Introduction:

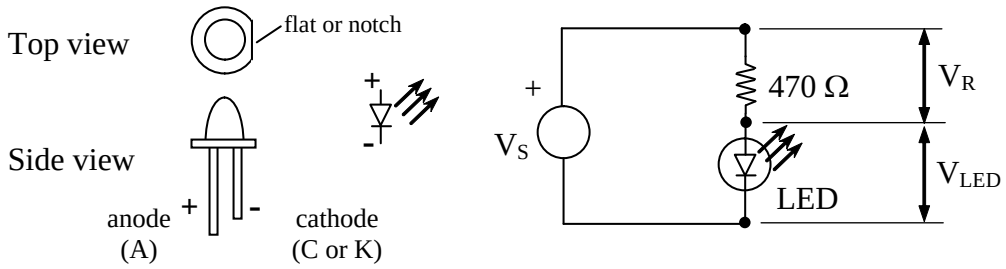
The resistance of the CdS (cadmium-sulfide) photoresistor while exposed to the ambient light in the room is around 0.568770 kOhm. When covered, its resistance is around 10.5 kOhm.

> The reading of the resistance while it was uncovered was very unstable, even if it was moved by a little. This is probably due to the varying light intensity of the fluorescent lights in the room for each different spot in the room.

470 Ohm == Yellow Purple Brown

> Actual reading of 470 Ohm resistor == 465.33 Ohm
> Anode is positive side of LED and is longer lead; put it at higher potential.

(Lab Report Q1) After the circuit was constructed, the supply voltage V_S was varied from 1 to 5 Volts in 1-Volt increments. At each voltage increment, the voltage across the LED V_{LED} was measured, and the voltage across the resistor V_R was measured. Additionally, the LED current was calculated by applying Ohm's Law across the resistor. All three of these values, and a comment on the LED's brightness, were input into the table below.



(Lab Report Q2) The current through the LED is the same as the current through the resistor because the two components are in series. The current through the LED was calculated by applying Ohm's Law across the 465.33 Ω resistor.

$$V_R = I \times R$$

$$I = \frac{V_R}{465.33\ \Omega}$$

V_S, [V]	V_{LED}, [V]	V_R, [V]	Current (Calc.), [mA]	Current (Measured), [mA]	Comment on LED Brightness
1	1.0050	0.00000	0	0.0001	Not visible
2	1.7147	0.33519	0.720328	0.7451	Extremely Dim
3	1.8220	1.2079	2.595792	2.6169	Bright
4	1.8979	2.1218	4.559775	4.6900	Brighter
5	1.9803	3.0875	6.635076	6.6258	Extremely Bright

Recall that to measure current with DMM:

a. Move the red cable from the HI terminal to the I terminal; keep the other cable (black cable) in the LO terminal
b. Break the circuit, putting the cables in between the resistor and the LED.

c. DON'T measure current across the resistor because current will avoid flowing through the resistor and flow through the DMM (current flows through the path of least resistance), causing a lot of current to go through the DMM and through the LED!

(Lab Report Q3) The theory of operation of the circuit in Figure 2 is that the LED should turn off as the CdS is covered. The reason is because the resistance of the CdS is high when there is less ambient light hitting it, i.e. when it is darker. The high resistance of the CdS would then impede current going into the LED in series.

(Lab Report Q4) A good supply voltage to use for this second circuit is 2.53 V. This voltage is high enough so that when the CdS is uncovered, it has low resistance, and enough current flows to light the LED; and the voltage is low enough so that when the CdS is covered, the LED is barely lit. The circuit from Figure 2 in the Lab Manual was then constructed to check its function. (Lab Report Q5) The function of the circuit in Figure 2 is a light-controlled LED. The voltage supply supplies power to activate the circuit. The CdS then absorbs the ambient light: its resistance is based upon the amount of light hitting the photocell – when the area is bright, the CdS resistance is low, and more current will flow to light up the LED brighter.

The Light-Controlled Switch Using a Transistor

(Lab Report Q6) *Calculation for Figure 3*

Given: V = 10V, R_{CdS} = 100 kΩ, R_C = 220 Ω, and transistor gain h_{fe} (or β_{DC}) = 100

Transistor Base Current "I_B" can be found by:

$$I_B = \frac{V_{IN} - V_{BE}}{R_B}$$

$$I_B = \frac{V - (V_B - V_E)}{R_{Cds}}$$

$$I_B = \frac{10\,V - (0.7\,V)}{100\,k\Omega}$$

$$I_B = \ 0.093\,mA$$

Then Transistor Collector Current "I_C" can be found by:

$$I_C = h_{fe} \times I_B$$

$$I_C = (100) \times \ (0.093\,mA)$$

$$I_C = \ 9.3\,mA$$

(Lab Report Q7) After determining the collector current, the circuit shown in Figure 4 of the Lab Manual was constructed. To do so, the following values were to be determined: a voltage supply "V_S" value; the value of the photoresistor's on-resistance "R_{ON}"; a properly selected "R_1" resistor was calculated and chosen; and a properly selected "R_C" resistor – the current limiting resistor – was calculated and chosen.

1. The first design choice made in a project, the supply voltage, used was V_S = 5 V.
2. The value of the photoresistor's on-resistance used was the value when it was uncovered, which was 0.568770 kΩ, so R_{ON} = 0.568770 kΩ.

R1 and the photoresistor form a voltage divider. Through Voltage Division across R_1, the value of R_1 was calculated to be 92.59 **Ω**, but the closest actual resistance used was R_1 = 100.13 **Ω**.

$$0.7\ V = V_S \times \left(\frac{R_1}{R_1 + R_{ON}}\right)$$

$$0.7 = 5 \times \frac{R_1}{R_1 + 568.770\ \Omega}$$

$$R_1 = 92.59\ \Omega$$

$$R_1 = 100.13\ \Omega$$

100 Ohm == Brown Black Brown

Or use the Trim Pot (connect the DMM to pins 1 and 2 of the trim pot, and rotate CCW to decrease its resistance).

3. The value of the current limiting resistor "R_C" had to be selected so that the LED current was limited to be a value between 10 mA and 15 mA. In the following equation, V_{LED} was the value of the forward voltage drop across the LED when it was at 5 V, which was 1.9803 V (from the first table). V_{sat} was the value of the saturation voltage across the transistor's collector and emitter, and a reasonable value for the 2N3904 transistor was 0.4 V**.**

$I_{MAX} = \frac{(V_S - V_{LED} - V_{sat})}{R_C}$	
$15\ mA = \frac{5\ V - 1.9803\ V - 0.4\ V}{R_C}$	$10\ mA = \frac{5\ V - 1.9803\ V - 0.4\ V}{R_C}$
$R_C = 174.6\ \Omega$	$R_C = 261.97\ \Omega$

The value of R_C had to be between these values:

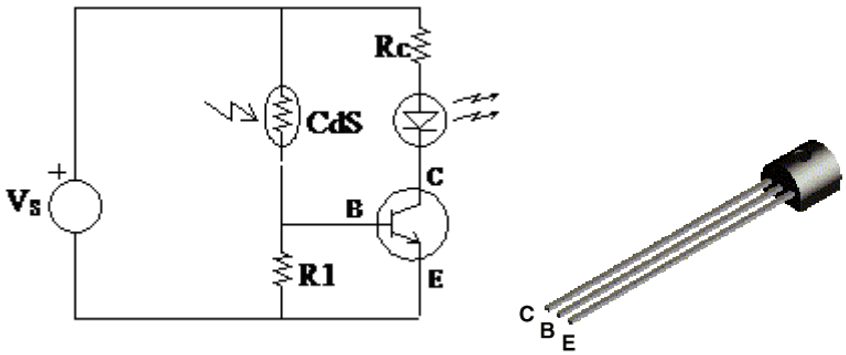
$$174.6 < R_C < 261.97$$

The chosen value for the current limiting resistor was R_C = 202.12 **Ω**, corresponding to a maximum current of 12.9623 mA.

200 Ohm == Red Black Brown

Or use the Trim Pot (connect the DMM to pins 1 and 2 of the trim pot, and rotate CCW to decrease its resistance). Got the trim pot to 202.72 Ohm, so Rc = 202.72 Ohm.

With properly selected values, the circuit from Figure 4 of the Lab Manual was constructed.

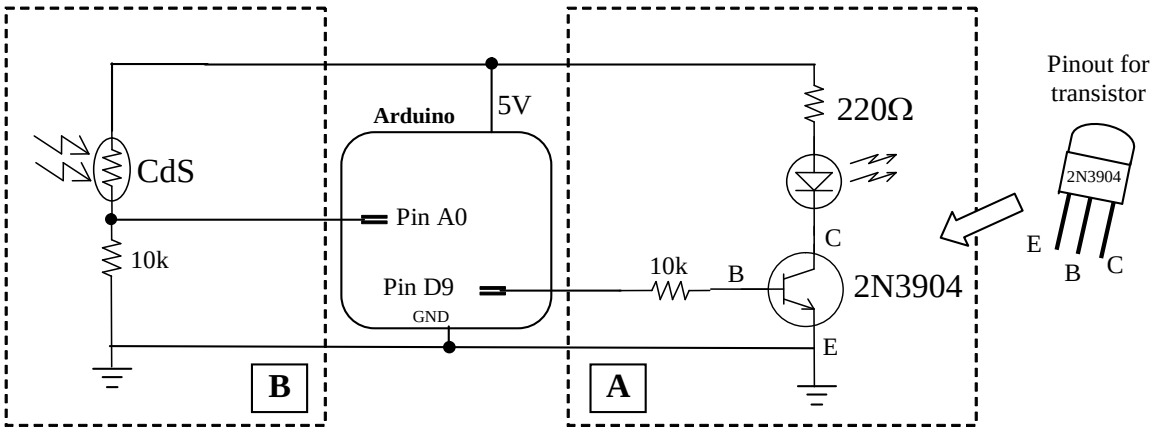


Insert picture of working circuit here.

When uncovered, LED == ON
When covered, LED == OFF
Tips:
Make sure Collector, Base, and Emitter are on SEPARATE Rails; don't want to short them
R1 and Emitter are connected to COM ground
Anode (+) is longer lead and is at higher potential, closer to Rc

Using the Arduino to Make a Programmable Light-Controlled Switch

10 kOhm == Brown Black Black Red
220 Ohm == Purple Red Black Black



(Lab Report Q11) Circuit "A" is the output circuit with respect to pin D9 because it contains the LED. Digital Pin 9 should be initialized as OUTPUT in setup().

(Lab Report Q12) 5 V on Digital Pin 9 will turn the transistor on.

For a transistor to turn on, the voltage difference between the Base and the Emitter "V_{BE}" must be greater than a threshold voltage, taken to be 0.7 V for an ideal transistor at room temperature. Additionally, saturation mode is the ON-mode of the transistor. In this mode, the transistor acts like a short circuit between the Collector and the Emitter, and both of the diodes in the transistor are forwards biased. This means: $V_{BE} > 0$, $V_{BC} > 0$, i.e. $V_B > V_C$ and $V_B > V_E$ ("Transistors") && (Alciatore & Histand, 2011, pp. 91-98).

Because the voltage difference between the Base and the Emitter "V_{BE}" is greater than 0.7 V, the transistor is on. To check whether the transistor is saturated when it is turned on, the following calculations were made:

When Digital Pin 9 is at 5 V:

I_C = 21.82 mA, as determined by the 220 Ohm resistor and V_{CE} = 0.2 V in Saturation.

I_B = 0.43 mA, as determined by the 10 kOhm resistor and V_{BE} = 0.7 V in Saturation.

$$I_C = \frac{V_S - V_{CE}}{R_C} = \frac{5\ V - 0.2\ V}{220\ \Omega} = 21.82\ mA$$

$$I_B = \frac{V_S - V_{BE}}{R_B} = \frac{5\ V - 0.7\ V}{10\ k\Omega} = 0.43\ mA$$

~~$$I_B = \frac{I_C}{100} = \frac{21.82\ mA}{100} = 0.2182\ mA$$~~

~~$$I_B = \frac{V_{in} - V_{BE}}{R_B}$$~~

~~$$V_{in} = I_B \times R_B + V_{BE} = (0.2182\ mA) \times (10\ k\Omega) + 0.7\ V$$~~

~~$$V_{in\,min} = 2.882\ V$$~~

~~2.882 V is the minimum input voltage required for saturation. Since the input voltage, the voltage at pin D9, is at 5 V and larger than 2.882 V, the transistor is in saturation mode. When PORTC0 is at 5V, I_B and I_C were calculated above.~~

The Blink program found in Appendix A of the Lab Manual was used for the following exercise. Refer to Appendix A of this Lab Report for the source code.

> **(Question 8-10 Points) After verifying, compiling, and downloading the sketch onto the Arduino UNO board, the LED blinked at a frequency of 1 Hz. This corresponds to the C-language's bitwise Exclusive-Or operator: '^='. The Exclusive-Or operator is commonly used either for switching the values of selected bits or for testing for inequivalence. This is done by determining whether a pair of bits are different (i.e. one of the bits is a 0, and the other is a 1); if the two bits are different, the result is a 1 (a value of 1 corresponds to a value of HIGH, or ON). The output on Pin D9, which is the LED connected to 10 kΩ resistor and the transistor, is toggled using the C-language's bitwise Exclusive-Or operator because the program continuously evaluates whether the bits are different. Then the LED is turned on using the bitwise Or operator: '`|=`'.**
>
> **Additionally, the variable called `ledIsOn` is declared as a `static byte` within the `loop()` function, or the `main()` function. When used within a function declaration (which is the `loop()` function, in this case), the reserved word `static` retains the value of `ledIsOn` throughout the execution of the program, so that the program remembers whether the LED is on during the next pass of the `loop()`. Another way to accomplish what the `static` variable does here is probably to declare the `ledIsOn` variable as a global variable. This way, the `ledIsOn` variable will not get rewritten every time at the beginning of the loop; it will also be available to all functions in the code, rather than just in `loop()`.**

Below is a slightly modified version of Appendix A. The changes made were: making ledIsOn a global variable, and ledIsOn ^= 1 was changed to ledIsOn = ledIsOn ^1 for better transparency.

> ~~**static global variable, outside of the `main()`, or `loop()`, function. When used for global variables, the reserved word `static` makes the variable visible to all functions from this point of declaration in this source file only. The variable is not visible to functions defined in other source files, and therefore cannot be accessed and modified by other programs.**~~

```
/*
 * Blink LED at 1000/(DELAY_TIME_MSECS * 2) Hz.
 * This code assumes that the LED is active-high.
 */

#define PORTB_LED_MASK 0b00000010 // LED is on PB1 (Arduino D9)

#define DELAY_TIME_MSECS 500      // msecs between toggles

byte ledIsOn = 0;
```

```
/*
 * initializations - just setup the one LED in this case.
 */
void setup()
{
  /*  setup Digital Pin 9 as an Output (1 indicates Output)  */
  /*  PORTB_LED_MASK has value 0b00000010  */
  DDRB  = PORTB_LED_MASK;    // LED on PORTB is an output

  /*  setup initial LED state to OFF  */
  PORTB = 0x00;
}

void loop()
{
  /*  The static keyword retains the value of ledIsOn
   *  throughout the execution of the program, so that
   *  the program remembers whether the LED is on
   *  during the next pass of the loop().   */
  //static byte ledIsOn=0;

  /*  If ledIsOn == 1 (true), turn LED on */
  /*  PORTB = PORTB | PORTB_LED_MASK      */
  /*  PORTB = 0b00000000 | 0b00000010     */
  if (ledIsOn)
  {
    PORTB &= ~PORTB_LED_MASK;   // LED -> off
  }

  /*  If ledIsOn == 0 (false), turn LED off */
  /*  PORTB &= ~PORTB_LED_MASK;             */
  /*  PORTB = 0b00000000 & ~0b00000010      */
  else
  {
    PORTB |= PORTB_LED_MASK;    // LED -> on
  }

  /*  ledIsOn = 0 ^ 1; ---> Result is a 1   */
  ledIsOn = ledIsOn ^ 1;
  //ledIsOn ^= 1;                 // toggle ledIsOn

  delay(DELAY_TIME_MSECS);
}
```

(Lab Report Q14) After Circuit "A" was constructed, and after the program from Appendix A was uploaded onto the Arduino UNO, Circuit "B" was constructed.

10 kOhm == Silver Black Black Red

The AnalogReadSerial that is found in the Example menu (Basics menu) of the Arduino IDE was used for this exercise. It was modified to read the input voltage on Pin A0, and then display this value on the Serial Monitor. Pin A0 was connected to the CdS. The modified AnalogReadSerial program was run. When the CdS photoresistor was covered, around 445 bits was displayed on the Serial Monitor, and this corresponds to a low voltage of 0 V. When the CdS was uncovered, around 934 bits was displayed on the Serial Monitor, and this corresponds to a high voltage of 5 V. These reading correlate to the changing value of the voltage divider (voltage divided between the CdS and the 10 kOhm resistor). The voltage divider is changing as a function of the photoresistor. The range of values displayed on the monitor were within 400 and 900 bits.

Note: Analog readings return something in bits, something within the range of 0 to 1023 bits, while digital readings return only either 0 V (LOW) or 5 V (HIGH) (unless it's a common anode LED display like in Lab 4)

```
/*  (I) Variable & Constants */
```

```
#define HUGE_LED 9
#define CDS_PHOTORESISTOR A0


/*  (II) setup()  */
void setup()
{
  pinMode(HUGE_LED, OUTPUT);

  /* initialize serial communication at 9600 bits per second:  */
  Serial.begin(9600);
}


/*  (III) loop()  */
void loop()
{
  /*  Read the input on analog pin 0:  */
  int sensorValue = analogRead(CDS_PHOTORESISTOR);

  /*  Print out the value you read:  */
  Serial.println(sensorValue);

  /*  delay in between reads for stability  */
  delay(1000);
}
```

(Lab Report Q15A) For the next exercise, a program was to be made that will turn the LED on when the photoresistor is covered. A threshold value of 850 Ω was chosen, above which the LED is in the off state. This value was to be between the 800 Ω and 967 Ω range. Source code is below; threshold value of 850 Ohms highlighted.

```
/*  (I) Variable & Constants */
#define HUGE_LED 9
#define CDS_PHOTORESISTOR A0
#define PORTB_LED_MASK 0b00000010 // LED is on PB1 (Arduino D9)


/*  (II) setup()  */
void setup()
{
  /*  setup Digital Pin 9 as an Output (1 indicates Output)  */
  /*  PORTB_LED_MASK has value 0b00000010  */
  DDRB = PORTB_LED_MASK;

  /*  setup initial LED state to OFF  */
  PORTB = 0b00000000;

  /* initialize serial communication at 9600 bits per second:  */
  Serial.begin(9600);
}


/*  (III) loop()  */
void loop()
{
  /*  Read the input on analog pin 0 (the CdS)  */
  int sensorValue = analogRead(CDS_PHOTORESISTOR);

  /*  Print out the value you read:  */
  Serial.println(sensorValue);
```

```
  /*  If CdS is covered, turn LED on  */
  /*  PORTB = PORTB | PORTB_LED_MASK  */
  /*  PORTB = 0b00000000 | 0b00000010 */
  if(sensorValue < 850)
  {
    PORTB |= PORTB_LED_MASK;    // LED -> on
  }

  /*  If CdS is uncovered, turn LED off */
  /*  PORTB &= ~PORTB_LED_MASK;         */
  /*  PORTB = 0b00000000 & ~0b00000010  */
  else if (sensorValue >= 850)
  {
    PORTB &= ~PORTB_LED_MASK;   // LED -> off
  }

  /*  delay in between reads for stability  */
  delay(50);
}
```

To have the LED stay on under bright ambient light conditions and turn off when a shadow falls upon the photoresistor (so that it is the opposite of the previous program is what should happen), one change has to be made. The logic test within the if-statement must be switched around to:

```
if (sensorValue > 850)
```

(Lab Report Q16) For the next exercise, a program, called photo_blinkie, was to be made that blinks the LED, where the LED blinks faster when the CdS is covered more completely, and blinks slower when more light falls upon the photoresistor's photocell. The Arduino `map()` function was used to scale the range of 550 – 850 bits to 100-200 milliseconds, respectively. The photo_blinkie program is below.

```
/*  (I) Variable & Constants */
#define HUGE_LED 9
#define CDS_PHOTORESISTOR A0
#define PORTB_LED_MASK 0b00000010 // LED is on PB1 (Arduino D9)


/*  (II) setup()  */
void setup()
{
  /*  setup Digital Pin 9 as an Output (1 indicates Output)  */
  /*  PORTB_LED_MASK has value 0b00000010  */
  DDRB = PORTB_LED_MASK;

  /*  setup initial LED state to OFF  */
  PORTB = 0b00000000;

  /* initialize serial communication at 9600 bits per second:  */
  Serial.begin(9600);
}


/*  (III) loop()  */
void loop()
{
  /*  The static keyword retains the value of ledIsOn
   *  throughout the execution of the program, so that
```

```
   *  the program remembers whether the LED is on
   *  during the next pass of the loop().   */
  static byte ledIsOn=0;

  /*  Read the input on analog pin 0 (the CdS)  */
  int sensorValue = analogRead(CDS_PHOTORESISTOR);

  /*  Print out the value you read:  */
  Serial.println(sensorValue);

  /*  LED blinks faster when CdS is more covered.
   *  So need a smaller delayTime
   *  CdS has lower resistance if covered.  */
  int delayTime;
  delayTime = map(sensorValue, 550, 850, 100, 200);

  /*  If CdS is covered, turn LED on  */
  /*  PORTB = PORTB | PORTB_LED_MASK  */
  /*  PORTB = 0b00000000 | 0b00000010 */
  if(sensorValue < 850)
  {
    if(ledIsOn)
    {
      PORTB |= PORTB_LED_MASK;     // LED -> on
    }
    else
    {
      PORTB &= ~PORTB_LED_MASK;    // LED -> off
    }
    ledIsOn = ledIsOn ^ 1;
    delay(delayTime);
  }

  /*  If CdS is uncovered, turn LED off */
  /*  PORTB &= ~PORTB_LED_MASK;         */
  /*  PORTB = 0b00000000 & ~0b00000010  */
  else if (sensorValue >= 850)
  {
    PORTB &= ~PORTB_LED_MASK;    // LED -> off
  }

  /*  delay in between reads for stability  */
  delay(50);
}
```

(Lab Report Q17) For the last exercise, the breadboard circuit was detached from the Arduino, and a servo-photoresistor combination device was connected to the Arduino. The four wires on the device are: red for 5 V, black for GND, white for servo control (in actuality, it was a green wire), and yellow for photoresistor sensing. A program was to be made that tracks a light source; the photoresistor detects from where the light is coming from, and then the servo turns towards the light source. The upper and lower analog photoresistor range were used.

```
/*  (I) Include libraries and other Tabs */
#include <Servo.h>

/*  (II) Variable & Constants & Objects  */
#define CDS_PHOTORESISTOR A2
Servo myservo;

/*  (III) setup()  */
void setup()
{
```

```
  myservo.attach(10);
  pinMode(CDS_PHOTORESISTOR, INPUT);
  Serial.begin(9600);
}

/*  (IV) loop()  */
void loop()
{
  int sensorValue = analogRead(CDS_PHOTORESISTOR);
  Serial.println(sensorValue);
  sensorValue = map(sensorValue, 0,1023, 180, 0);
  myservo.write(sensorValue);
  delay(10);
}
```