Lecture 12

CATCH-UP

Agenda

00. OscilloSorta V1.3 Demo

01. ADC on ATmega 32U4 (Teensy)

02. Software filters

03. Q&A

Stuff

- Selecting music for Lab 3 Waldo dance… post suggestions on Piazza to pinned thread. On Monday we'll take a Piazza poll to vote for your favorite.

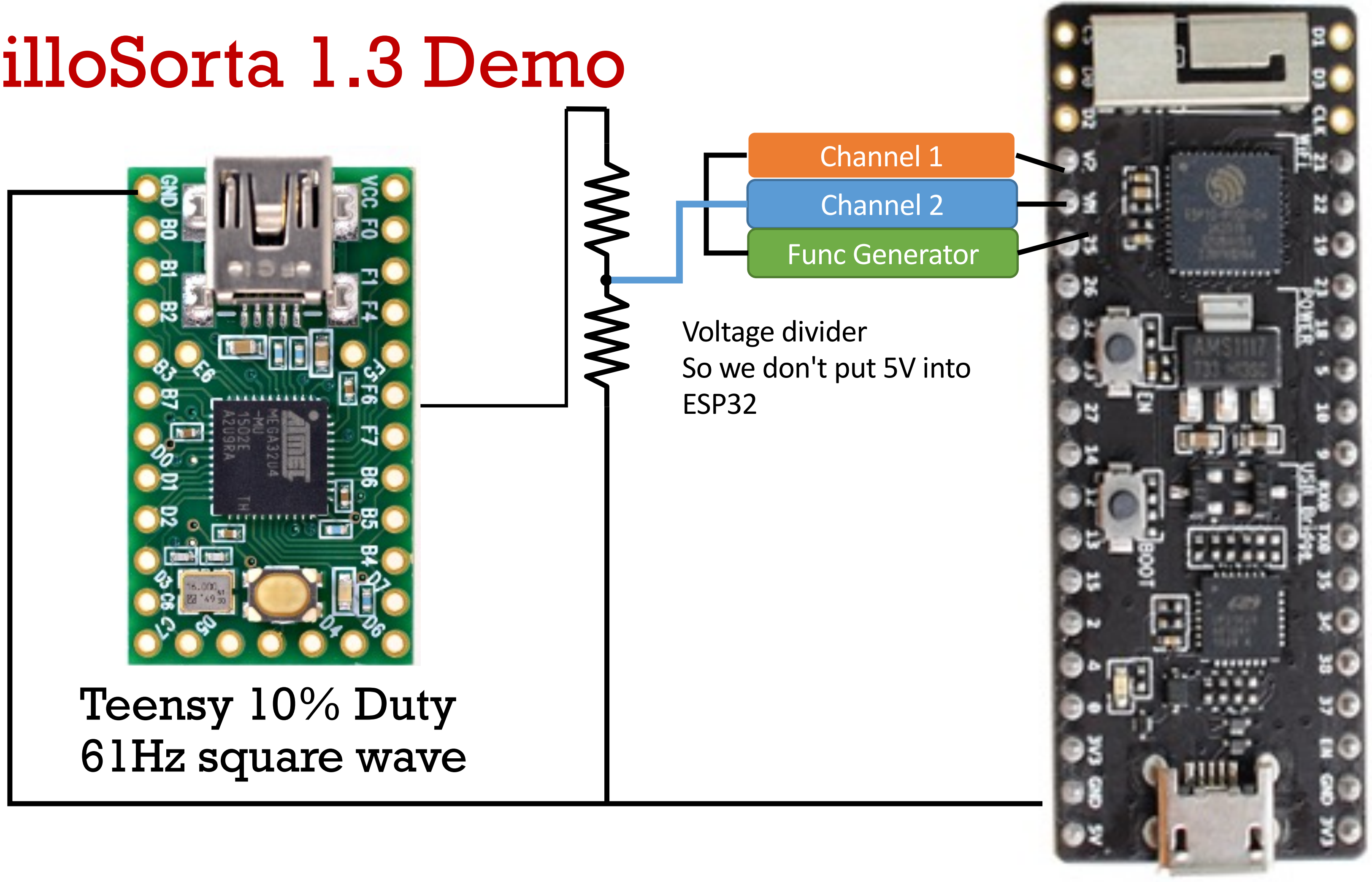
- Today is your opportunity to ask Lab 3 questions like:

  How do we setup ADC to read two at the same time?

  What Teensy commands set the position of the servo?

- Fall break starts tomorrow – no recitation – check calendar for OH.

OscilloSorta 1.3 Demo



Channel 1

Channel 2

Func Generator

Voltage divider
So we don't put 5V into
ESP32

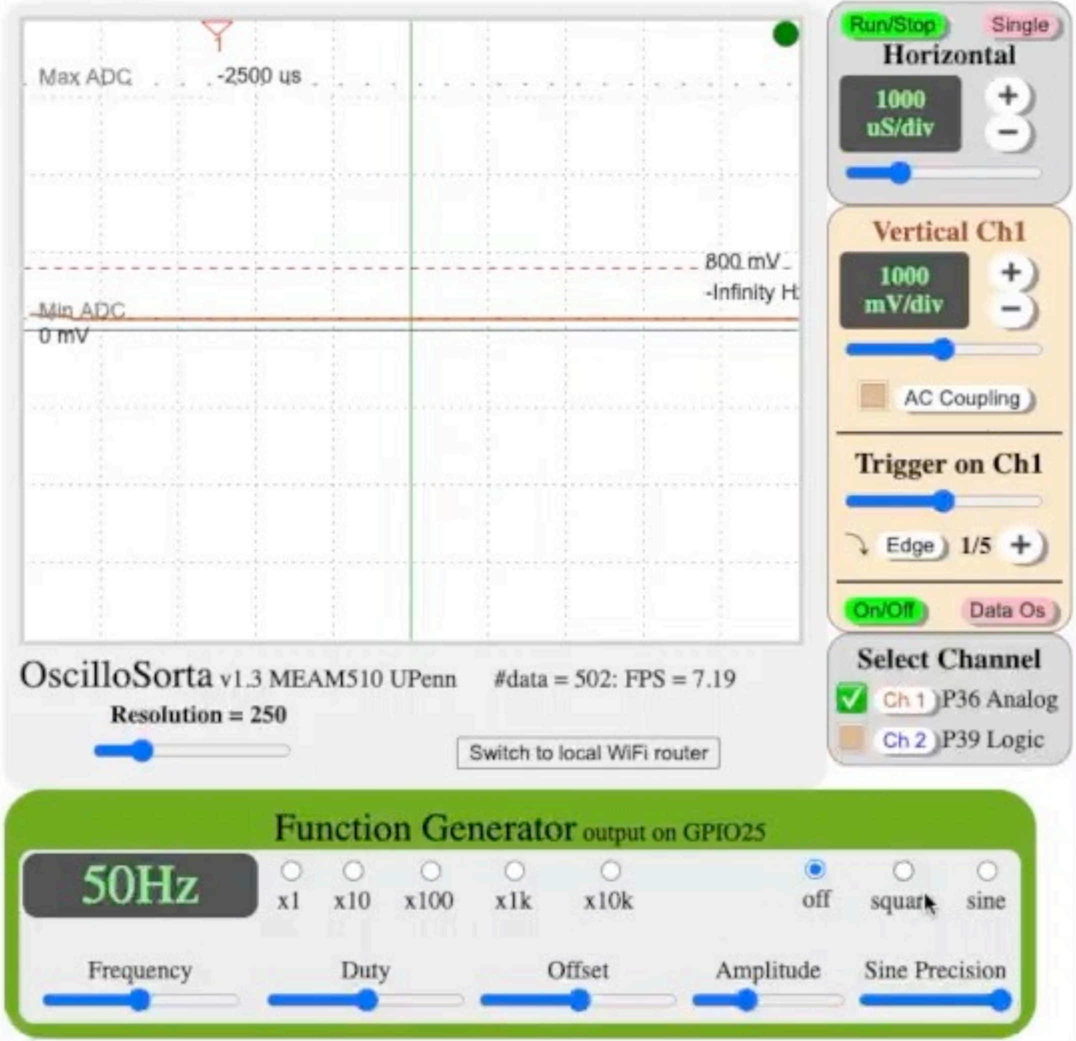
Teensy 10% Duty
61Hz square wave

Features

- Voltage Range
- Time Range
- Measurements
- Trigger on edges
- 2 Channel mode
- Single shot mode
- Aliasing caveats
- Sinewave params
- Resolution

Caveats:

- Ch 2 digital only
- Ch 1 and 2 sync not very good
- AC coupling ch 1 only



Horizontal: 1000 uS/div

Vertical Ch1: 1000 mV/div

□ AC Coupling

Trigger on Ch1: Edge 1/5

OscilloSorta v1.3 MEAM510 UPenn #data = 502: FPS = 7.19

Resolution = 250

Select Channel: ☑ Ch 1 P36 Analog; □ Ch 2 P39 Logic

Function Generator: 50Hz

○ x1 ○ x10 ○ x100 ○ x1k ○ x10k ● off ○ square ○ sine

01

ADC on ATmega 32U4

Round 2

Steps to use ADC

1. set the voltage reference
2. set the ADC clock prescaler
3. disable ADC digital input
4. set up interrupts and triggering, if desired
5. select the desired analog input
6. enable conversions
7. start the conversion process
8. wait for conversion to finish
9. read the result
10. clear the conversion flag

Setup registers

Get reading from one pin

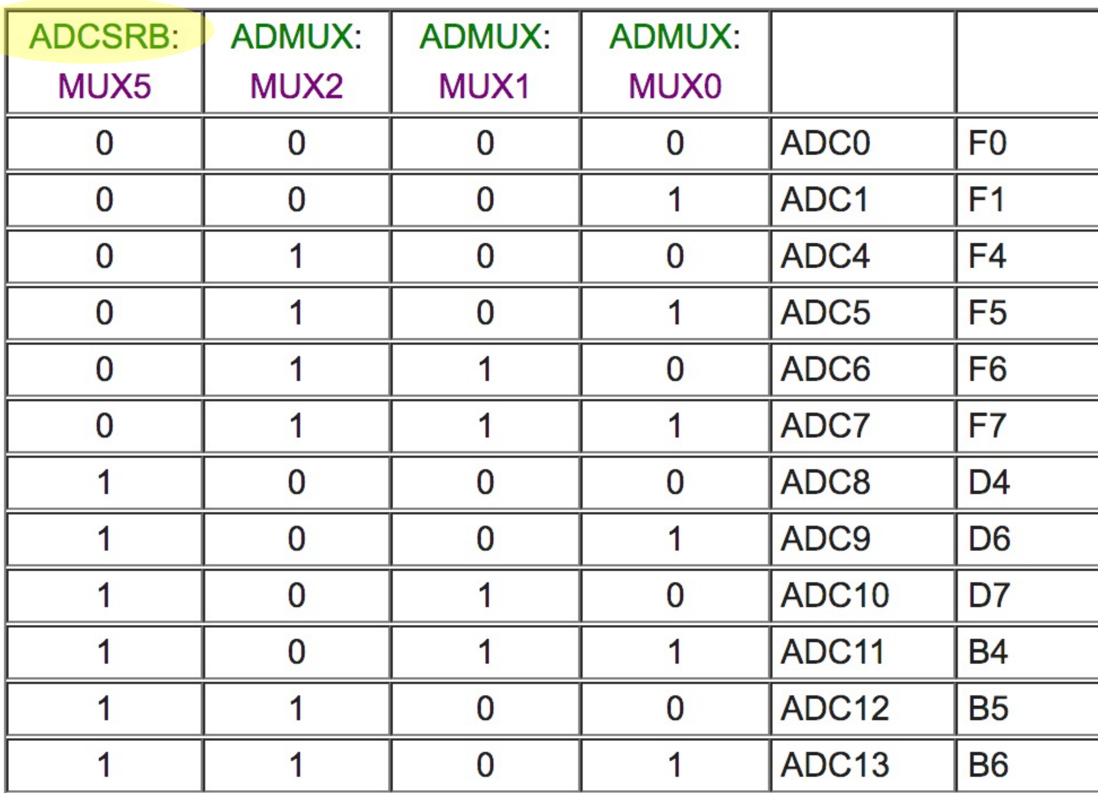
repeat

All of this using registers.

See http://medesign.seas.upenn.edu/index.php/Guides/MaEvArM-adc

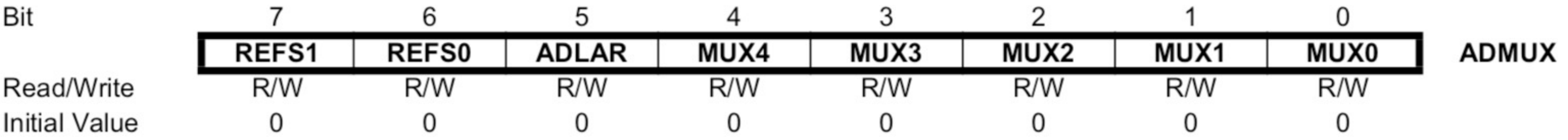
Step 5: Choose ADC channel

(skip step 4 for now)



ADCSRB: MUX5	ADMUX: MUX2	ADMUX: MUX1	ADMUX: MUX0		
0	0	0	0	ADC0	F0
0	0	0	1	ADC1	F1
0	1	0	0	ADC4	F4
0	1	0	1	ADC5	F5
0	1	1	0	ADC6	F6
0	1	1	1	ADC7	F7
1	0	0	0	ADC8	D4
1	0	0	1	ADC9	D6
1	0	1	0	ADC10	D7
1	0	1	1	ADC11	B4
1	1	0	0	ADC12	B5
1	1	0	1	ADC13	B6

Q1: Why can we initially set the ADC channel to ADC0 without any set / clear commands?



Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
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	

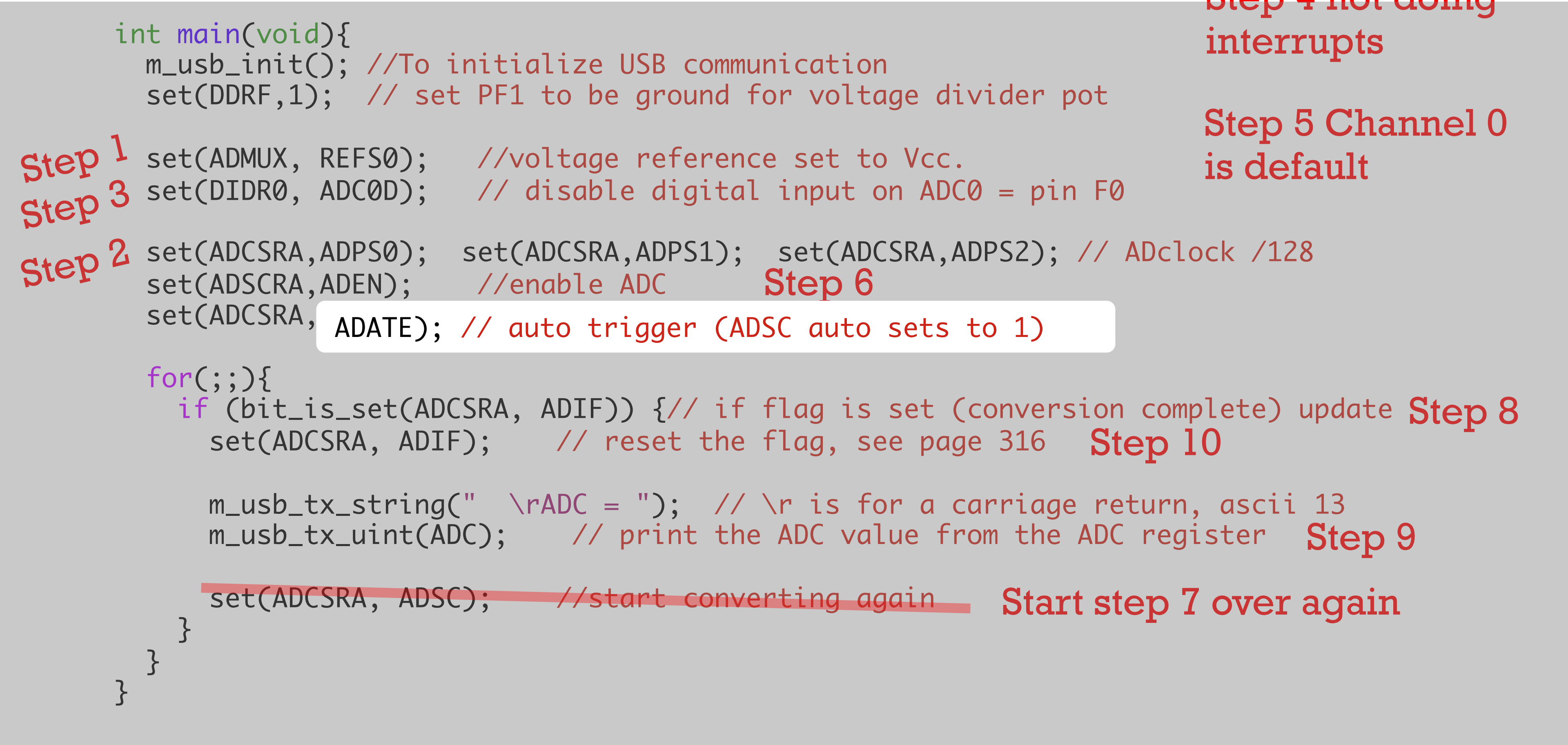
Simplest ADC code (uses ADC0)

Step 4 not doing interrupts

Step 5 Channel 0 is default

```
        int main(void){
          m_usb_init(); //To initialize USB communication
          set(DDRF,1);  // set PF1 to be ground for voltage divider pot

Step 1    set(ADMUX, REFS0);   //voltage reference set to Vcc.
Step 3    set(DIDR0, ADC0D);   // disable digital input on ADC0 = pin F0

Step 2    set(ADCSRA,ADPS0);  set(ADCSRA,ADPS1);  set(ADCSRA,ADPS2); // ADclock /128
          set(ADSCRA,ADEN);    //enable ADC        Step 6
          set(ADCSRA, ADSC);   //start it converting  Step 7

          for(;;){
            if (bit_is_set(ADCSRA, ADIF)) {// if flag is set (conversion complete) update  Step 8
              set(ADCSRA, ADIF);    // reset the flag, see page 316   Step 10

              m_usb_tx_string("  \rADC = ");  // \r is for a carriage return, ascii 13
              m_usb_tx_uint(ADC);    // print the ADC value from the ADC register   Step 9

              set(ADCSRA, ADSC);    //start converting again    Start step 7 over again
            }
          }
        }
```

Optional AutoTrigger ADATE



```
int main(void){
  m_usb_init(); //To initialize USB communication
  set(DDRF,1);  // set PF1 to be ground for voltage divider pot

  set(ADMUX, REFS0);   //voltage reference set to Vcc.
  set(DIDR0, ADC0D);   // disable digital input on ADC0 = pin F0

  set(ADCSRA,ADPS0);  set(ADCSRA,ADPS1);  set(ADCSRA,ADPS2); // ADclock /128
  set(ADSCRA,ADEN);    //enable ADC
  set(ADCSRA, ADATE); // auto trigger (ADSC auto sets to 1)

  for(;;){
    if (bit_is_set(ADCSRA, ADIF)) {// if flag is set (conversion complete) update
      set(ADCSRA, ADIF);    // reset the flag, see page 316

      m_usb_tx_string("  \rADC = ");  // \r is for a carriage return, ascii 13
      m_usb_tx_uint(ADC);    // print the ADC value from the register

      ~~set(ADCSRA, ADSC);    //start converting again~~
    }
  }
}
```

Step 1

Step 3

Step 2

Step 6

Step 8

Step 10

Step 9

Start step 7 over again

Step 4 not doing interrupts

Step 5 Channel 0 is default

Reading multiple channels

```
int main(void){
  m_usb_init(); //To initialize USB communication
  set(DDRF,1);  // set PF1 to be ground for voltage divider pot

  set(ADMUX, REFS0);   //voltage reference set to Vcc.
  set(DIDR0, ADC0D);   // disable di

  set(ADCSRA,ADPS0);  set(ADCSRA,ADP
  set(ADSCRA,ADEN);    //enable ADC
  set(ADCSRA, ADSC);   //start it converting

  for(;;){
    if (bit_is_set(ADCSRA, ADIF)) {/
      set(ADCSRA, ADIF);    // reset

      m_usb_tx_string("  \rADC = ")
      m_usb_tx_uint(ADC);    // prin

      set(ADCSRA, ADSC);    //start converting again
    }
  }
}
```

```
setDIDR(yourchannel); // part of Lab 3.1.2
```

```
clear(ADCSRA,ADEN); // stop ADC
if (ch==7) ch = 0;  // alternate ch 7 and 0
setChannel( ch );   // this is Lab 3.1.2
set(ADCSRA, ADEN);  // enable the ADC
```

Step 3 Disable Digital Input
DIDR0/DIDR2 Digital Input Disable Register

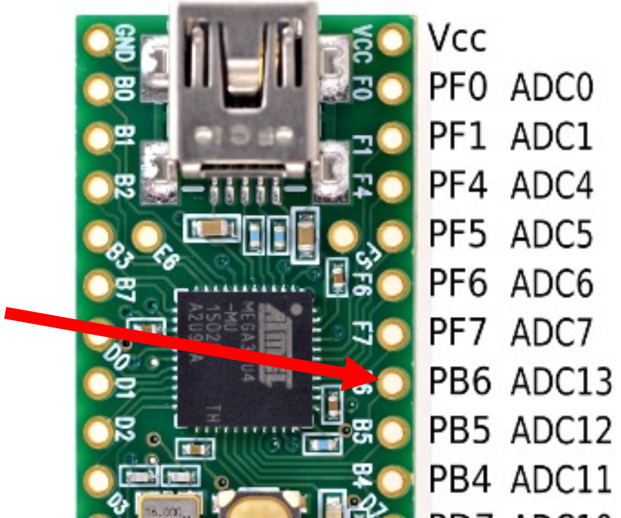
Bit	7	6	5	4	3	2	1	0	
	ADC7D	ADC6D	ADC5D	ADC4D	-	-	ADC1D	ADC0D	DIDR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	-	-	ADC13D	ADC12D	ADC11D	ADC10D	ADC9D	ADC8D	DIDR2
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	

ed. -

- **Bit 5:0 – ADC13D..ADC8D: ADC13:8 Digital Input Disable**

Q2: What command will disable the Digital Input for this pin?



ADC Multiplexing (Simplest free run)

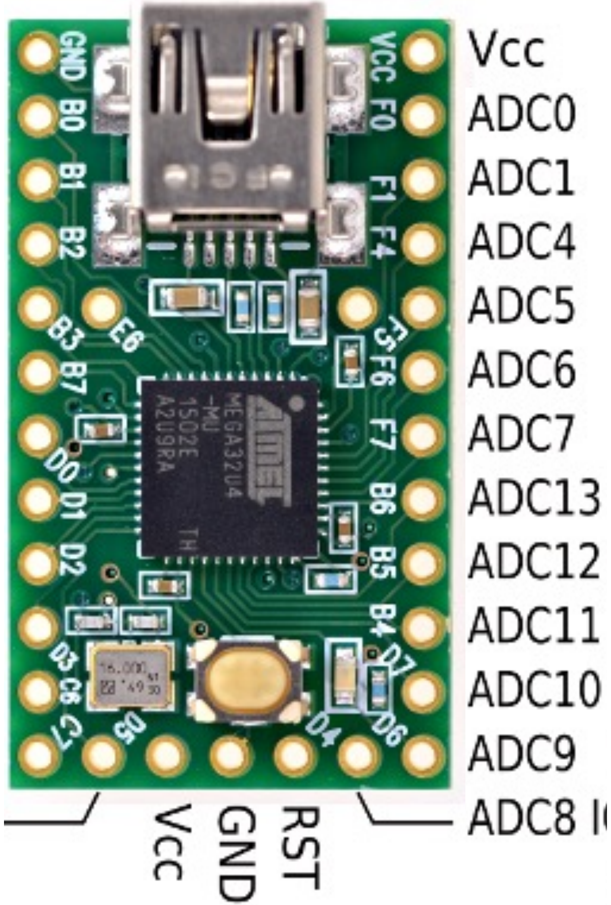
- Simplest using free run mode

  Setup

  1. set the voltage reference
  2. set the ADC clock prescaler
  3. disable some digital inputs
  4. set ADCSCRA:ADATE = 1 (free run)

  Loop at ~5Khz

  4. disable conversion (clear ADEN)
  5. select the analog channel (change to new one if desired ADMUX)
  6. enable conversions (set ADEN)
  7. wait for conversion to finish
  8. read the result for the channel selected



ADC Multiplexing (Faster)

- Faster than simple (about double speed)

  Setup

  1. set the voltage reference
  2. set the ADC clock prescaler
  3. disable some digital inputs
  4. select the analog input for 1st read

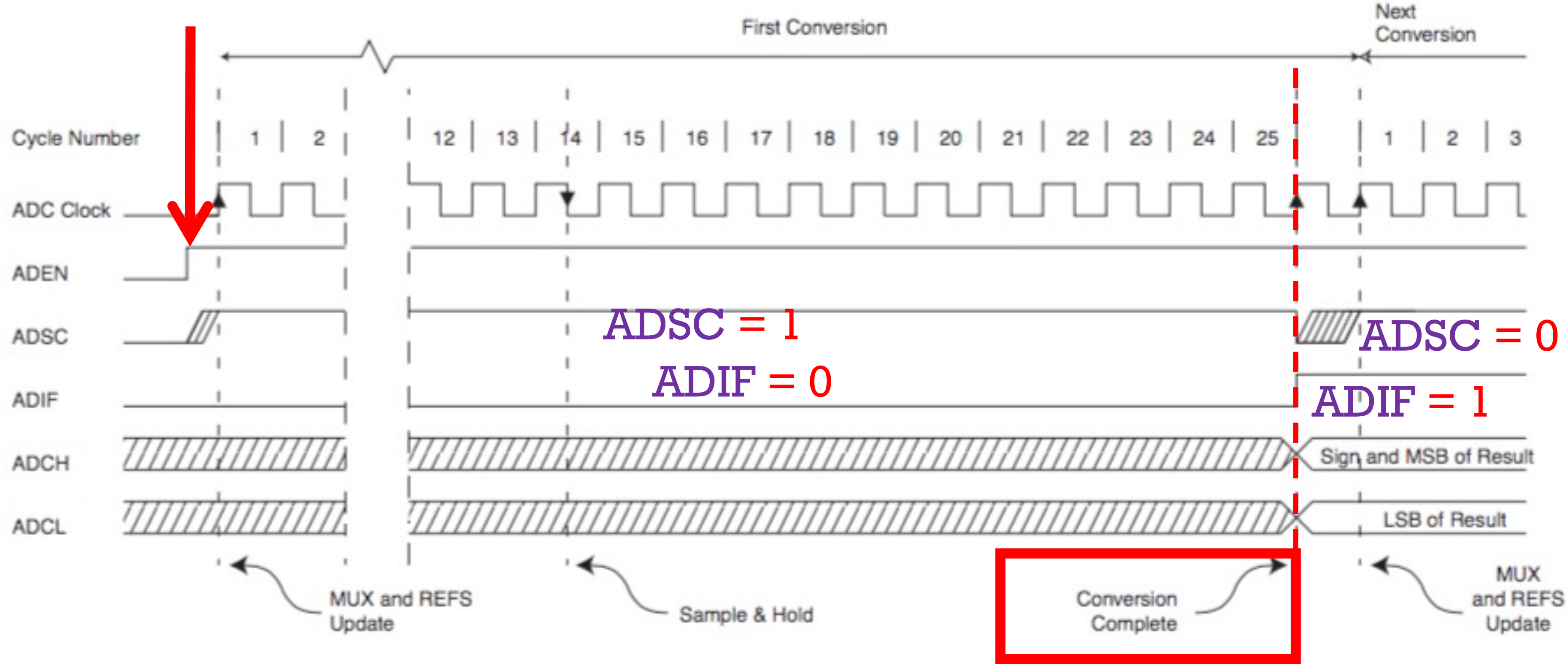
  Loop @ ~10Khz

  5. start conversion for nth read (set ADSC)
  4. select/change analog input for (nth+1) read
  6. wait for conversion to finish
  7. read nth result

see Atmel docs starting at p 300 for more info.

ADC single conversion timing diagram

Difference between (ADCSRA:ADSC) and (ADCSRA:ADIF) ?



Demo

- Single channel (`minimum_ADC.c`)

- Multichannel (`dual_ADC.c`)

03

Software Filters

Round 2

Moving Average Software Filter

```
int movingavg(int newvalue,int WEIGHT) {
  int update;
  static int lastvalue;
  update = (WEIGHT*lastvalue + newvalue)/(WEIGHT+1);
  lastvalue = update;
  return update;
}
```

Example input	6	7	4	6	5	100	4	5	5
Running avg (weight = 1)	3	5	4	5	5	52	28	16	10
Running avg (weight = 3)	2	3	3	3	3	27	21	17	14

Exponential Moving Average (EMA) [Infinite Impulse Response (IIR) filter]

$$y[n] = (1-\alpha)y[n-1] + \alpha x[n]$$

Example: `update = (1 - 0.3) lastvalue + 0.3 * newvalue;`

```
update = (WEIGHT*lastvalue + newvalue)/(WEIGHT+1);
```

$$f_{3dB} = \frac{f_s}{2\pi}\cos^{-1}\left[1 - \frac{\alpha^2}{2(1-\alpha)}\right]$$

f_{3dB} **is the corner (cutoff) frequency**

f_s **is the sampling frequency**

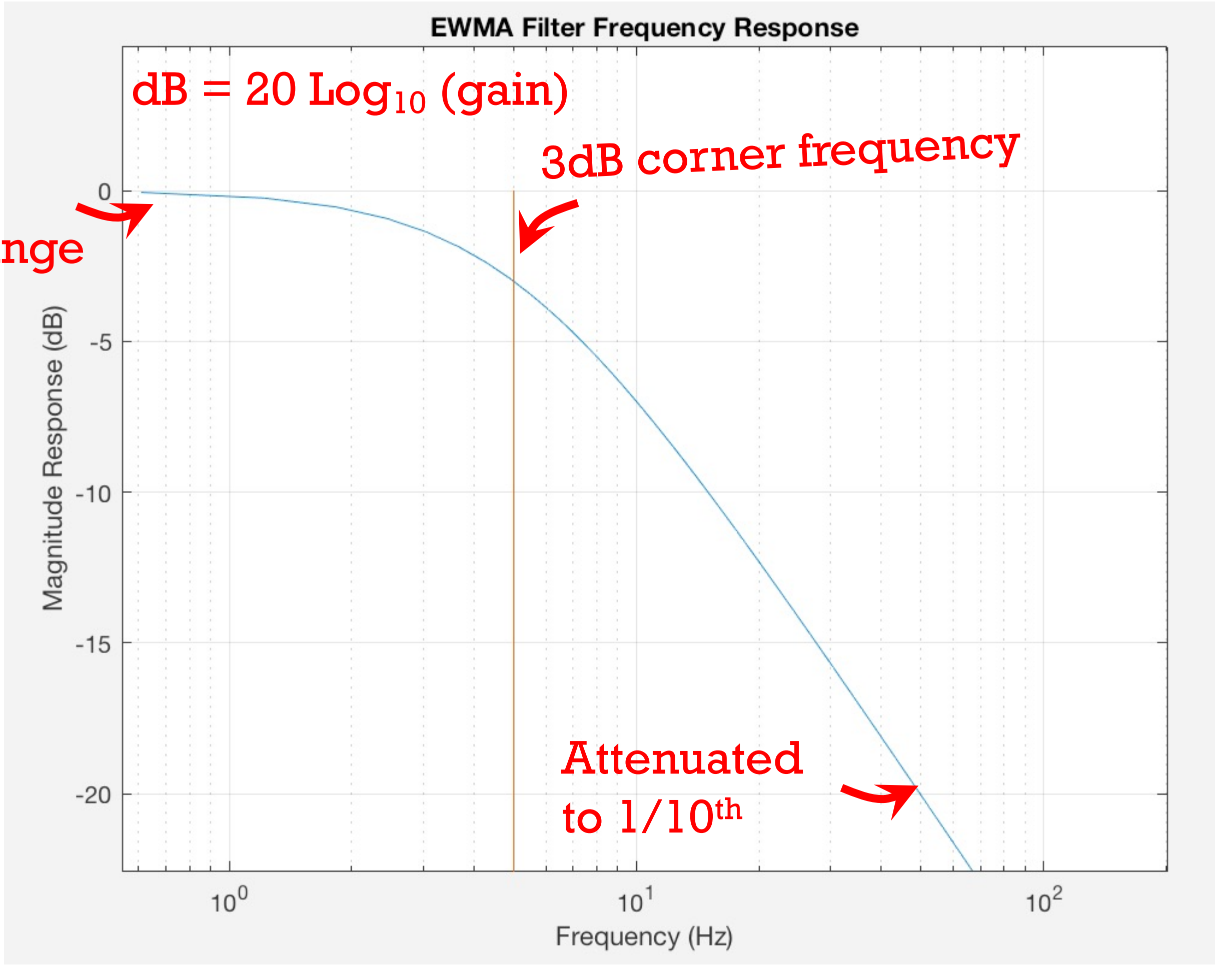
α **is** `1-(WEIGHT/(WEIGHT+1))` **in the previous code**

`WEIGHT` $= (1-\alpha)/\alpha$

https://dsp.stackexchange.com/questions/40462/exponential-moving-average-cut-off-frequency

EMA Example

Fs = 40000Hz
F3db = 5Hz
WEIGHT = 1273
(α = 7.85 e-4)



dB = 20 Log_{10} (gain)

3dB corner frequency

No change

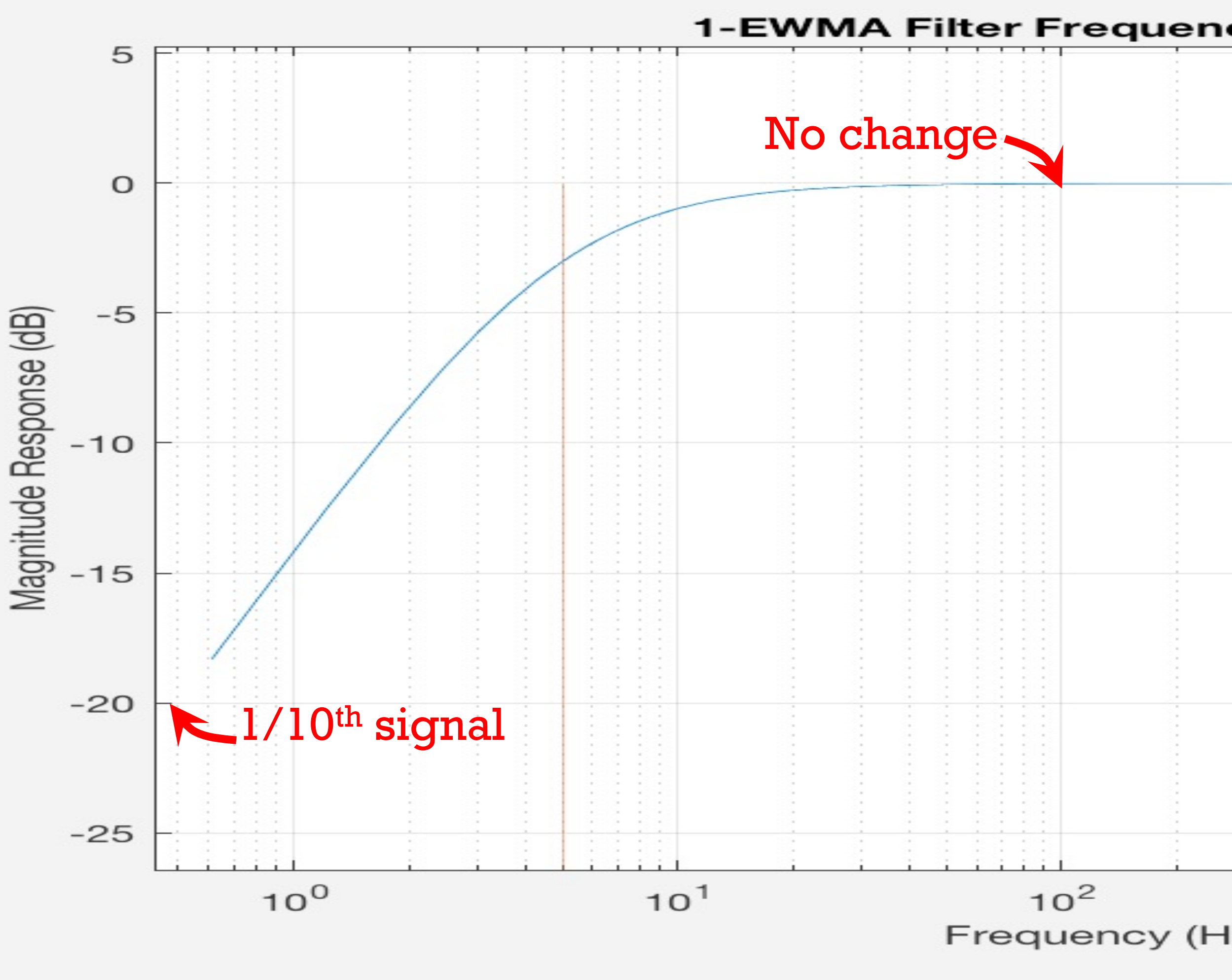
Attenuated to 1/10th

Log / Log plot

High Pass?

Fs = 40000Hz
F3db = 5Hz
WEIGHT = 1273
(α = 7.85 e-4)

Plot of 1 – EMA response



High Pass Filter (just subtract lowpass)

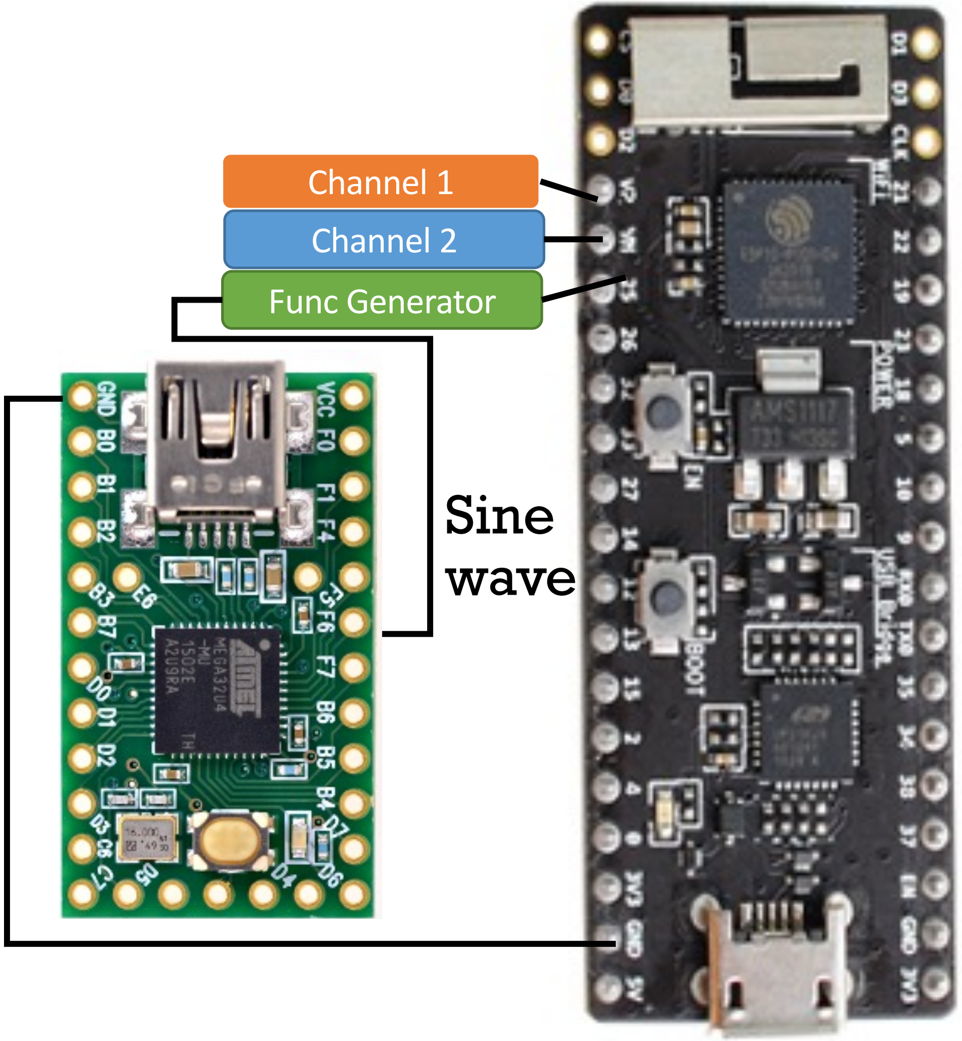
```
int highpass(int newvalue,int WEIGHT){
  static int lastvalue;

  lastvalue = WEIGHT*lastvalue + newvalue)/(WEIGHT+1);
  return newvalue - lastvalue;
}
```

Example input	6	7	4	6	5	100	4	5	5
Running avg (weight = 1)	3	5	4	5	5	52	26	15	10
Highpass	3	2	0	1	0	48	-22	-10	-5

https://www.norwegiancreations.com/2016/03/arduino-tutorial-simple-high-pass-band-pass-and-band-stop-filtering/

Demo

- Moving Average low pass filter
  - (movingAvg.c)
  - Changing weight
- Moving Average high pass filter
- Moving Average band pass filter
- Median Filter
  - (medFiltADC.c)



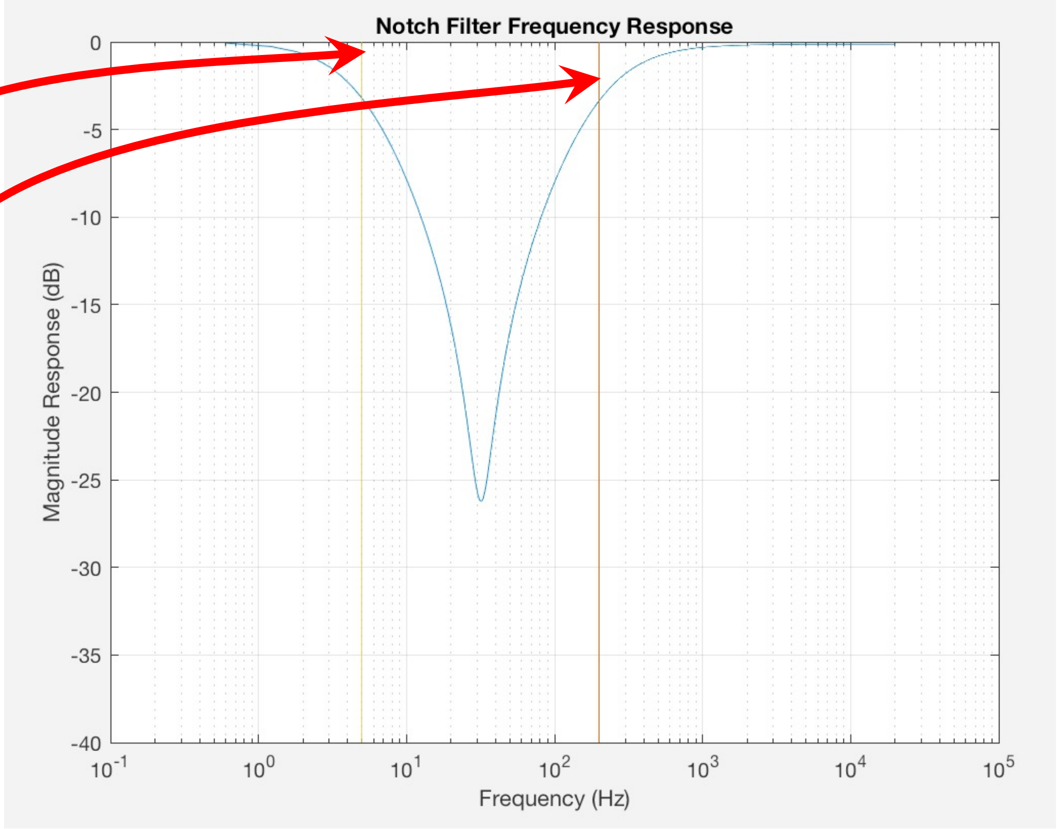
Band Stop AKA Notch Filter

Fs = 40000Hz

F3db1 = 5Hz
WEIGHT = 1273
(α = 7.85 e-4)

F3db1 = 200Hz
WEIGHT = 31
(α = 0.0309)

Plot of EMA2 – EMA1 response



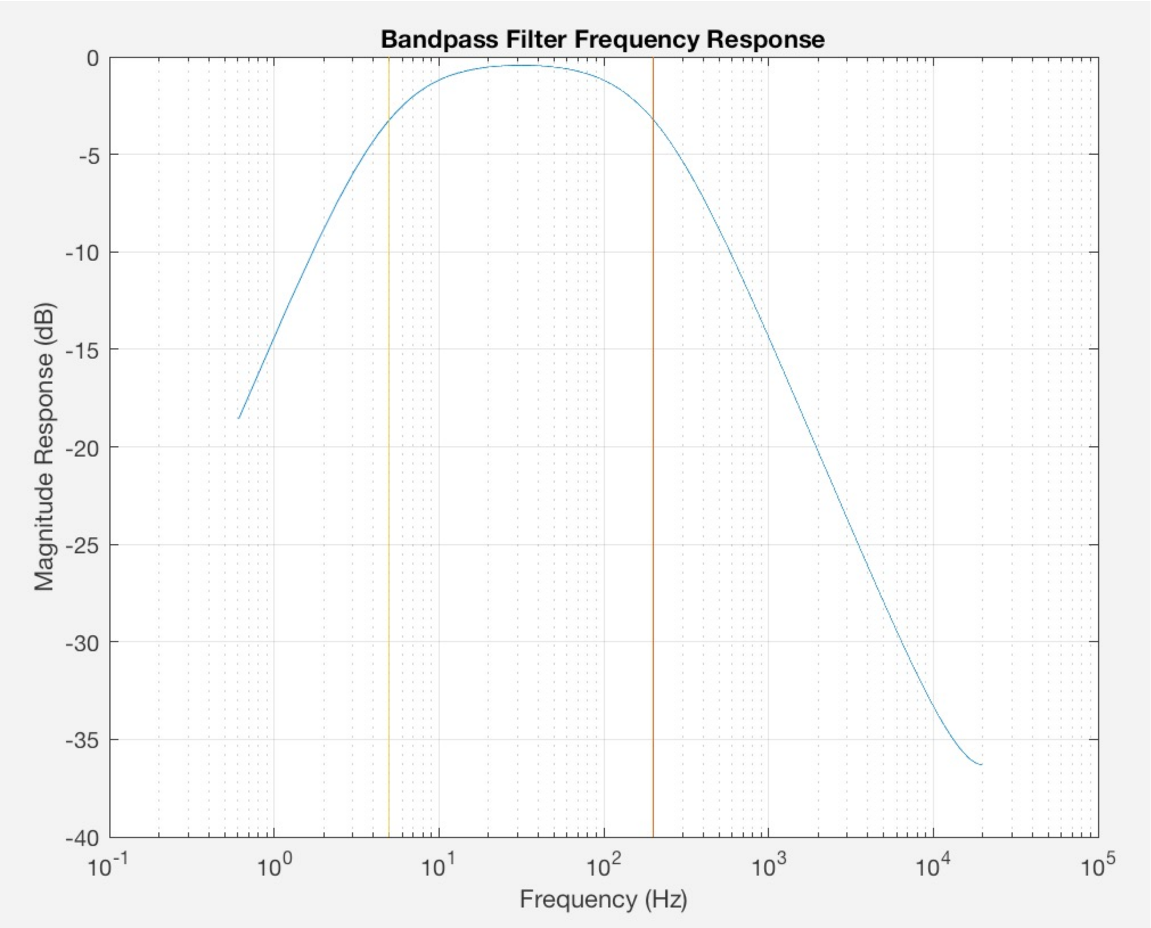
Band Pass?

Fs = 40000Hz

F3db1 = 5Hz
WEIGHT = 1273
(α = 7.85 e-4)

F3db1 = 200Hz
WEIGHT = 31
(α = 0.0309)

Plot of 1- Band stop



Designing for WEIGHT cutoff frequency

f_{3dB} is the corner (cutoff) frequency

f_s is the sampling frequency

$$\Omega = \frac{2\pi}{f_s} f_{3dB} = \frac{2\pi}{40KHz} 60Hz = 0.0094$$

$$\alpha = \cos(\Omega) - 1 + \sqrt{\cos^2(\Omega) - 4\cos(\Omega) + 3}$$

$$\texttt{WEIGHT} = (1 - \alpha)/\alpha = 105$$

f_s = 40,000Hz

Designing for `WEIGHT` cutoff frequency

f_{3dB} is the corner (cutoff) frequency
f_s is the sampling frequency

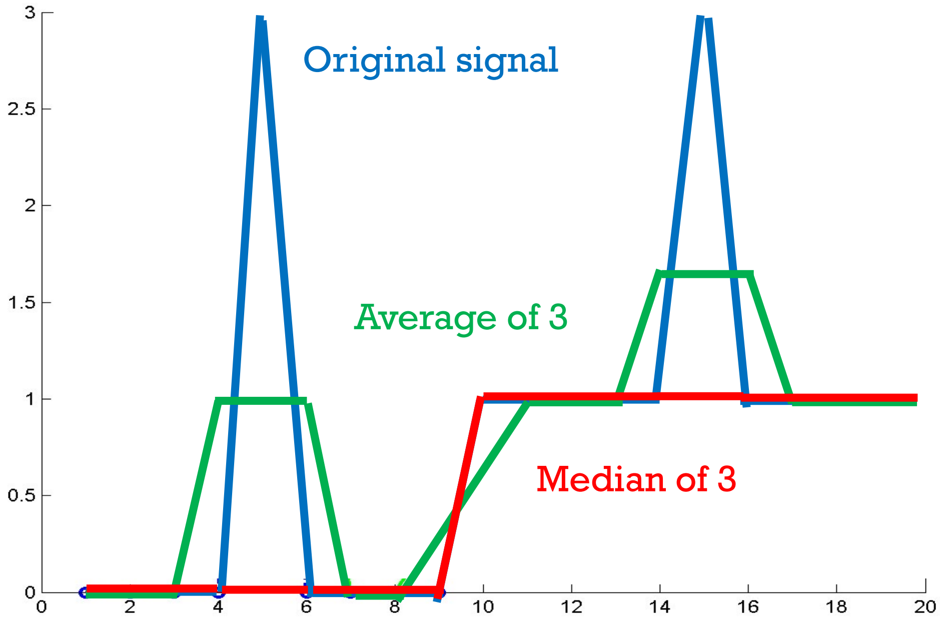
$$\Omega = \frac{2\pi}{f_s} f_{3dB} \quad = \frac{2\pi}{1KHz} 60Hz = 0.376$$

$$\alpha = \cos(\Omega) - 1 + \sqrt{\cos^2(\Omega) - 4\cos(\Omega) + 3}$$

$$\mathtt{WEIGHT} = (1-\alpha)/\alpha \quad \sim = 2$$

f_s = 1,000Hz

Median filter



Example of median of 3 values:

```
x = [2 80 6 3 ]
```

The median filtered output signal y will be:

```
y[1] =                        Median[2 2 80] = 2
y[2] = Median[2 80 6] = Median[2 6 80] = 6
y[3] = Median[80 6 3] = Median[3 6 80] = 6
y[4] = Median[6 3 3]  = Median[3 3 6]  = 3
i.e. y = [2 6 6 3].
```

Simplest Code Examples

```
int med3Filt(int a, int b, int c) {
  int middle;
  if ((a <= b) && (a <= c))
    middle = (b <= c) ? b : c;
  else if ((b <= a) && (b <= c))
    middle = (a <= c) ? a : c;
  else    middle = (a <= b) ? a : b;
  return middle;
}
```

```
int movingAvg(int newvalue, int WEIGHT){
  static int lastvalue;
  lastvalue = (WEIGHT*lastvalue + newvalue)/(WEIGHT+1);
  return lastvalue;
}
```

04

Servos and Teensy

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 output to clear at rollover TCCR3A register : COM3Ax bits
  - Set timer prescaler to get 61Hz TCCR3B register: CS3x bits

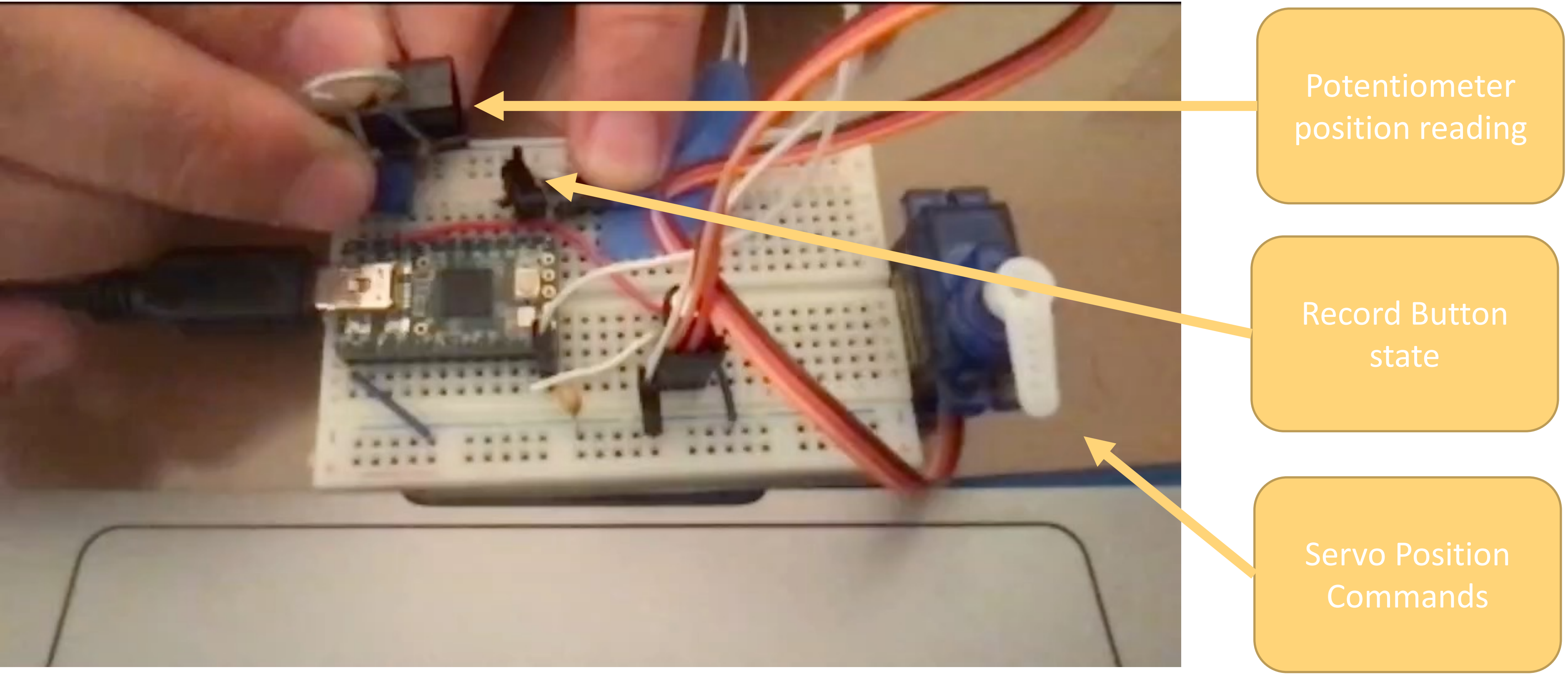
Q2: What commands will clear at rollover with mode 5?

- 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

Servo Loop Demo using Output Compare ISR



Potentiometer position reading

Record Button state

Servo Position Commands

Servo Output Compare Demo

```
ISR(TIMER3_COMPA_vect) {
  static int i=0, j=0;

  if (recording) {
    b[(step++)%endtime] = servoNow;
    OCR3A = servoNow;  // command servo duty
  }
  else {
    int tmp = b[(step++)%endtime] +
                      servoNow - 100;
    if (tmp < 2) tmp = 2;
    OCR3A = tmp; // command servo duty
  }
}
```

```
int main(void){
  initADC();
  initServo();  // interrupt on TCNT3=OCR3A
  initButton();
  set (DDRD,7);  // for debugging
  sei();         // enable all interrupts

  for(;;) {
    if (ADCready()) {
      potPosition = map(filter(ADC));
      restartADC();
      toggle(PORTD,7);  // for debugging
    }
    recording = bit_is_clear(PINC,7);
    if (recording) { teensy_led(ON); }
    else { teensy_led(OFF); }
  }
}
```

Q3: What commands will enable interrupts for the output compare A for timer3?

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

05

Other Questions

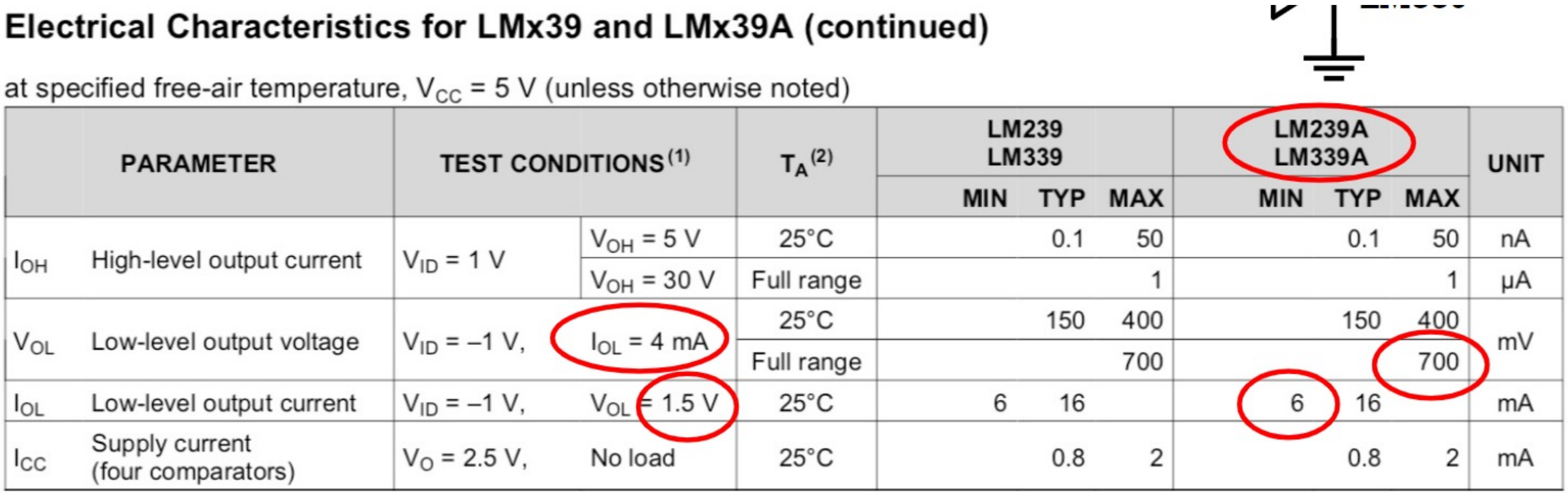
Filters:

- When to use an inductive filter?
  - [RL similar to RC but N/A for this class – we don't have inductors in ministore]
- Which R and C to choose for RC filter?
- How can you tell when notch or physical filters are working and why would you use a hardware filter over a software filter?
- I did notice that with long cables on my breadboard, my finger would cause some interference when I touched the long cables. Would a filter solve this?
- Do we need to filter out 60 Hz noise even though our IR phototransistors are not that sensitive to the overhead lights?

Servos:

- What Teensy commands are we supposed to use to be able to set the position of the servo? [answered in previous slides]
- How can we determine the step and speed at which we change the duty cycle to achieve a very smooth and fine movement on our servo? Besides just trial and error with the PWM signal?

Comparators



Electrical Characteristics for LMx39 and LMx39A (continued)

at specified free-air temperature, V_{CC} = 5 V (unless otherwise noted)

PARAMETER		TEST CONDITIONS[1]		T_A[2]	LM239 LM339			LM239A LM339A			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
I_{OH}	High-level output current	V_{ID} = 1 V	V_{OH} = 5 V	25°C		0.1	50		0.1	50	nA
			V_{OH} = 30 V	Full range			1			1	μA
V_{OL}	Low-level output voltage	V_{ID} = –1 V,	I_{OL} = 4 mA	25°C		150	400		150	400	mV
				Full range			700			700	
I_{OL}	Low-level output current	V_{ID} = –1 V,	V_{OL} = 1.5 V	25°C	6	16		6	16		mA
I_{CC}	Supply current (four comparators)	V_O = 2.5 V,	No load	25°C		0.8	2		0.8	2	mA

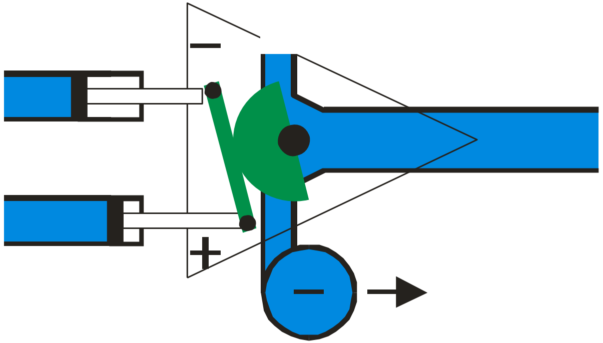
- Why did we choose to see the low-level output current and not the high-level output current? I want to understand how to select what parameters matter.

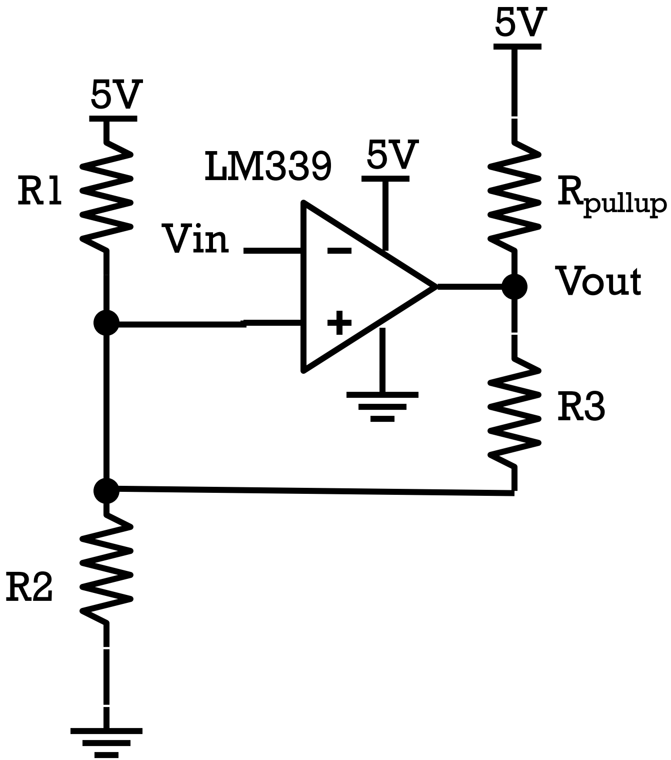
- What happens if the value of the positive feedback resistor in comparator is very high?

Comparator / Pullup Resistors

Need for Pull up resistor.

Open collector output





- What happens if the value of the positive feedback resistor in comparator is very high? [smaller hysteresis]

Pointers

- What are some situations where pointers are better to use compared to arrays?

Arrays can only access elements in the array.

Pointers can point to any address dynamically.

```
int array1[10], array2[20];
int *pointer;
pointer = array1;  // assign pointer
pointer = array2+3; // point to 4th element in array2
array1 = array2+3; // generates an error
```

https://www.geeksforgeeks.org/difference-pointer-array-c/

Other Questions?

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. Software filters

B. ADC on Teensy

C. Would it have been better to have a normal lecture than this review? (1=Lecture better, 4=Review better)