Lecture 05

Debouncing Switches, Photosensors and Input Capture

Agenda

00. Misc Lab 1 Issues and OscilloSorta demo
01. Capacitor Transient Behavior and Filters
02. Switches
03. Timer Input Capture
04. Photosensors
05. Phototransistors (CdS cell if enough time)

Lab 1 issues:

- Ideal submission format:
  - One pdf file, with links to youtube of vimeo videos. Shared files are okay if you properly share (many people had problems with this)
- `teensy_wait(variable);` doesn't work. Calls a builtin delay function that takes only constants.
- Programming using `floats` and `ints`.

What does `f` evaluate to?

```
int i;
float f;

i = 10;
f = i/40;
```

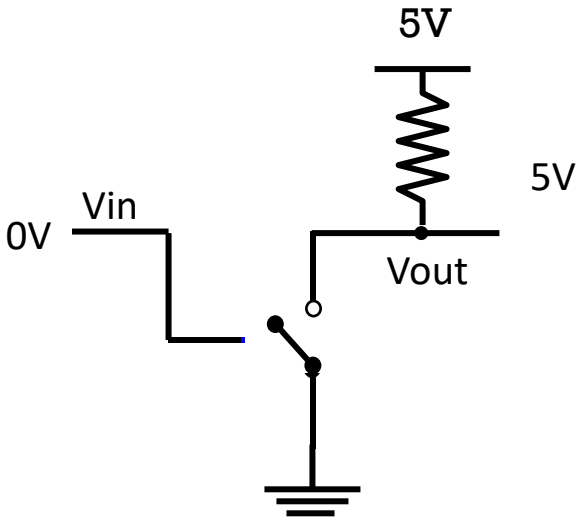
```
int i;
float f;

i = 10;
f = i/40.0;
```

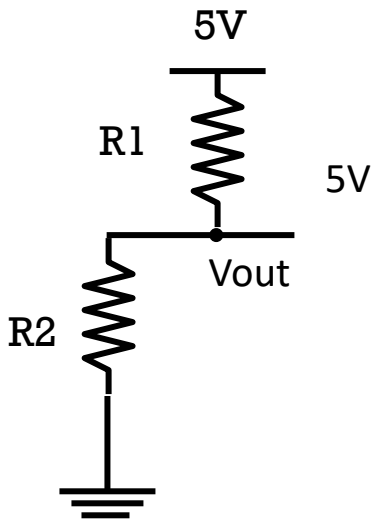
```
int i;
float f;

i = 10;
f = (float)i/40;
```

Promotion happens at evaluation

Pullup Logic as Voltage Divider

- When to use pullups? Only when your logic output drivers don't drive both high and low.





$$Vout = 5V \frac{R2}{R1+R2}$$

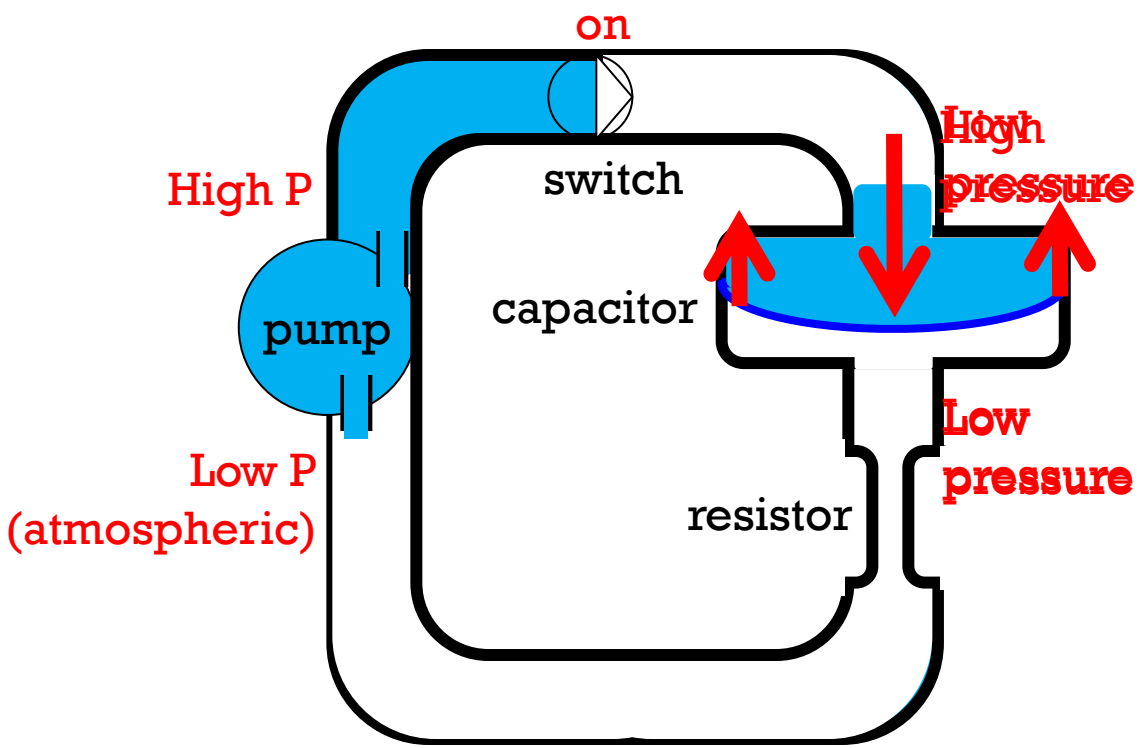
R2 has two states:
0 Ω and infinity Ω

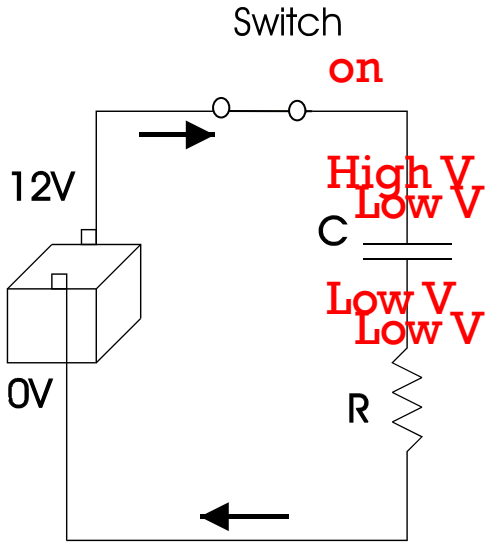
Oscillosorta 1.2 Demo:

- Lab 2 use OscilloSorta function generator for 3.3V Logic Level square wave
- Demo on smartphone:
  - Access point SSID: "Oscillosorta"
  - Website: 192.168.1.2
  - View function generator (with 25 to VP) on scope

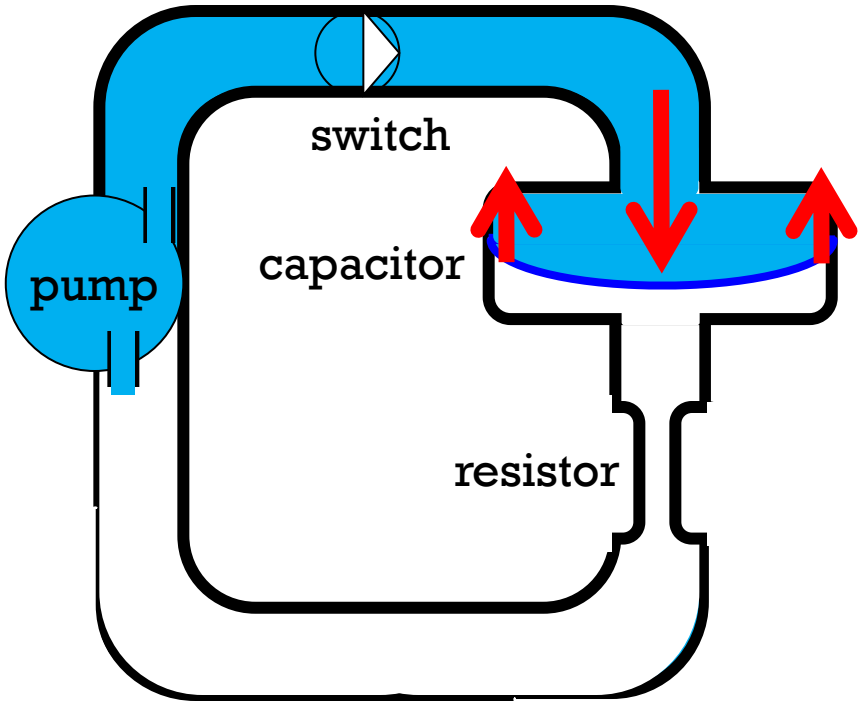

01
Capacitor Transient Behavior
85°C
SK
85°C
SK
85

Capacitor Transient Behavior





At Steady State, Full Capacitor



Membrane is fully stretched

Force on membrane up equals force from pump pressure down

Stops flow

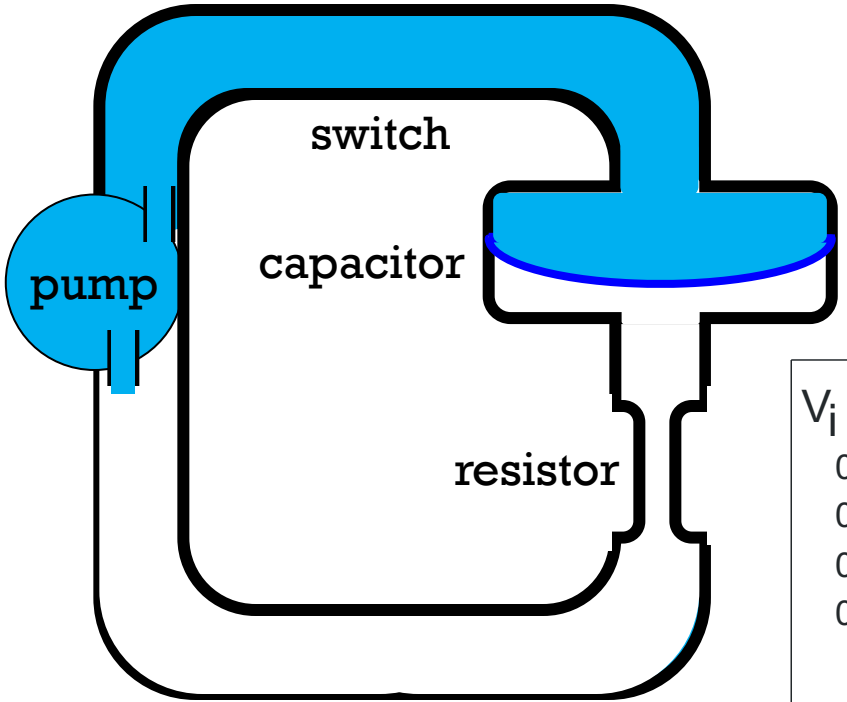
No water crosses membrane

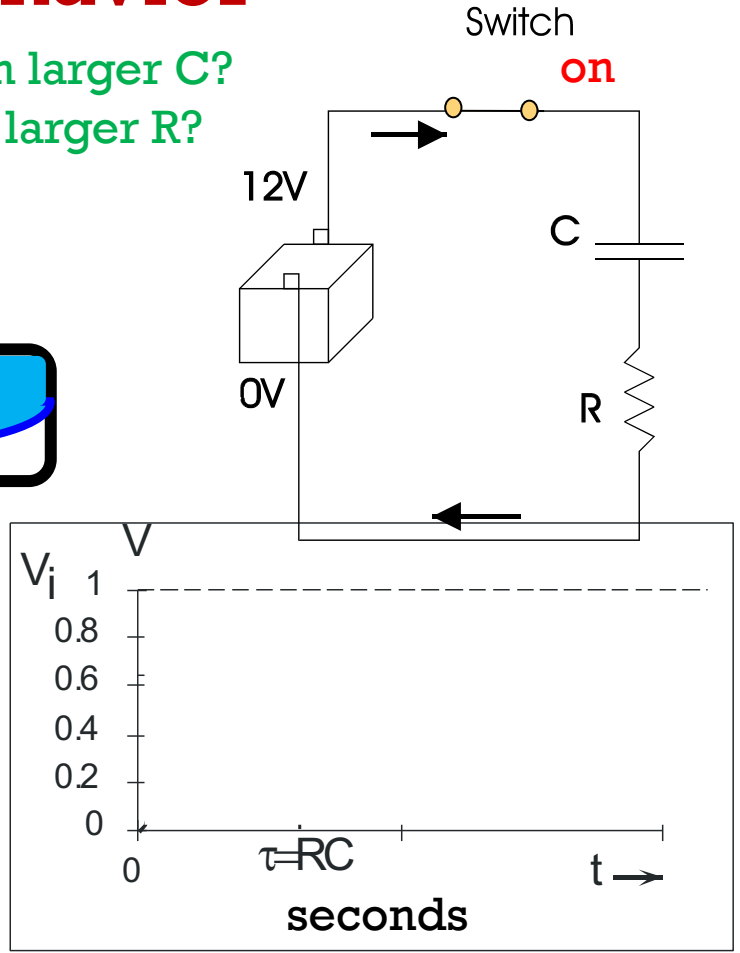
No electrons cross gap of capacitor

Capacitor Transient Behavior

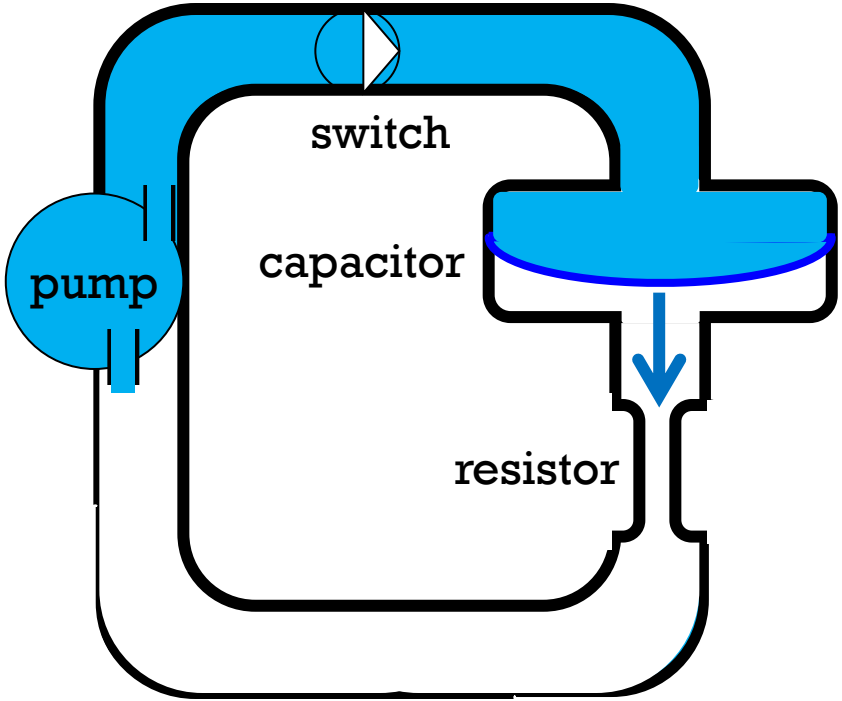
Q1: Draw & hold what happens to graph with larger C?

Q2: Draw&hold what happens to graph with larger R?





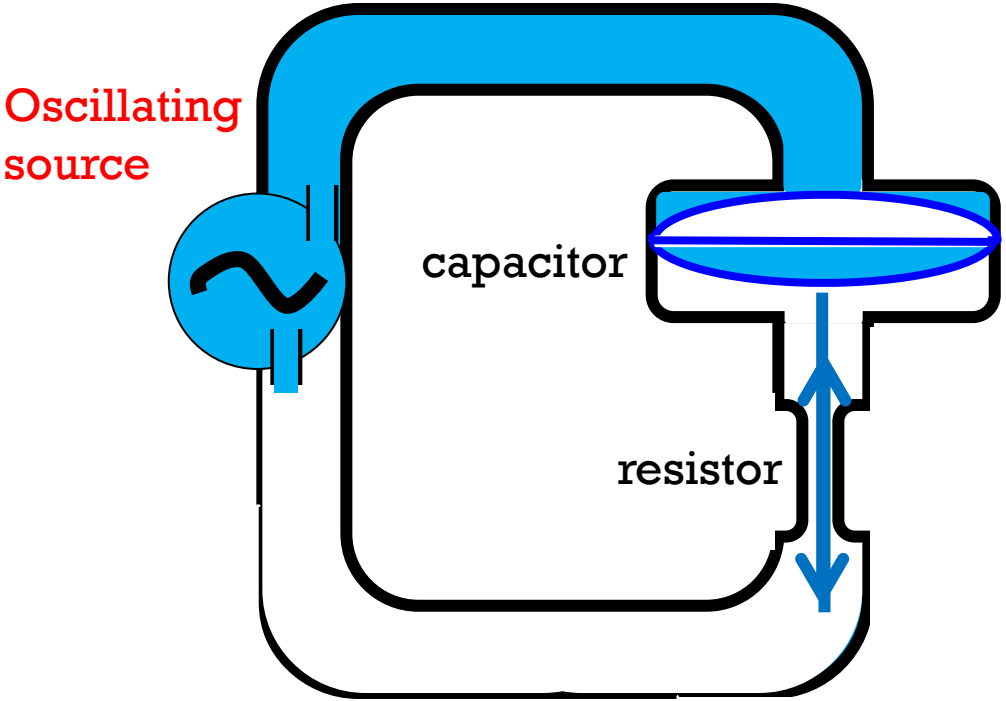
Flow "through" capacitor



The initial motion of membrane pushes flow out of capacitor. How much depends on size of capacitor.

But it stops flow in steady state.

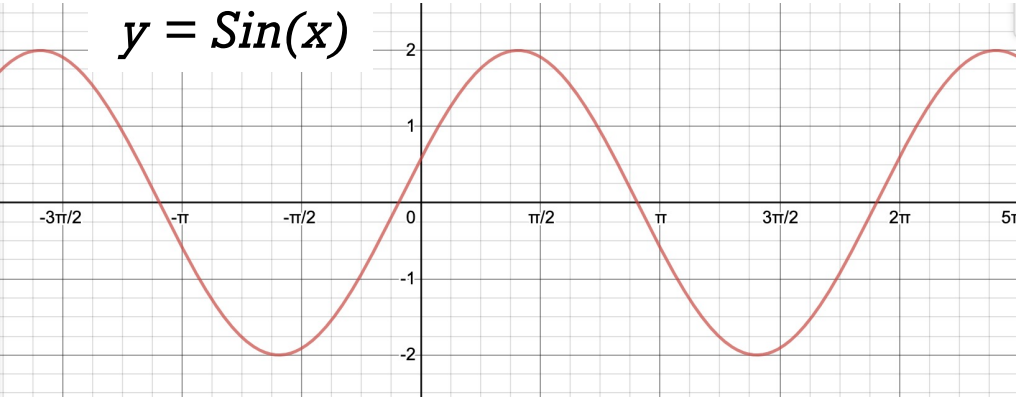
Capacitor with oscillating voltage source

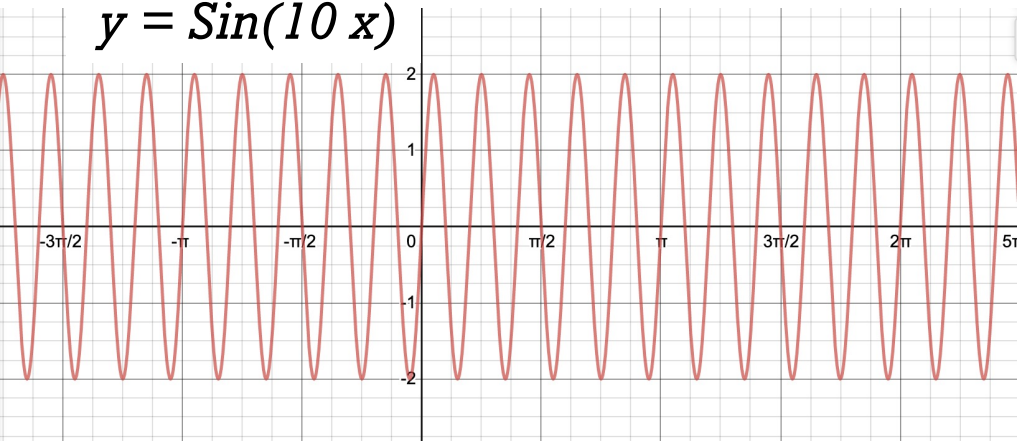


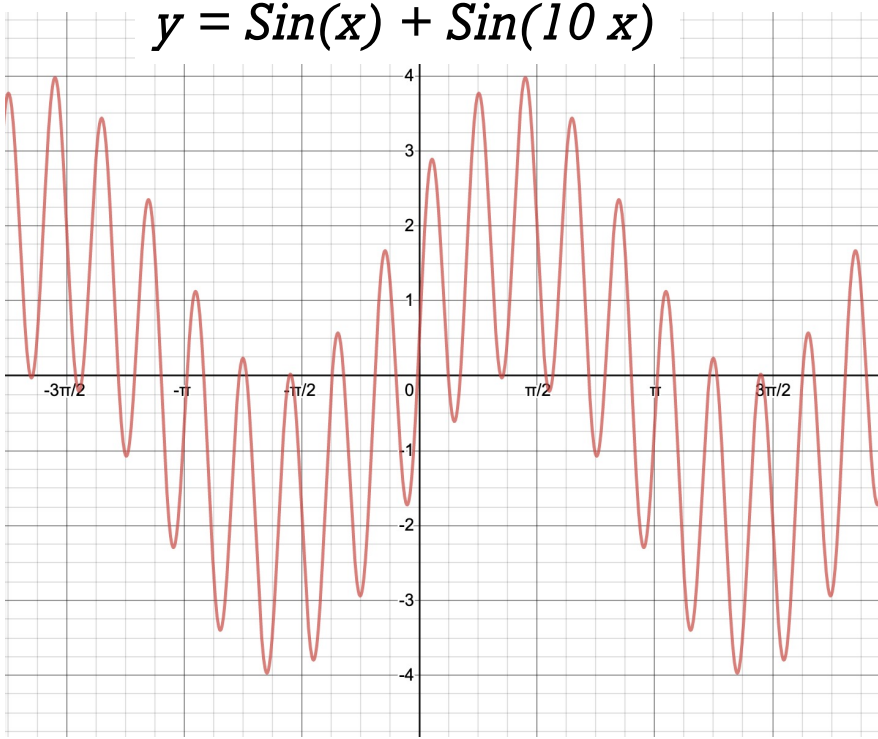
Capacitance C is like thinness of membrane with same sized chamber

Resistance slows down flow

Combining oscillating signals

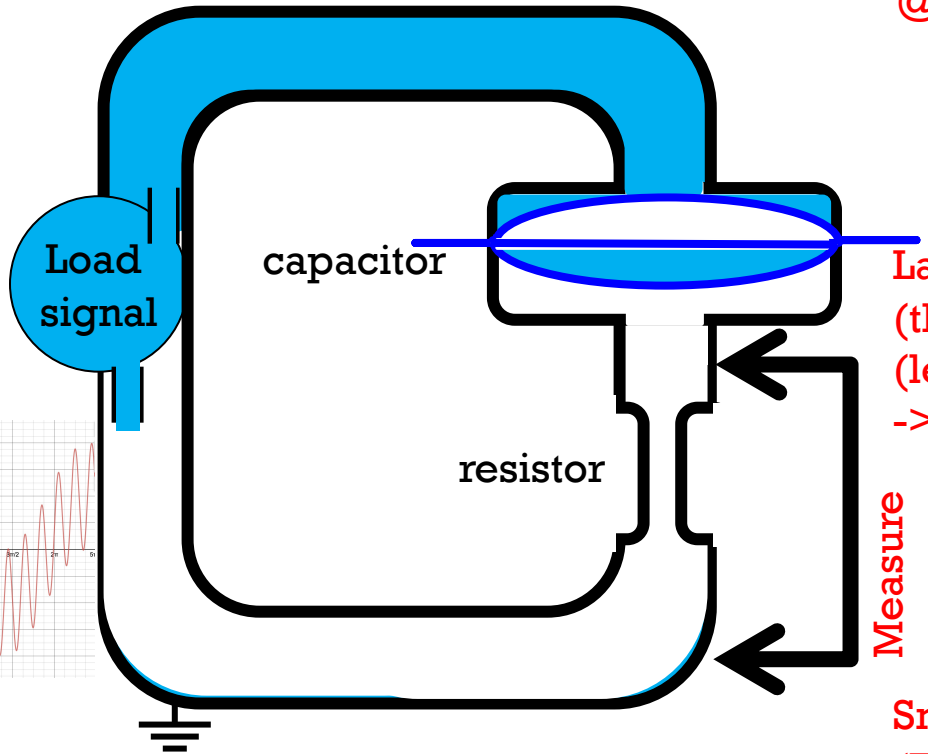






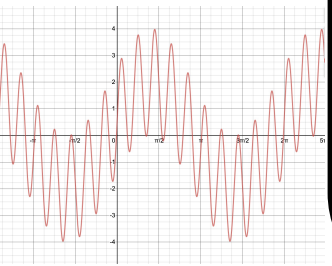
Capacitor in a filter

@ steady state: stops flow
(Large voltage drop across cap)
@ high freq: low resistance on C
(flow across cap as if it isn't there)

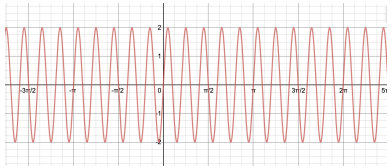


High pass filter!

Larger C,
(thinner membrane)
(less pressure for given motion)
->Slower time constant

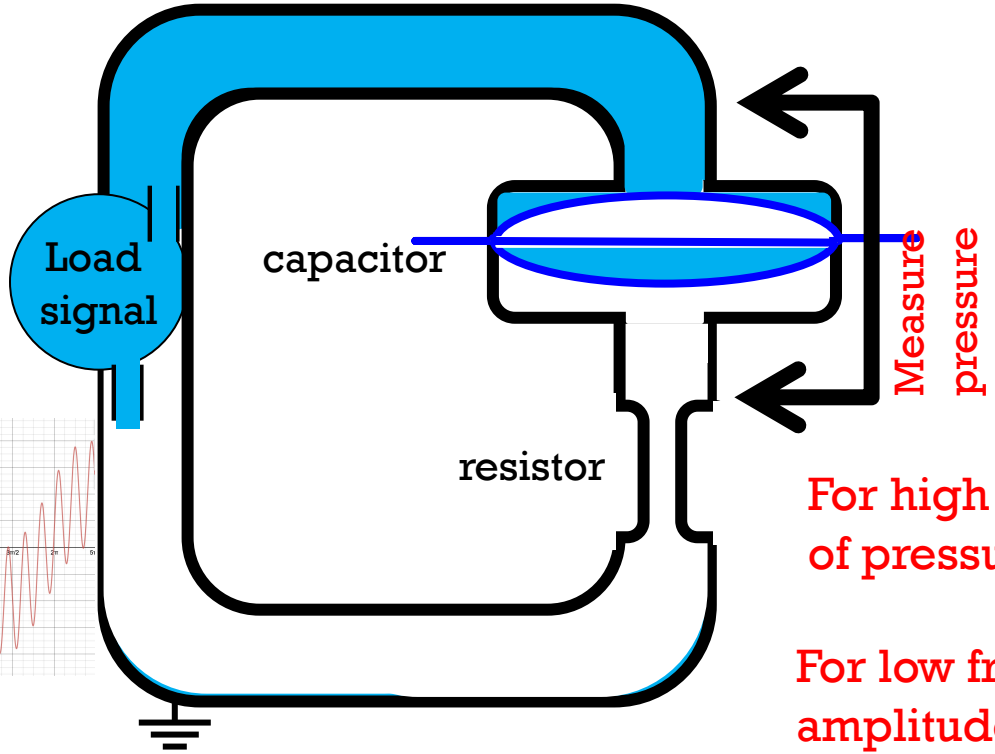


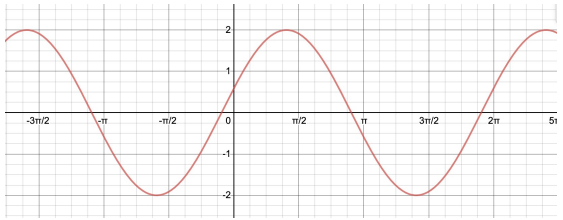
Measure pressure

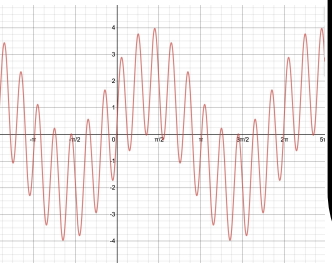


Smaller R,
(Easier to flow)
(allows more motion in C)
Faster time constant

Another filter configuration



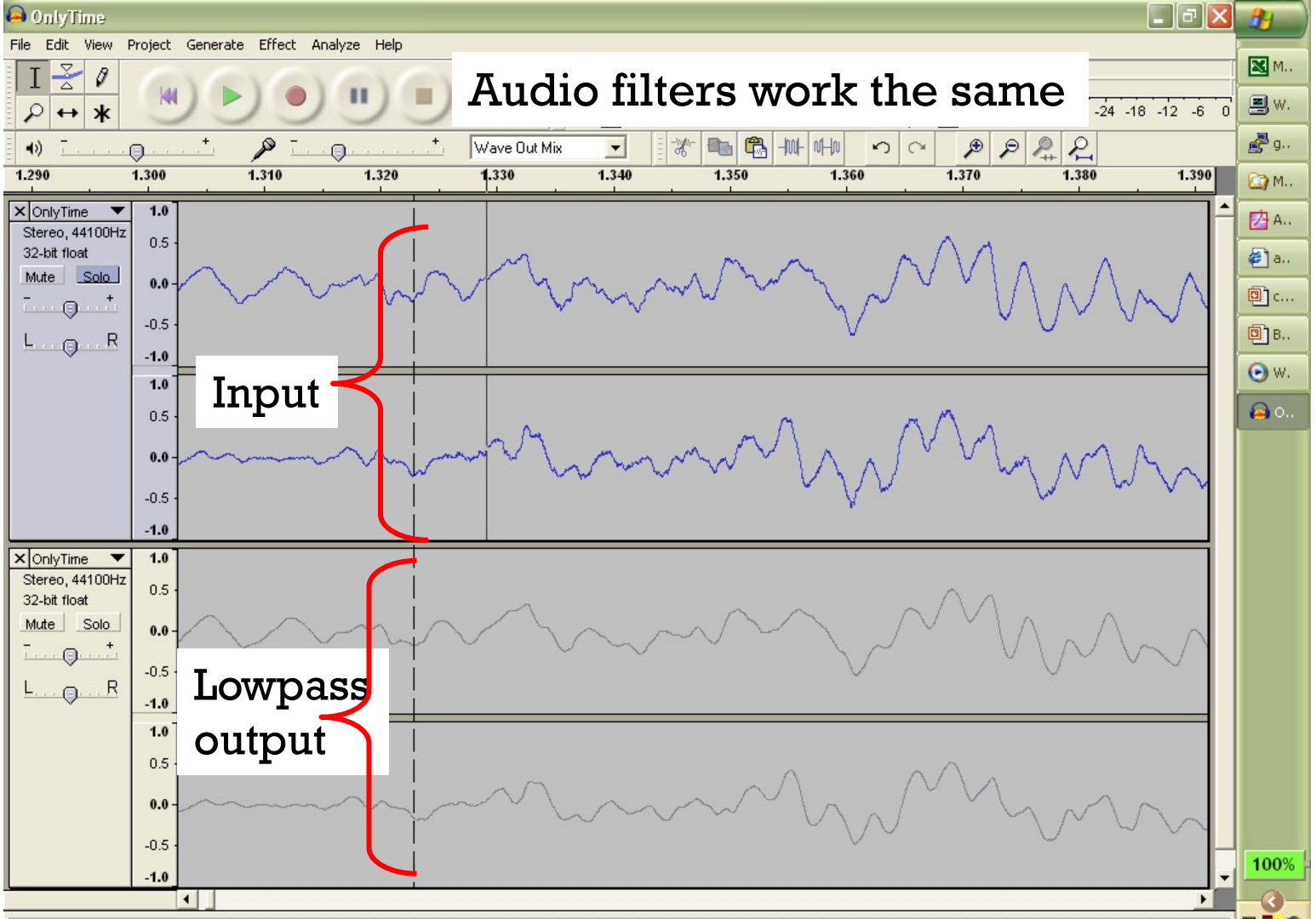




For high freq, what does the amplitude of pressure across capacitor appear?

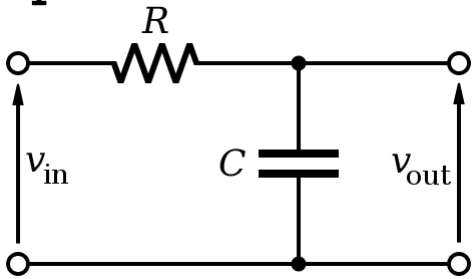
For low freq, what does the amplitude look like?

Low Pass filter!


OnlyTime
File Edit View Project Generate Effect Analyze Help
Audio filters work the same
Wave Out Mix
1.290 1.300 1.310 1.320 1.330 1.340 1.350 1.360 1.370 1.380 1.390
OnlyTime
Stereo, 44100Hz
32-bit float
Mute Solo
L R
1.0 0.5 0.0 -0.5 -1.0
Input
OnlyTime
Stereo, 44100Hz
32-bit float
Mute Solo
L R
Lowpass output
100%

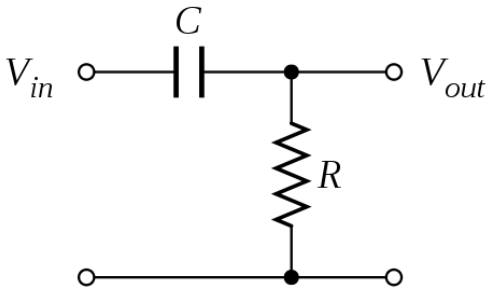
Capacitors in filters

- Two main configurations
  - Capacitors in parallel with source



Low Pass filter

  - Capacitors in series with source

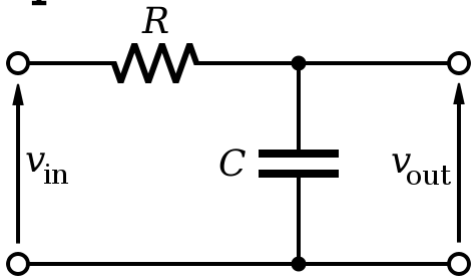


High Pass filter

Images from wikipedia

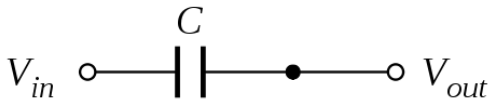
Capacitors in filters

- Two main configurations
  - Capacitors in parallel with source



Low Pass filter

  - Capacitors in series with source

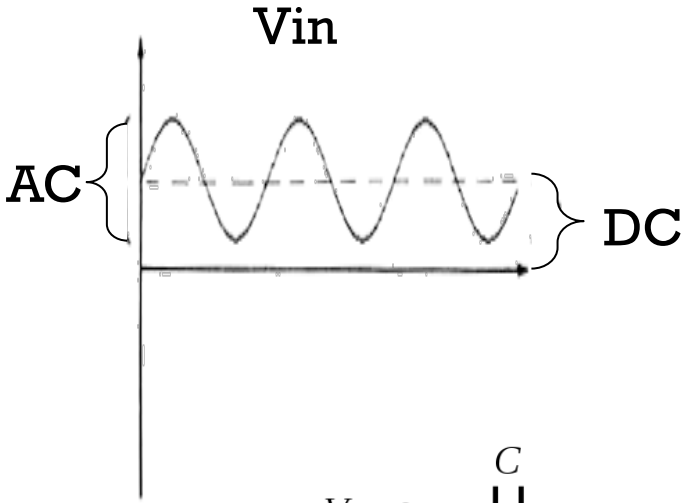


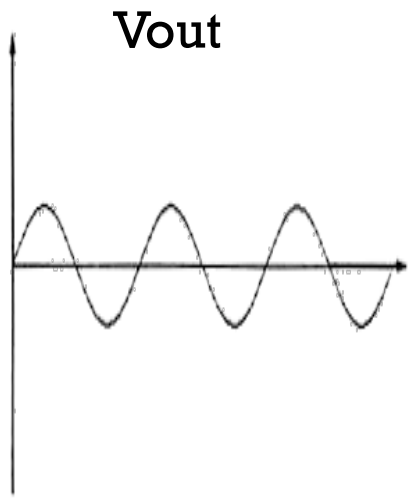
AC coupling

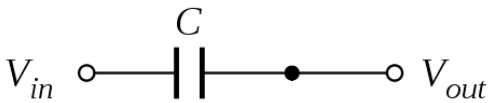
Images from wikipedia

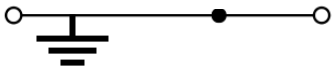
AC coupling

- Removes DC component.

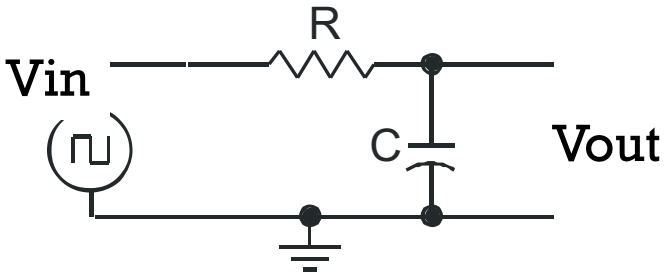








Filter effect on square wave

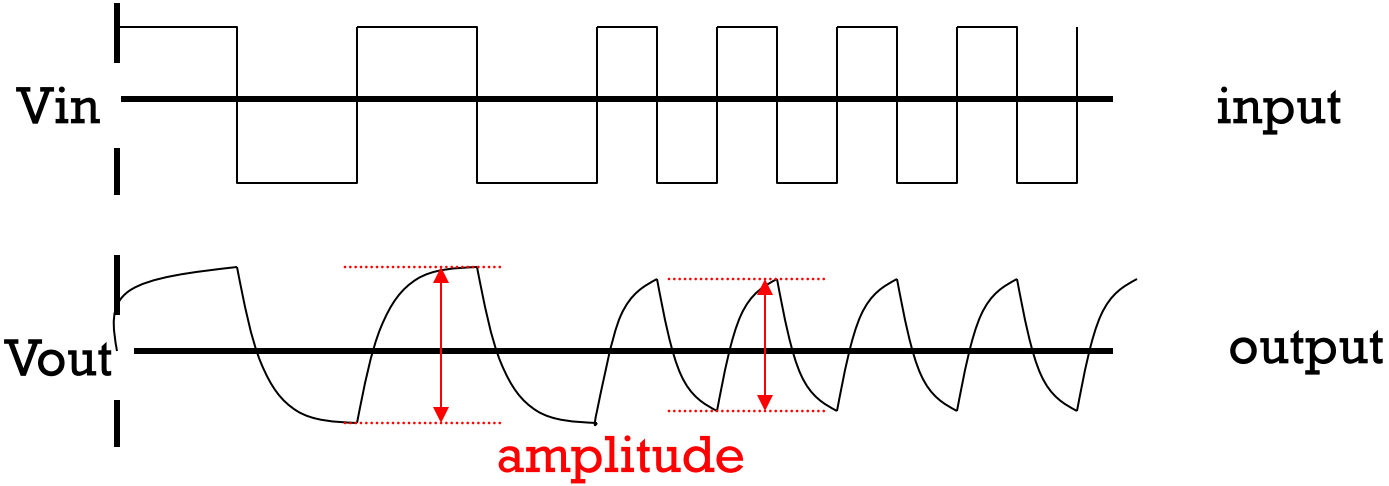


Low pass filter

Vout = Vin / $\sqrt{(1+\omega^2R^2C^2)}$

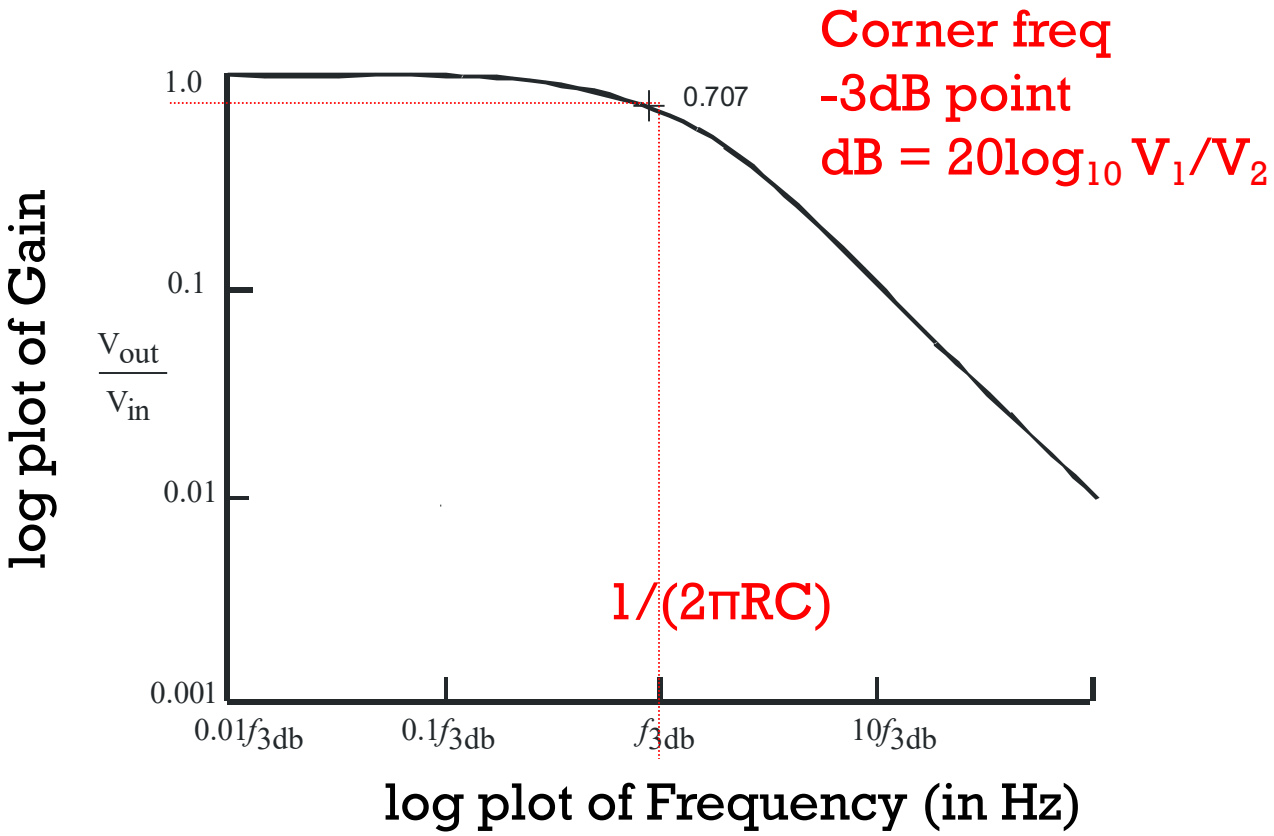
(for sinusoid)

Vout/Vin is the ***gain***



High frequencies are attenuated

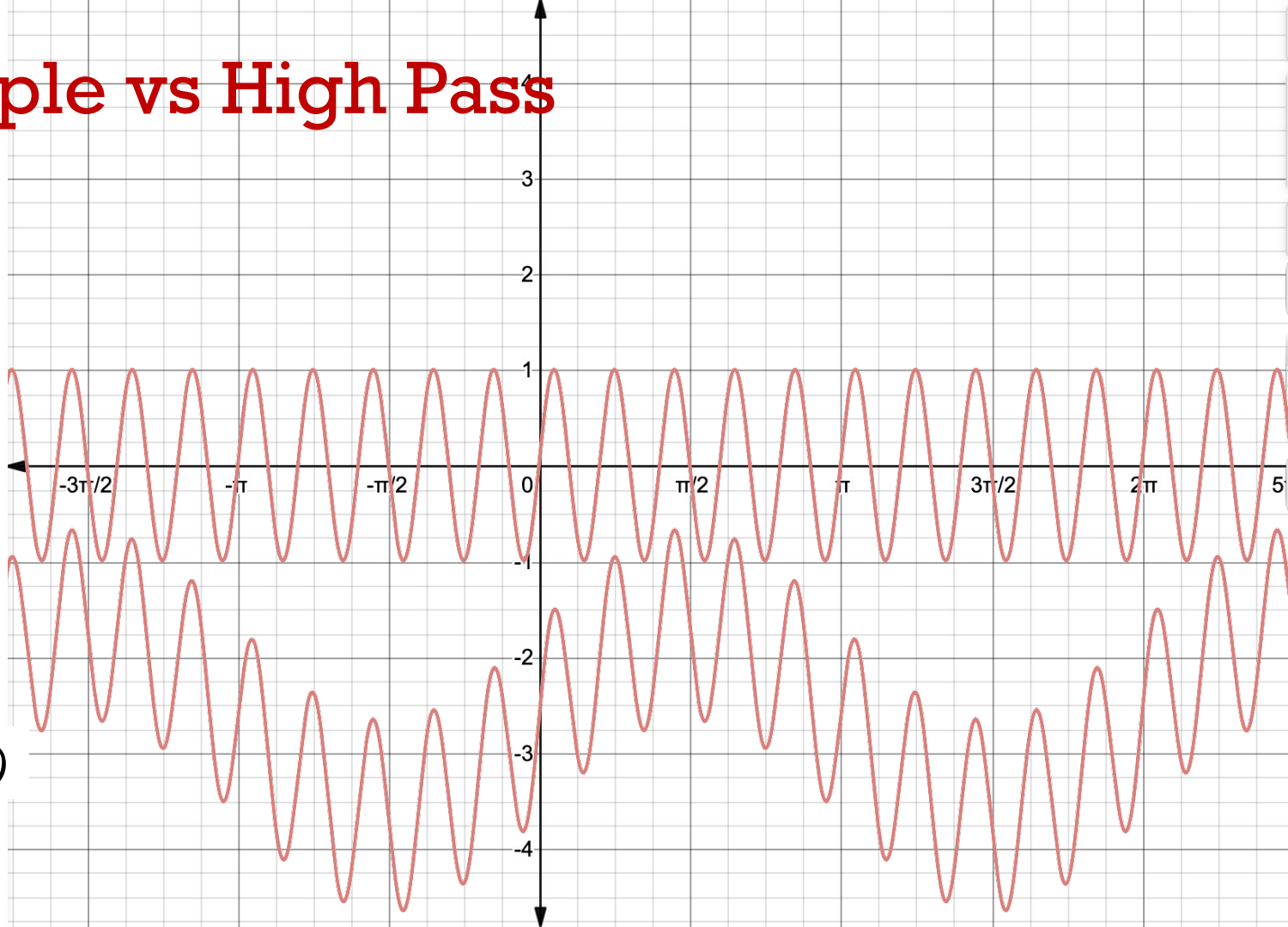
Frequency Domain of Low Pass Gain



AC couple vs High Pass

Sin(10 x)

Sin(x) + Sin(10 x)



02

Switches

Switch Terms

Poles — # of circuits

Throws — # of possible circuits

Single pole (SP)
Double pole (DP)
Multipole (MP)

Single throw (ST)
Double throw (DT)
Multithrow (MT)

NC — Normally closed

NO — Normally open

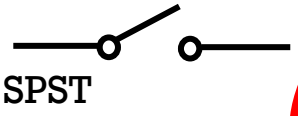
Momentary — Switch holds state while held

Toggle — Two stable positions

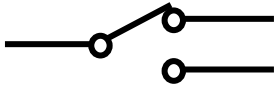
Slide / Rocker / Push button — Type of action

Switch Types

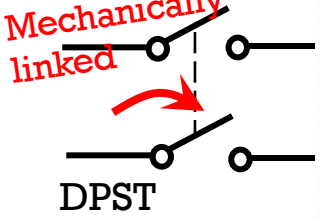
Available in GM lab



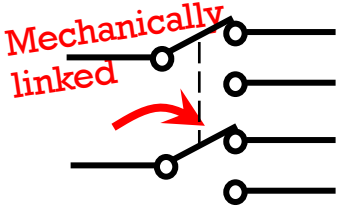
SPST



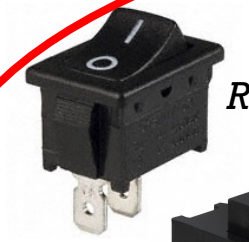
SPDT



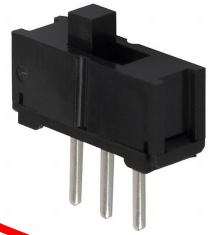
DPST



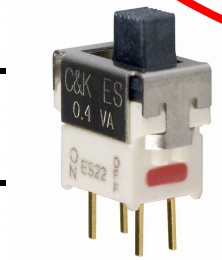
DPDT

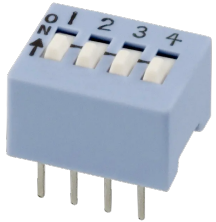


Rocker



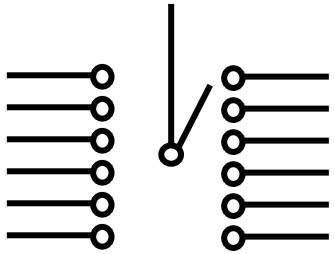
Slide





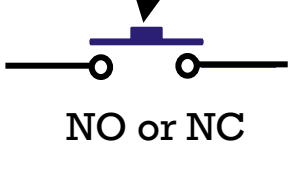
DIP switch

Toggle



SPMT (SP12T)

MPMT



Momentary

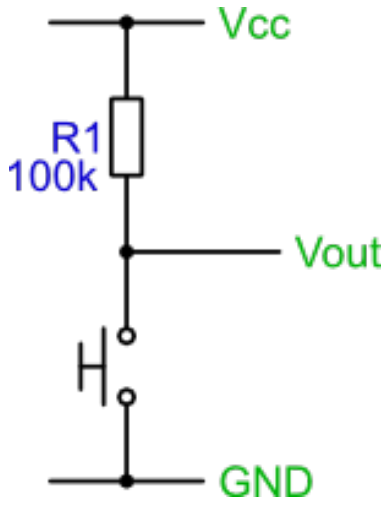
NO or NC

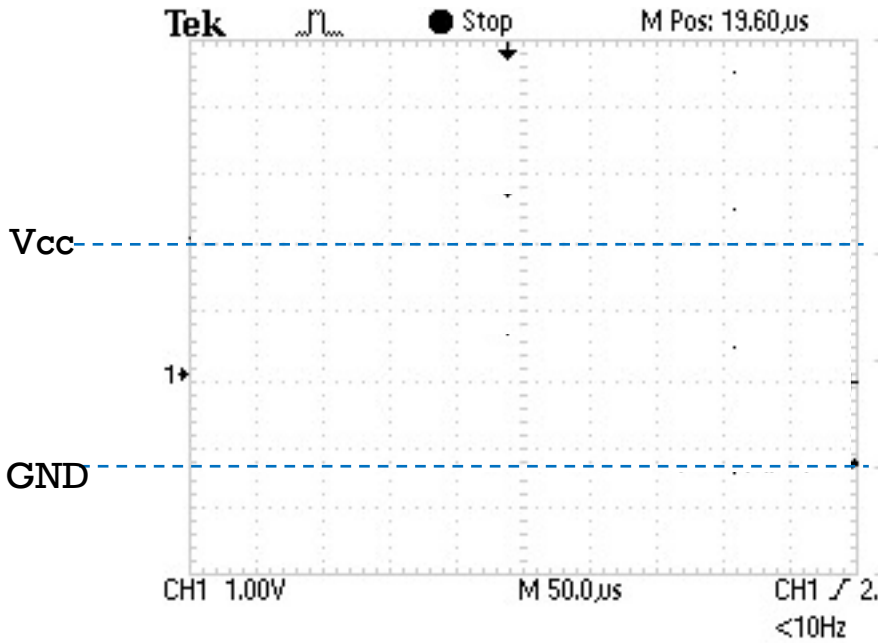
Limit switch

Rotary

Switch bouncing

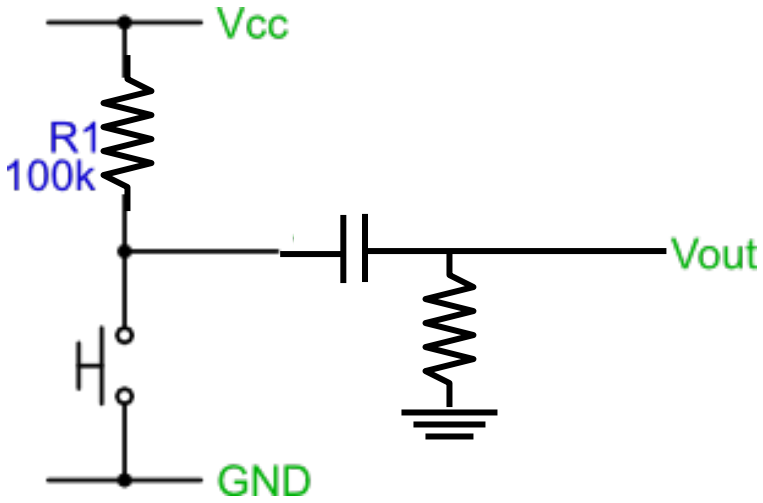
Q3 Draw and hold what the scope will show as the button is pushed down if scope is rising edge trigerred



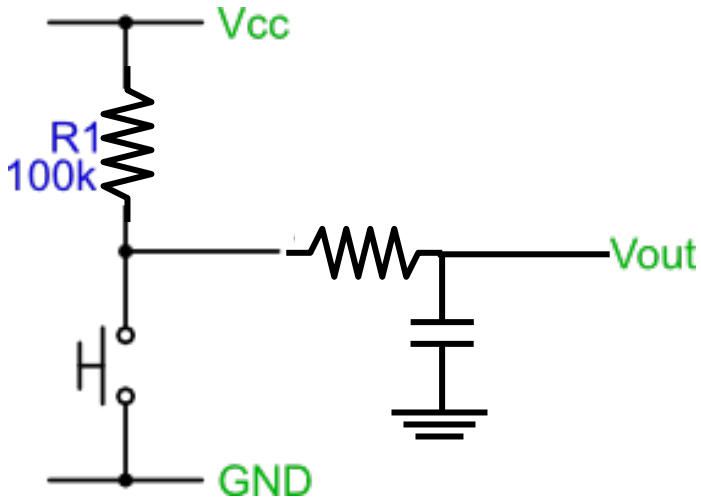


Switch debouncing in hardware (RC)

Q4: In chat type A or B for which circuit that will debounce the switch



Circuit A



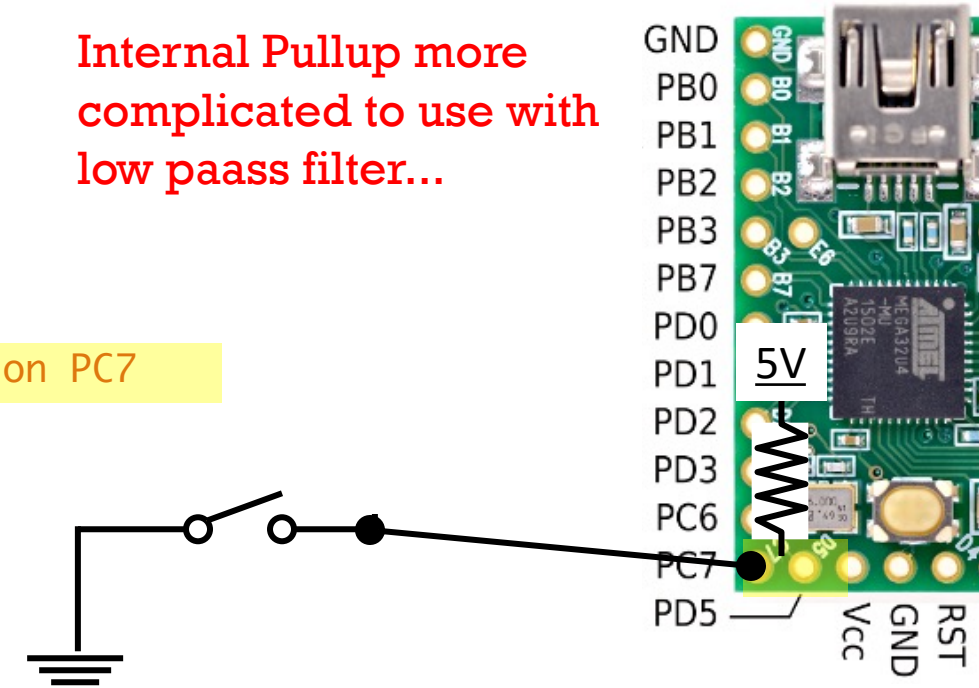
Circuit B

Note these filters are slightly different than the low pass/ high pass in earlier section when thinking about corner freq.

Detecting Button Presses

```
#include "teensy_general.h"
#include "t_usb.h"

#define PRINTNUM(x) m_usb_tx_uint(x); m_usb_tx_char(10); m_usb_tx_char(13)

void checkPC7() {
  int pinstate = bit_is_set(PINC,7);
  PRINTNUM(pinstate);
}

int main(){
  m_usb_init();
  set(PORTC,7);     // turn on pull up on PC7

  while(1) {
    checkPC7();
  }
}
```

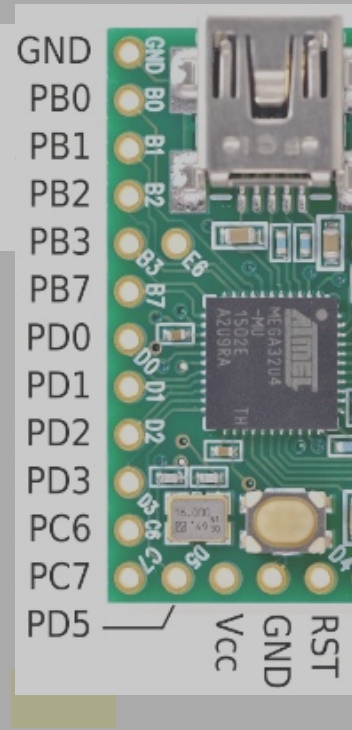
Internal Pullup more complicated to use with low paass filter...



Detecting Button (on/off) State Change

```
#include "teensy_general.h"
#include "t_usb.h"

#define PRINTNUM(x) m_usb_tx_uint(x); m_usb_tx_char(10); m_usb_tx_char(13)

void checkPC7() {
  int pinstate = bit_is_set(PINC,7);
  PRINTNUM(pinstate);
}

int main()
  m_usb_ir
  set(POR

  while(1)
    checkP
  }
}
```

```
void checkPC7() {
  static int oldstate;
  int pinstate = bit_is_set(PINC,7);

  if (pinstate != oldstate) {
      PRINTNUM(pinstate);
  }
  oldstate = pinstate;
}
```



03

Timer Input Capture

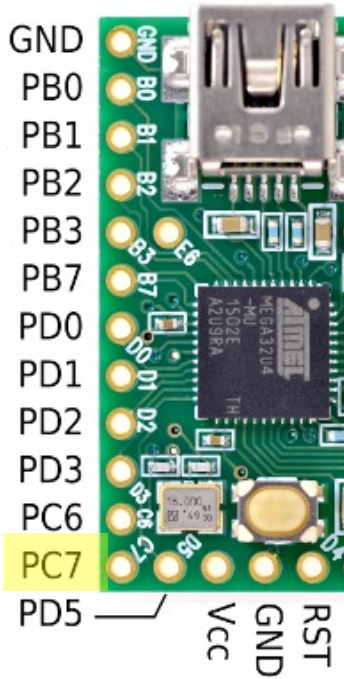
Timer Input Capture

- Recording a time stamp for events.
- Uses input functions on a pin combined with timer.
- Automatically and precisely captures timer with a hardware event.
- Can be more precise than polling.

Using Timer 3 to measure time of event (no input capture)

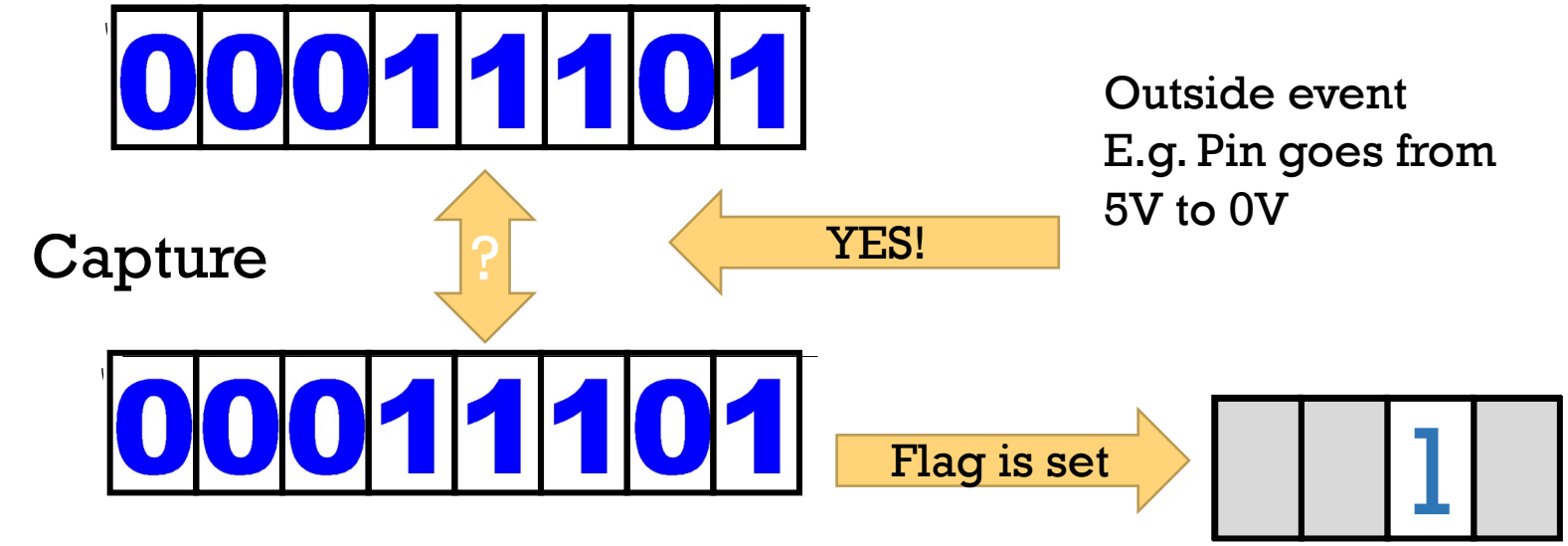
```
// same includes  and #define as previous program...
void waitforpress() {
  while( bit_is_set(PINC,7)) ; // wait while PC7 is high
  while(!bit_is_set(PINC,7)) ; // wait until PC7 stops being low
  PRlNTNUM(TCNT3);
}


int main(){
  m_usb_init();                          // init USB for print statements
  set(TIMERBITS); set(MORETIMERBITS); // Start timer3 with /1024 prescaler
  set(PORTC,7);                          // turn on pull up on PC7

  while(1) {
    m_usb_tx_string("first hit ");
    waitforpress();
    m_usb_tx_string("second hit ");
    waitforpress():
  }
}
```

DON'T COPY AND PASTE THIS CODE FOR LAB 2.
THERE ARE ERRORS IN THIS CODE



Timer/Counter Input Capture

"Free running" counter

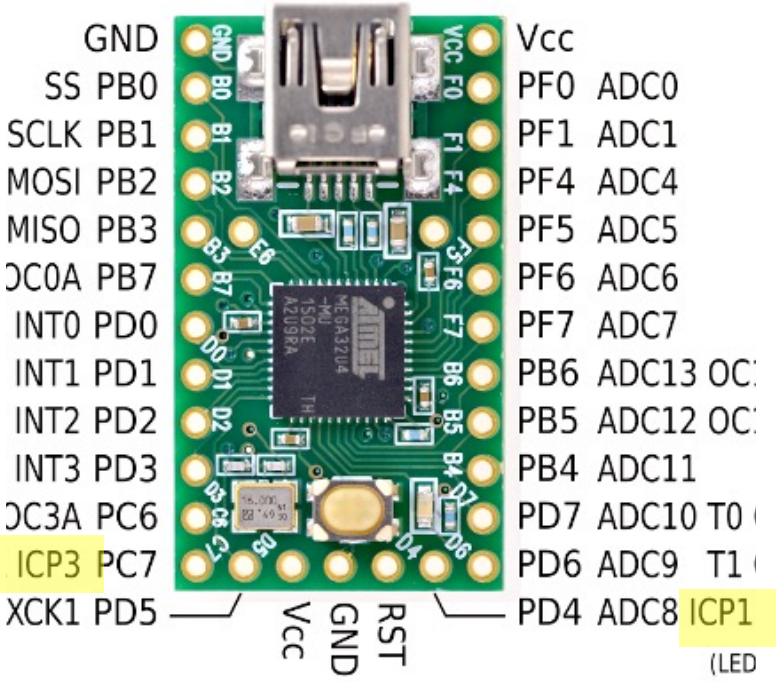


Input Capture Register

(Teensy has two 16bit registers `ICR1, ICR3`)

Input Capture Registers

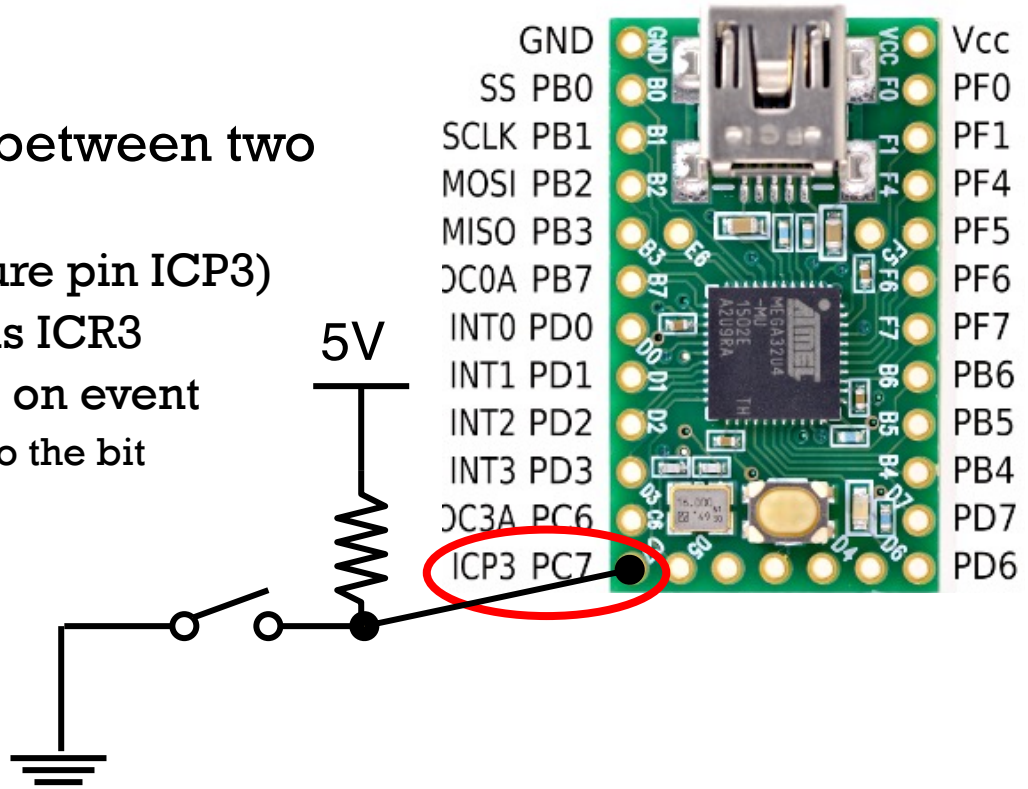
- Only need to start timer counting
- Two channels
  - Timer 1 ICP1 (Pin D4); ICR1 Register
    - ICF1 flag bit set in TIFR1
  - Timer 3 ICP3 (Pin C7); ICR3 Register
    - ICF3 flag bit set in TIFR3
- Options:
  - Rising or Falling Edge



Timer 1	Timer 3	
TCCR1B: ICES1	TCCR3B: ICES3	Function
0	0	store timer value on falling edge (default)
1	1	store timer value on rising edge

Input Capture on Timer 3 Example:

- Use Timer 3 to printout time between two button presses
  - Switch to PC7 (also input capture pin ICP3)
  - Timer Input Capture Register is ICR3
  - IC Flag is set in TIFR3, bit ICF3 on event
    - Flag is cleared by **writing** a 1 to the bit



Using Input Capture to Measure Event time

```
// same includes  and #define as previous program...
void waitforpress() {
  while( bit_is_set(PINC,7)) ; // wait while PC7 is high
  while(!bit_is_set(PINC,7)) ; // wait until PC7 stops being low
  PRlNTNUM(TCNT3)
}
```

```
int ma
  m_us
  set(
  set(

  whil
    m_
    wa
    m_
    wa
  }
}
```

```
// same includes as previous program...
void dosomething() {   // dummy routine to pretend that we have other things to do
  _delay_ms(500);      // same as teensy_wait(500);
}


void waitforpress() {
  while(!bit_is_set(TIFR3,ICF3))  dosomething() ;     // check input capture flag
  set (TIFR3, ICF3);                                  // clear flag by writing 1 to flag (!)
  PRINTNUM(ICR3):
}
```

Measuring time between two events

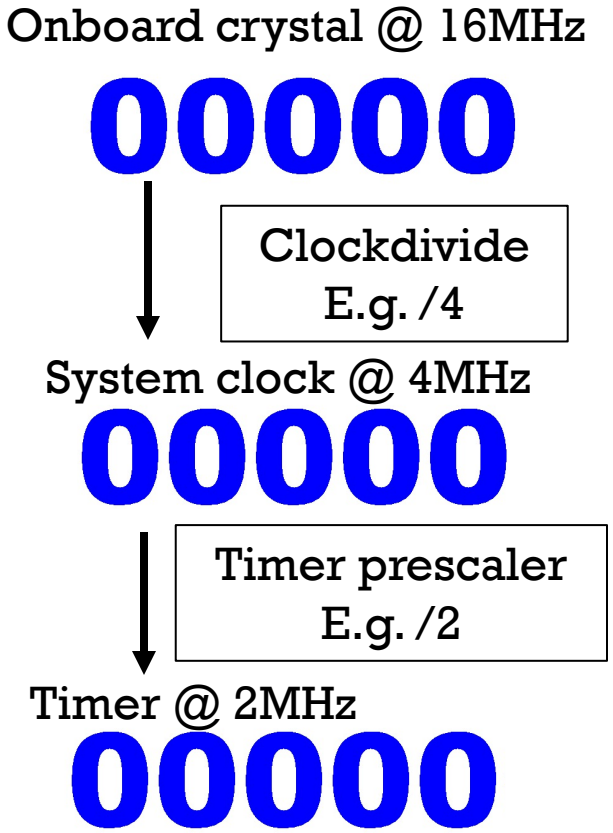
```
// same includes  and #define as previous program...
void waitforpress() {
  while( bit_is_set(PINC,7)) ; // wait while PC7 is high
  while(!bit_is_set(PINC,7)) ; // wait until PC7 stops being low
  PRlNTNUM(TCNT3)
}
```

```
int ma
  m_us
  set(
  set(

  whil
    m_
    wa
    m_
    wa
  }
}
```

```
// same includes as previous program...
void dosomething() {   // dummy routine to pretend that we have other things to do
  _delay_ms(500);      // same as teensy_wait(500);
}

unsigned int oldtime, tperiod;

void waitforpress() {
  while(!bit_is_set(TIFR3,ICF3))   dosomething();
  set (TIFR3, ICF3);  // clear flag by writing 1 to flag (!)
  tperiod = ICR3-oldtime;
  oldtime = ICR3;
  PRINTNUM(tperiod);       // time between sequential presses
}
```

Timer Overflow

- Overflow occurs when Timer has counted to largest number.
  - Depends on clockspeed
  - We can slow down the system clock
  - We can prescale the clock into the timer
  - Timer 1,3 have 16bit register (counts to 65535) -> @16MHz overflow every 4ms
- Max timer prescaler is /1024
  - @16Mhz maximum value corresponds to 4.2sec



- Q5: If we gain max time by slowing down thee clock, what do we lose?

Timer Register Wrapping Math Issue?

```
void waitforpress() {
  while(!bit_is_set(TIFR3,ICF3))   dosomething();
  set (TIFR3, ICF3);  // clear flag by writing 1 to flag (!)
  tperiod = ICR3-oldtime;
  oldtime = ICR3;
  PRINTNUM(tperiod);
}
```

- Normally `ICR3 > oldtime` as `ICR3` is read after oldtime has been stored and the timer is free running.

```
tperiod = ICR3-oldtime;
```

Q6: In Chat: What happens if ICR3 wraps (>65535) in the calculation? TCNT3 Overflow: Changes from 0xFFFF to 0x0000

Timer Register Wrap Issue?

- 2's complement math takes care of this issue.
- TCNT and the variables are unsigned integers. They can't go negative.

When you subtract a larger `int` from a smaller `int`:

```
tperiod = ICR3 - oldtime;
```

It becomes the same as if the smaller number had 65536 added to it.

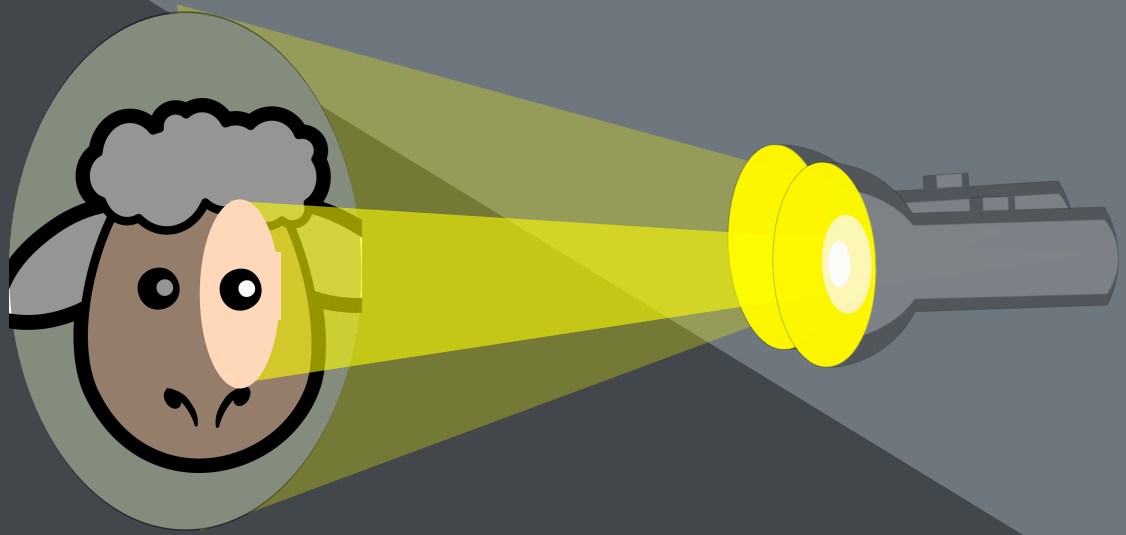
```
tperiod = 65536 + ICR3 - oldtime;
```

- Issue is if **more than one full** overflow has occurred.

04

Photosensors

Illuminance vs Flux

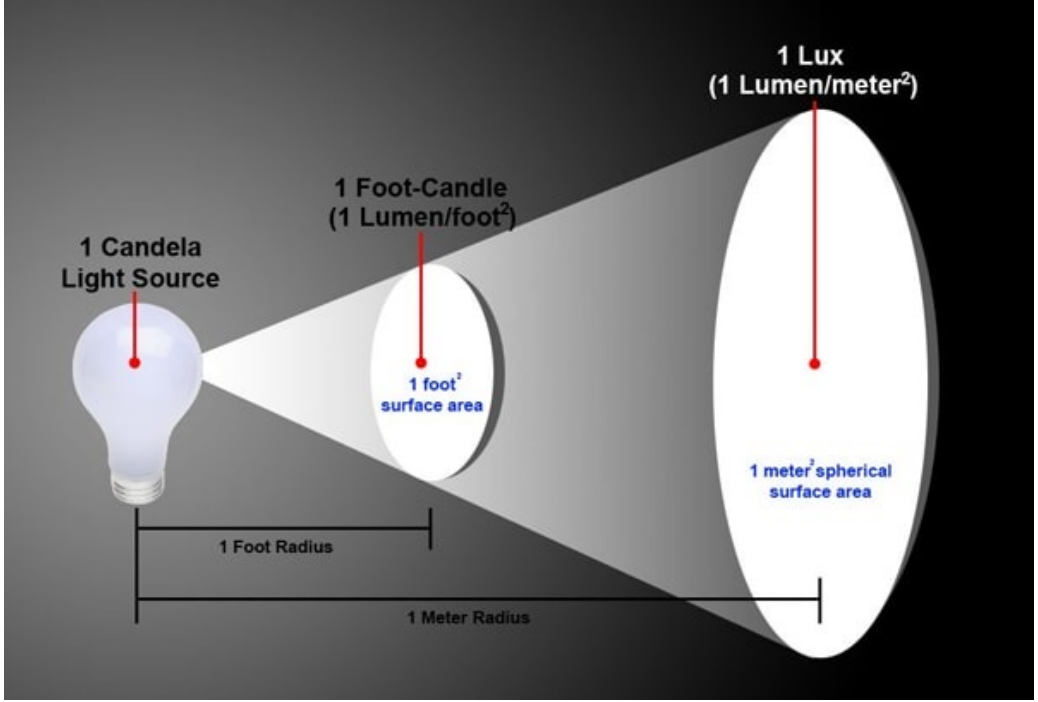


- Luminous Flux: Lumens
  - Visible light frequencies
- Radiant Flux: Watts
  - All frequencies
- Illuminance: Lux

Flashlight with "zoom lens"

- Constant luminous flux (total light energy) (Lumens)
- Varying brightness (Lux)

Order of Magnitude Examples



Low light
50 lux

Office
500 lux

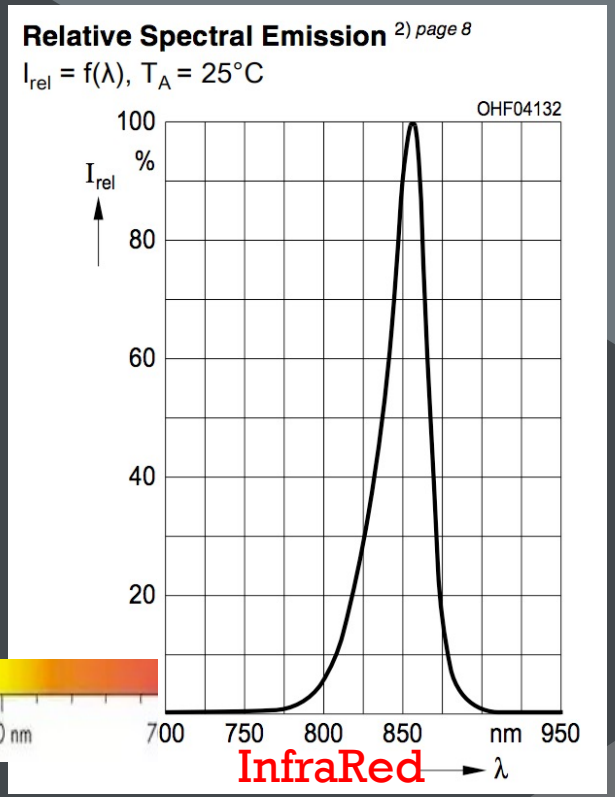
Rain
10,000 lux

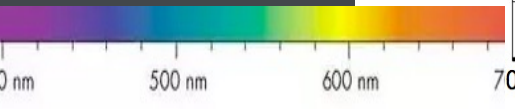
Direct Sun
100,000 lux

Light Frequencies

- Many semiconductor are sensitive or emit at infrared frequencies.

IR LED





TEPT4400 PhotoTransistor

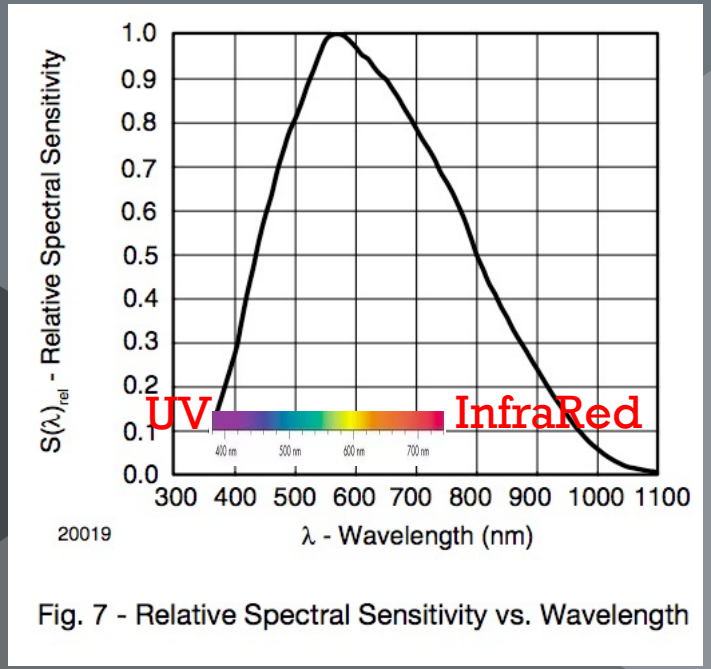
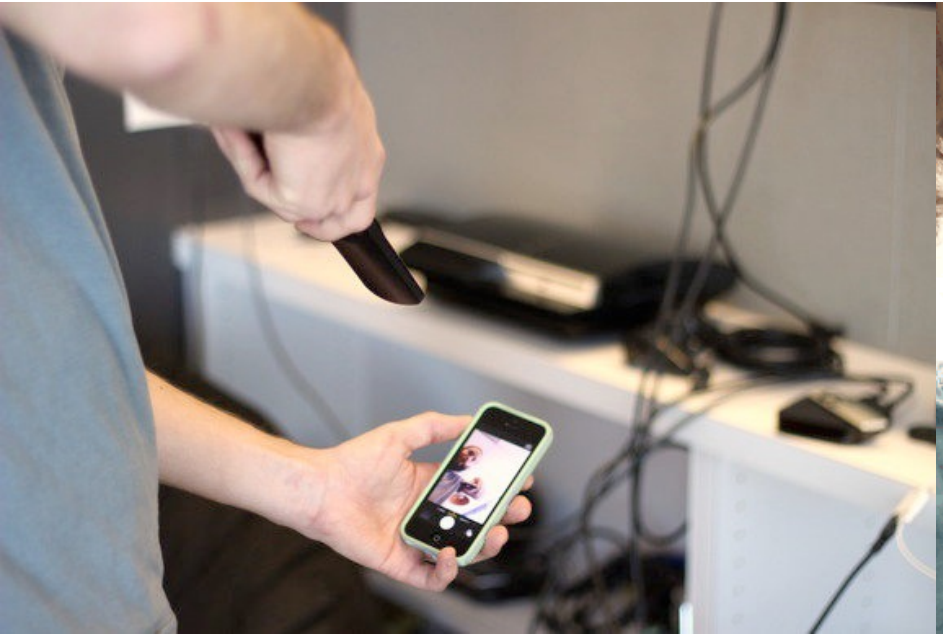


Fig. 7 - Relative Spectral Sensitivity vs. Wavelength

Phone camera IR debugging tool.



https://www.techhive.com/article/2090126/use-your-smartphone-to-check-your-remote-control-batteries.html

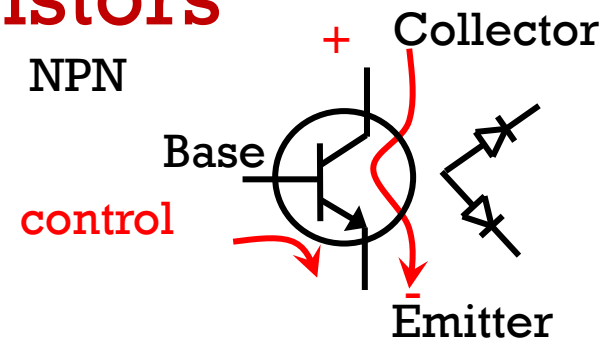
Photo Sensor Examples

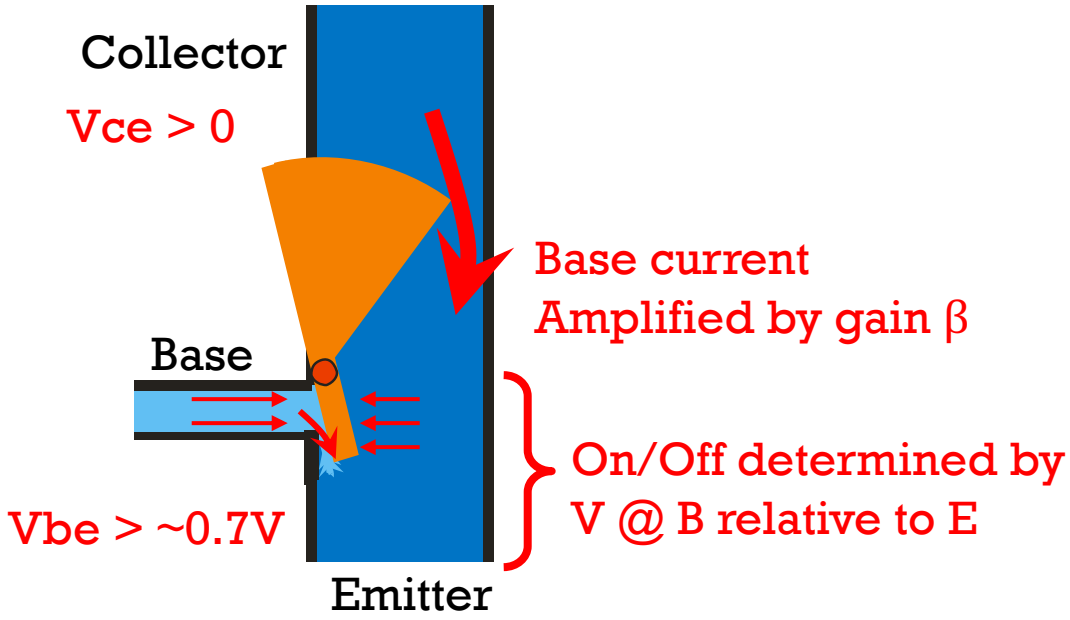
- Phototransistor
  - Easy to use, relatively high sensitivity
  - Cheap and robust
- Photodiode
  - Very Fast (nS)
  - High dynamic range
- CdS photo cell
  - Color response like human eye
  - Very slow (~1-100mS)
- Photovoltaic cell
  - Linear response to incident light
  - Fragile
  - Slow (depending on size)

05

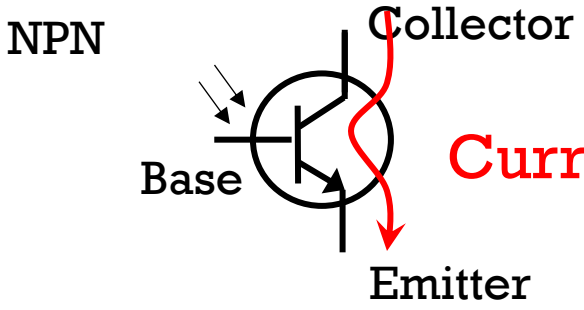
Phototransistors

BJT Transistors



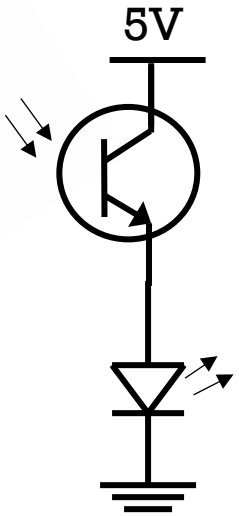


Phototransistors



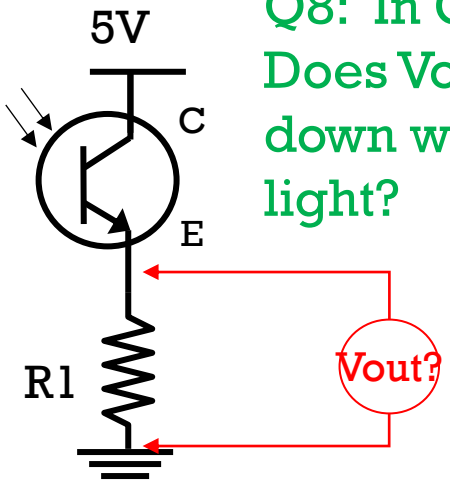
Current = I_{CE}

Q7: What happens if circuit is exposed to a bright room?

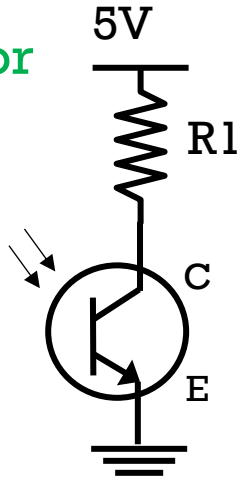


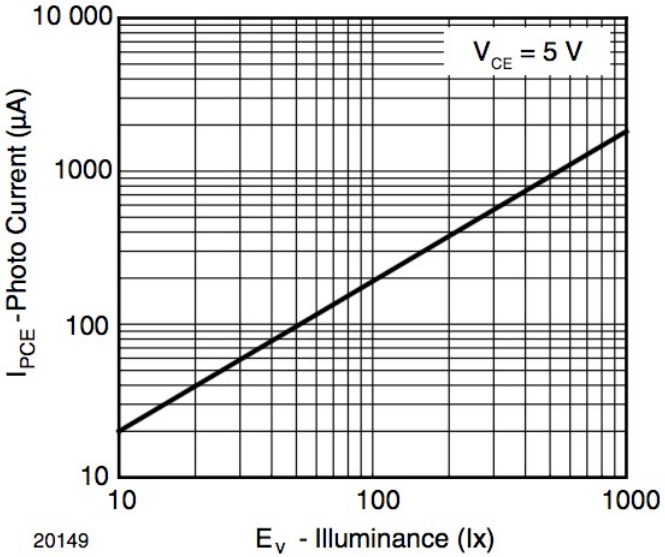
- Just two leads (no base)
  - Looks just like a diode…
- Light into base causes current to flow.
- If V_C is larger than V_E, (typically > 0.4V) then more light means more more current I_{CE}

Phototransistors: How to use?



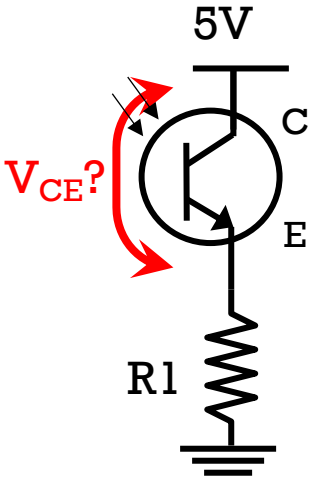
Q8: In Chat
Does Vout go up or down with more light?

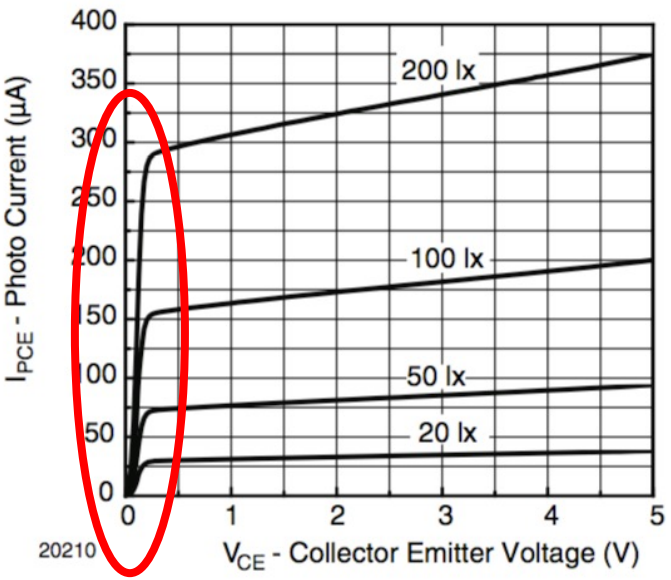


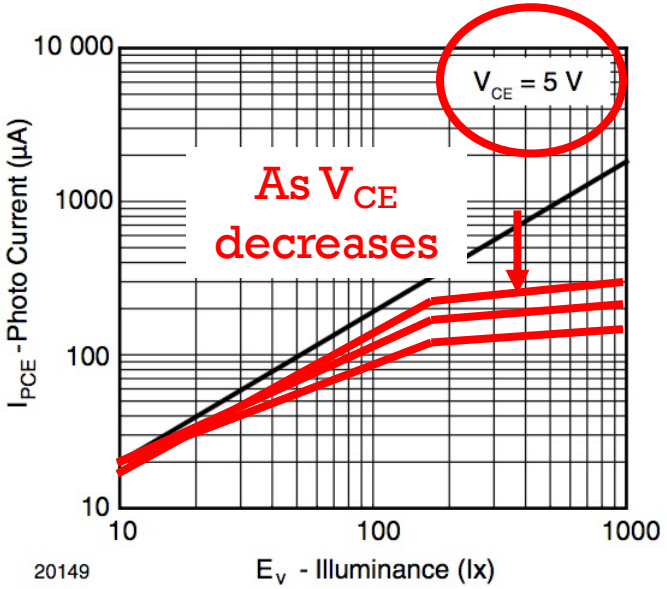


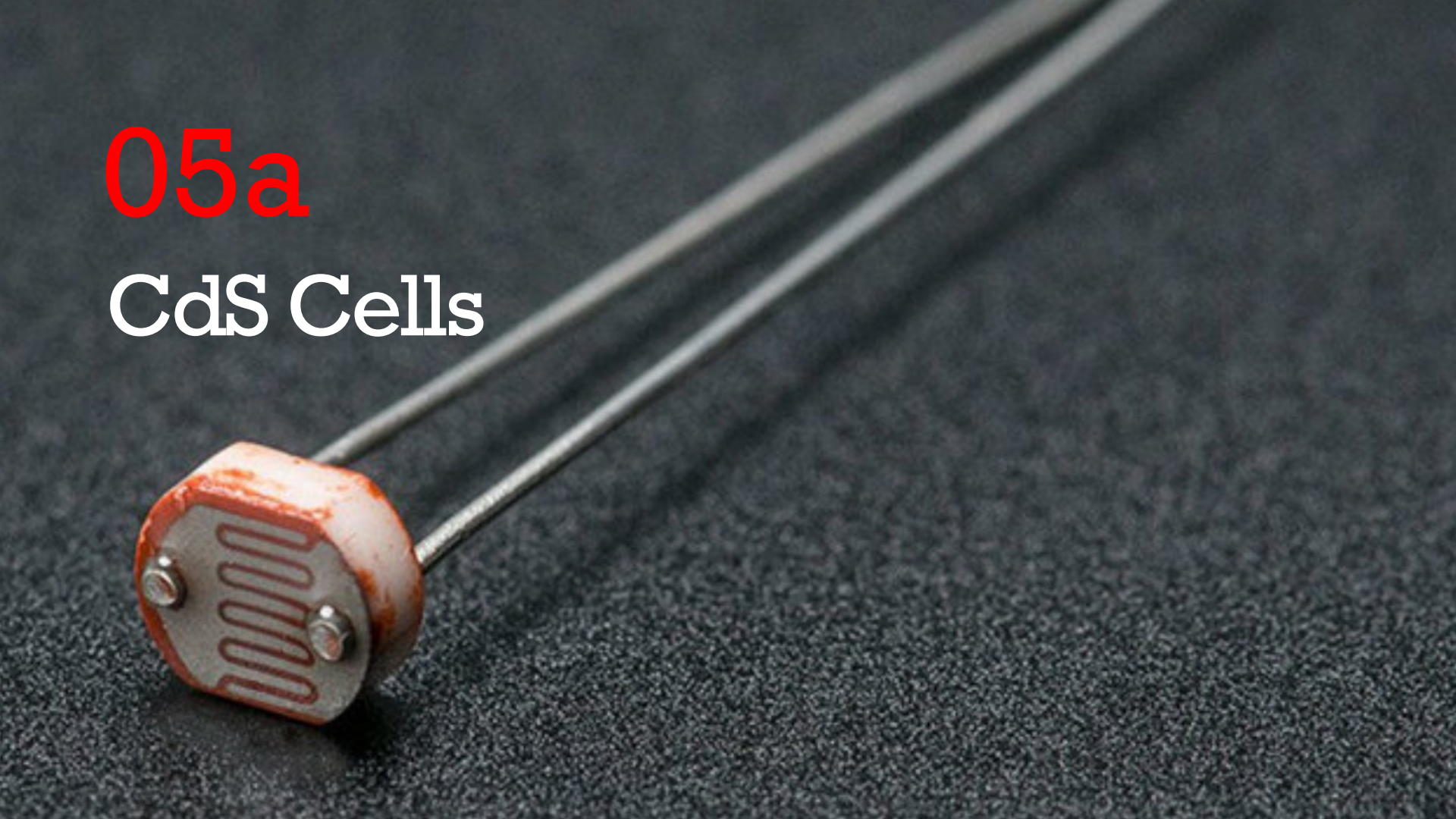
- Q9: In Chat: How does value of R1 effect sensitivity?
- Q10: Can R1 be too big?

Phototransistors: How to use?









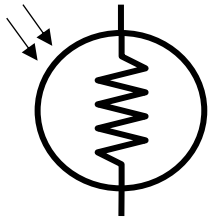
05a

CdS Cells

CdS PhotoCell

Not solar cells

Symbol

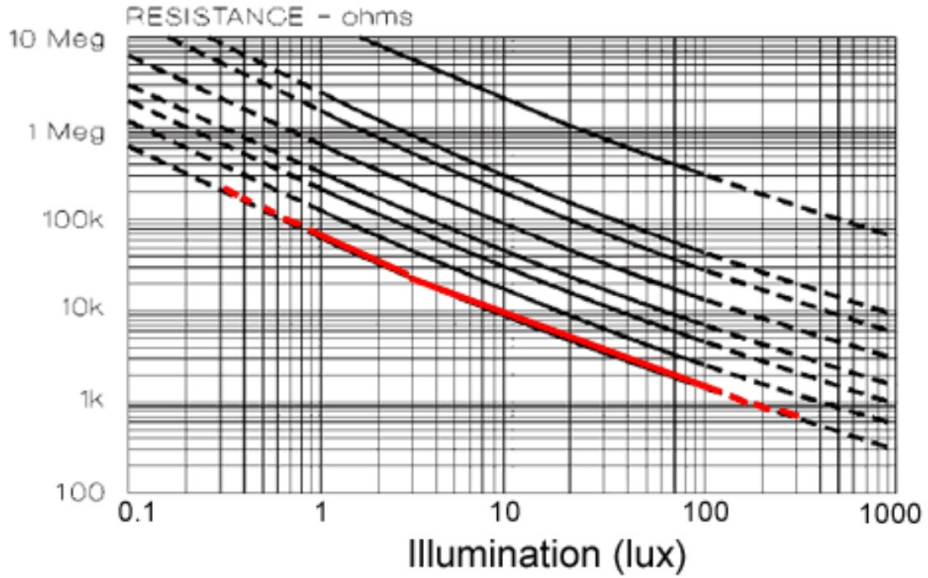


Variable resistor

CdS Resistance Varies with Incident Light

More Light = Lower Resistance

- Spectral Response



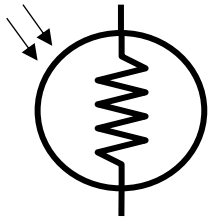
- Dynamic Response is slow

- Power Rating

CdS PhotoCell

Not solar cells

Symbol



Variable resistor

CdS Resistance Varies with Incident Light

More Light = Lower Resistance

- Spectral Response

  Approximates Human Eye

- Dynamic Response is slow

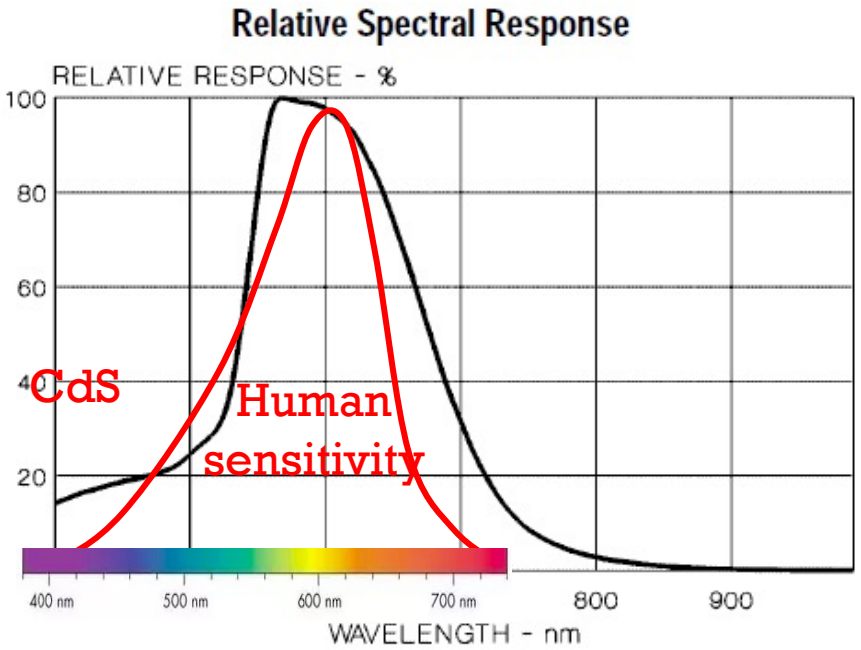
  (~1's – 10's of mS)

  Can act as filter!

- Power Rating

  Exceeding power generate heat

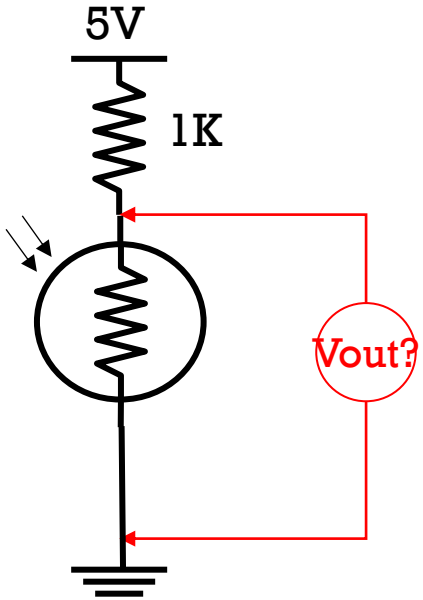
  Changes sensitivity



How do we use a CdS Cell?

- How is a variable resistor different than a potentiometer?
- If our light conditions vary from 10 to 500 lux, what is our voltage output range?

- How can we get a more sensitivity (increase our output range for the same input range?)



Summary

- Input capture can get high resolution timing independent of what processor is doing – as long as you handle multiple overflow and don't miss events.
- Phototransistors vary current (in voltage divider-like setup)
- Bigger resistances convert small currents to larger voltages (ohm's law). Can increase gain on current output sensors like phototransistors.

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. Capacitors Highpass/Lowpass

B. Timer Input Capture

C. Phototransistors