Lecture 07

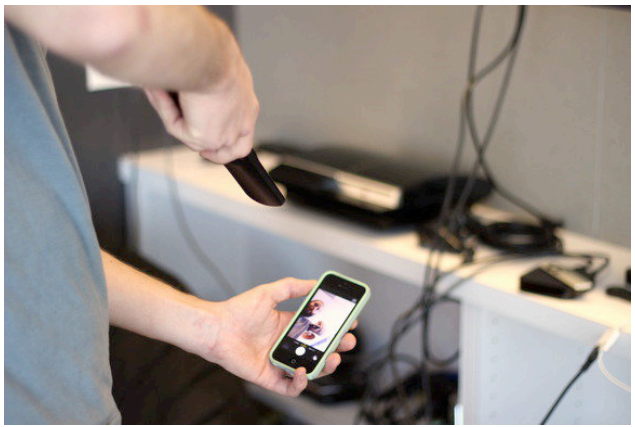
Comparators

Agenda

00. Coding Practice: Subroutines
01. Comparators
02. CdS Cells
03. Resistor Colors

Kit check

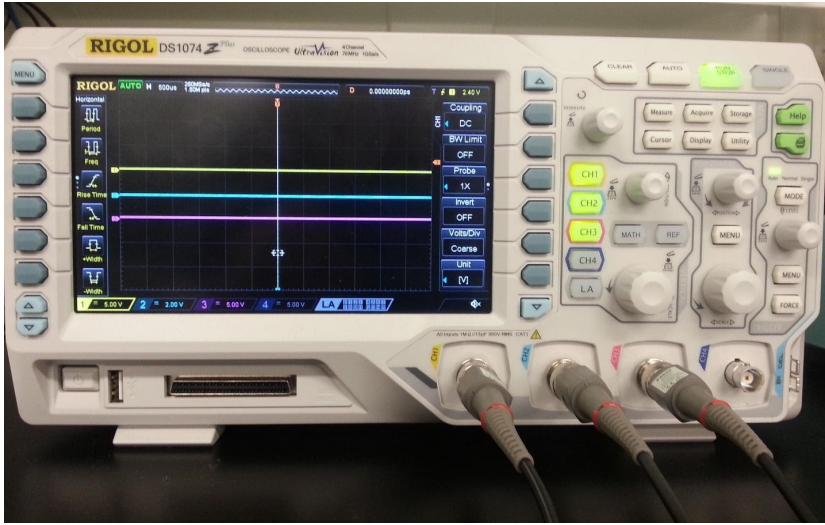
- For 2.3 make sure you have an IR LED and an IR phototransistor. The IR LED looks a lot like the TEPT4400. Check by putting seeing if the device glows when viewed with a cell phone, when powered in series with a resistor.



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

Real Oscilloscope Assignment (tba)

- Sign up to borrow one of the 10 from GMlab for up to two days. Late day penalty applies for returning late.
- There is an extra oscilloscope exercise – due by end of semester.
- Exercise should take about 15 minutes.
- These oscilloscopes will be available for you to borrow (for up to two days) as well.
- Use the GM signup sheet (oscilloscope tab) to reserve one.



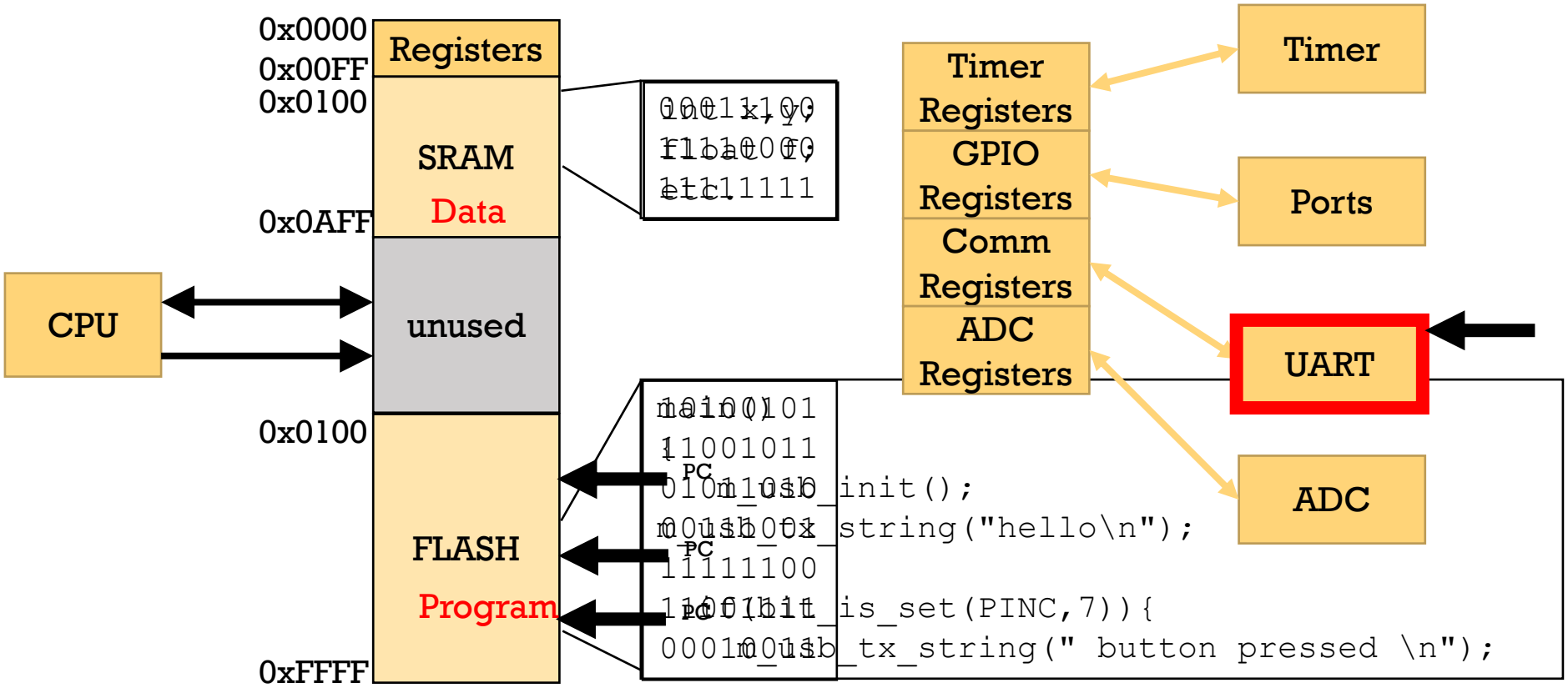
Print statements are slow

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

int main()
{
  m_usb_init(); //initializes usb connection

  m_usb_tx_string("hello\n");
  if(bit_is_set(PINC,7)){           //if the switch is pressed...
    m_usb_tx_string(" button pressed \n");
  }
  else {
    m_usb_tx_string(" button not pressed \n");
  }
  return 0;
}
```

Peripherals run concurrent with main code



01

Basic Coding Practice
Subroutines
Psuedo code

Structured programming to avoid spaghetti code…

There is a tendency in mechatronics students to write code that becomes difficult to follow… adding more and more onto a piece of code that initially worked for some parts, but as it grew in complexity became very hard to follow and harder to debug or add onto because it wasn't modular, or well thought out in terms of architecture at the beginning.

Structured programming to avoid spaghetti code…

Mechatronics students tend to write difficult to follow code…

Modular code will be easier to follow, debug, and add onto as it gets larger.

Thinking about the code structure beforehand helps

Subroutines

```
#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#include "t_usb.h"

int main(){
  m_usb_init();                          // init USB for print statements
  set(TCCR3B,CS30); set(TCCR3B,CS32); // Start timer3 with /1024 prescaler

  m_usb_tx_string("first hit ");
  while( bit_is_set(PINC,7)) ;// wait while PC7 is high
  while(!bit_is_set(PINC,7)) ;// wait until PC7 stops being low
  m_usb_tx_uint(TCNT3); m_usb_tx_char(10); m_usb_tx_char(13);

  m_usb_tx_string("second hit ");
  while( bit_is_set(PINC,7)) ;// wait while PC7 is high
  while(!bit_is_set(PINC,7)) ;// wait until PC7 stops being low
  m_usb_tx_uint(TCNT3); m_usb_tx_char(10); m_usb_tx_char(13);

  m_usb_tx_string("first hit ");
  while( bit_is_set(PINC,7)) ;// wait while PC7 is high
  while(!bit_is_set(PINC,7)) ;// wait until PC7 stops being low
  m_usb_tx_uint(TCNT3); m_usb_tx_char(10); m_usb_tx_char(13);

  m_usb_tx_string("second hit ");
  while( bit_is_set(PINC,7)) ;// wait while PC7 is high
  while(!bit_is_set(PINC,7)) ;// wait until PC7 stops being low
  m_usb_tx_uint(TCNT3); m_usb_tx_char(10); m_usb_tx_char(13);
```

Subroutines

```
#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#include "t_usb.h"

int main(){
  m_usb_init();                          // init USB for print statements
  set(TCCR3B,CS30); set(TCCR3B,CS32); // Start timer3 with /1024 prescaler

  while(1) {
    m_usb_tx_string("first hit ");
    while( bit_is_set(PINC,7)) ;// wait while PC7 is high
    while(!bit_is_set(PINC,7)) ;// wait until PC7 stops being low
    m_usb_tx_uint(TCNT3); m_usb_tx_char(10); m_usb_tx_char(13);

    m_usb_tx_string("second hit ");
    while( bit_is_set(PINC,7)) ;// wait while PC7 is high
    while(!bit_is_set(PINC,7)) ;// wait until PC7 stops being low
    m_usb_tx_uint(TCNT3); m_usb_tx_char(10); m_usb_tx_char(13);
  }
}
```

Subroutines

```
#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#include "t_usb.h"

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


int main(){
  m_usb_init();                          // init USB for print statements
  set(TCCR3B,CS30); set(TCCR3B,CS32); // Start timer3 with /1024 prescaler

  while(1) {
    m_usb_tx_string("first hit ");
    while( bit_is_set(PINC,7)) ;// wait while PC7 is high
    while(!bit_is_set(PINC,7)) ;// wait until PC7 stops being low
    PRINTNUM(TCNT3);

    m_usb_tx_string("second hit ");
    while( bit_is_set(PINC,7)) ;// wait while PC7 is high
    while(!bit_is_set(PINC,7)) ;// wait until PC7 stops being low
    PRINTNUM(TCNT3);
  }
}
```

Subroutines

```
#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#include "t_usb.h"

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

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
  PRINTNUM(TCNT3);
}

int main(){
  m_usb_init();                          // init USB for print statements
  set(TCCR3B,CS30); set(TCCR3B,CS32); // Start timer3 with /1024 prescaler

  while(1) {
    m_usb_tx_string("first hit ");
    waitforpress();

    m_usb_tx_string("second hit ");
    waitforpress();
  }
}
```

Subroutines

```
#include "teensy_general.h"  // includes the resources included in the teensy_general.h file
#include "t_usb.h"

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

int waitForC7RisingEdge() {
  while( bit_is_set(PINC,7)) ;// wait while PC7 is high
  while(!bit_is_set(PINC,7)) ;// wait until PC7 stops being low
}

void waitforpress() {
  waitForC7RisingEdge();
  PRINTNUM(TCNT3);
}

int main(){
  m_usb_init();                          // init USB for print statements
  set(TCCR3B,CS30); set(TCCR3B,CS32); // Start timer3 with /1024 prescaler

  while(1) {
    m_usb_tx_string("first hit ");
    waitforpress();

    m_usb_tx_string("second hit ");
    waitforpress();
```

Probably too much,
Don't need to make this
a subroutine

Subroutines – proper use benefits

- Makes the code clearer to understand
- Remove duplication (simpler is better)
- Makes the code more modular
  - Can reuse
  - Can change elements more easily
  - Breaking down into components makes it easier to build
- May effect speed, (sometimes slightly slower, but usually negligible)

Pseudocode

- Artificial and informal language to help programmers develop algorithms.
- Indent for dependencies or loops

Example:

```
Set grade counter to one
While grade counter is less than ten
     Input the next grade
     Add the grade into the total
Set the class average to the total divided by nine
```

This is NOT Pseudo-Code

```
for (signal = GetSignal(); signal == (new_signal = GetSignal());)
    ;

for ( this_time = GetTime(), i=0; i < SAMPLE_SIZE; i++ )
{
    signal = new_signal;
    last_time = this_time;
    while ( signal == ( new_signal = GetSignal() ) )
      ;
    this_time = GetTime();
    delta_Ts[i] = this_time - last_time;
}

avgtime = average(delta_Ts);
if (avgtime > NOISE) openDoor();
```

For this class, Pseudocode is detailed yet readable

Wait for signal to change states

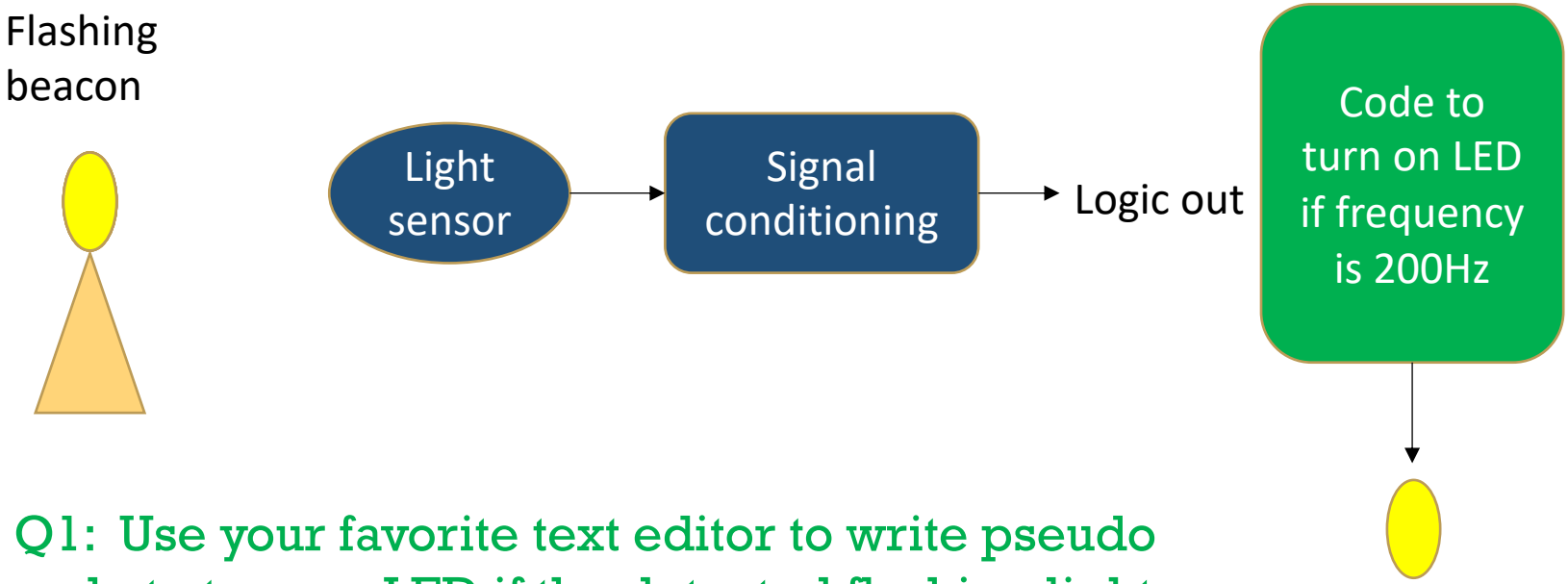
Repeat SAMPLE_SIZE times
    wait for signal to change states
    save the delta-T into an array of samples

Calculate average time
If signal is not noise open door

Pseudocode Becomes Comments

```
/*  Wait for signal to change states  */
for (signal = GetSignal(); signal == (new_signal = GetSignal());)
    ;
/*  Repeat SAMPLE_SIZE times  */
for ( this_time = GetTime(), i=0; i < SAMPLE_SIZE; i++ )
{
    signal = new_signal;
    last_time = this_time;
    /*  wait for signal to change states  */
    while ( signal == ( new_signal = GetSignal() ) )
      ;
    this_time = GetTime();
    /*  save the delta-T into an array of samples  */
    ticks[i] = this_time - last_time;
}
/* Calculate average time */
avgtime = average(times);
if (avgtime > NOISE) openDoor();   // If  signal not noise open door
```

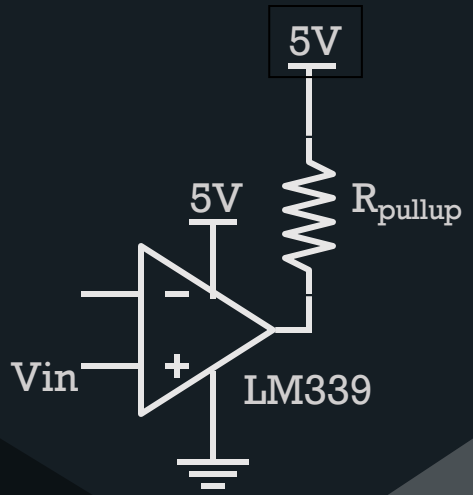
Pseudocode Practice (prelude to 2.3.3)



Q1: Use your favorite text editor to write pseudo code to turn on LED if the detected flashing light frequency is 200Hz.

03

Comparators

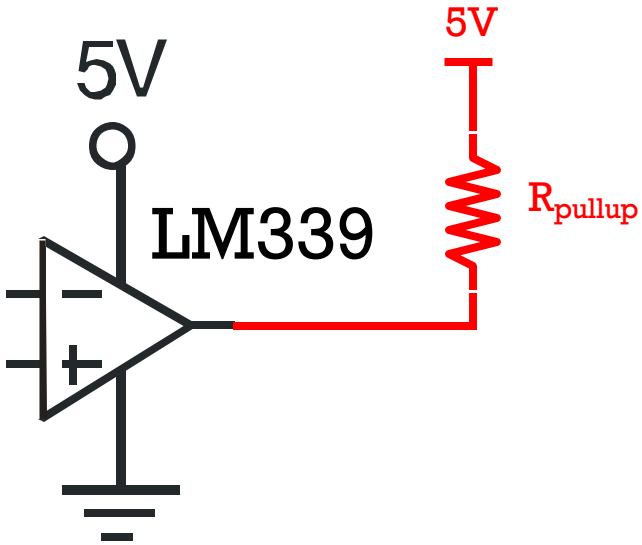


Comparator

Looks like an opamp,
but it's a comparator

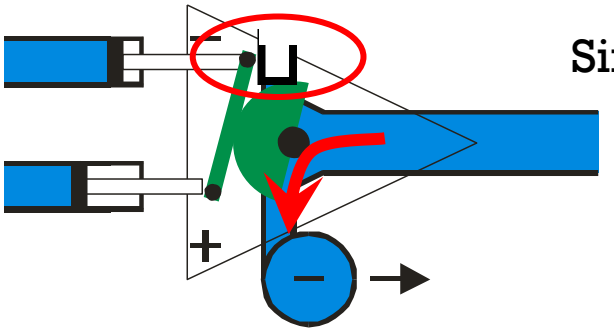
Acts like an opamp
but designed for driving
output to a rail not linearity.

Compares the two inputs
swings output to negative
rail or pulled up voltage.

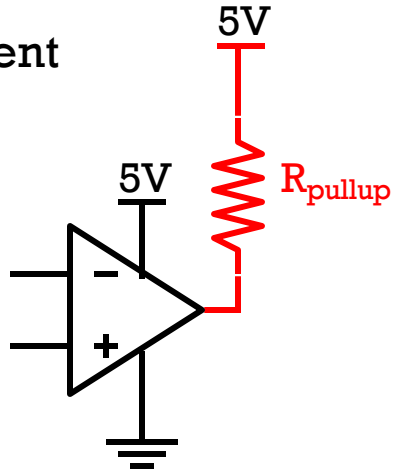


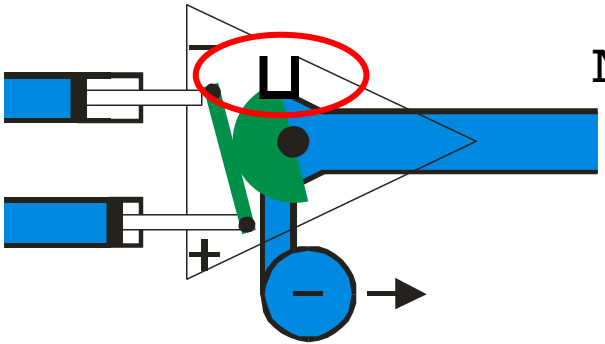
Output is often
"open collector" or "open drain."
(need pull-up resistor)

Two states of open collector comparator



Sinking current





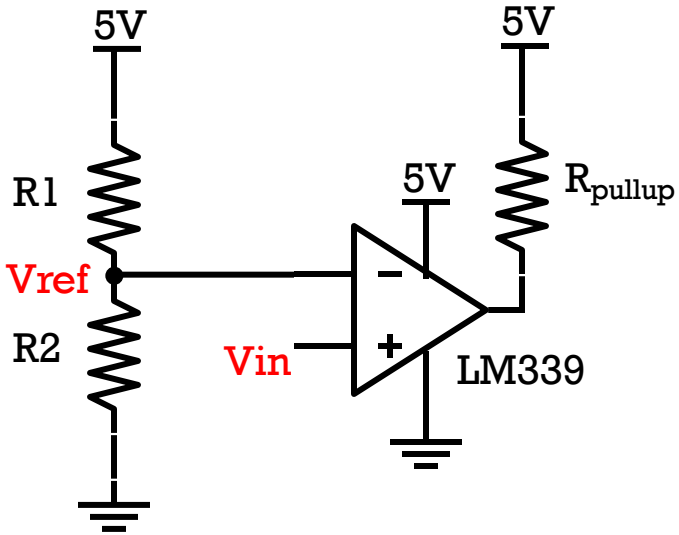
No flow

Output is "floating"

High impedance state

How do you use a comparator?

R1 and R2 are a voltage divider setting a reference voltage

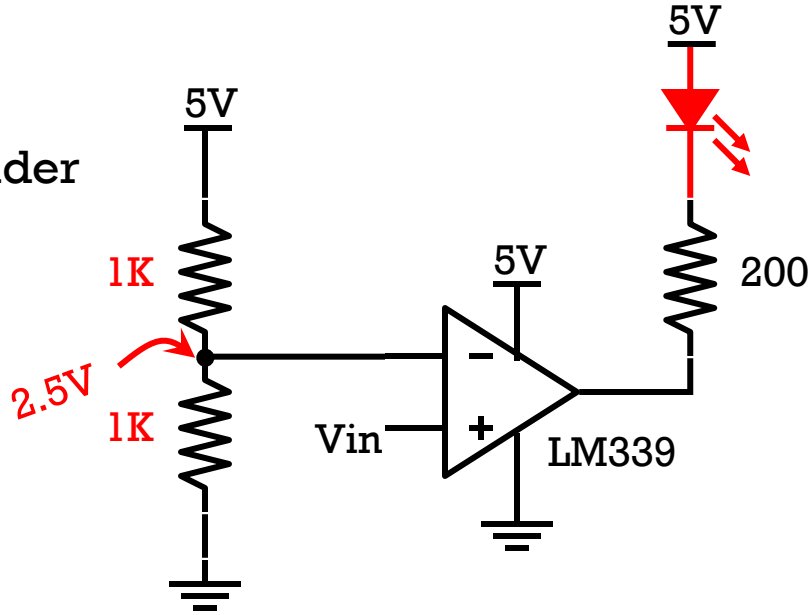


Input	Output
Vin < Vref	driven LOW
Vin > Vref	pulled HIGH

Non-inverting

What would this do?

R1 and R2 are a voltage divider setting a reference voltage

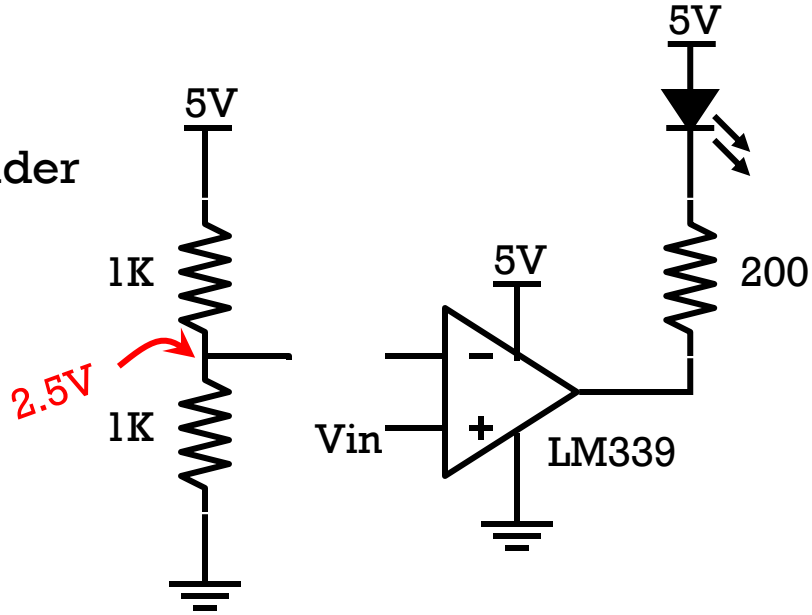


Non-inverting

Q2: in chat: What happens to the LED if Vin is 1.5V?

What would this do?

R1 and R2 are a voltage divider setting a reference voltage

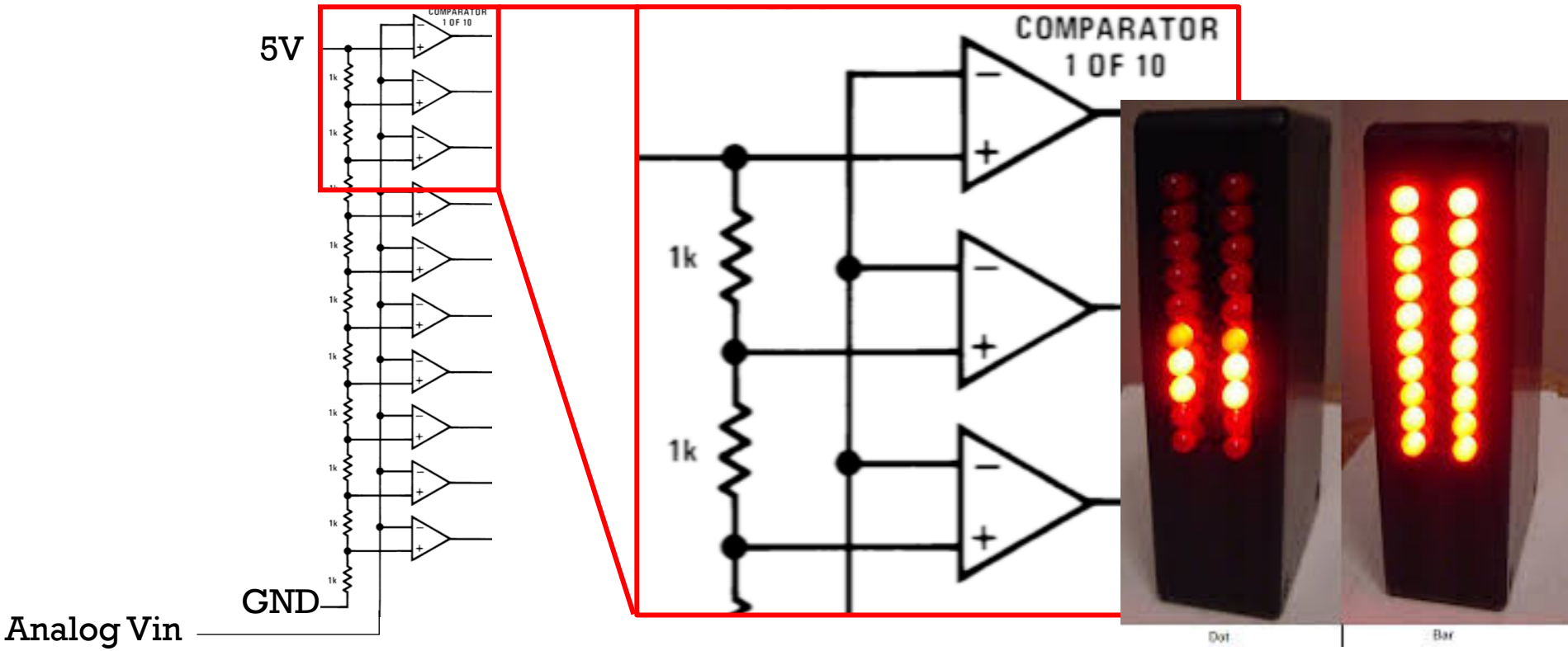


If Vin < 2.5V? LED on

If Vin > 2.5V? LED off

Inverting configuration

How does this circuit behave?



LM339

- Driving LED's with LM339A? What do we need to know?

Q3: Draw and hold: Circle the important value on this spec sheet.

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

When would you use a comparator?

- Converting analog to digital.
- Analog signal crosses a threshold voltage, set output high or low.

Q4: unmute: Is there a part in Lab 2.1 or 2.2 where this could be used?

Versus Op amp with very high gain?

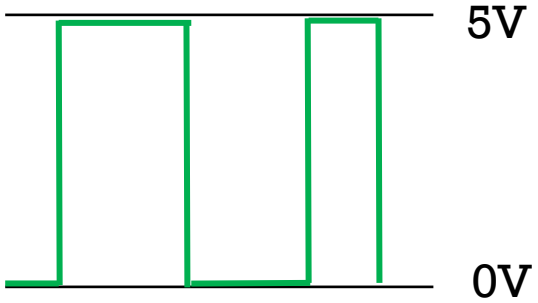
- When you need fast signals (input or output in uS)
- When you want to tie many outputs together (wired OR)
- When you have large voltage swing output requirements.
- Op amps designed for negative feedback, not railed output, may have stability issues.

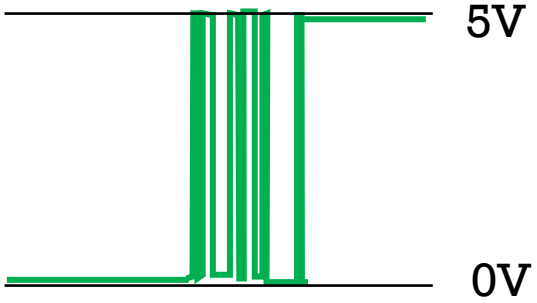
01a

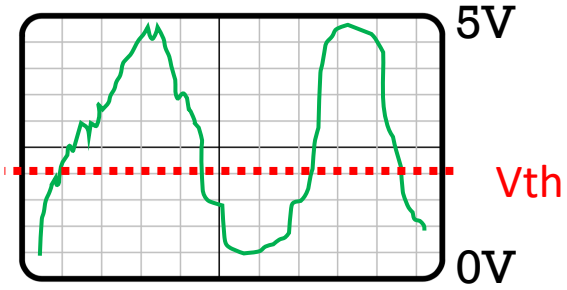
Comparators Hysteresis

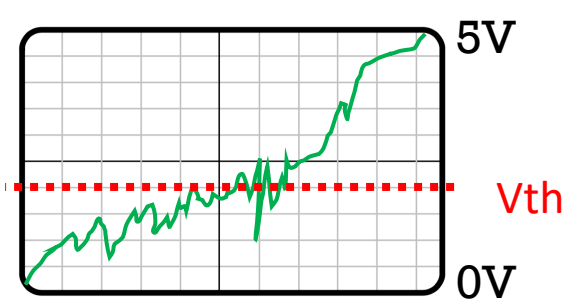
Comparators

Q5 Draw and hold: What would the output be with threshold Vth?

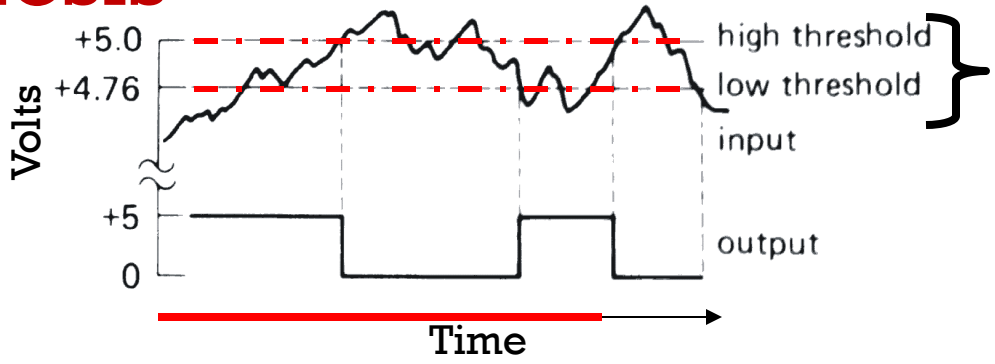




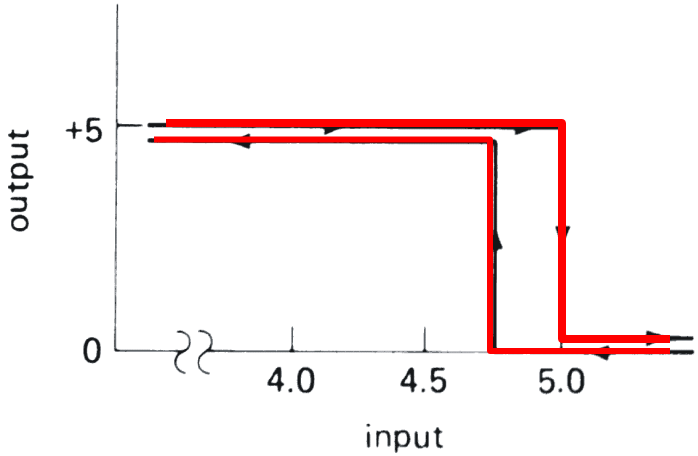




Hysteresis

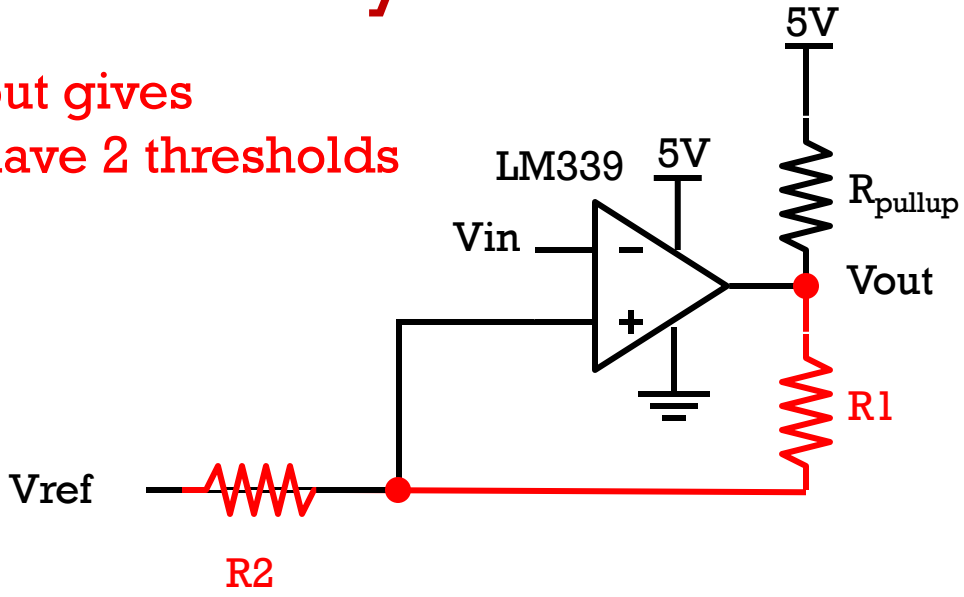


Two Thresholds, Which one applies depends on past history



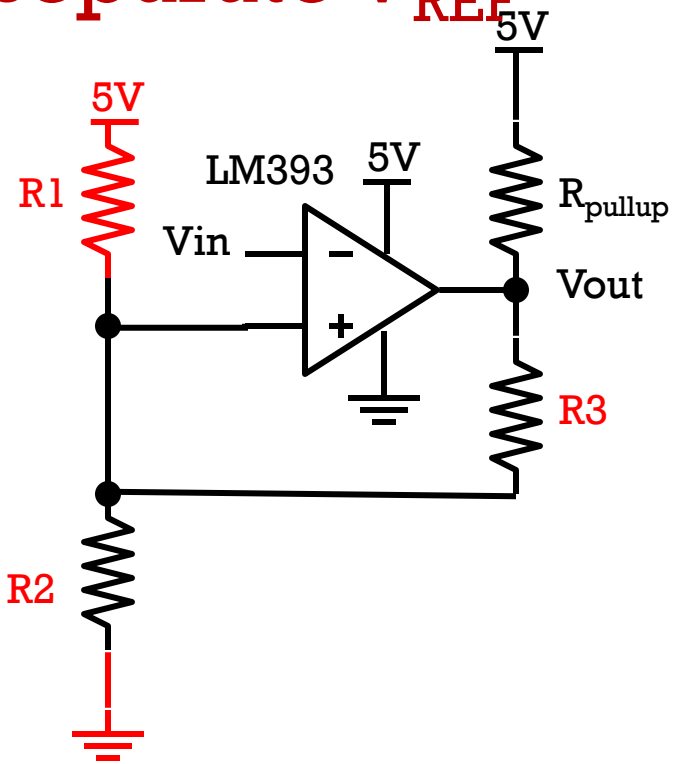
How do you add hysteresis?

Two states of Vout gives mechanism to have 2 thresholds



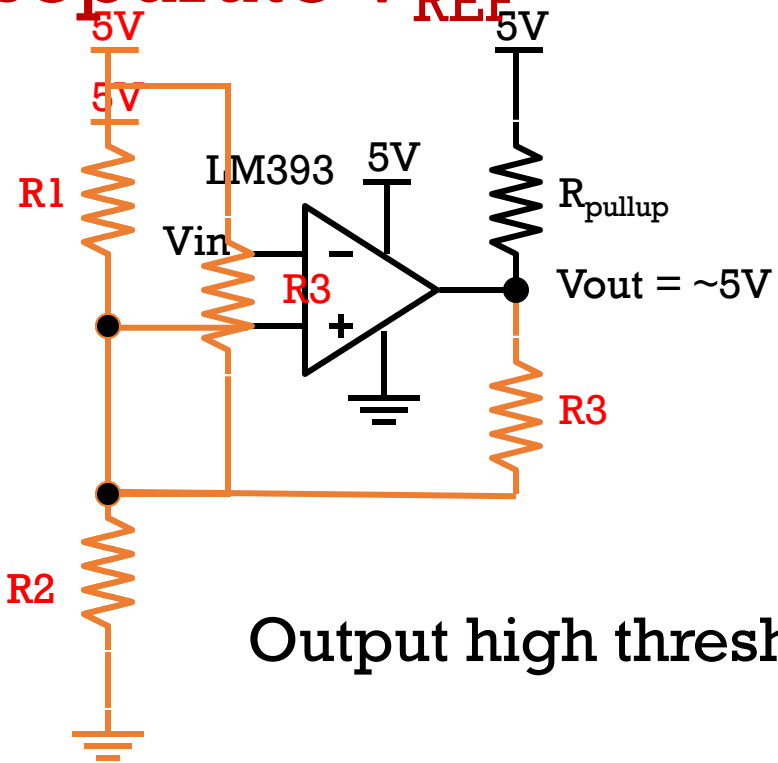
Hysterisis is about Vout R2/R1

Eliminating the separate V_{REF}



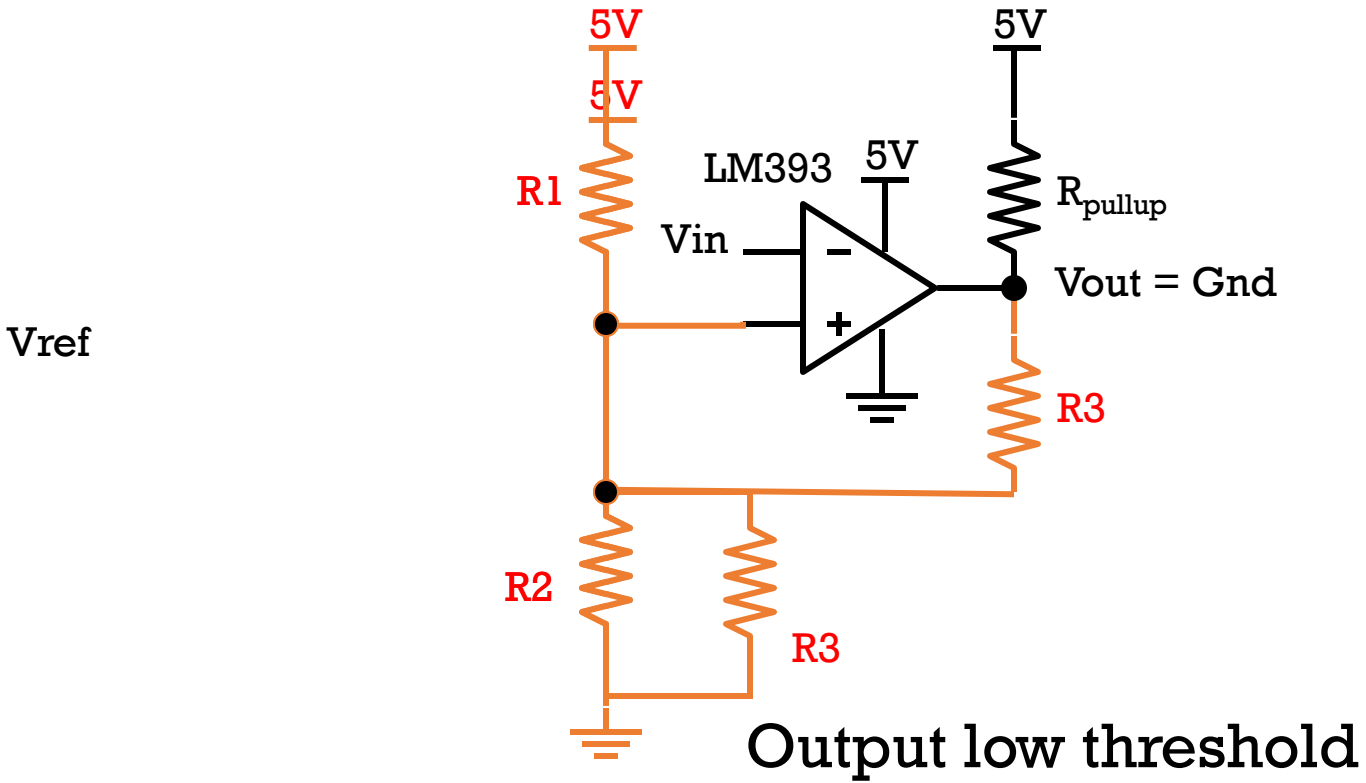
Eliminating the separate V_{REF}

Vref

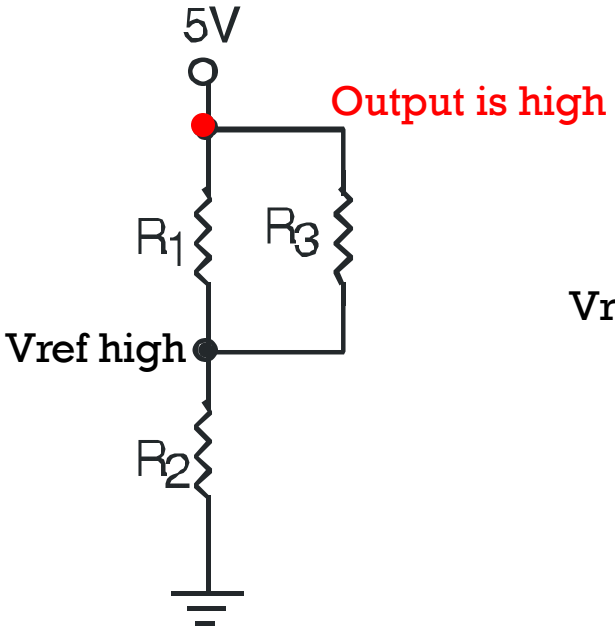


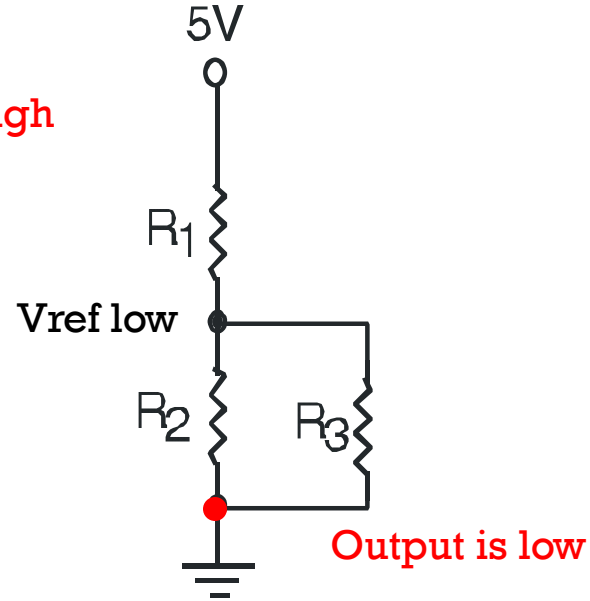
Output high threshold

Eliminating the separate V_{REF}



What are the two thresholds?





$$V_{ref\ high} = V_{cc} \frac{R2}{(R1 \,||\, R3) + R2}$$

$$V_{ref\ low} = V_{cc} \frac{R2 \,||\, R3}{(R2 \,||\, R3) + R1}$$

Inverting Comparator Design Procedure

1) $R_{pullup} << R_{load}$

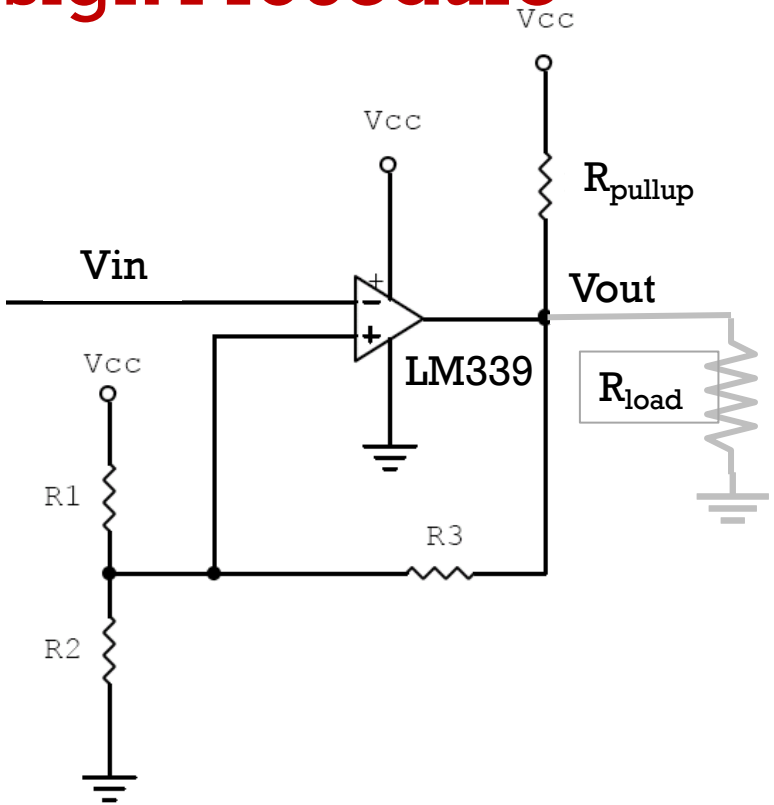
2) R3 >> Rpullup

3) Choose thresholds

ΔV = Vref high - Vref low

4) $R1 = \frac{R3\ (\Delta V)}{Vref\ low}$

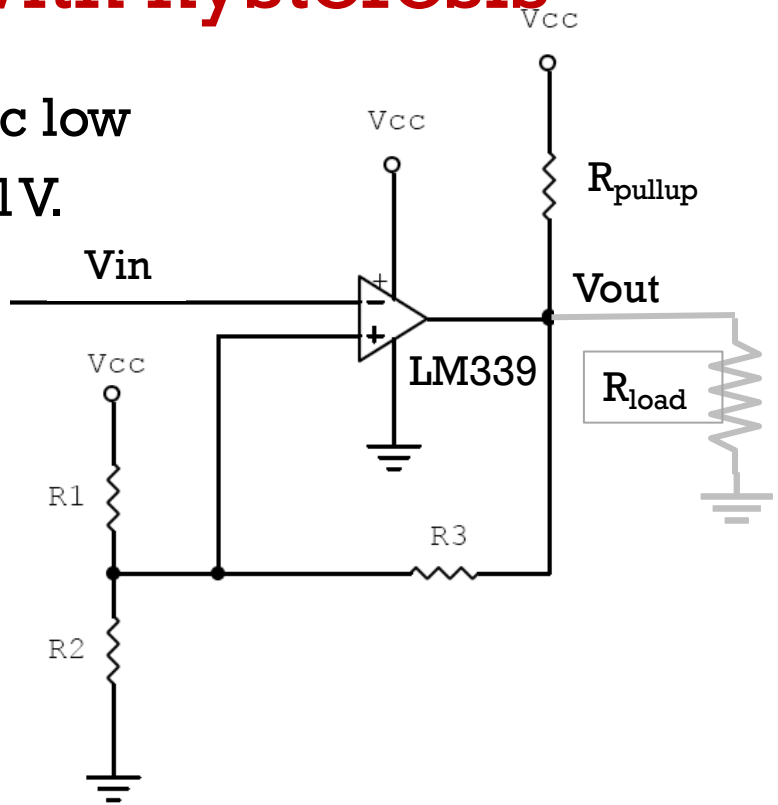
5) $R2 = \frac{R1\ ||\ R3}{(Vcc/Vref\ high) - 1}$



Start with R_{pullup} = 3.9K,
R3 = 1.0M

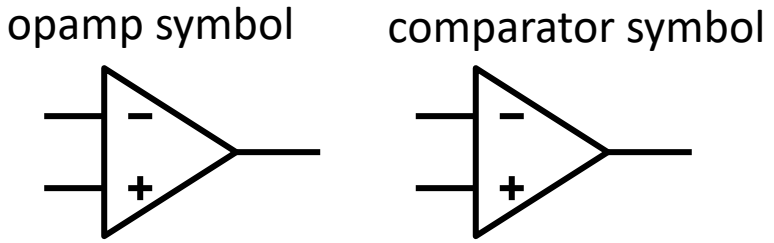
Example comparator with hysteresis

- Conditions: 3.3V logic high. 0V logic low
- Choose Vref high = 2V, Vref low = 1V.
- Choose pullup of 5kΩ
- Choose R3 = 100KΩ

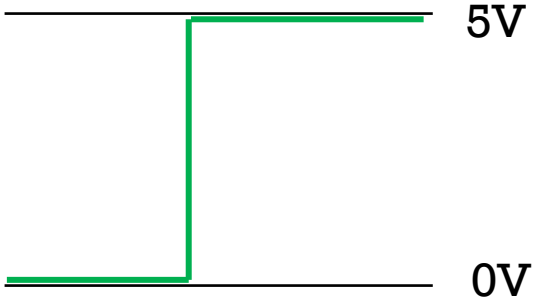


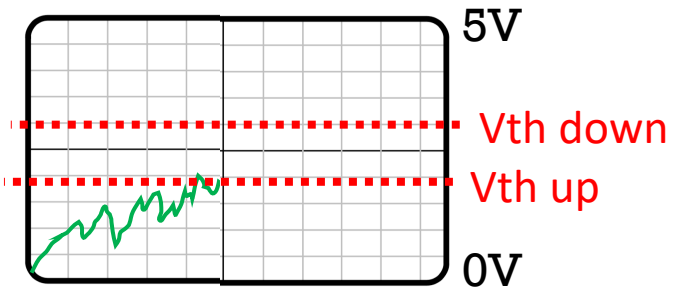
- R1 = 100KΩ 1V/1V
- R1 = 100KΩ
- R2 = 100KΩ || 100KΩ = 50KΩ
  (3.3V/2V – 1) 1.65
- R2 = 30KΩ

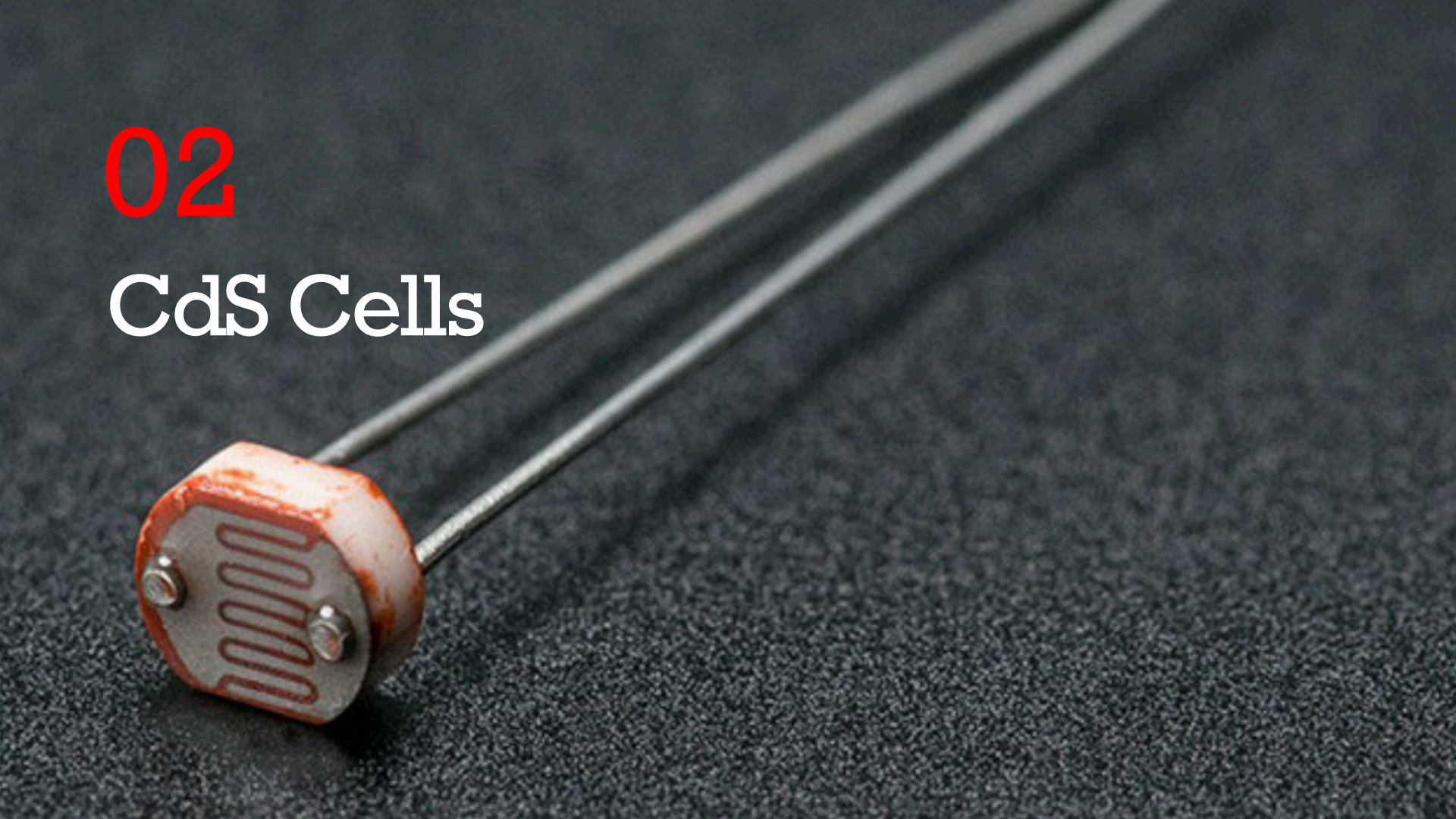
Comparators vs opamp



- High gain difference amp - Like an opamp
- Comparators need pullup, opamps don't
- Op amps not designed to be at rails (only subset of opamps are rail-to-rail)
- The MCP6044 is rail-to-rail and could theoretically be used as comparator even with hysteresis
- LM393 and LM339 available in GM lab







02

CdS Cells

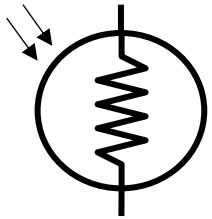
Photo Sensor Examples

- 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)

CdS PhotoCell

Not solar cells

Symbol

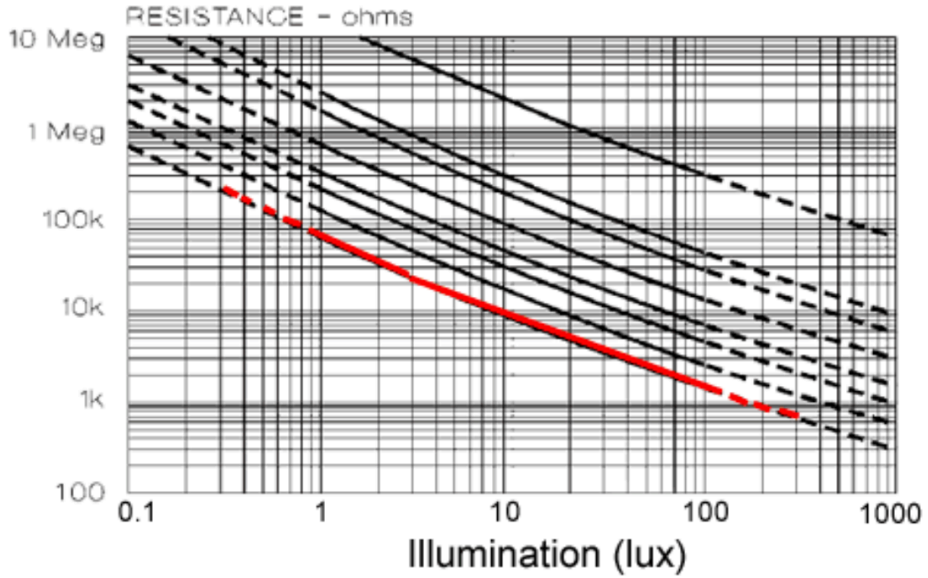


Variable resistor

CdS Resistance Varies with Incident Light

More Light = Lower Resistance

- Spectral Response



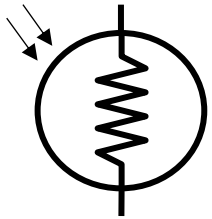
- Dynamic Response is slow

- 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

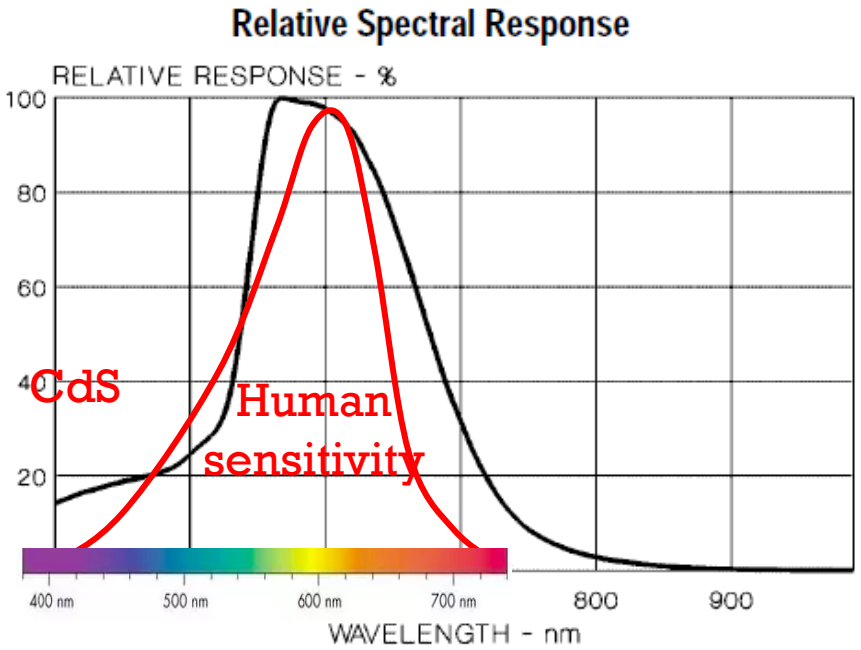
  (~1's – 10's of mS)

  Can act as filter!

- Power Rating

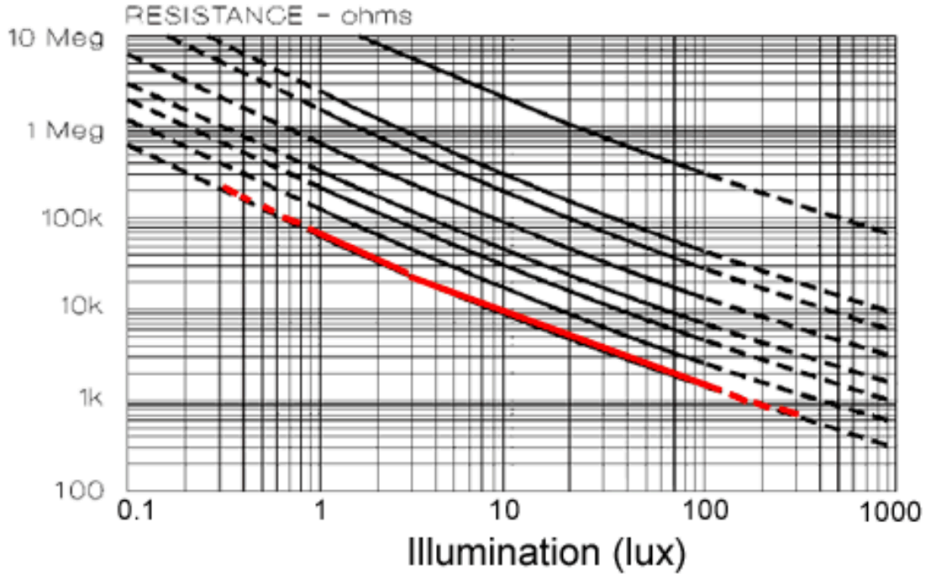
  Exceeding power generate heat

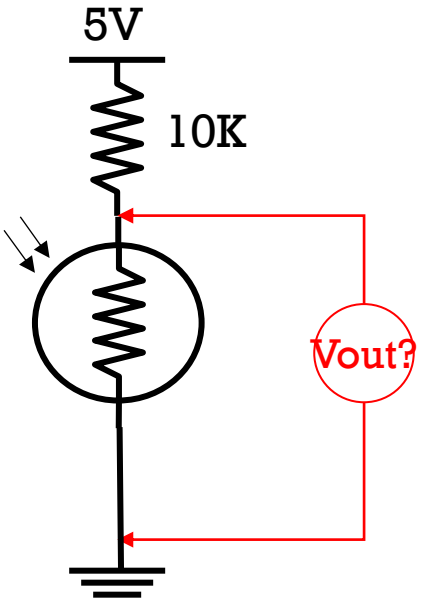
  Changes sensitivity



How do we use a CdS Cell?

- How is a variable resistor different than a potentiometer?
- Q6: If our light conditions vary from 10 to 500 lux, what is our voltage output range?
- Q7: I ase our (")





Summary

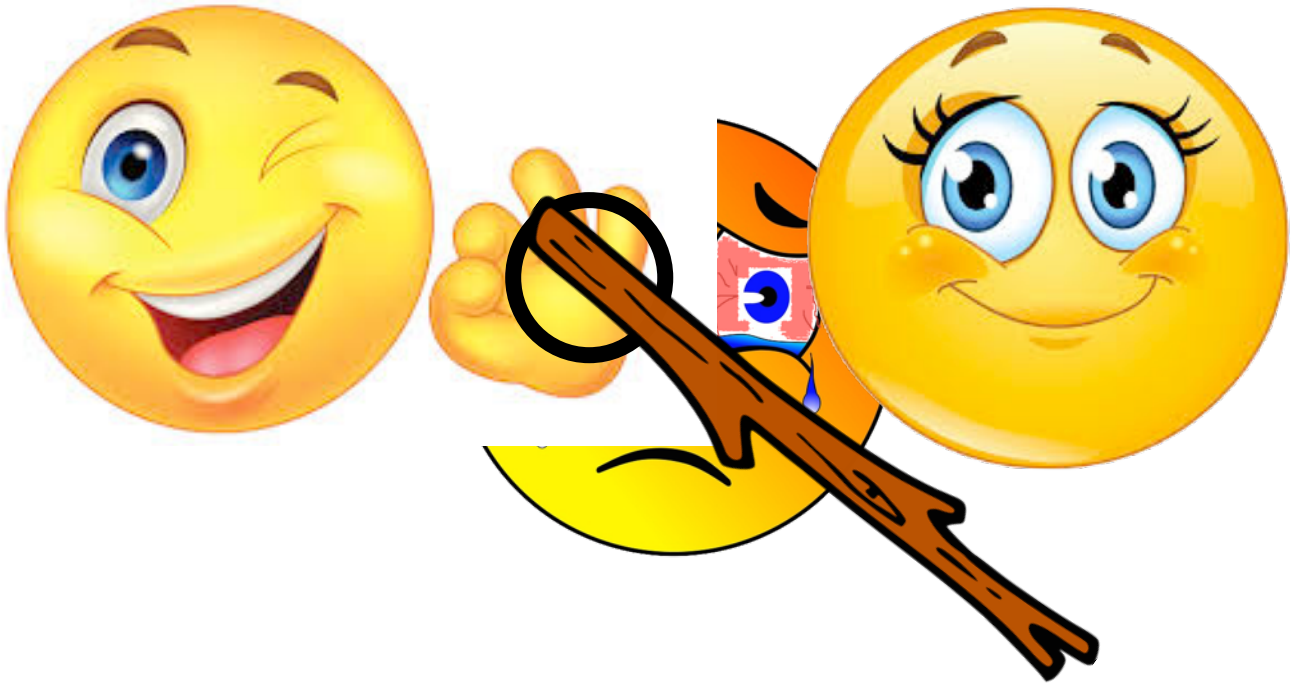
- Comparators can be used to output digital logic from analog thresholds
- Positive feedback resistors can add hysteresis
- CdS cells change resistance relative to visible light

03

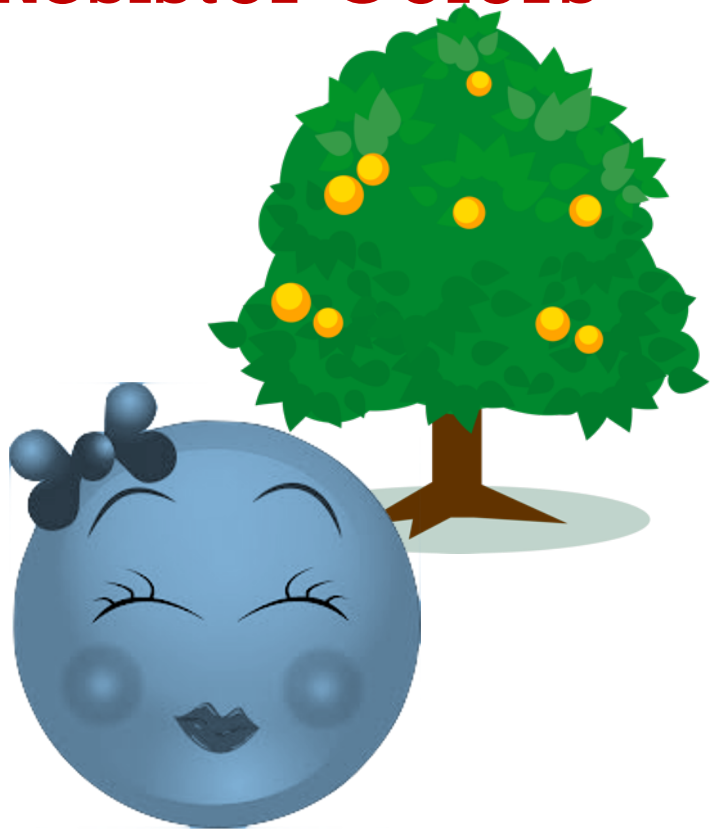
Resistor Colors (a dumb story)

Sorting resistors

Resistor Colors



Resistor Colors



Resistor Colors

Resistor Colors

Resistor colors

- What number is?
  - grey
  - yellow
  - red
  - orange
  - green
- What color is
  - six
  - zero
  - nine
  - one
  - seven

- What number is?
  - white
  - blue
  - brown
  - black
  - violet
- What color is
  - four
  - two
  - five
  - three
  - eight

Quiz Question

- Q8: What value is this resistor?

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

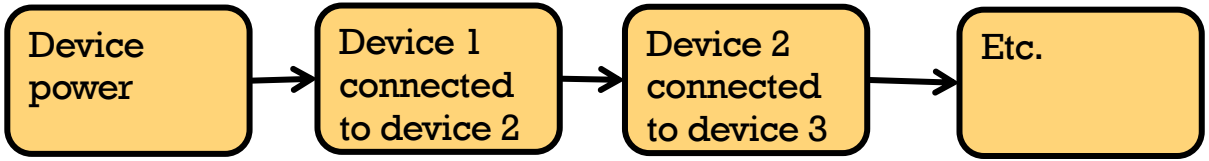
A. OpAmps

B. Comparators

C. Basic Coding Practice

Debugging Electronics

- Hardware and Software piece - need to check both
- Break your system apart into pieces
  - viewed as linked via inputs and outputs
  - Ideally a chain of parts with 1 output from one part going to 1 input in the next part.



- Check known voltage at each part starting with power and ground.

Debugging Code

- If you feel you need help with C or programming in general.

  `https://www.onlinegdb.com/`

- Has an online debugger that lets you track the flow of code.
- Note: ATmega specific things won't work (PORTD, Timers etc.)
- Looking into setting it up so we can actually debug ATmega code (e.g. put includes and fake stubs so things can compile).