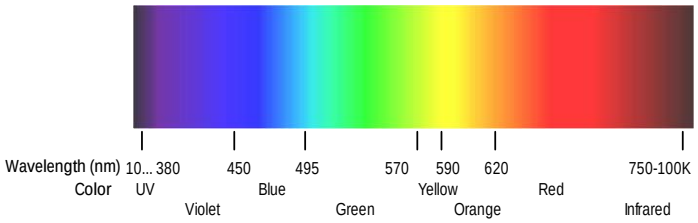
MECHATRONICS I

LABORATORY LECTURE & EXERCISES

LAB 3 - LIGHT DETECTOR (H)



PARTS LIST

(1) Computer, Arduino, & USB cable
(1) Breadboard
(n) Hook up or jumper wire
(1) Flashlight
(1) Partition

(2) Resistor (~ 2 kΩ)
(2) Phototransistor (or photo-resistors)

EXERCISE

Here we will learn how to obtain light data from a phototransistor using Arduino. The general goal with obtaining sensor data is to obtain an electrical signal that changes with the physical variable. We will refer to the sensor circuits as "light detectors".

BACKGROUND

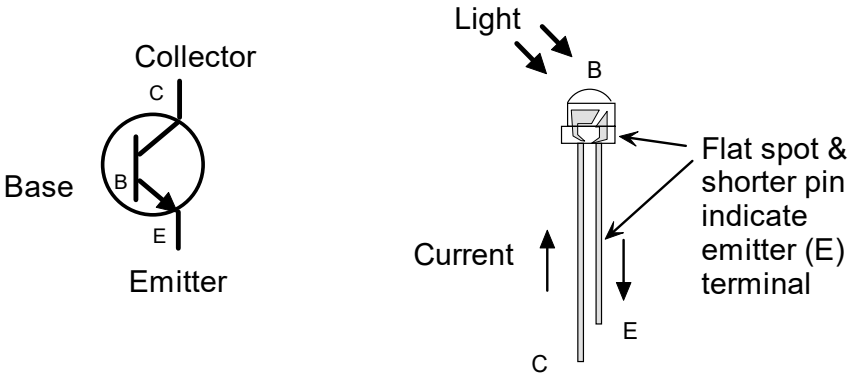
REFER to CT Guide (Ch 6.9 - transistors).

A transistor is an electronic switch with 3 legs. The legs are the base (B), collector (C), and emitter (E). In regular transistors, an electrical signal at B (base) determines a much higher current flowing between C (collector) & E (emitter).

Light is electro-magnetic radiation that oscillates in waves that have varying frequencies. Frequency is associated with wavelength.

If you are using phototransistors, note that they may look like clear LEDs, but they are not LEDs. They are transistors that use light as the "base" signal. If the phototransistor receives light at B, then it allows current to flow between C and E. Less light means less current. More light means more current. Our

phototransistor is most sensitive at 850 nm wavelengths (IR range) and visible light < 450 nm. Visible light sources (halogen, incandescent lights, the sun) emit lots of IR. The phototransistor works ideally indoors with fluorescent lighting. It works poorly outdoors or with halogen lights because these have too much IR interference.

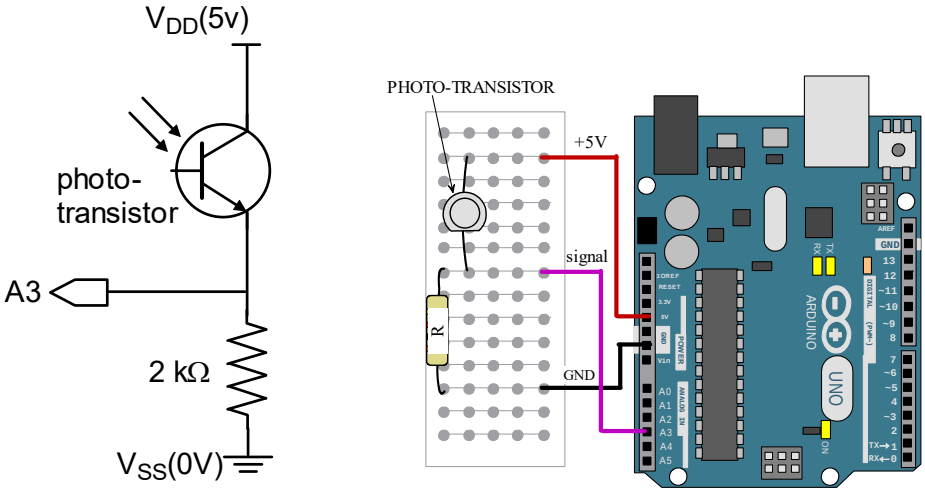


3.1 ANALOG VOLTAGE

EXERCISE

In this activity we will use a simple phototransistor (or photo-resistor) circuit to produce an analog voltage that represents the amount of light received by the phototransistor. This voltage will be passed to one of Arduino's analog inputs. When there is little light, the voltage should be around 0V. When there is bright light detected, the voltage should approach 5 V. The sensed voltage value will be output to the Serial Monitor.

CIRCUIT



Connect the components as shown. A3 is an analog input on Arduino. Arduino analog inputs use 10-bit analog-to-digital conversion over a range of 0 – 5 V. This means they break up the voltage (range: 0 – 5 V) into 1024 (2^{10}) levels. Vdd is 5 VDC. While the phototransistor controls how much current flows, the resistor is needed to transform current into a readable voltage. The resistor should be at least 2kΩ to restrict current draw on the Arduino to just a couple of mA.

HOW IT WORKS

Current flowing into the Arduino analog input is nearly zero (we'll estimate it is exactly 0). Thus, the current flowing through the phototransistor will be essentially the same the current flowing through the resistor R.

When no light is detected, no current flows through the phototransistor, and also through R. Ohm's Law relates voltage, current, and resistance in a resistor (V = IR). So if no current flows through R (I = 0), then there is no voltage difference across R (V = 0). Since R is connected to ground (GND) on one side (and GND has 0V). Then the other side of R must also be at 0V. This means 0V is passed to the analog input of Arduino.

When light is applied to the phototransistor, it allows current to pass. This same current passes through R. As current passes thru R, a voltage will form across R (Ohm's Law) that is proportional to the current flowing through it. Thus, a non-zero voltage will be passed to the analog input pin. A small voltage appears across the phototransistor, so the voltage at A3 will approach 5V but it won't go all the way to 5V.

Thus, an analog voltage will be applied to pin A3 that is related to how much light is received by the phototransistor. You can adjust the sensitivity of this circuit by changing the value of R. Higher R values mean less current will produce more voltage. This makes the circuit more sensitive to light. Lower R values will make the circuit less sensitive to light. If you are working with low amounts of light, it may help to use a higher R value.

CODE

```
// light1_volts.ino -----------------------
// display analog voltage output
// of phototransistor circuit (light detector) in serial monitor or plotter

void setup()
{
  Serial.begin(9600);                       //Set baud rate for serial monitor
}
void loop()                                  // Main loop auto-repeats
{
  Serial.print("A3 = ");         // Display "A3 = "
  Serial.print(volts(A3));            // Display measured A3 volts
  Serial.println(" volts");           // Display " volts" & newline
  delay(200);                       // update display ea second
}

float volts(int adPin)                // create function that measures volts
{                                     // Returns floating point voltage
  return float(analogRead(adPin)) * 5.0 / 1024.0;
}
```

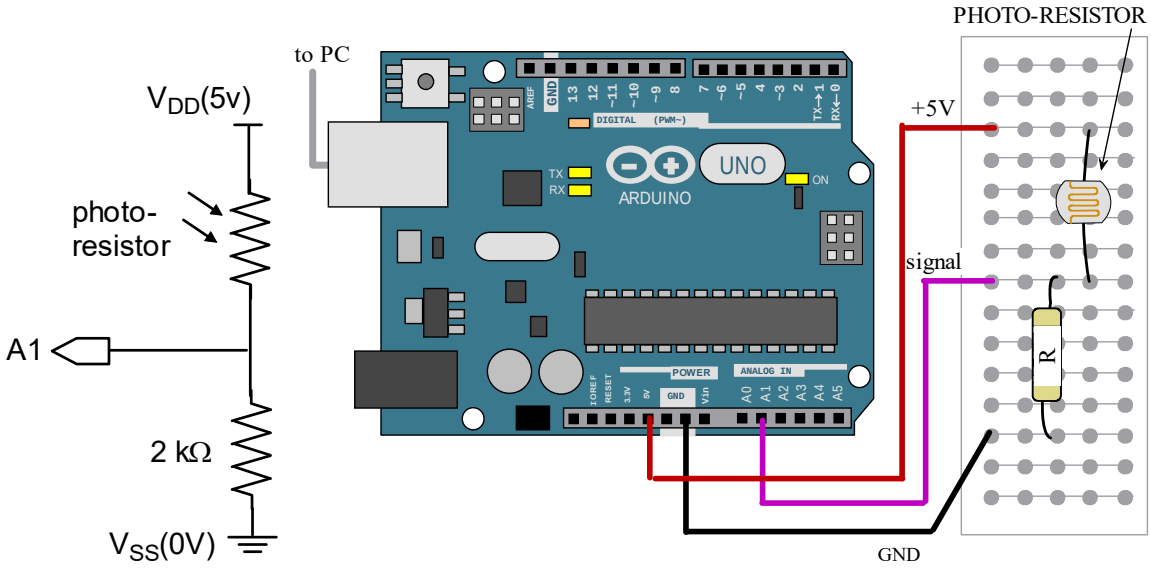
OUTPUT

In this case, use the Serial Plotter instead of the serial monitor. Variables written to the Serial Monitor can also be plotted on the Serial Plotter (Tools > Serial Plotter, or Ctrl-Shift-L). Watch how the plot responds to light applied to the phototransistor (use a flashlight). The range of values should be roughly 0 to 5V.

You can alternately plot "levels" (0 – 1023) using `Serial.println(analogRead(adPin));`
(don't use the "volts( )" function).

PHOTO-RESISTOR OPTION

The PHOTO-RESISTOR (or light-dependent resistor or LDR) can be used instead of a photo-transistor. It is a resistor whose resistance changes based on the amount of light received (~ 50 kΩ in near darkness, ~ 50 Ω in bright light). If we insert this into a voltage divider circuit, the voltage output will depend on the light received. We simply pass this voltage to an Arduino analog input. LDR's are becoming less common as they contain cadmium sulfide, which is considered a hazardous material and is not RoHS compliant (restriction of hazardous substances). However, LDR's are often used in Arduino kits. The photo-resistor circuit will work pretty much the same as that of the phototransistor. More light results in higher voltage at the analog input pin. Low light results in low voltage. This is due to the voltage divider rule for resistors in series.



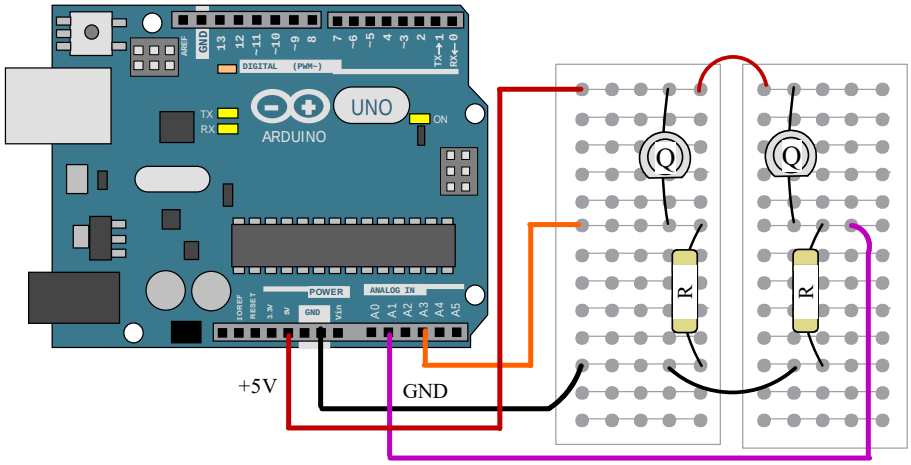
3.2 TWO-LIGHT DETECTORS

EXERCISE

Our solar tracker will need separate 2 light detector circuits. In this activity we will set up 2 light detectors and both signals sent to the serial monitor (or plotter).

CIRCUIT

Connect the components as shown. Basically you are making a duplicate of the prior circuit.



CODE

In this case we'll display the levels rather than the voltage.

```
// light2_two.ino -----------------------
//Display output of 2 phototransistor circuits

int sensorPin1 = A1;
int sensorPin2 = A3;

void setup() // -----------------------------------------
{
  Serial.begin(9600);
}

void loop() // ------------------------------------------
{
  int sensor1 = analogRead(sensorPin1);
  int sensor2 = analogRead(sensorPin2);

  Serial.print(sensor1);
  Serial.print("   ");
  Serial.println(sensor2);
  delay(100);
}


//end ----------------------------------------------------------
```

OUTPUT

Try the serial monitor but also the serial plotter. The plotter is easier to see. Shine a light on the two sensors separately and watch how the plot responds. Cover the photo-transistor with your finger to cut off light. The range of values should be 0 to about 900 or 1000.

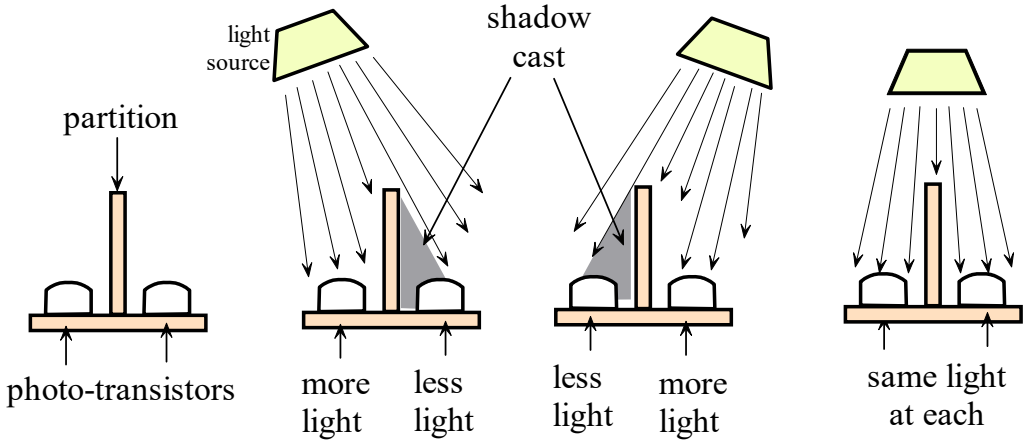
3.3 LIGHT DIFFERENCES

INTRO

Here you will collect data from 2 light detector circuits, compute the difference, and display that difference on the Serial Monitor. No servo is used here yet.

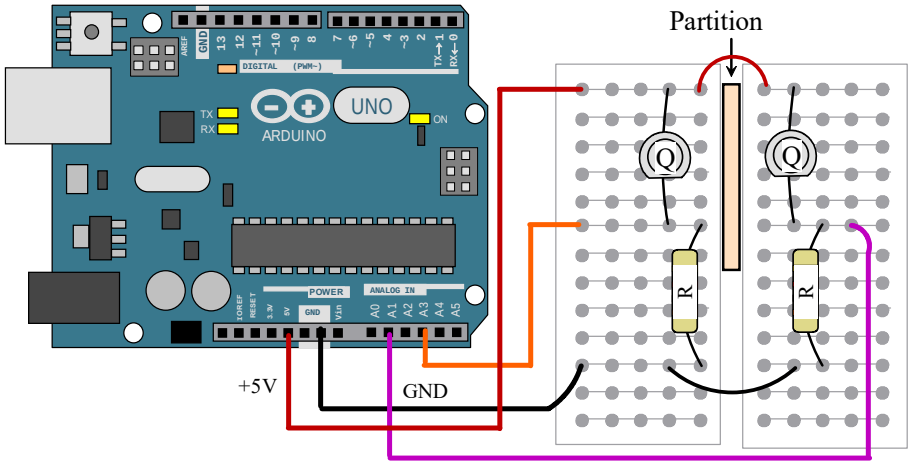
BACKGROUND

We are working towards making a solar tracker by moving in small steps. Here we need 2 light detector circuits with a separating partition between the two (e.g., a piece of cardboard). The separator or partition is essential for obtaining the control we seek. If the light is angled relative to the partition, the partition will cast a shadow on the far side, causing that sensor to receive less light than the sensor on the same side as the light source. The difference detected is an indication of how well the platform aligns with the light source (essentially what a solar tracker will do). The variable of interest here is the DIFFERENCE between the 2 signals.



CIRCUIT & CONNECTIONS

We still need 2 light detector circuits, but we will add a PARTITION to separate the 2 sensors.



CODE

In code, we will compute and display the differences in the output of those 2 circuits. In this case we are not converting the output into voltage, but leaving them as integers ranging from 0 to 1023 (10-bit conversion).

```
// light3_diff.ino --------------------
//Display DIFFERENCE in voltage of 2 phototransistor circuits
```

```
int sensorPin1 = A1;
int sensorPin2 = A3;

void setup() // ----------------------------------------
{
  Serial.begin(9600);
}

void loop() // ----------------------------------------
{
  int sensor1 = analogRead(sensorPin1);
  int sensor2 = analogRead(sensorPin2);
  int diff = sensor2 - sensor1;

  Serial.println(diff);
  delay(100);
}


//end ---------------------------------------------------------------
```

OUTPUT

While moving the light back and forth across the partition, observe the serial plotter. The difference should range between -1023 to 1023. It should be zero when equal amounts of light appear on each detector.

3.4 LIGHT DIFFERENCES SMOOTHED

EXERCISE

The plot from the last exercise likely shows a lot of noise. Noise appears as random jagged ups and downs in the plot. Nearly all analog signals will have noise. Later when we use this signal to drive a servo, the noise will cause the motor to "twitch", which is not desirable. In this exercise we will smooth the data using computational techniques. Wiring and connections are unchanged. No servo is used here.

BACKGROUND

The hardware is unchanged here from the last exercise, but the code changes. We want to reduce the motor twitchiness that comes from signal noise. There are a variety of ways to solve this problem. One way is to FILTER our sensor data to smooth it out. There are many filters. The simplest ones take a running average of the sampled data. Running averages take an average of the last "n" data points ("n" = number of data points... you get to choose how many!). This should smooth the data but it will also cause a delay because you cannot write commands to the motor until all n data points are collected. Thus, a larger "n" smoothes data more but it also delays the response more.

The running average filter described above works exclusively with the incoming (unfiltered) data. Another filter is an EXPONENTIAL FILTER. Exponential filters works with both the raw data (unfiltered) and the filtered data. A single weight variable w (ranging from 0 to 100%) weights how much raw vs. filtered data is used. Higher w weights the raw data more, making the result fast but less smooth. Lower w weights filtered data more, making the result smoother but less responsive. The exponential filter equation is quite similar to the result obtained when using so-called TRANSFER FUNCTIONS (beyond the scope of this class).

Here is a nice article on exponential filters
https://www.megunolink.com/articles/coding/3-methods-filter-noisy-arduino-measurements/

CIRCUIT

The hardware is mostly unchanged from the last exercise, but always double check that your physical pin connections match that of the code below.

CODE

The code below includes functions for both the running average and the exponential filter. However, the running average function is not used (it's not called). A variety of Serial.print statements appear. These are there to allow you to debug or view the response in the serial plotter.

```
// light4_smoothed.ino --------------------
// drive servo based on smoothed diff in light signal (using simple voltage circuit)
// via expon or running avg
// smooth both diff and servoAngle

int sensorPin1 = A1;
int sensorPin2 = A3;

void setup() // ----------------------------------------
{
  Serial.begin(9600);
}


void loop() // -----------------------------------------
{
  int light1 = analogRead(sensorPin1);
  int light2 = analogRead(sensorPin2);
  int diff = light2 - light1;
  int diffexpon = exponFilter(diff); //call exponFilter(), assign result to diffexpon

  Serial.print(diff);
  Serial.print("  ");                  // this space is important
  Serial.println(diffexpon);

  delay(40);
}

int exponFilter(int lightData) //-----------------------------------
{
  static int lastFilteredData;
  float w = .30;
  int y;
  y = w * lightData + (1 - w) * lastFilteredData;
  lastFilteredData = y;
  return y;
}

int runningAvg(int lightData) //---------------(not used) ------------
{
```

```
  static int i = 0;                   // index
  static int tot = 0;                 // running total
  const int imax = 5;                 // specify # readings
  static int readings[imax] = {0};    //initialize to 0's
  int avg = 0;

  tot = tot - readings[i];    //rem last reading
  readings[i] = lightData;    //insert new reading
  tot = tot + lightData;      //sum new set of readings
  i++;

  if (i >= imax)
  {
    i = 0;                    //reset index when get end of array
  }
  avg = tot / imax;           //compute avg
  return avg;
}

//end ----------------------------------------------------------------
```

OUTPUT

While shining light onto and off of he light sensor, observe both the unfiltered and filtered plots on the serial plotter (not serial monitor). The filtered data should appear much smoother than the unfiltered data. The signal should range from -1023 to +1023. Try changing the "w" value in the exponential filter function to see how it affects performance.

Upon completion of this exercise, do NOT take apart your circuit. You will need it for the next lab.

LAB SIGN-OFF FORM
(for students to print out)

Student Name: _____

1. _____	light1_volts	light detector outputs analog voltage
2. _____	light2_two	2 light detectors
2. _____	light3_diff	2 light detectors, display difference
3. _____	light4_diff-smoothed	2 light detectors, smoothed, displayed

SUBMISSION:

Students will demonstrate the working exercises to the instructor in class.

A possible option is uploading videos to Canvas.
The instructor will let you know which is acceptable.
See the rules for file uploads to Canvas.