Lecture 10

RC Servos / Inductive Spikes

Agenda

00. Important Lab 2.3 and 2.4 clarity, Arduino updates.
01. RC Servos
02. PWM Generation on Teensy
03. Power Requirements
04. Inductive Loads
05. Pointers and Arrays in C (if there is time)

Stuff

- Towne Building access is closed from 10PM to 7:30AM, everyday
- If you are sick:
  - You can work from home. Use the Oscillosorta.
  - Lab Check off by Zoom – setup a time with a TA to zoom.
- Tracking replacement ESP32 - $10 and Teensy - $20 to project budgets or you can pay direct.
- Lab Instructions vs Rubric. Always go by instructions. Rubrics are subject to change.

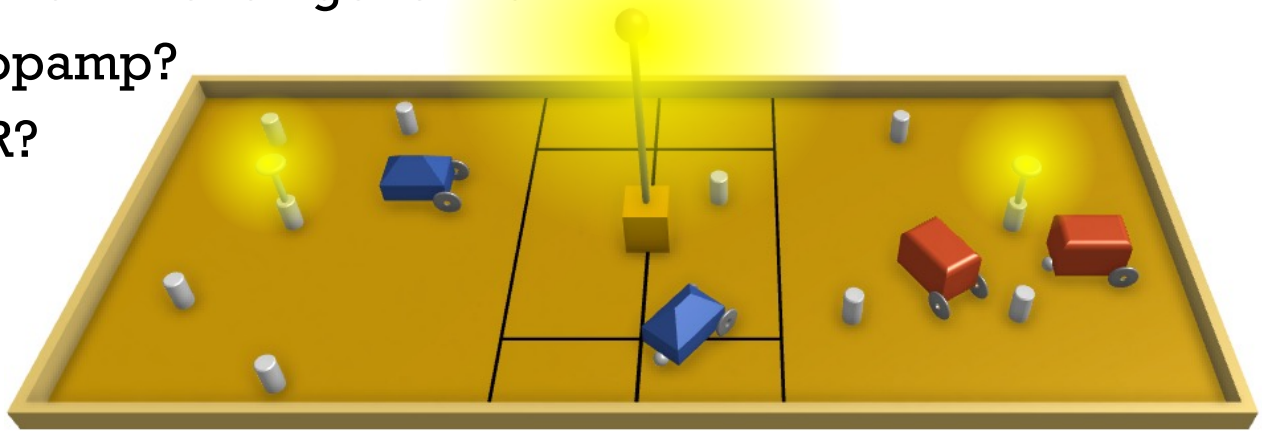
Planning for rest of semester

- Lab 2 is probably most difficult lab in terms of new material.
- Lab 3 and 4 will start to emphasize integration and coding.
- Final project will use everything from previous labs – but integrated.

Sensing with IR phototransistor

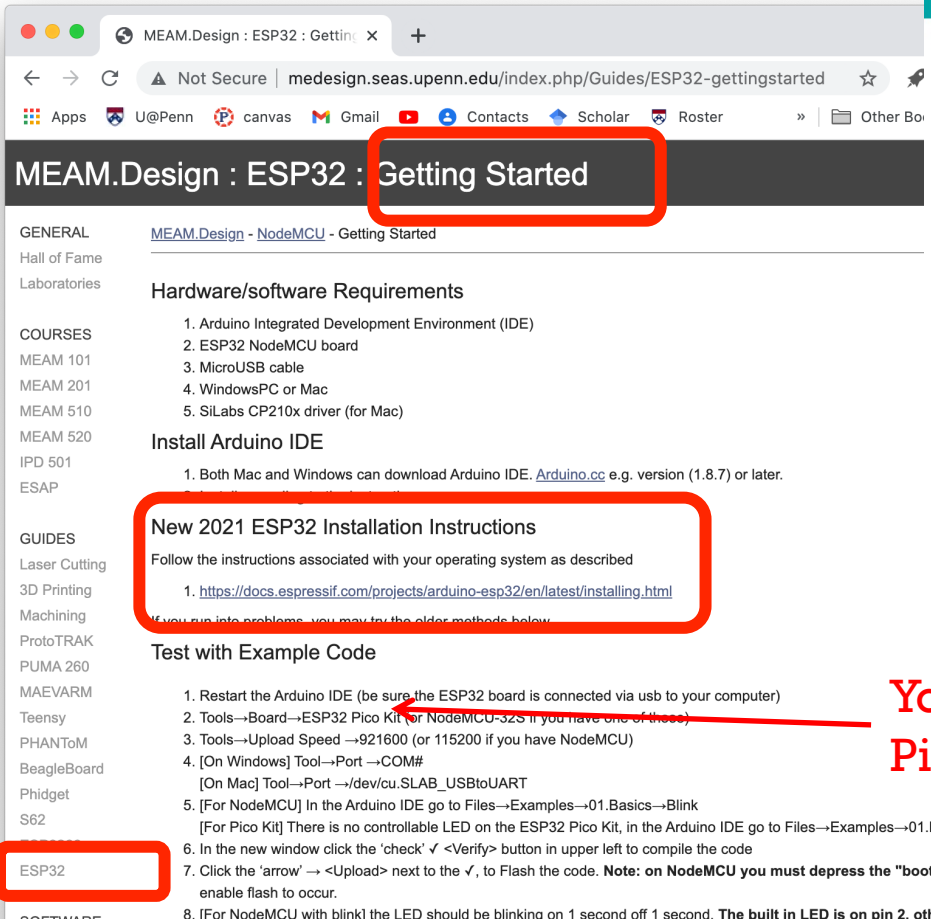
- Why use ESP32 instead of function generator?
- Why are we using an opamp?
- Why are we sensing IR?

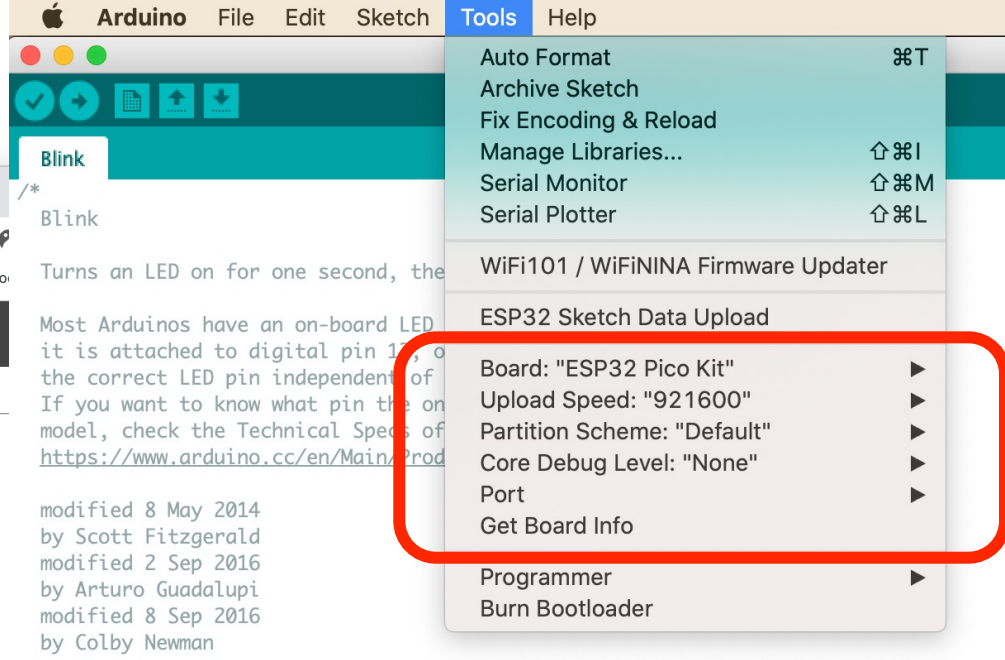
The final project will have IR beacons



- Hints for completing 2.4 - you need to distinguish only 3 states: 700Hz signal, 23Hz signal, no signal – ideally display via Teensy.
  - If you sense some frequency and it's over 150Hz assume is 700Hz.
- Use AC coupling to remove DC, then amplify alot.
- Due to oscillosorta issues and limited phototransistors, we will add 3 late days to every student (effectively due date is Friday).

Arduino Set-up





You have the Pico Kit

Espressif recently released V2.0 which breaks some things

http://medesign.seas.upenn.edu/index.php/Guides/ESP32-gettingstarted

Loading OscilloSorta V1.2 to ESP32

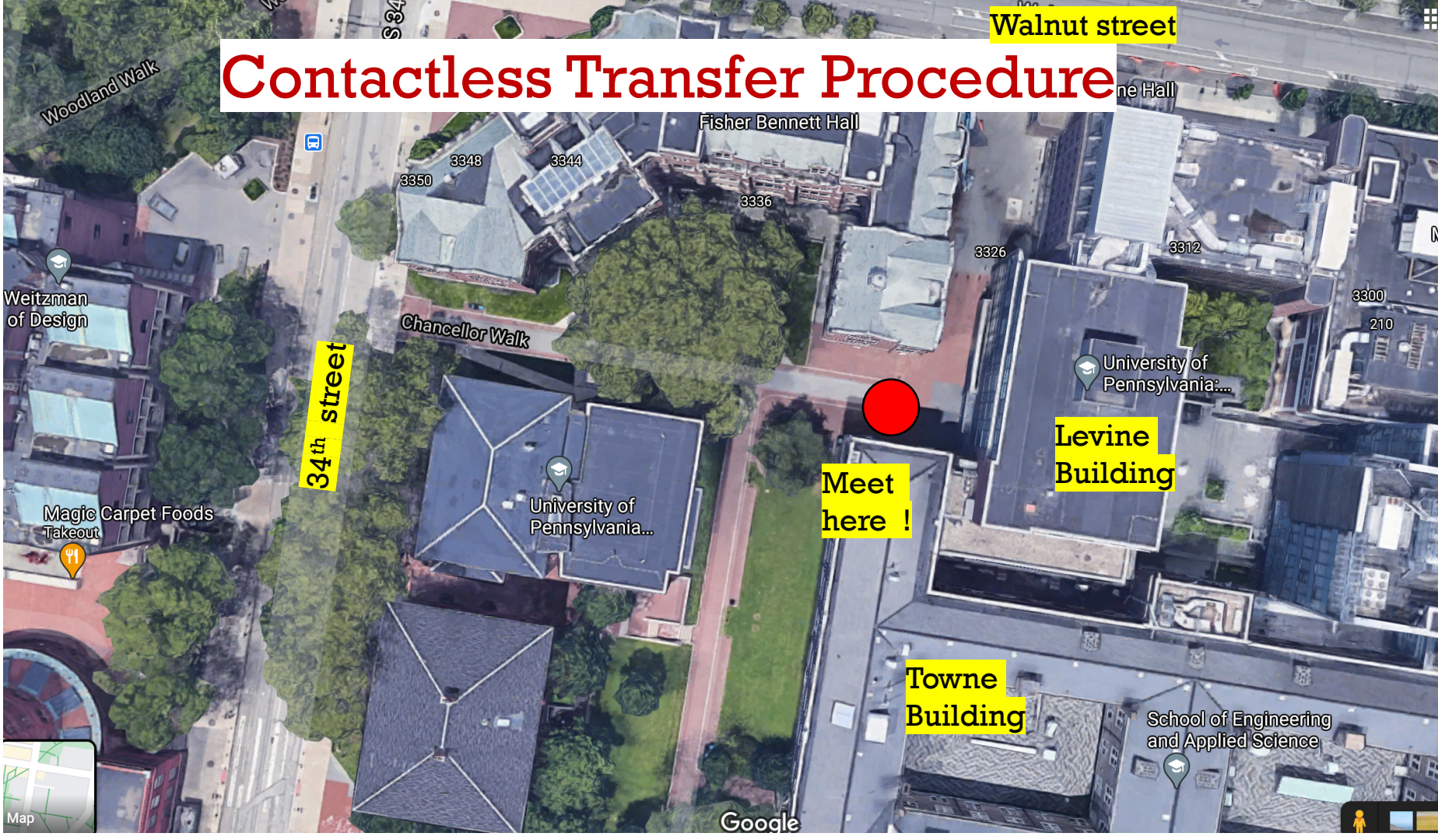
- You may need to install ESP32 Arduino V1.0X not V2.0.
- Find:
  - Oscillosorta1_2 folder in canvas -> files -> resources
- Copy whole folder (9 files inside) to your Arduino directory
- Double click *Oscillosorta1_2.ino*
- Edit the line near the top

```
String ssid = "your_SSID_NAME";
```

- Connect the micro-USB cable to the ESP32
- Click (➔) <Upload> to Flash code to Arduino.

Contactless Transfer Protocol

1. Setup time with TA to transfer (email or piazza, we will post to the leadership slack if needed.)
2. Call TA when you arrive.
3. Wave hello to each other from far away.
4. Item should be dropped on ground.
5. Deliverer walks away.
6. Deliveree picks up item.
7. Yell thank you from far away - through your mask.
8. No-one gets close to the other.


Contactless Transfer Procedure
Walnut street
Woodland Walk
Fisher Bennett Hall
3348
3344
3350
3336
3326
3312
3300
210
Weitzman of Design
Chancellor Walk
34th street
University of Pennsylvania:...
Levine Building
Meet here !
Magic Carpet Foods
Takeout
University of Pennsylvania...
Towne Building
School of Engineering and Applied Science
Map
Google

01

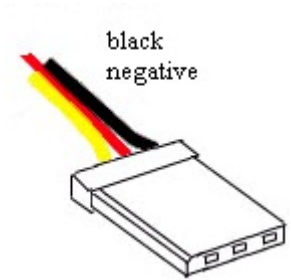
RC Servos

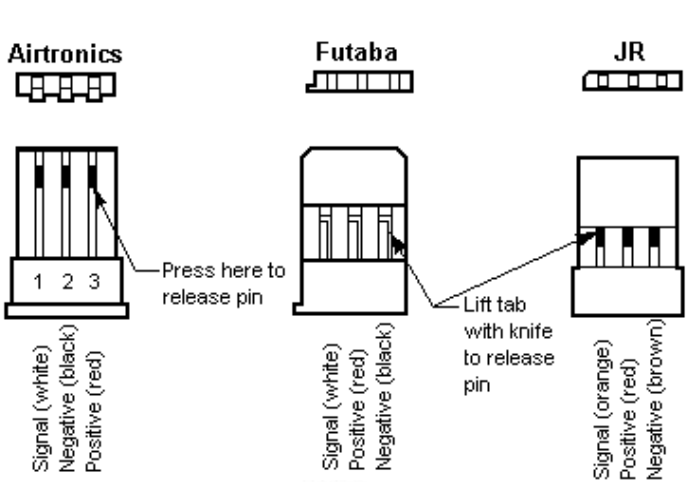
What is a servo?

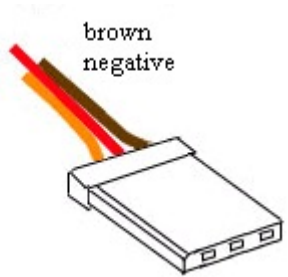
- Motor with a sensor and control feedback.
- Commands to servo control position (usually)
- In the toy or hobbyist community sometimes referred to as RC servo (radio control).

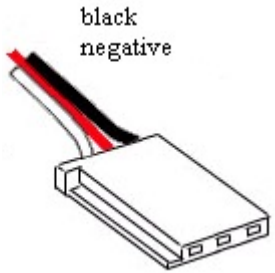
Hobby Servo

- Position control





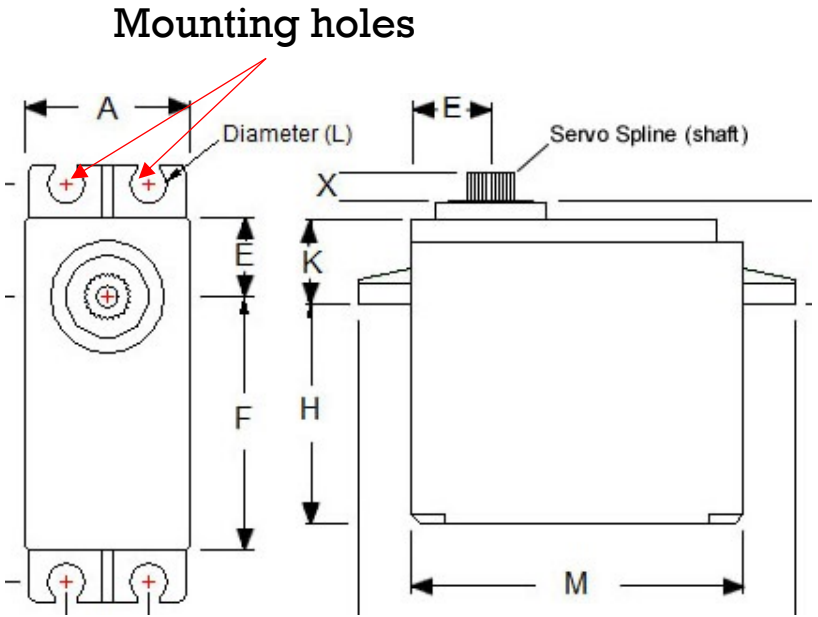




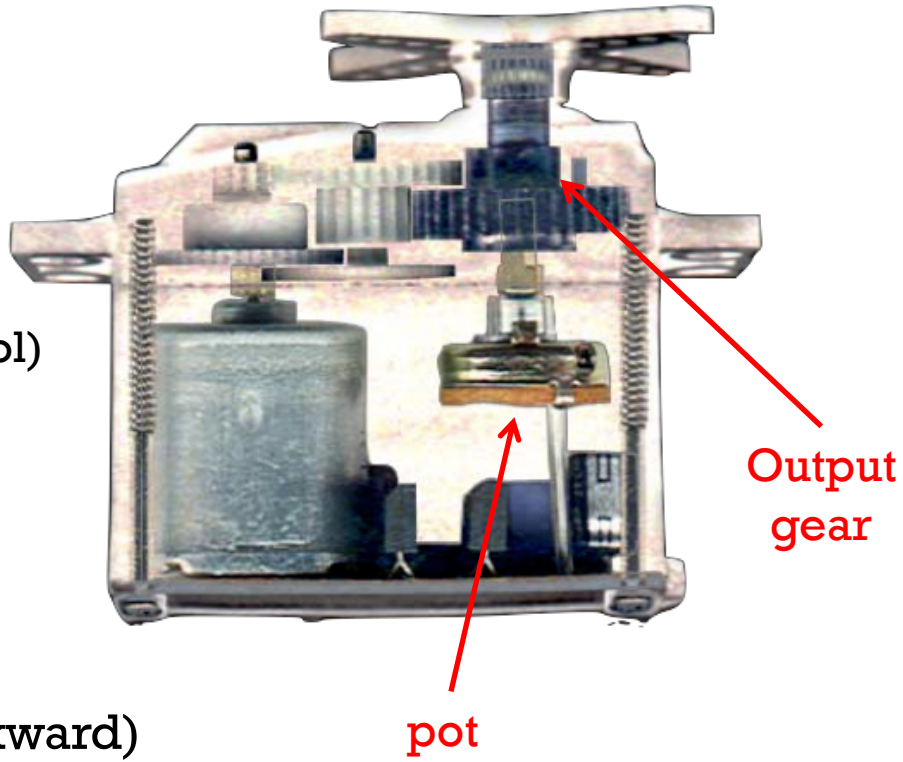
- 3 wire
  - Ground
  - Power (typ 5 or 6 V) [always red]
  - Control Signal (5V pwm @ 50hz)

Specifications

- Most take 4.8V to 6V
- Speed (time to move 60 degrees)
  - e.g. 0.11sec/60
- Torque
  - 1 kg cm to 30 kg cm
- Size
  - 1" side to 6"
- Weight
  - 8gm to 800gm
- Deadband (control uS)
  - 0uS (rare) to 10 uS
- Gearing (metal or plastic)
- Mate with "servo horn" spline interface.



Servo caveats

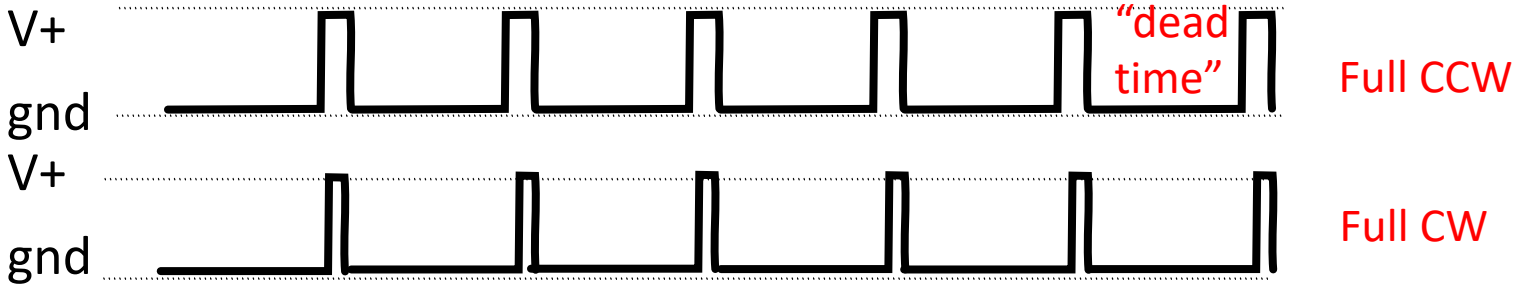


- Control type is fixed
  - Closed loop position control
  - Continuous rotation (speed control)
  - No force control.
- Never enough precision
  - Deadband
- Never light/small enough
- Never enough torque
- Difficult to mount to (spline is awkward)
- Failure modes
  - Output gear failure (with large spike in torque)
  - Motor stall-> coils melting/seizure (when over-torqued for too long)

Duty Cycle For Hobby Motors

- 20ms nominal period (can vary 10-70ms)
  - Typically 20ms = 50Hz (can vary sometimes ~15Hz to ~100Hz)
- For one direction speed contoller
  - 1ms = 5.5% duty cycle, Motor Full off
  - 2ms = 11.1% duty cycle, Motor Full on
- For servos
  - 1ms = 5.5% duty cycle, Full clockwise
  - 2ms = 11.1% duty cycle, Full counter clockwise

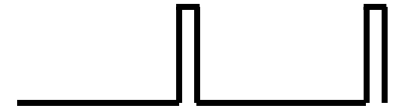
Used to multiplex multiple servos



Duty Cycle For Hobby Motors

- Actual range of motion:
  - min < ~1ms and max > ~2ms
- Range of motion examples:
  - +- 90 degrees
  - +- 135 degrees
  - Multiple turn +/630 degrees
- Note: servo horn can be attached at random angles

2mS

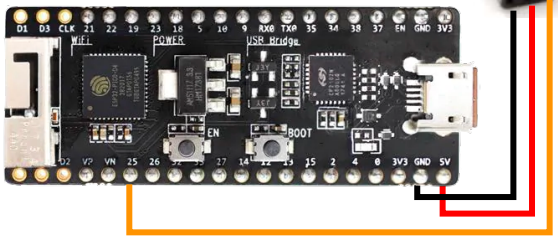


1.5mS

1mS

OscilloSorta demo

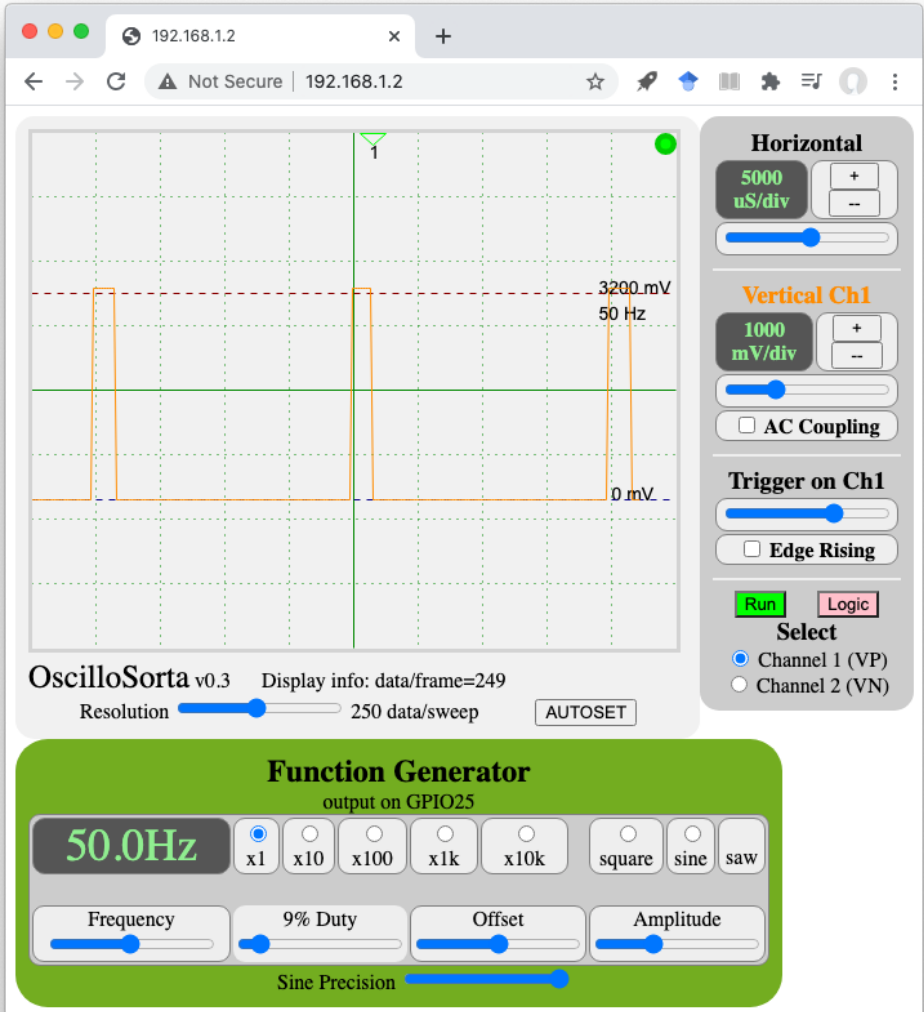
- Use 5V from USB port
  - 3.3V will also work but be a little weaker/slower
- OscilloSorta generates 3.3V
  - Not to spec, but
  - Probably okay for SG90
- PWM on OscilloSorta:
  - Freq can vary
  - Resolution ~5



Gnd

+5V (from USB)

Pin 25 FG output



02

PWM generation on Teensy

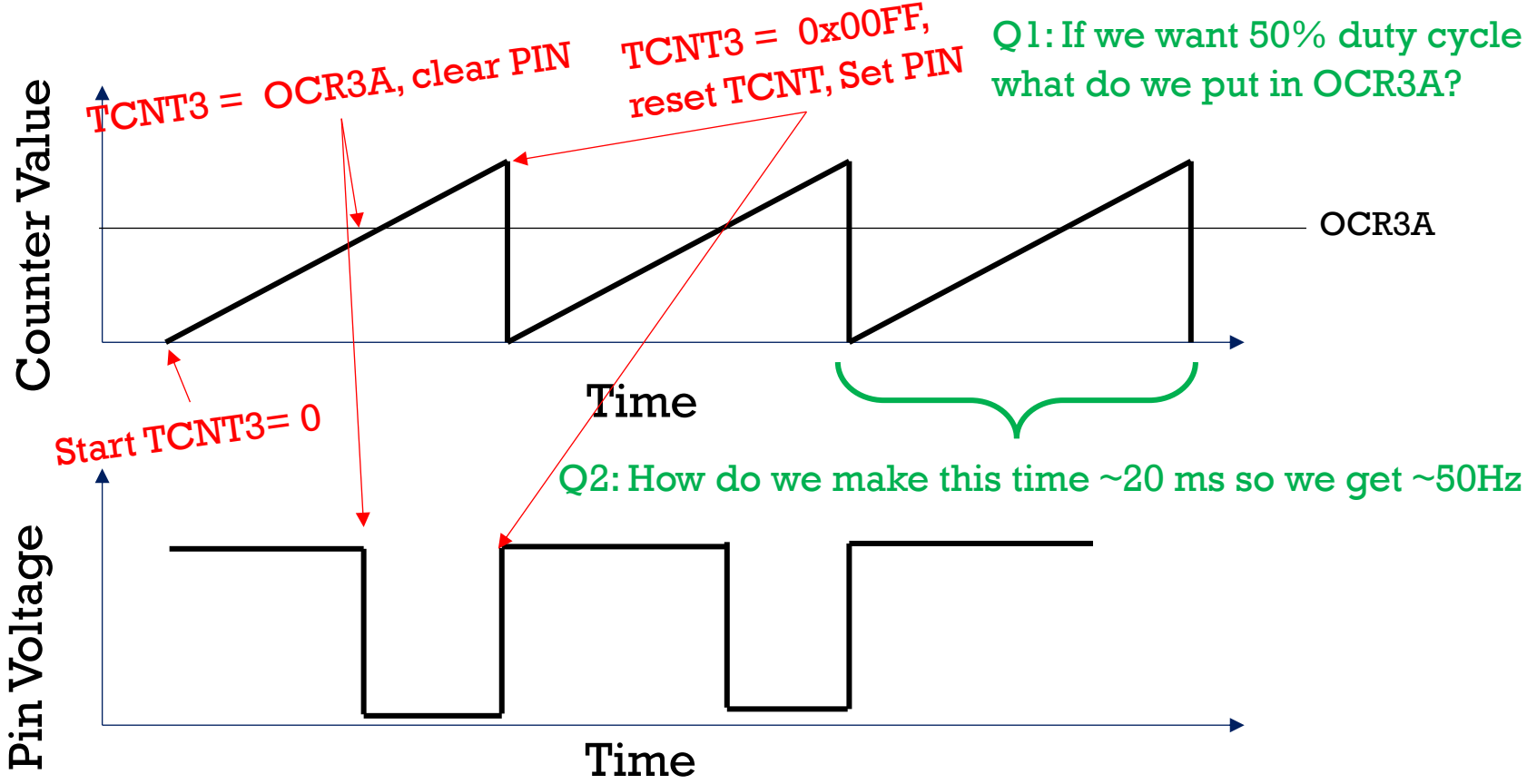
Timer 3 Control Registers TCCR3A TCCR3B

- Waveform Generator Modes

TCCR3B: WGM33	TCCR3B: WGM32	TCCR3A: WGM31	TCCR3A: WGM30	Function
Normal: Timer UP to a value, reset to 0x0000:				
0	0	0	0	(mode 0) UP to 0xFFFF (16-bit)
0	1	0	0	(mode 4) UP to OCR3A
1	1	0	0	(mode 12) UP to ICR3
Single-Slope: Timer UP to a value, reset to 0x0000 (set/reset PWM):				
0	1	0	1	(mode 5) UP to 0x00FF (8-bit), PWM mode
0	1	1	1	(mode 7) UP to 0x03FF (10-bit), PWM mode
1	1	1	0	(mode 14) UP to ICR3, PWM mode
Dual-Slope: Timer UP to a value, DOWN to 0x0000 (set/reset PWM):				
0	0	0	1	(mode 1) UP to 0x00FF, DOWN to 0x0000, PWM mode
0	0	1	1	(mode 3) UP to 0x03FF, DOWN to 0x0000, PWM mode
1	0	1	0	(mode 10) UP to ICR3, DOWN to 0x0000, PWM mode

OCR3A

PWM Waveform Mode 5 (8bit resolution)



Timer 3 Control Registers TCCR3A TCCR3B

Bit	7	6	5	4	3	2	1	0	
	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare

COM3A1	COM3A0

Clock Source

CS32	CS31	CS30

Waveform Generation Mode

WGM33	WGM32	WGM31	WMG30

Timer 3 Control Registers TCCR3A TCCR3B

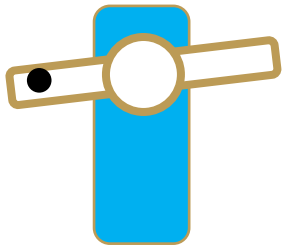
- Q3 What clock prescaler will work for generating a RC servo signal?

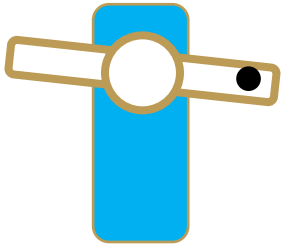
TCCR3B: CS32	TCCR3B: CS31	TCCR3B: CS30	Prescaler
0	0	0	OFF
0	0	1	/1
0	1	0	/8
0	1	1	/64
1	0	0	/256
1	0	1	/1024

Clock Frequency		Mode 5 PWM Frequency (wraps at 0xFF)
Source Off		Source Off
16Mhz	/256	62.5khz
2Mhz		7.8khz
250khz		976hz
62.5khz		244hz
15.625khz		61hz

Servo resolution (with mode 5 which is 8bit)

	Full Left	Full Right
Angle	-90	+90
PWM pulse	1mS	2mS
PWM Timer @ falling edge	255 * 1ms/20ms = 12	255 * 2ms/20ms = 24

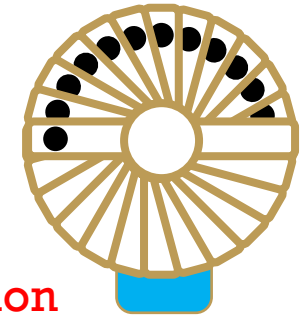


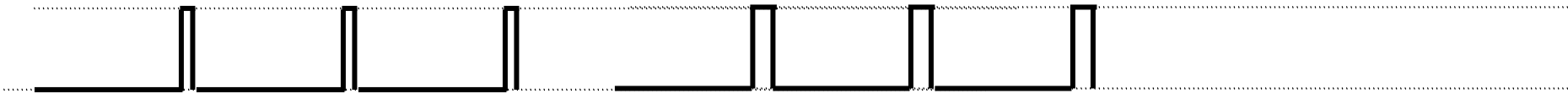


Angle increment = 180/12 = 15°
#positions = 12

Not smooth!

Use Mode 7 or 14
For higher resolution





Steps for PWM example pin C6 mode 5

- Initialization for timer 3 mode 5
  - Set mode TCCR3A TCCR3B registers : WGM3x bits
  - Set pin to be output DDRC and PORT6 bit
  - Set polarity for output compare TCCR3A register : COM3Ax bits
  - Set timer prescaler TCCR3B register: CS3x bits

- Setting PWM duty cycle
  - Write to OCR3A register (0-255 for mode 5)

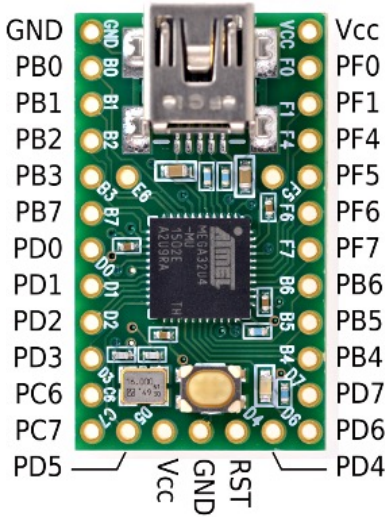
- Recommend you use the oscilloscope to see if you are getting proper signal on output pin.

Refer to website for values: http://medesign.seas.upenn.edu/index.php/Guides/MaEvArM-timer3

03

Power Requirements

Current driving for servos



- How much current does the servo require?
- How much current can be supplied?
  - From GPIO output pin (on ATMEGA32U4?)
  - From VCC (5V) of Teensy?
  - From USB port?

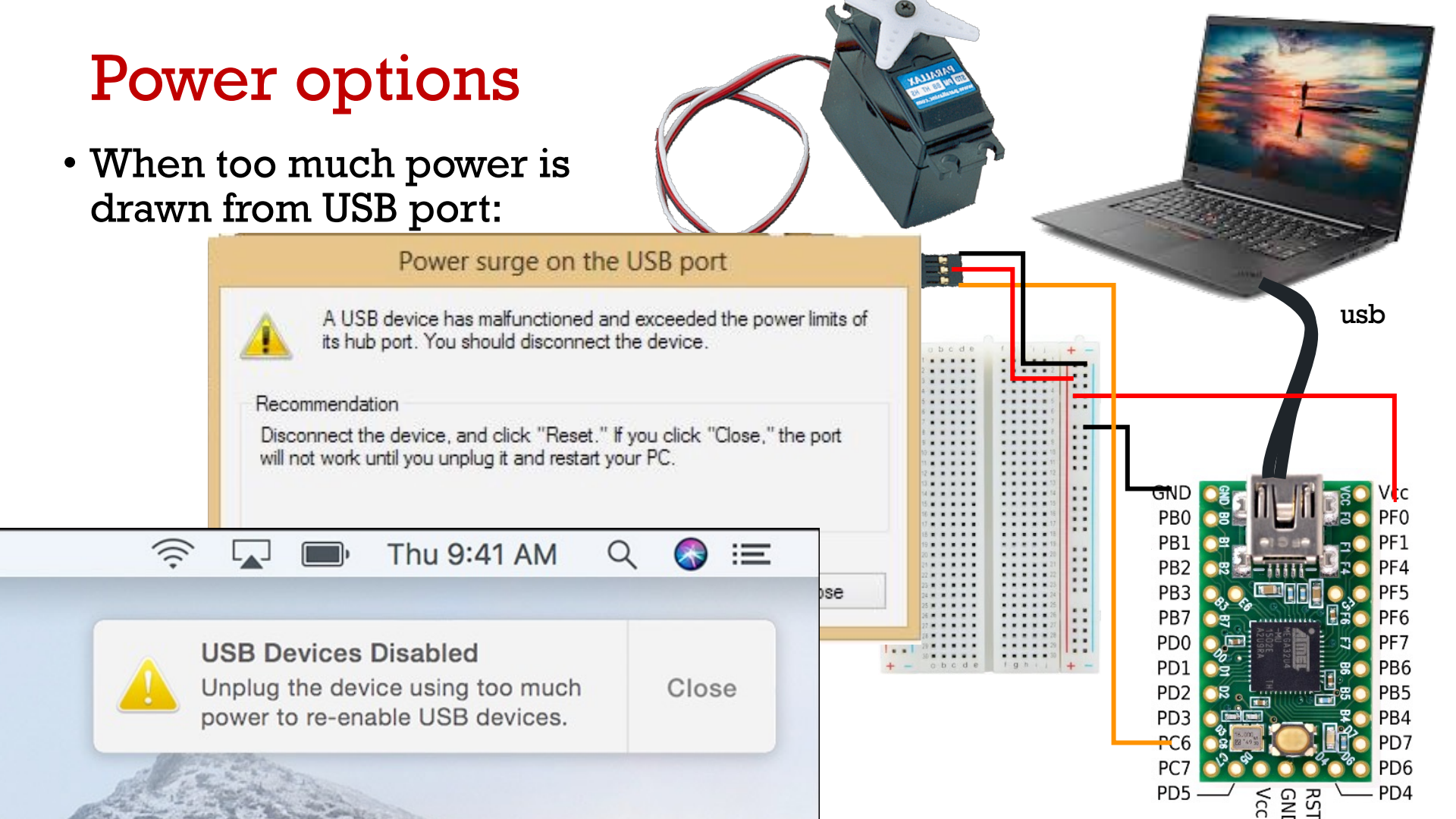
Q4: What happens if servo power is connected to VCC of Teensy and it requires more than VCC can supply?

A. The servo works weakly
B. The servo does not work and neither does the Teensy
C. The Teensy blows up
D. The USB Power source (laptop or battery) blows up
E. Some combination of all of the above

- What can we do to supply more current?

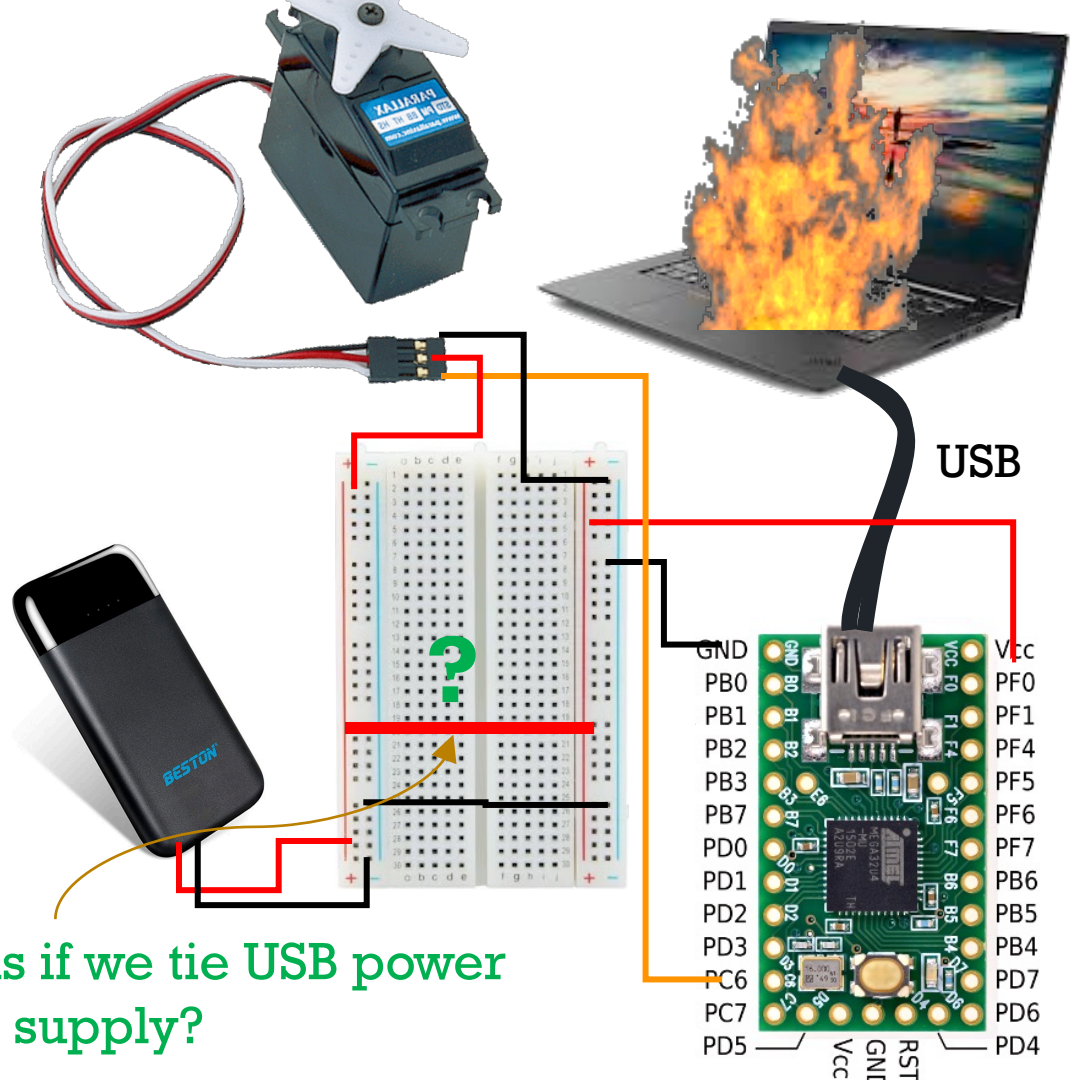
Power options

- When too much power is drawn from USB port:



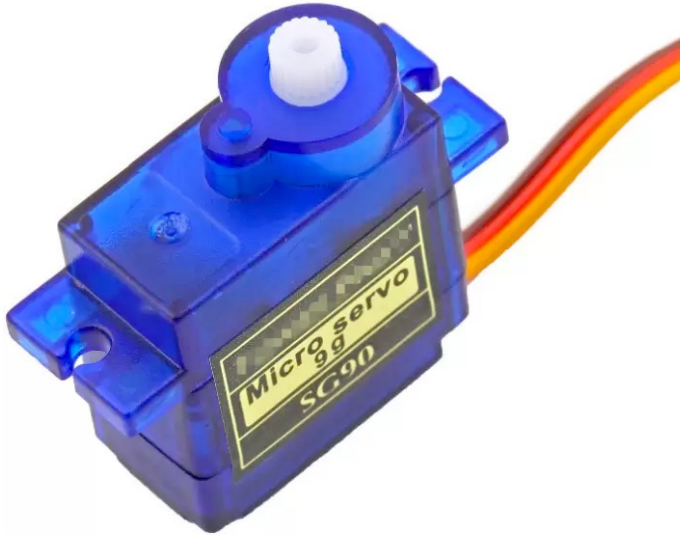
Power options

- Using two power sources (e.g. use Battery)
- Is it safe to connect external power supply while being powered by USB?
- Need to share common ground



Q5: What happens if we tie USB power to external power supply?

RC SERVO
SG90 (also smaza S51)



Everyone receives 2 servos as part of Lab 3

Digital?:	N
Free-run current @ 6V:	120 mA
Stall current @ 6V:	800 mA
Speed @ 6V:	0.10 sec/60°
Stall torque @ 6V:	1.5 kg·cm
Speed @ 4.8V:	0.12 sec/60°
Stall torque @ 4.8V:	1.3 kg·cm
Lead length:	10 in

RC SERVO HD 1581HB

Digital?:	Y
Free-run current @ 6V:	300 mA
Stall current @ 6V:	1200 mA
Speed @ 6V:	0.16 sec/60°
Stall torque @ 6V:	2.6 kg·cm
Speed @ 4.8V:	0.21 sec/60°
Stall torque @ 4.8V:	2.2 kg·cm
Lead length:	10 in

RC SERVO HD 1810MG

Digital?:	Y
Free-run current @ 6V:	240 mA
Stall current @ 6V:	1400 mA
Speed @ 6V:	0.13 sec/60°
Stall torque @ 6V:	3.9 kg·cm
Speed @ 4.8V:	0.16 sec/60°
Stall torque @ 4.8V:	3.1 kg·cm

RC SERVO MG 996R

Relatively inexpensive. We may make bulk class purchase for final project.

Digital?:	Y
Free-run current @ 6V:	600 mA
Stall current @ 6V:	2500 mA
Speed @ 6V:	0.14 sec/60°
Stall torque @ 6V:	11 kg·cm
Speed @ 4.8V:	0.17 sec/60°
Stall torque @ 4.8V:	9.4 kg·cm

USB power standards

	Specification	Current	Voltage	Power (max.)
USB 1.0	Low-power device	100 mA	5 V[a]	0.50 W
	Low-power SuperSpeed (USB 3.0) device	150 mA	5 V[a]	0.75 W
USB 2.0	High-power device	500 mA[b]	5 V	2.5 W
USB 3.0	High-power SuperSpeed (USB 3.0) device	900 mA[c]	5 V	4.5 W
	Multi-lane SuperSpeed (USB 3.2 Gen 2) device	1.5 A[d]	5 V	7.5 W
	Battery Charging (BC) 1.1	1.5 A	5 V	7.5 W
	Battery Charging (BC) 1.2	5 A	5 V	25 W
	USB-C	1.5 A	5 V	7.5 W
		3 A	5 V	15 W
	Power Delivery 1.0 Micro-USB	3 A	20 V	60 W
	Power Delivery 1.0 Type-A/B	5 A	20 V	100 W
	Power Delivery 2.0/3.0 Type-C[e]	5 A	20 V	100 W

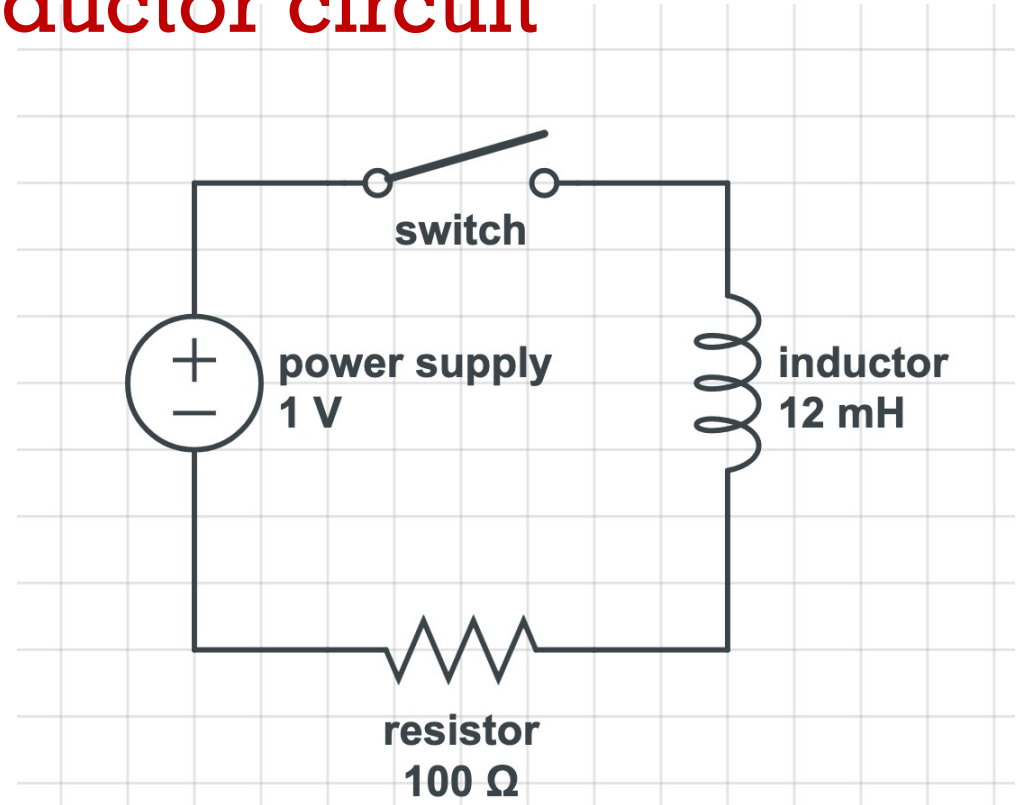
a. ^ *a b* The V_{BUS} supply from a low-powered hub port may drop to 4.40 V.
b. ^ Up to five unit loads; with non-SuperSpeed devices, one unit load is 100 mA.
c. ^ Up to six unit loads; with SuperSpeed devices, one unit load is 150 mA.
d. ^ Up to six unit loads; with multi-lane devices, one unit load is 250 mA.
e. ^ Requires active PD 5 A cable.

http://en.wikipedia.org/wiki/USB

04

Inductive loads

Inductor circuit



Inductance, L, is measured in Henrys

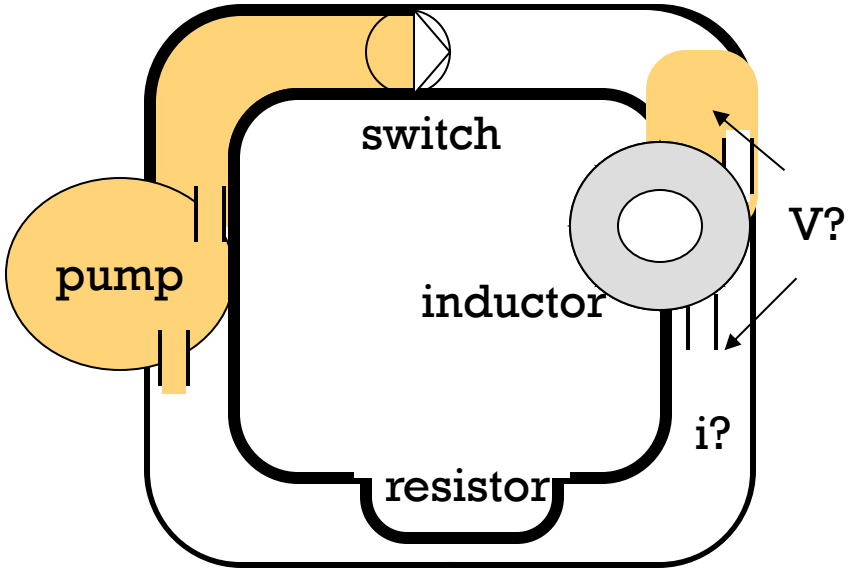
$$v(t) = L\frac{di(t)}{dt}$$

Like capacitors, interesting things happen with transients (not DC signals)

For reference Capacitor eqn:

$$i(t) = C\frac{dV(t)}{dt}$$

Large inductance



Inductance, L, is measured in Henrys

L is similar to mv, momentum, like water wheel

Initial state, switch is off
no voltage across inductor
no current in inductor

Switch turns on
hi voltage seen on one side
no current in inductor

Steady state on
voltage equal on both sides
current in inductor

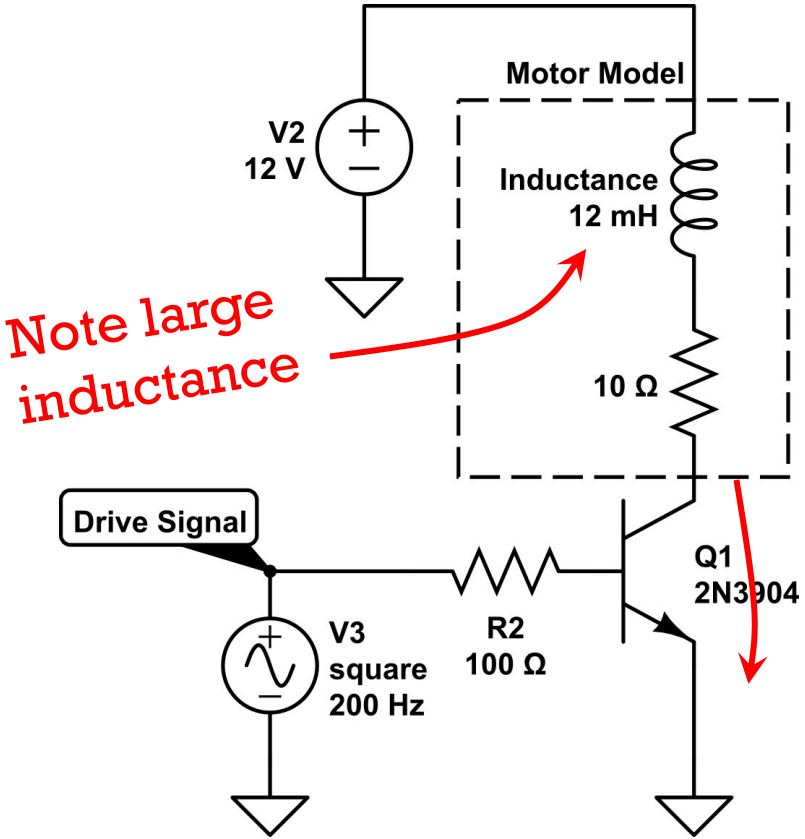
Switch turns off
hi voltage on other side
no current in inductor

Steady state off
voltage equal on both (0)
no current in inductor

Effect of Large Inductance

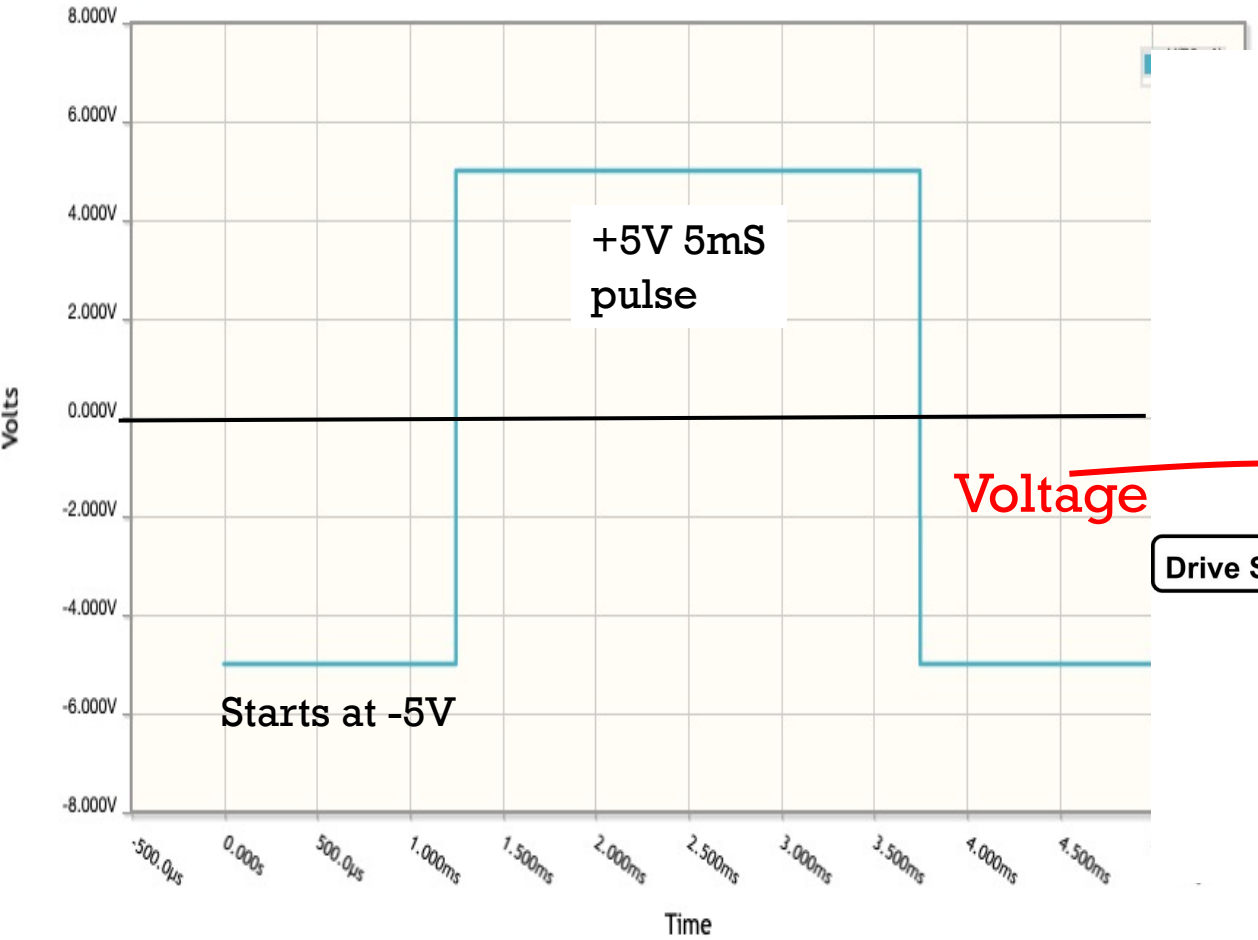
We can analyze the transient voltage and current behavior motor model driven by a transistor.

A drive signal turns on and off a transistor that turns on and off a motor.

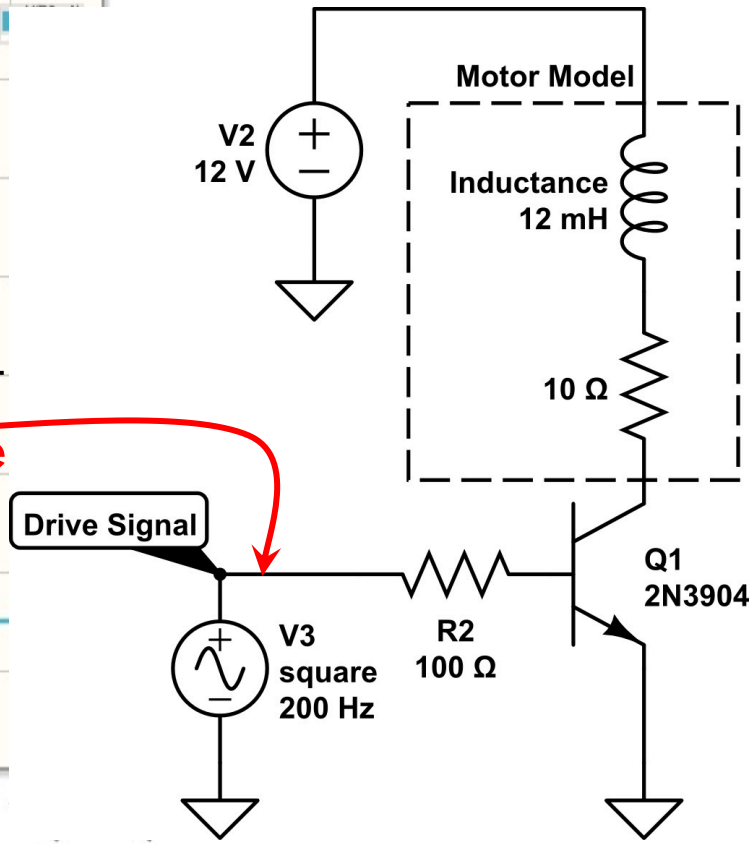


Circuitlab simulation

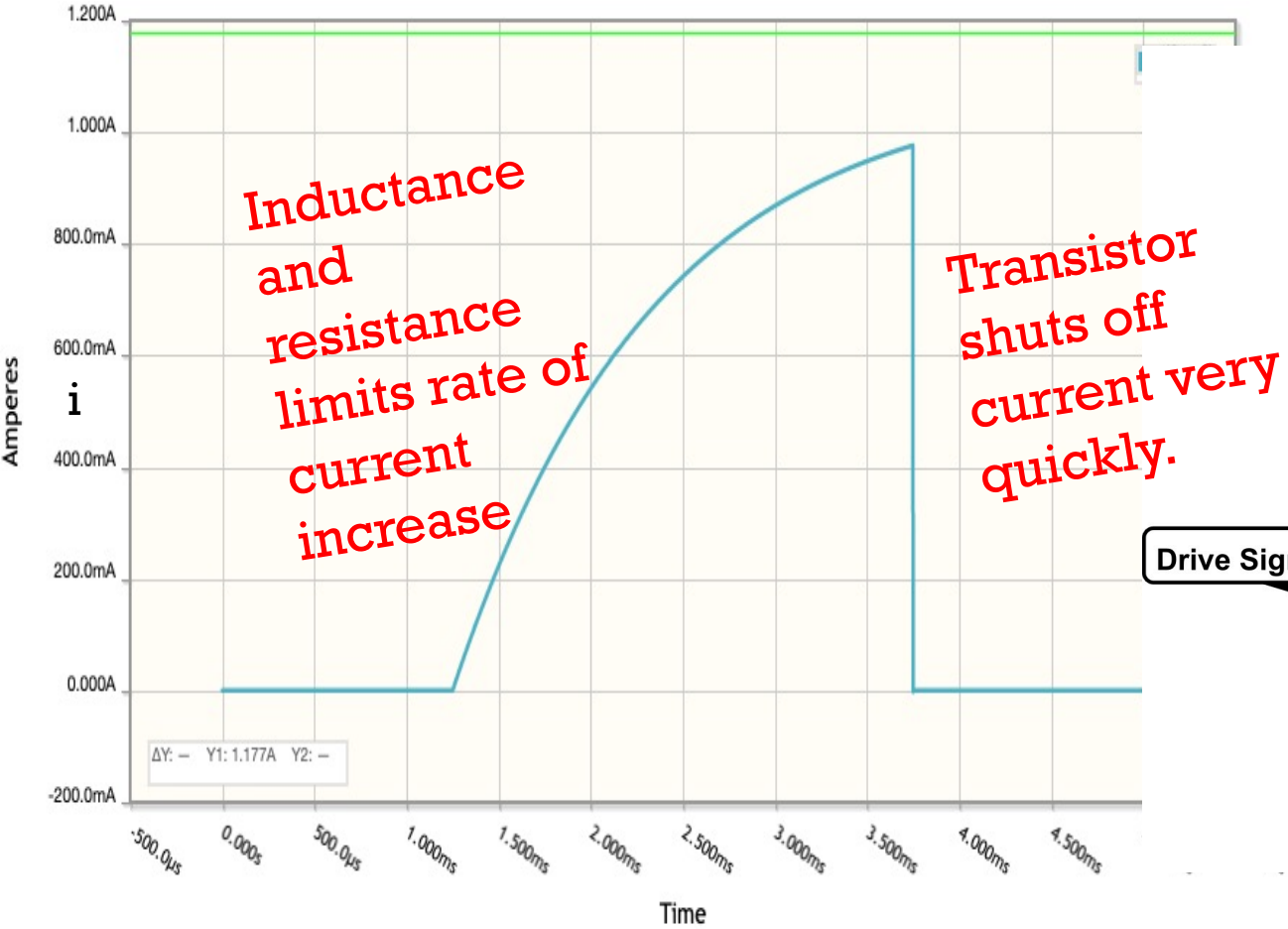
Drive Waveform Voltage

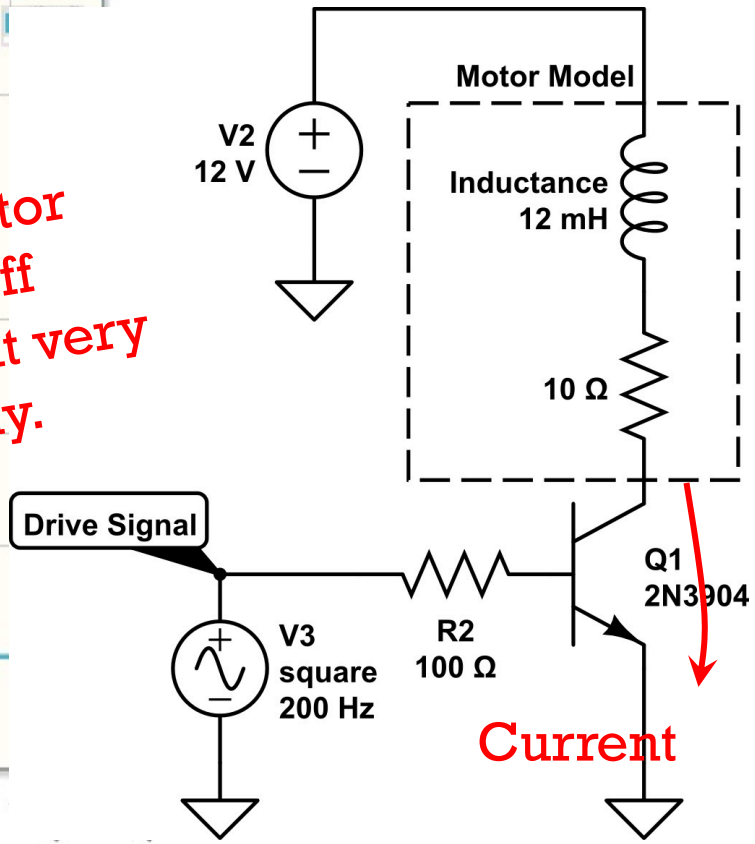


Voltage

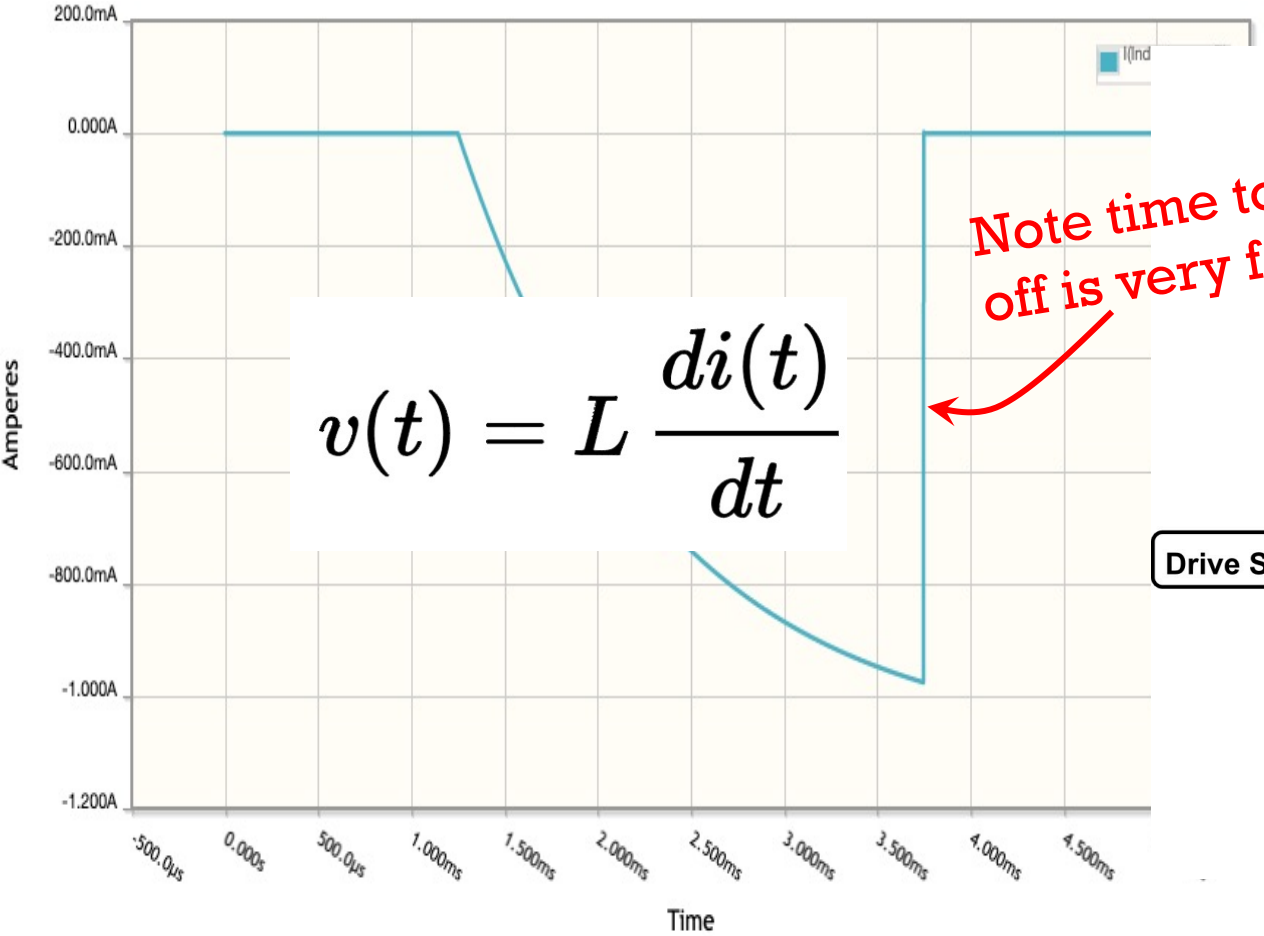


Transistor Current





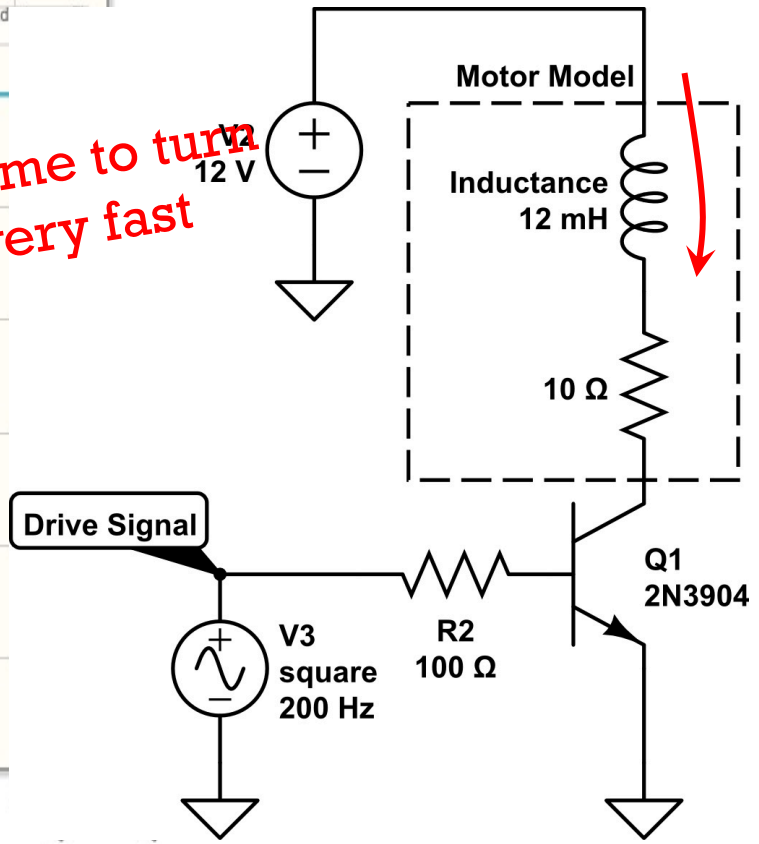
Inductor Current (ignore that it's upside down)



$$v(t) = L\,\frac{di(t)}{dt}$$

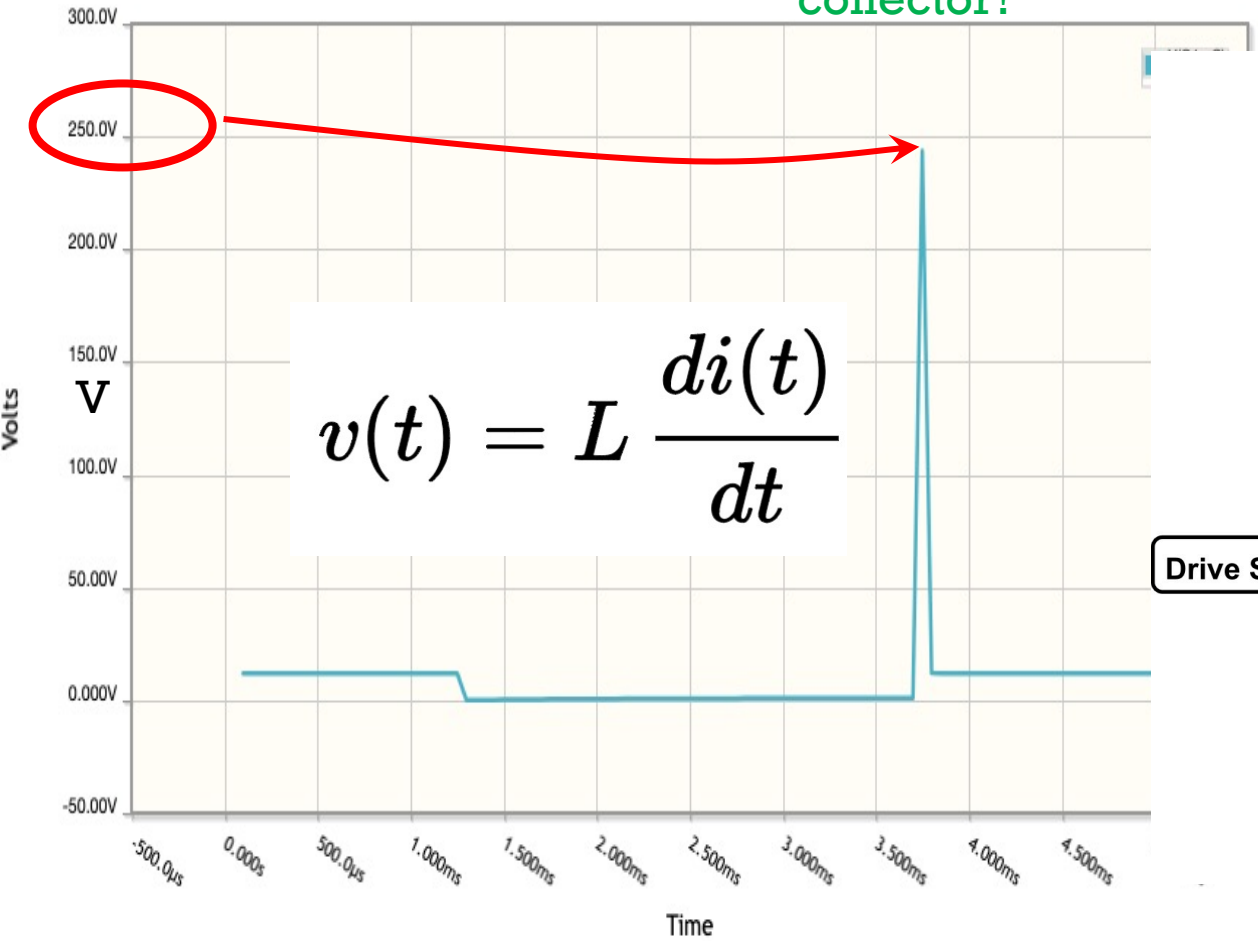
Note time to turn off is very fast

Current

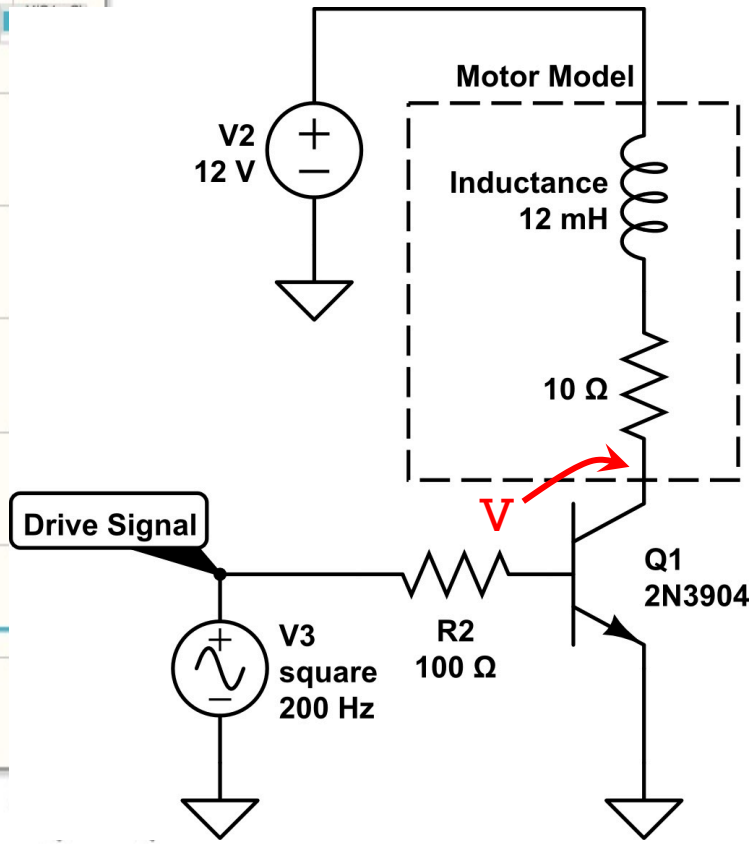


Collector Voltage

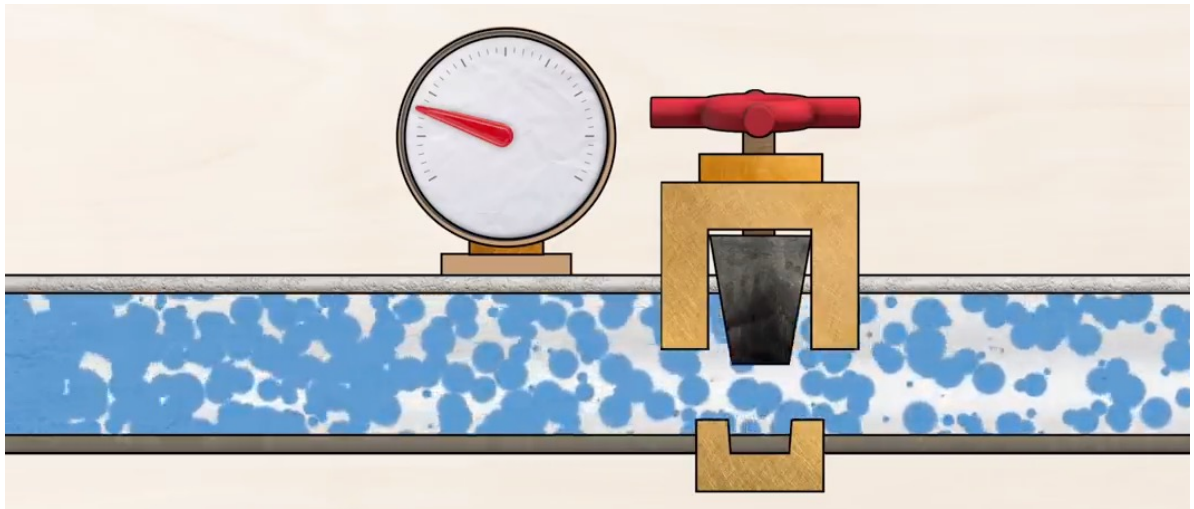
Q6: What is the peak voltage we see at the collector?



$$v(t) = L\frac{di(t)}{dt}$$



Inductive Spikes in the water model – Water hammer!



DC motors are in all servos. But nearly all servos have protection circuits so these huge voltage spikes are not passed on to your control circuit or power supply.

https://www.youtube.com/watch?v=xoLmVFAFjn4

Summary

- RC Servos are driven with 50Hz PWM signals (with 1 – 2 ms pulse width)
- RC Servos and motors draw high currents
- Large inductances (e.g. found in DC motors) can generate huge voltage spikes.
- Be careful of connecting external power supplies when USB is connected to the Teensy.

Answer in CHAT

Answer how you feel about each topic below with:

1. I don't understand this topic at all
2. I don't know now, but know what to do to get by
3. I understand some, but expect to get the rest later
4. I understand completely already

A. RC Servos

B. Inductive Loads

C. PWM on Teensy

05

Pointers and Arrays in C

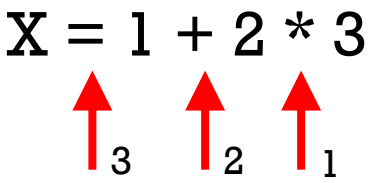
Declarations and scoping

```
int myGlobalVariable;
int i, j, k;
```

```
int main() {
  int i; // note this “i” is different than the global one above
  for (i=0; i<10 ; i++) {
    int j, k;
    subroutine(j);
  }
}
```

Operators and precedence

1. `( ) [ ] -> .` — Scope operators
2. `++ -- ! ~ - * (dereference) & (address of) cast` — Unary operators
3. `* / %` — Math operations
4. `+ -` — Math operations
5. `<< >>` — Math operations
6. `< <= >= >` — Comparison operations
7. `== !=` — Comparison operations
8. `&` — Bitwise operators
9. `^` — Bitwise operators
10. `|` — Bitwise operators
11. `&&` — Logical operators
12. `||` — Logical operators
13. `? :`
14. `= *= /= %= += -= <<= >>= &= |= ^=` — Assignment operators
15. `,`



Arrays

- Hold multiple pieces of data and reference them individually:

```
int myArrayofInts[4];

int myArrayofInts[] = {2, 3, -1, 5}; // initialize
myArrayofInts[0]= 4;
// first index is 0

x = myArrayofInts[1]; // make x = 3
char word[] = "hello";
```

Using array's

```
int x[8];

for (int i=0; i<8; i++) {
  x[i] = i*i;
}
m_usb_tx_uint(x[3]);
```

Q7: What is printed?

0 indexed -> it starts with 0 (unlike MATLAB which starts at 1)

`x[0] = 0`, x[] holds 8 values up to `x[7] = 49`

First element is `x[0]`, and does **NOT** go to `x[8]`

Pointers

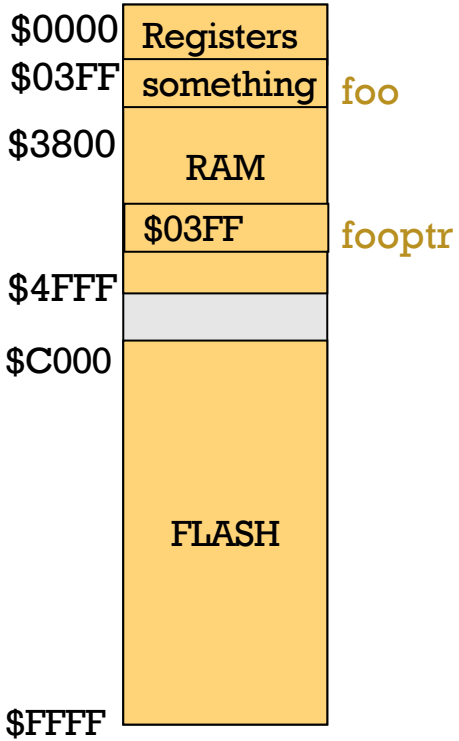
- A pointer is a variable whose value is the address of something.

- Example:

```
int     foo;
int     *fooptr = &foo;
```

& is "address of operator"

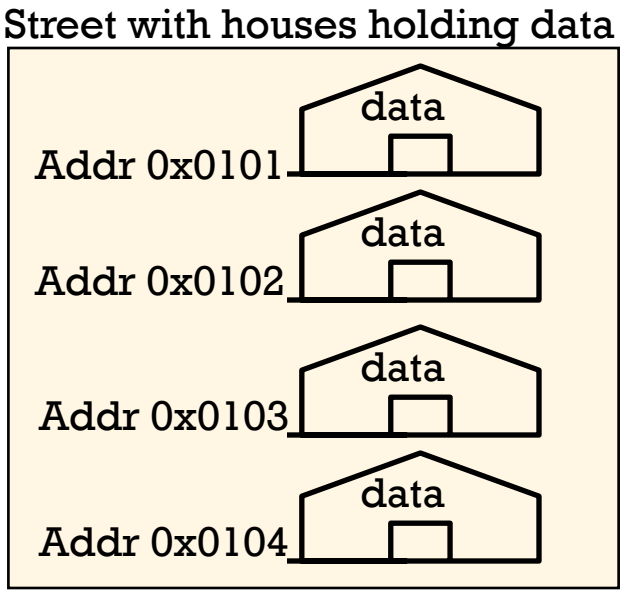
* is dereference operator



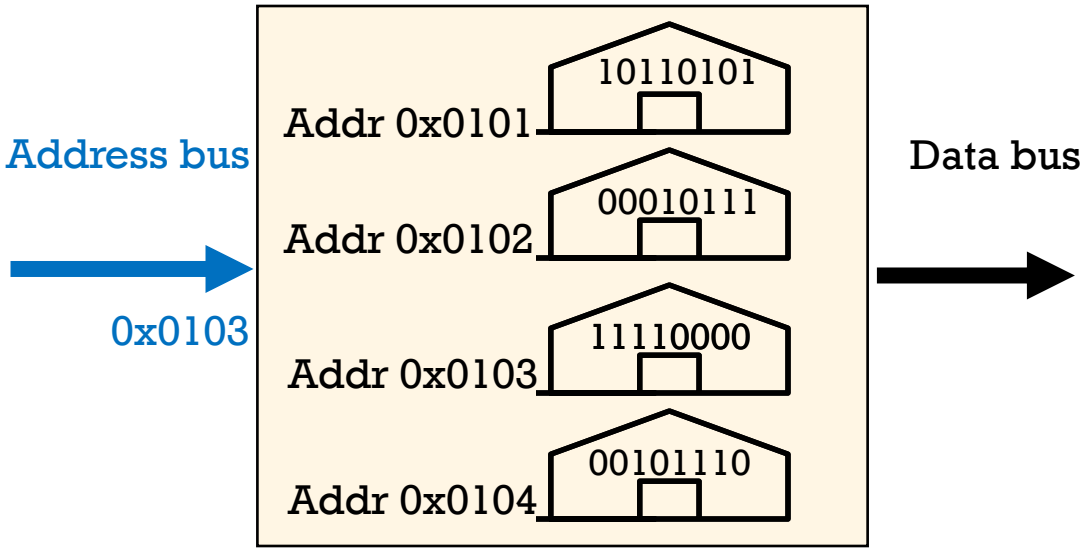
What is memory?
(from Lecture 3)

Memory holds addressable data that you can READ and/or WRITE to

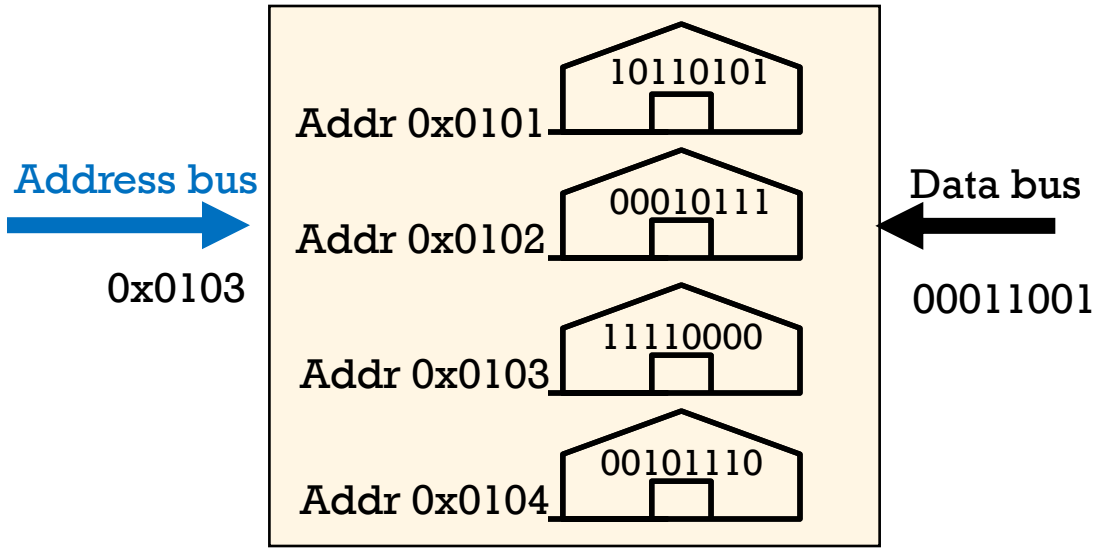
Addresses enable access to specific data



Memory READ


Address bus
0x0103
Addr 0x0101
10110101
Addr 0x0102
00010111
Addr 0x0103
11110000
Addr 0x0104
00101110
Data bus

Memory WRITE

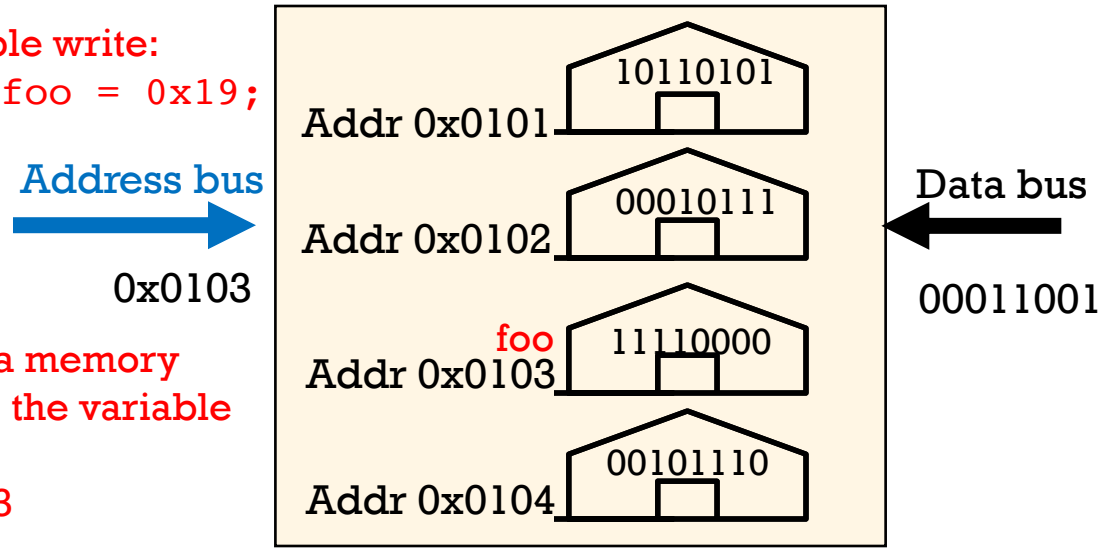


What is a variable?

A variable is the name assigned to a memory location that holds the variable's data

Example write:

```
char foo = 0x19;
```



The compiler finds a memory location to assign to the variable (this time 0x103).
So, foo lives at 0x103

What is a pointer?

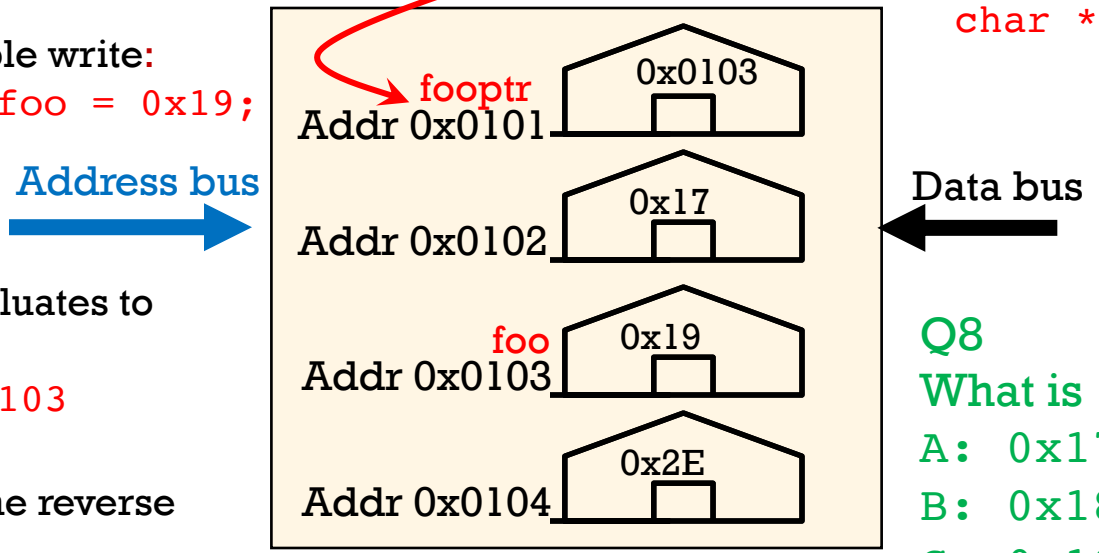
A pointer is the address of a variable (memory location).

The compiler allocates memory for fooptr

```
char *fooptr = &foo;
```

Example write:

```
char foo = 0x19;
```



`&` operator evaluates to (Address of)

```
&foo is 0x0103
```

`*` operator is the reverse (contents of) :

```
*(&foo) = 0x19;
```

Q8

What is `*(fooptr-1)`?

A: 0x17

B: 0x18

C: 0x19

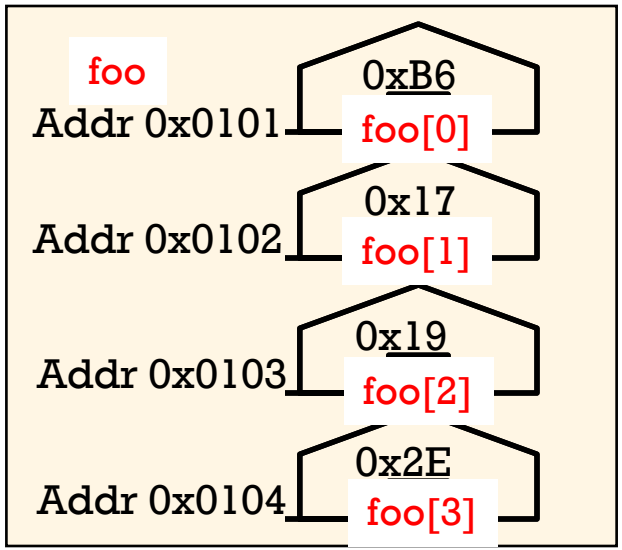
D: 0x2E

How is an Array related to Pointers?

```
char foo[4]; // declaration allocates 4 sequential bytes
```

`foo` is pointer
that points to `0x0101`

`*foo` is `foo[0]`



Q9: Which is correct?

A `*foo+1 = foo[1]`

B `(*foo)+1 = foo[1]`

C `*(foo+1) = foo[1]`

Q10: Which is correct?

A `foo+2 = &foo[2]`

B `*foo+2 = &foo[2]`

C `&(foo+2) = &foo[2]`

Pointer declarations

- Pointer declarations use * combined with a variable type

```
int    *ip;    // pointer to an integer  (2 bytes)
double *dp;    // pointer to a double    (8 bytes)
float  *fp;    // pointer to a float     (4 bytes)
char   *cp;    // pointer to a character (1 byte)
// Note: char* cp; is the same as char * cp; and char *cp;
```

- Size of pointers are all the same (16bit for ATmega)
- Size of the things they point to may vary (important for arrays)

Using Pointers

- Assignment of declared pointer:

```
fooptr = 42;
```

Will generate a warning when compiling

- Dereferencing

```
*fooptr = 42;
```

Stores 42 into the location at fooptr

```
int bar = *fooptr;
```

Loads the contents in fooptr into bar

- Passing pointers to functions

```
int subroutine1( int *fooptr) {
  *fooptr = 20;   // changes content of pointer (like a global)
  return 1;
}
```

Using Pointers

- Assignment

```
fooptr = 42;
```

Will generate a warning when compiling

- Dereferencing

```
*fooptr = 42;
```

Stores 42 into the location at fooptr

```
int bar = *fooptr;
```

Loads the contents in fooptr into bar

- Pointer arithmetic (4 operations)
  - ++ Increment address by one word (sizeof variable)
  - -- Decrement address by one word (sizeof variable)
  - + Add to address # of words (sizeof variable)
  - - Subtract from address # of words (sizeof variable)

Pointers and Arrays

- Arrays can be treated as pointers almost always

```
int array[];
array == &array[0] == &array
int *iptr;
*iptr == iptr[0]
```

- Some exceptions

```
int *arrayA, arrayB[8];
arrayA = arrayB;
```

`arrayA[i]` is the same as `arrayB[i]`

`*(++arrayA)` is the same as `arrayB[1];` but

`*(++arrayB)` will give an error. You can't change the value of the address of an array.

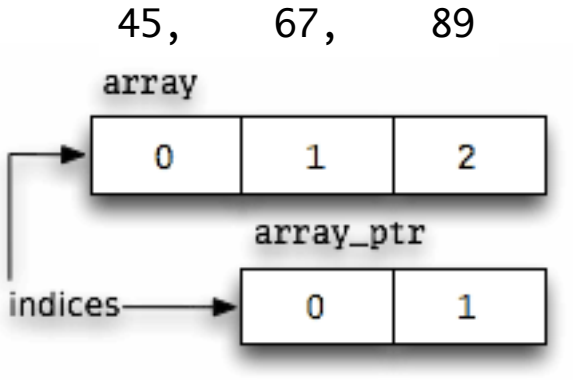
- Strings are arrays of characters that are NULL terminated, adding extra

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
char greeting[] = "Hello";
```

Pointers and Arrays

- Q11: What does the following print?

```
int array[] = { 45, 67, 89 };
int *array_ptr = &array[1];
printf("%i\n", array_ptr[1]);
```



Pre-increment and post-increment

```
void somefunction()
{
  int x = 10, a;
   a = ++x;   // Value of x will change before assignment

  m_usb_tx_uint( a);
  m_usb_tx_uint( x);
}
```

Output:

```
11
11
```

```
void somefunction()
{
  int x = 10, a;
   a = x++;   // Value of x will change after assignment

  m_usb_tx_uint( a);
  m_usb_tx_uint( x);
}
```

```
a = x;
x = x+1;
```

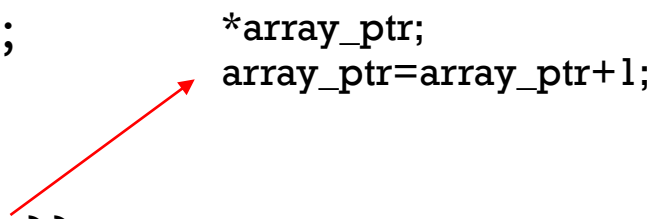
Output:

```
10
11
```

Q12: Pointer Arithmetic (what will it print?)

- Indexing an array with pointer math

```
int array[] = { 45, 67, 89 };
int *array_ptr = array;
somecode() {
  m_usb_tx_uint(*(array_ptr++));
  m_usb_tx_uint(*(array_ptr++));
  m_usb_tx_uint(*array_ptr);
  m_usb_tx_uint(*(array+2));
}
```



Q13: Pointer Arithmetic (what will it print?)

- Indexing an array with pointer math

```
int array[] = { 45, 67, 89 };
int *array_ptr = array;
somecode() {
  m_usb_tx_uint(*(array_ptr++));
  m_usb_tx_uint(*(array_ptr++));
  m_usb_tx_uint(*array_ptr);
  m_usb_tx_uint(*(array+2));
}
```

*array_ptr;
array_ptr=array_ptr+1;

parentheses has no effect in many post increment cases

Extra stuff about pointers

- In general, you can probably get through this course without using pointers.
- Pointers which point to NULL or 0 are a special case. Usually NULL is an invalid address, often indicating uninitialized pointers or other special cases.
- Multiple indirection is valid (I don't recommend it…)

```
int    a =  3;
int   *b = &a;
int  **c = &b;
int ***d = &c;
```

- Function pointers. You can treat functions as variables by passing pointers to functions. (advanced topic – not necessary in this course).

http://boredzo.org/pointers/

Summary

- Arrays are like pointers (variables that hold addresses)
- For MEAM510, most functions can be achieved without pointers, but pointers are the mechanism that makes accessing to registers work. It's all hidden from you by people who set up the compiler.