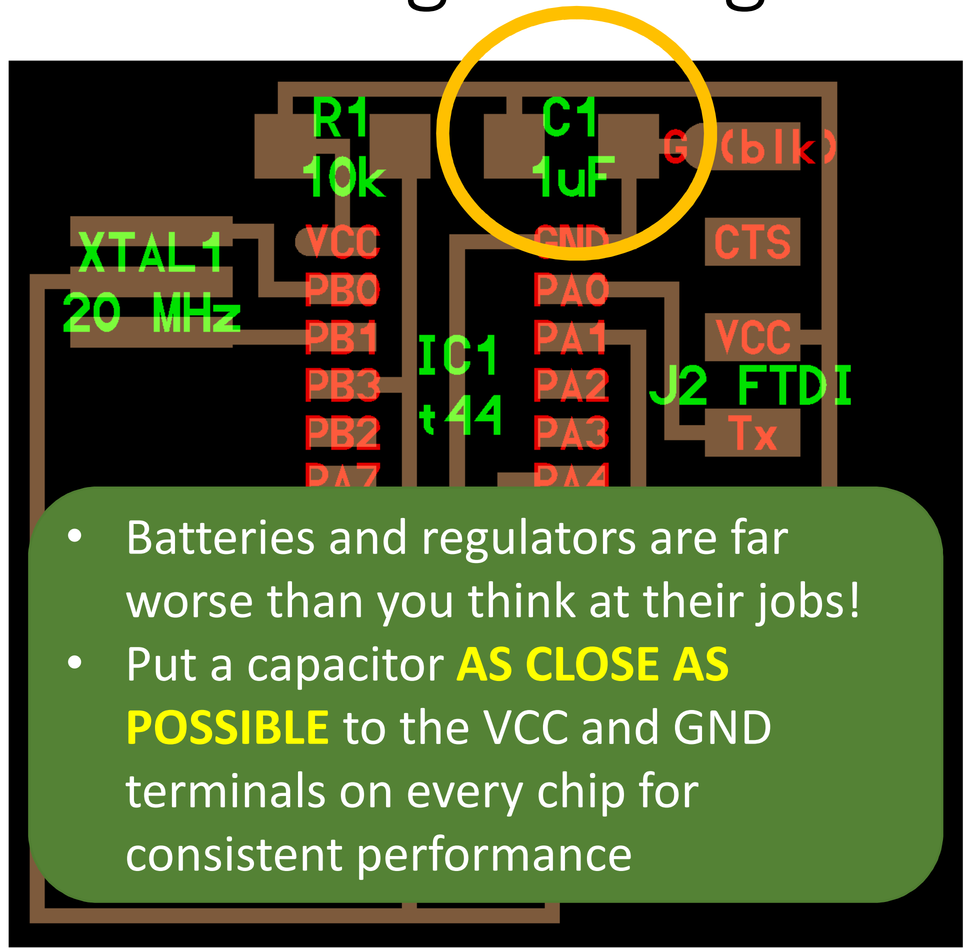


Lets talk some basic electric engineering…

$$I = C * \frac{dv}{dt}$$

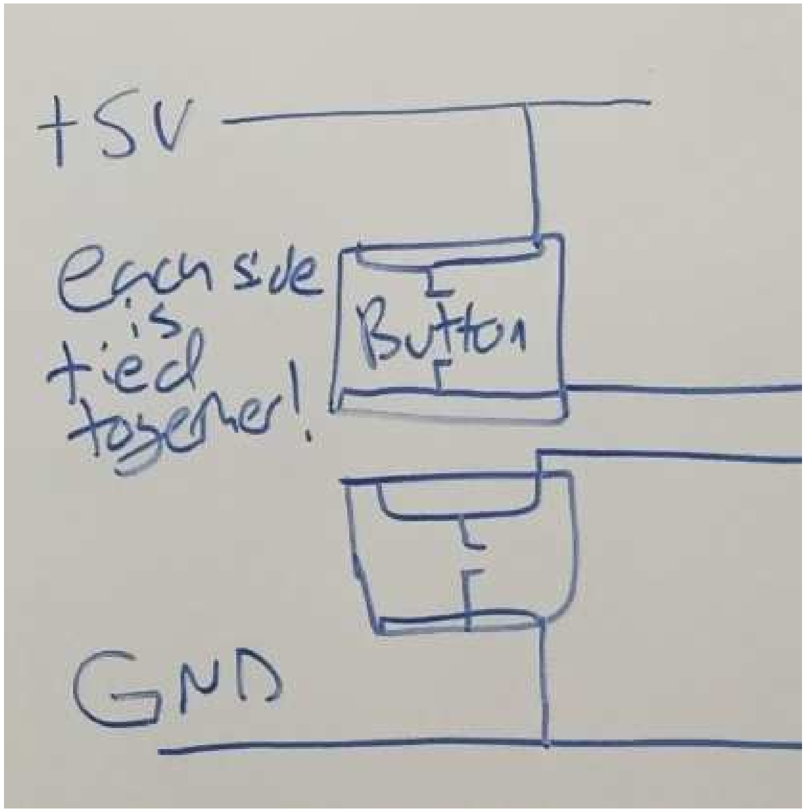
I: Current
Capacitance
dv/dt : change in voltage

Think of it as a tiny **backup battery** for changing spikes down in voltage (it ***filters*** them out)!



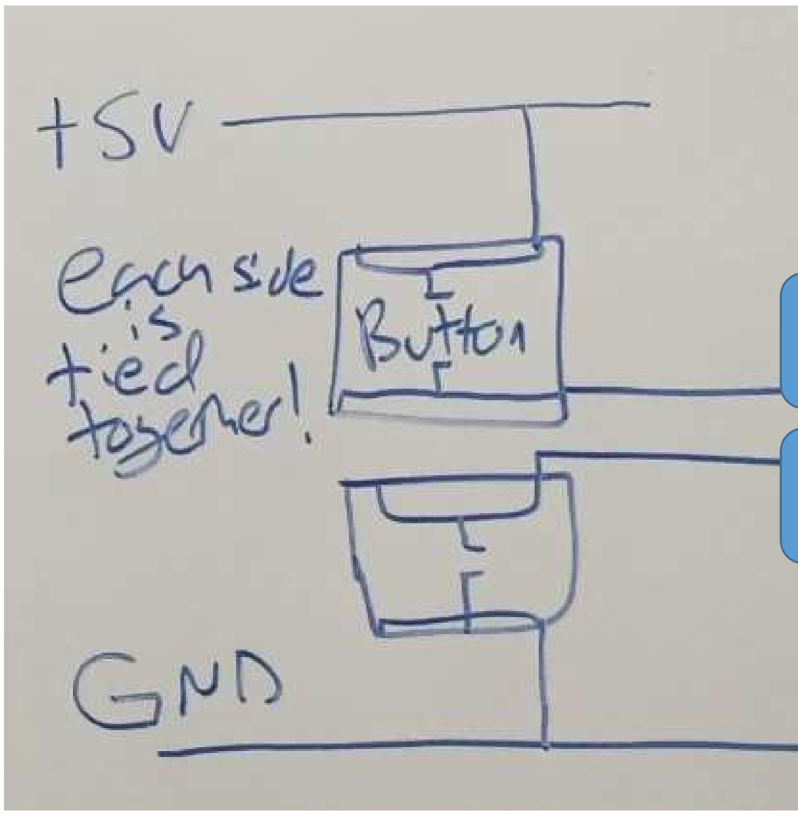
- Batteries and regulators are far worse than you think at their jobs!
- Put a capacitor **AS CLOSE AS POSSIBLE** to the VCC and GND terminals on every chip for consistent performance

Lets talk some basic electric engineering…



What about a button?

Lets talk some basic electric engineering…

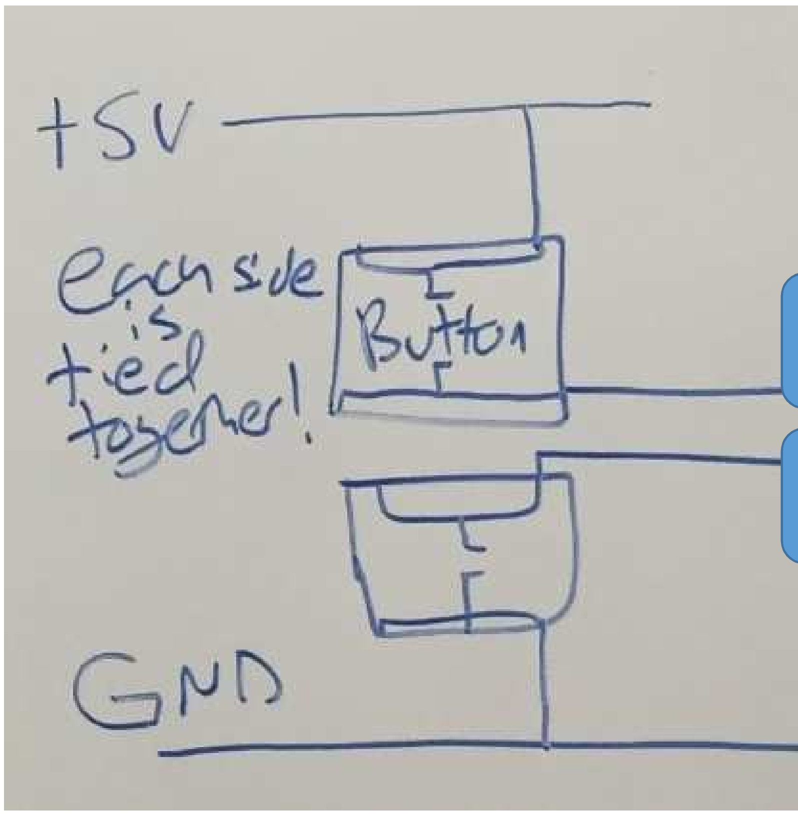


PAX

PAY

Well we could connect the button to +5V or GND (0V) and then into the Attiny – how do we pick?

Lets talk some basic electric engineering…

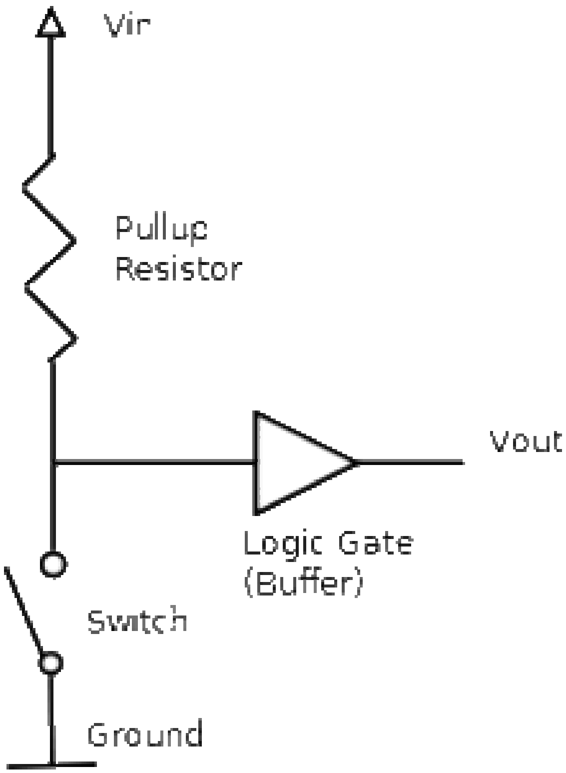


PAX

PAY

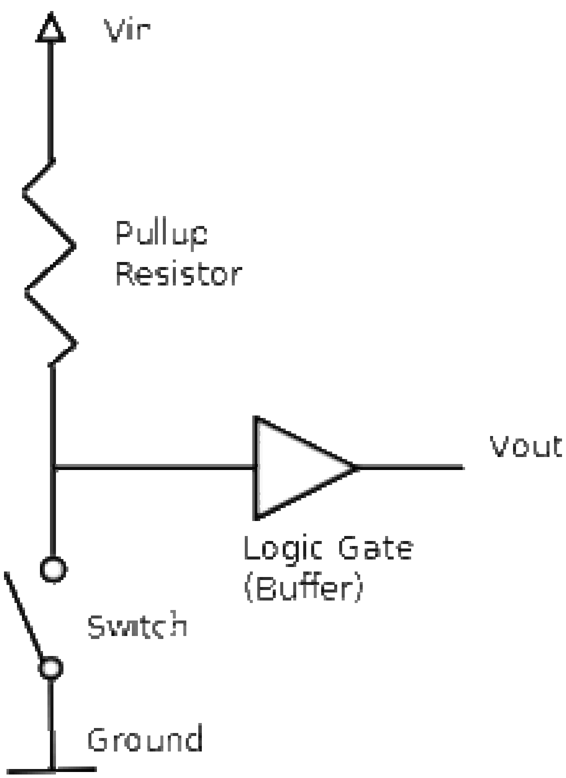
It turns out connecting to GND is better (more power efficient) BUT….

Lets talk some basic electric engineering…



You need a **pullup resistor** – the good news is that this pattern is so common there is a built-in pullup in the ATtiny you can turn on via software so no need to put it in your design!

Lets talk some basic electric engineering…



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

We got smarter! We know the basics now!

Ok so now how do we go about actually doing this...

Ok so now how do we go about actually doing this...

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

Tips for designing and routing boards:

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

Tips for designing and routing boards:

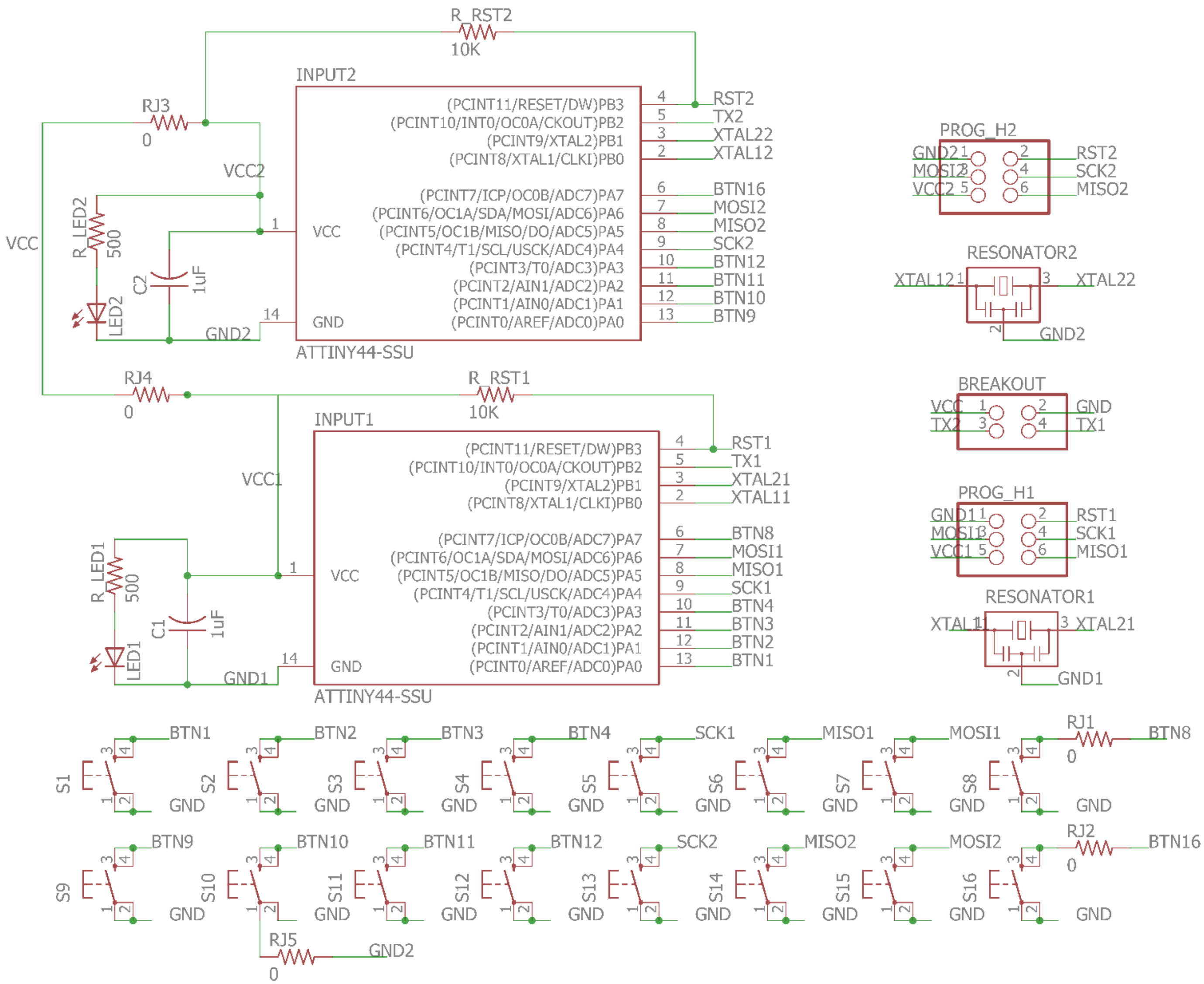
1. Do the schematic first (and finish it before moving on to routing)
2. Use lots of names to keep the schematic clean and readable

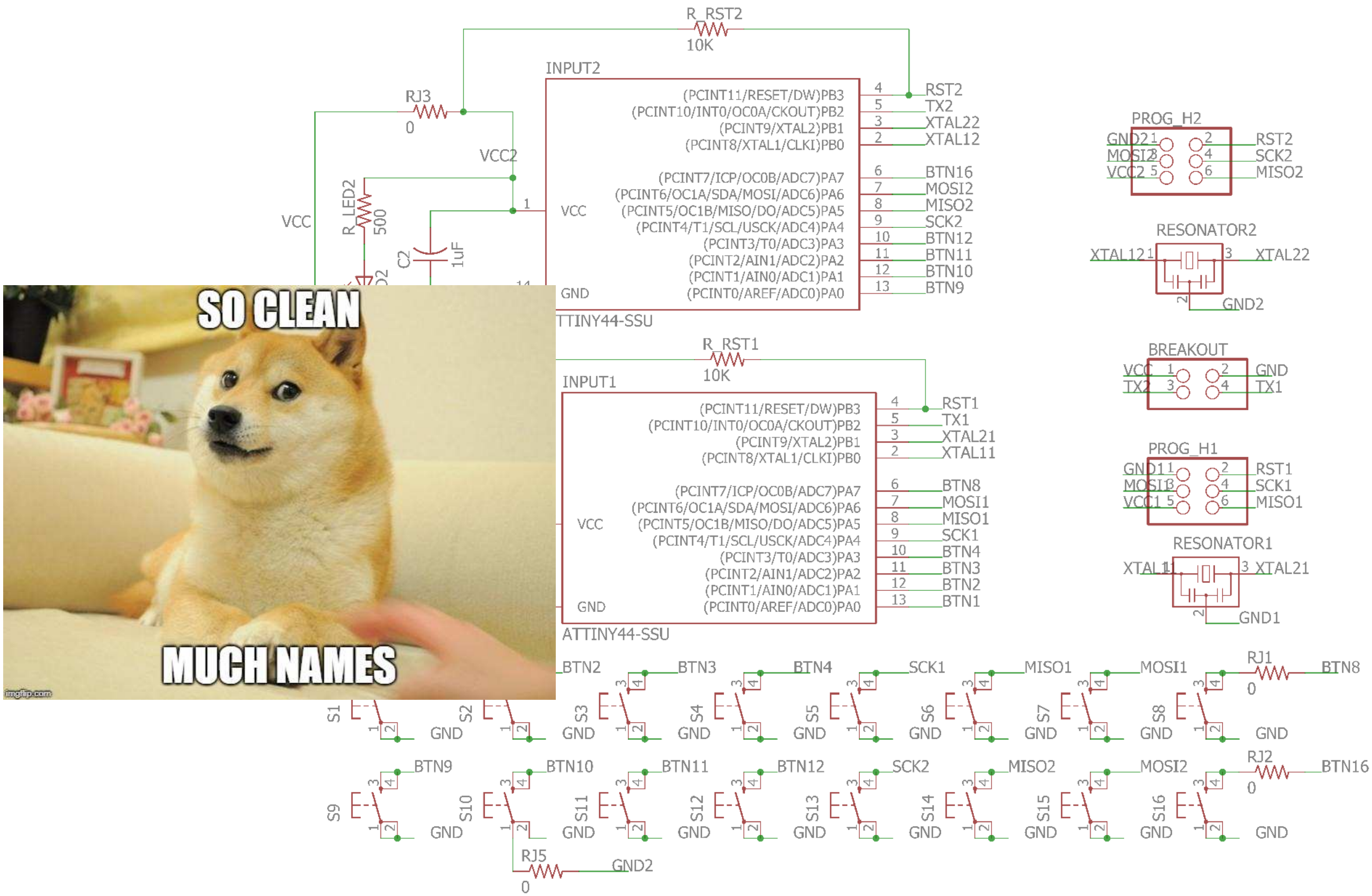
Tips for designing and routing boards:

1. Do the schematic first (and finish it before moving on to routing)
2. Use lots of names to keep the schematic clean and readable
3. Triple check the schematic before moving onto the board file (and have someone else check it if you are unsure)

Tips for designing and routing boards:

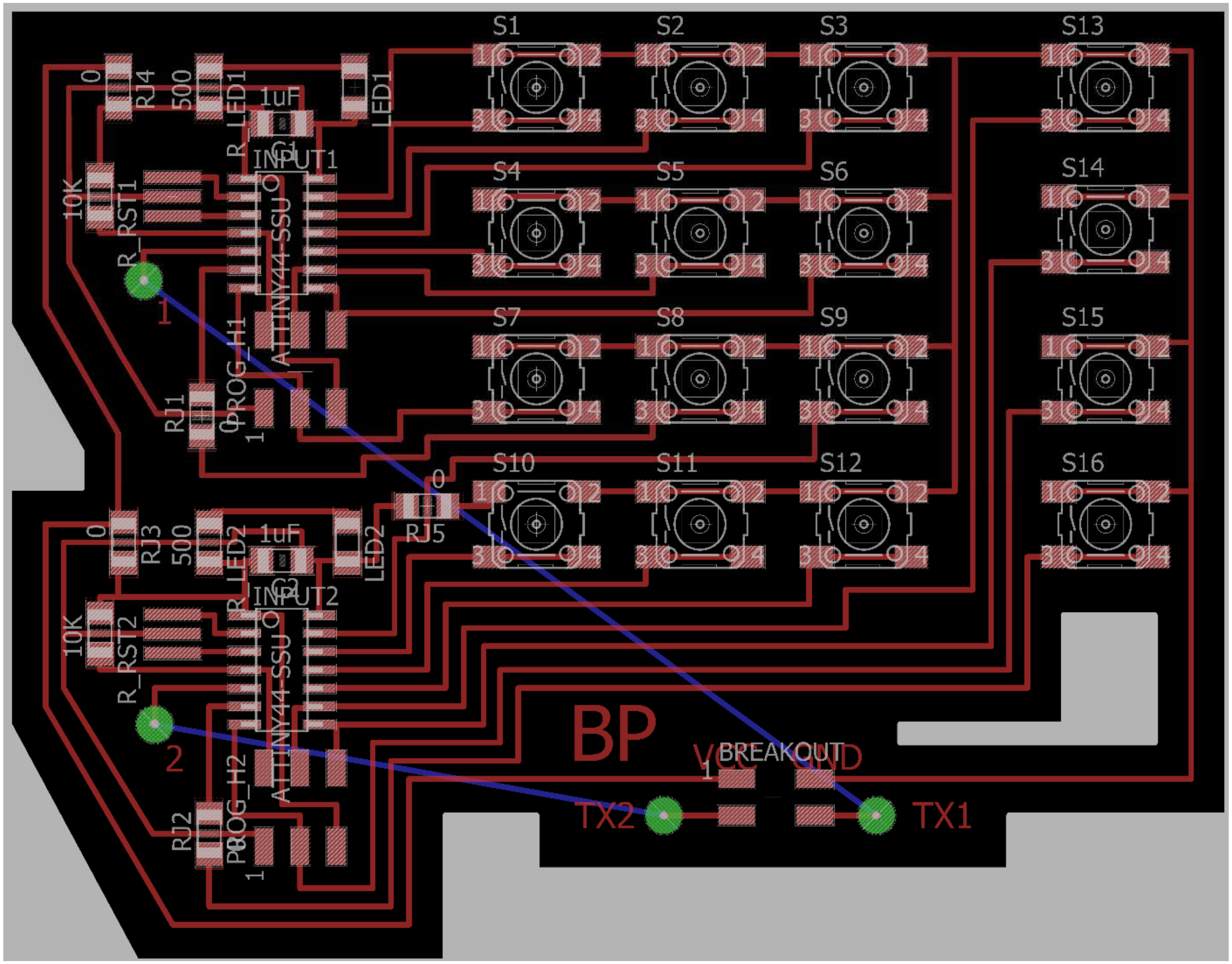
1. Do the schematic first (and finish it before moving on to routing)
2. Use lots of names to keep the schematic clean and readable
3. Triple check the schematic before moving onto the board file (and have someone else check it if you are unsure)
4. Add an LED from power to ground as a first sanity check when you are done with the board that it is “working”


R_RST2
10K
INPUT2
RJ3
0
VCC2
VCC
R_LED2
500
C2
1uF
LED2
GND2
ATTINY44-SSU
(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
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
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

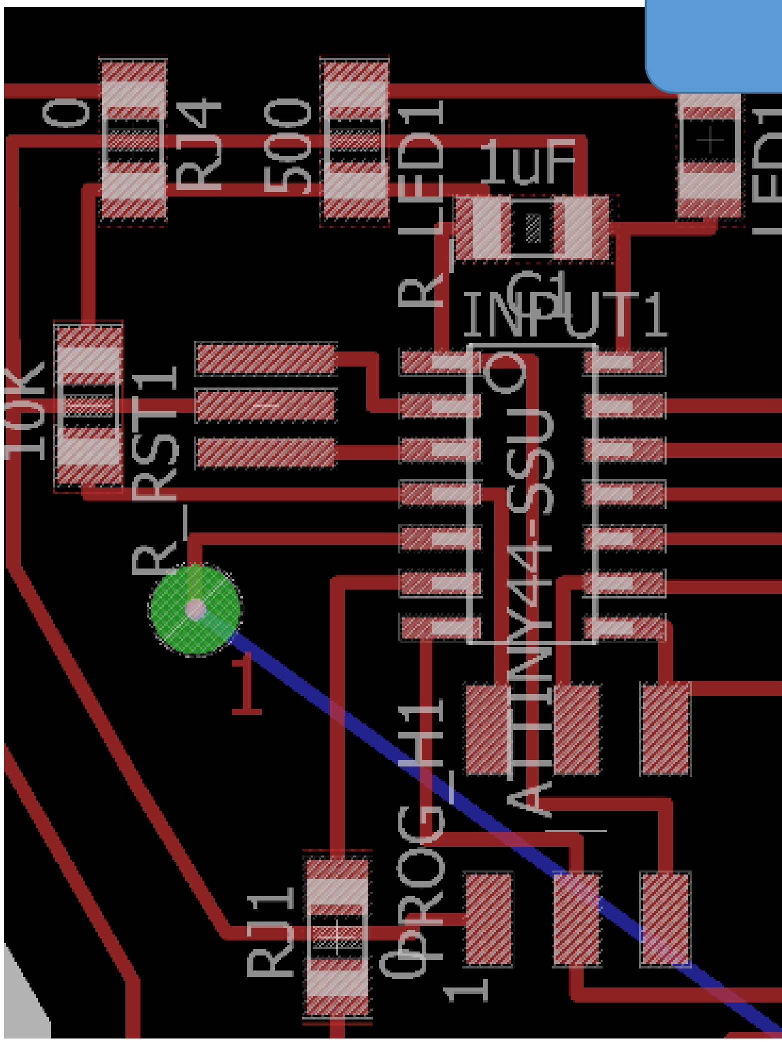

R_RST2
10K
INPUT2
RJ3
0
VCC2
VCC
R_LED2
500
C2
1uF
VCC
GND
(PCINT11/RESET/DW)PB3
(PCINT10/INT0/OC0A/CKOUT)PB2
(PCINT9/XTAL2)PB1
(PCINT8/XTAL1/CLKI)PB0
(PCINT7/ICP/OC0B/ADC7)PA7
(PCINT6/OC1A/SDA/MOSI/ADC6)PA6
(PCINT5/OC1B/MISO/DO/ADC5)PA5
(PCINT4/T1/SCL/USCK/ADC4)PA4
(PCINT3/T0/ADC3)PA3
(PCINT2/AIN1/ADC2)PA2
(PCINT1/AIN0/ADC1)PA1
(PCINT0/AREF/ADC0)PA0
RST2
TX2
XTAL22
XTAL12
BTN16
MOSI2
MISO2
SCK2
BTN12
BTN11
BTN10
BTN9
TTINY44-SSU
PROG_H2
GND2
MOSI2
VCC2
RST2
SCK2
MISO2
RESONATOR2
XTAL12
XTAL22
GND2
R_RST1
10K
INPUT1
RST1
TX1
XTAL21
XTAL11
BTN8
MOSI1
MISO1
SCK1
BTN4
BTN3
BTN2
BTN1
ATTINY44-SSU
BREAKOUT
VCC
GND
TX2
TX1
PROG_H1
GND1
MOSI1
VCC1
RST1
SCK1
MISO1
RESONATOR1
XTAL11
XTAL21
GND1
BTN2
BTN3
BTN4
SCK1
MISO1
MOSI1
RJ1
0
BTN8
S1
S2
S3
S4
S5
S6
S7
S8
GND
BTN9
BTN10
BTN11
BTN12
SCK2
MISO2
MOSI2
RJ2
0
BTN16
S9
S10
S11
S12
S13
S14
S15
S16
GND
RJ5
0
GND2
SO CLEAN
MUCH NAMES
imgflip.com

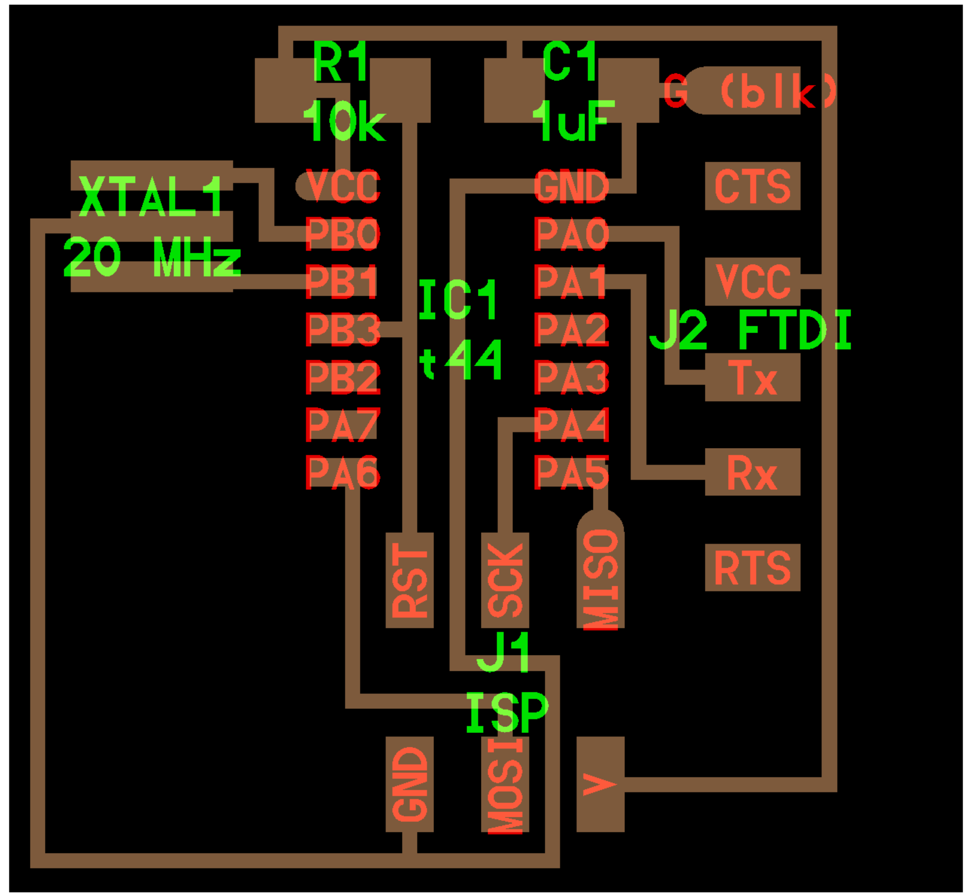
Tips for designing and routing boards:

1. Do the schematic first (and finish it before moving on to routing)
2. Use lots of names to keep the schematic clean and readable
3. Triple check the schematic before moving onto the board file (and have someone else check it if you are unsure)
4. Add an LED from power to ground as a first sanity check when you are done with the board that it is “working”
5. When routing the board file copy the kinds of routing patterns Neil or other people have used


S1
S2
S3
S13
S4
S5
S6
S14
S7
S8
S9
S15
S10
S11
S12
S16
0
RJ4
500
R_LED1
1uF
C1
LED1
INPUT1
10K
R_RST1
ATTINY44-SSU
PROG_H1
RJ1
1
0
RJ3
500
R_LED2
1uF
C2
LED2
RJ5
INPUT2
10K
R_RST2
ATTINY44-SSU
PROG_H2
RJ2
2
BP
BREAKOUT
VCC
GND
TX2
TX1

Not so different after all…

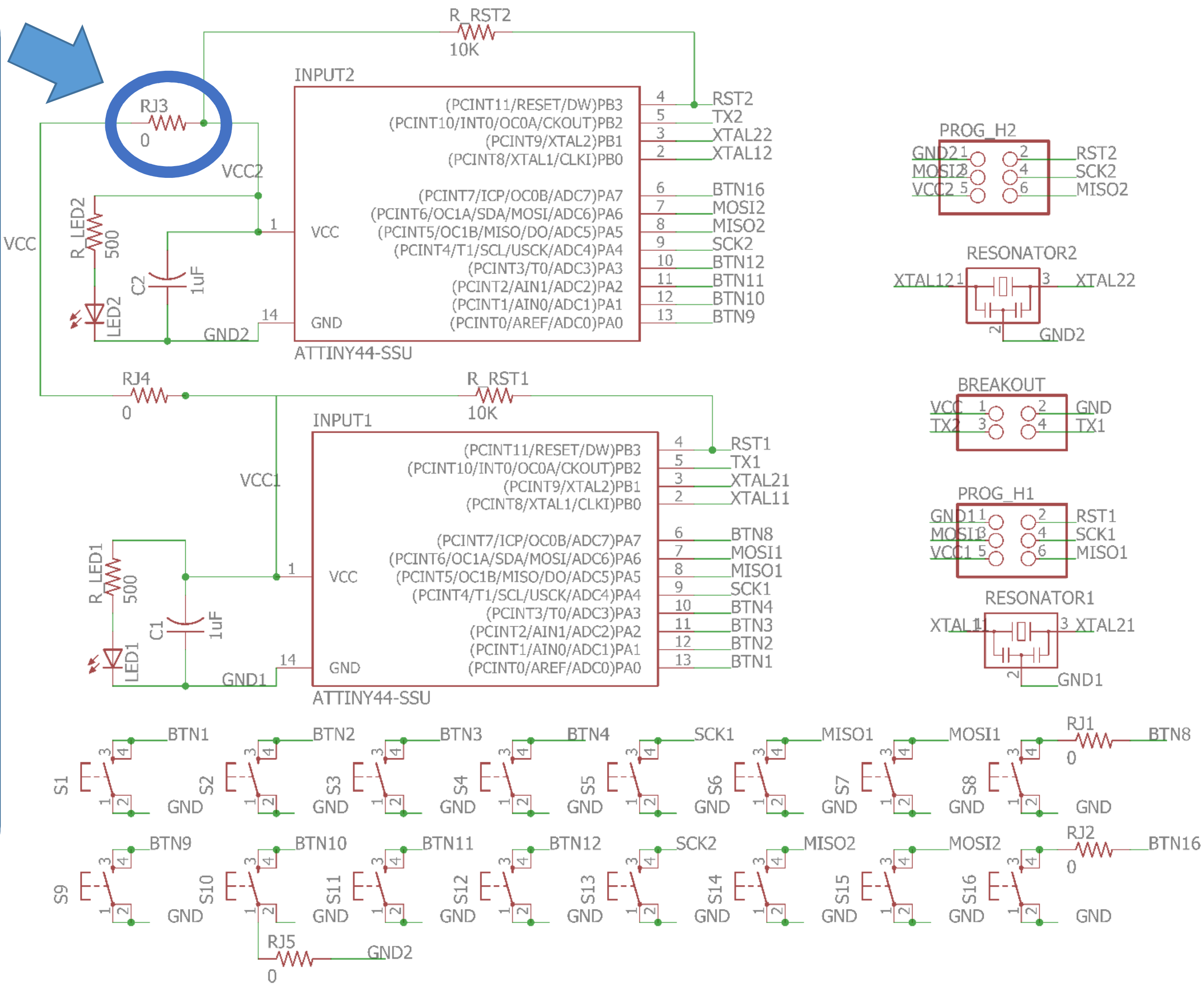


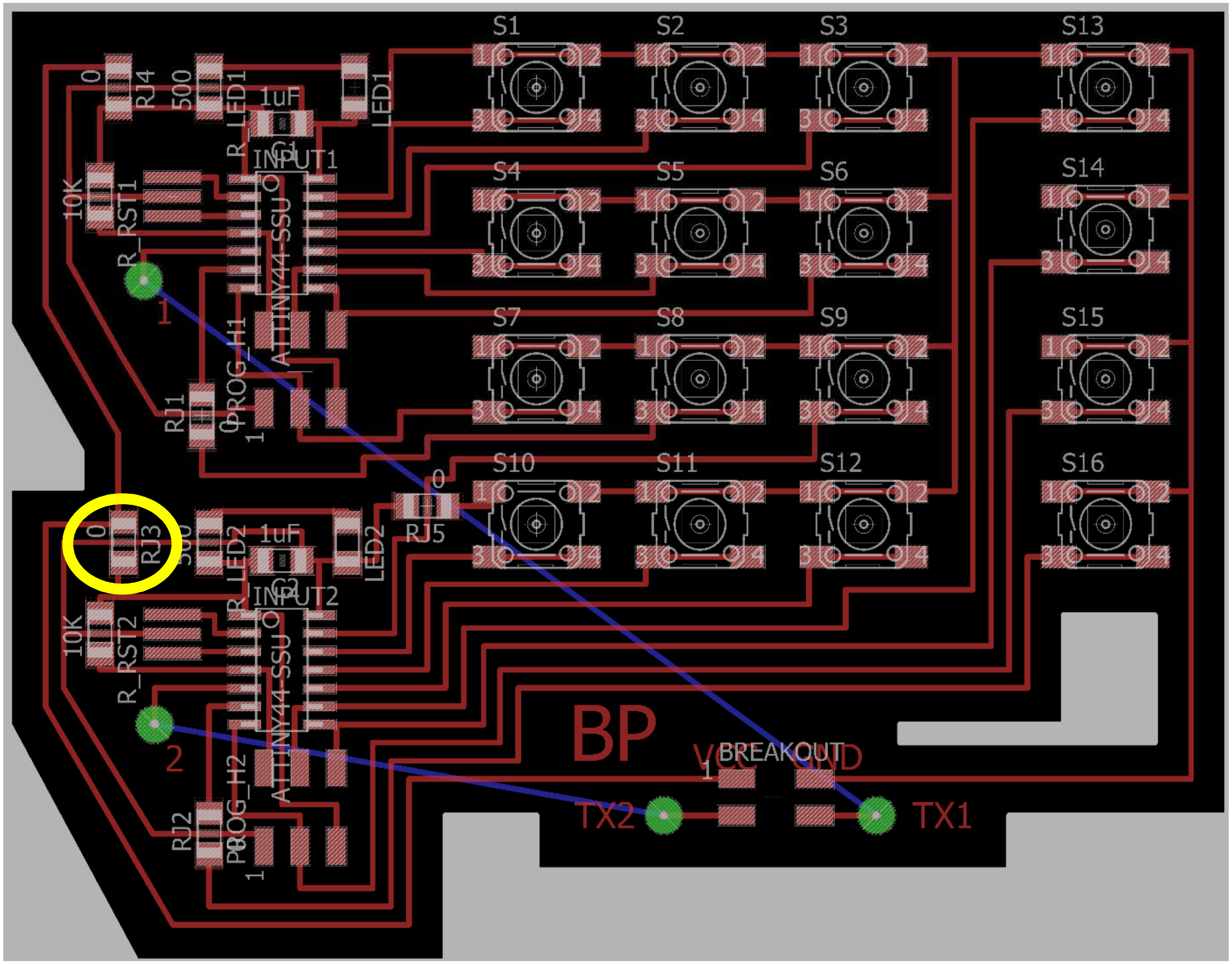


Tips for designing and routing boards:

1. Do the schematic first (and finish it before moving on to routing)
2. Use lots of names to keep the schematic clean and readable
3. Triple check the schematic before moving onto the board file (and have someone else check it if you are unsure)
4. Add an LED from power to ground as a first sanity check when you are done with the board that it is “working”
5. When routing the board file copy the kinds of routing patterns Neil or other people have used
6. If you get super stuck routing add 0 ohm resistors!

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



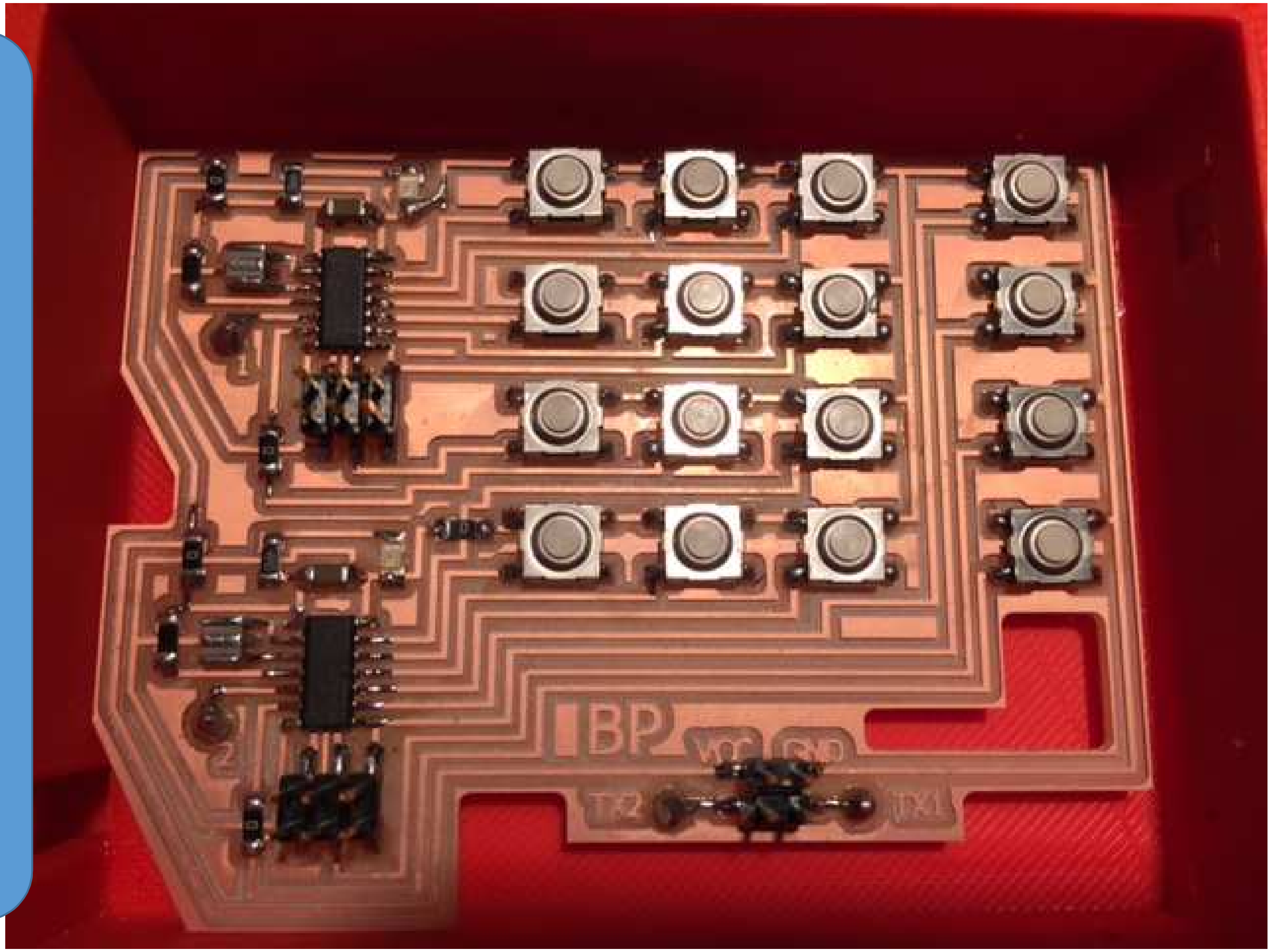


S1
S2
S3
S13
S4
S5
S6
S14
S7
S8
S9
S15
S10
S11
S12
S16
RJ4
RJ1
RJ2
RJ3
RJ5
LED1
LED2
R_LED1
R_LED2
R_RST1
R_RST2
1uF
C1
C2
INPUT1
INPUT2
ATTINY44-SSU
PROG_H1
PROG_H2
10K
500
BP
BREAKOUT
VCC
GND
TX1
TX2

Tips for designing and routing boards:

1. Do the schematic first (and finish it before moving on to routing)
2. Use lots of names to keep the schematic clean and readable
3. Triple check the schematic before moving onto the board file (and have someone else check it if you are unsure)
4. Add an LED from power to ground as a first sanity check when you are done with the board that it is “working”
5. When routing the board file copy the kinds of routing patterns Neil or other people have used
6. If you get super stuck routing add 0 ohm resistors!
7. I promise it gets way way easier after you do this a couple times.

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



Eagle Demo

http://fab.academany.org/2018/labs/fablaboshanghai/students/bob-wu/Fabclass/week7_electronic_design/eagle_practice.html