

digitalRead()

Now we would like to get information into the micro-controller. A first step in the direction is to use the digital pins to a *digital* measurement of the voltage applied to a pin.

A digital measurement is rather crude in that it only determines if the voltage on the pin is HIGH or LOW. HIGH and LOW are measured with respect to 2.5 V — i.e. halfway between 0 and 5 V. If the voltage on a pin is less than 2.5 V, then it is measured as being LOW, and if it is higher than 2.5 V, then it is HIGH. If it happens to be exactly 2.5 V, then the micro-controller will probably be confused and it might indicate either result.

Although the measurement is binary, this is often enough to make decisions about the external environment.

.

The first step in using `digitalRead` is to use `pinmode()` to set the pin function as an INPUT.

```
pinmode(8, INPUT);
```

To read the input, use the `digitalRead` function, which returns an integer value, either 0 or 1 (LOW or HIGH).

```
inPut = digitalRead(8);
```

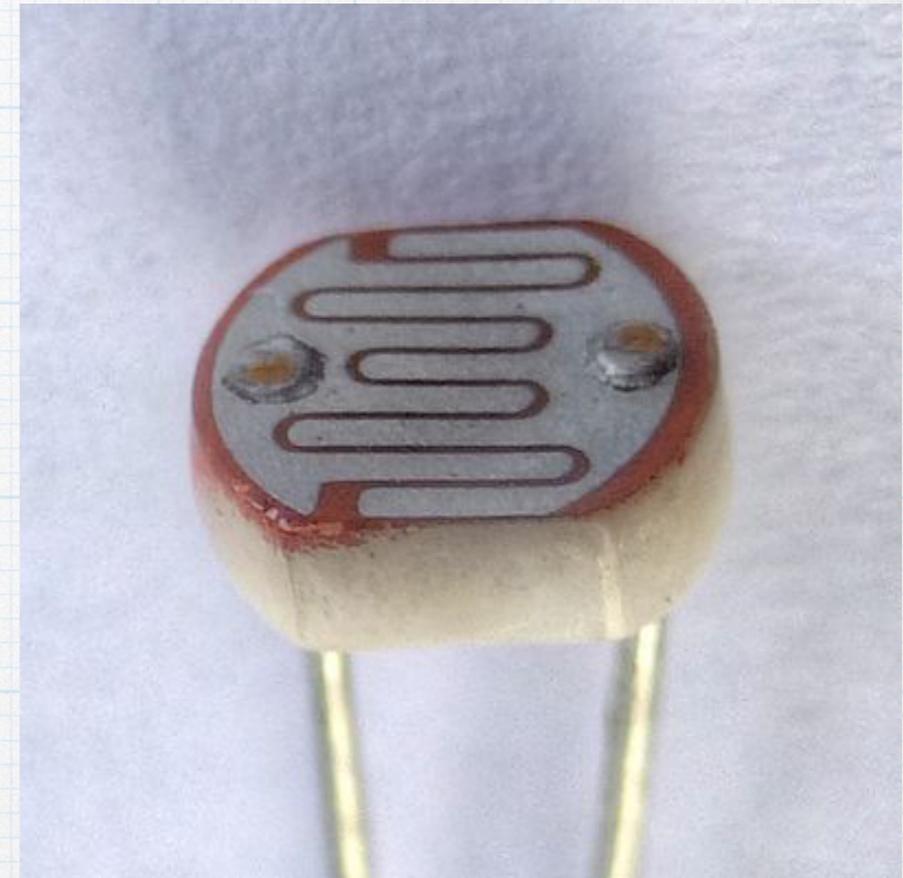
It's that easy.

For further details about `digitalRead` check the Language Reference.

Important note: `pinmode()` commands are not restricted to being in the `setup()` function. You can change the modes as the main part of the program is repeating in the `loop()` portion.

Example - photoresistive light sensor

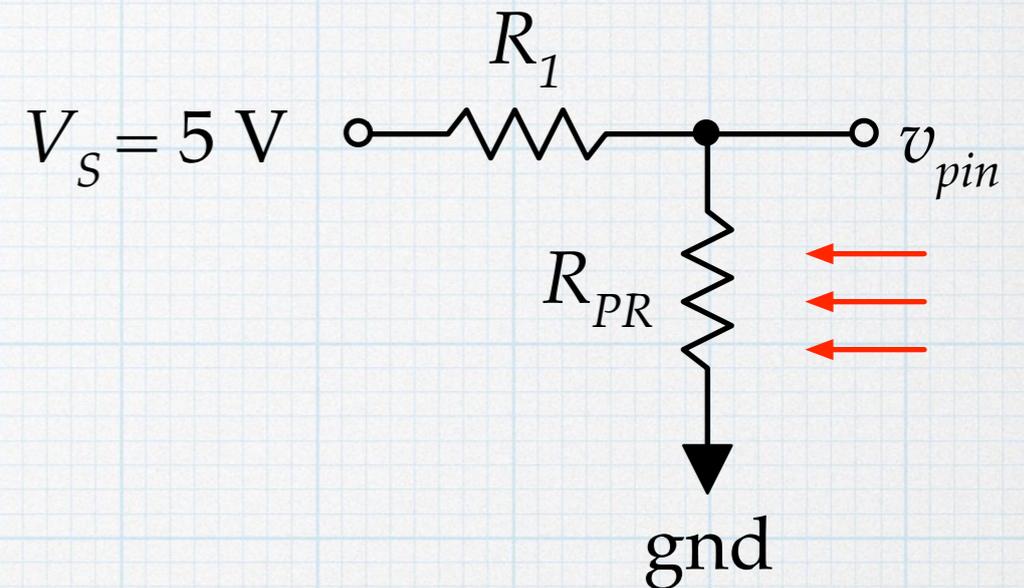
A photoresistor is a very simple type of light detector. It consists of a resistor made with a semiconducting material — cadmium sulfide (CdS) or similar — which is exposed to ambient light. The semiconducting material can absorb the incident photons, which create extra charge carriers (electrons and holes). As the charge carrier concentration increases, the resistance will go down. In essence, this is a light-dependent variable resistor.



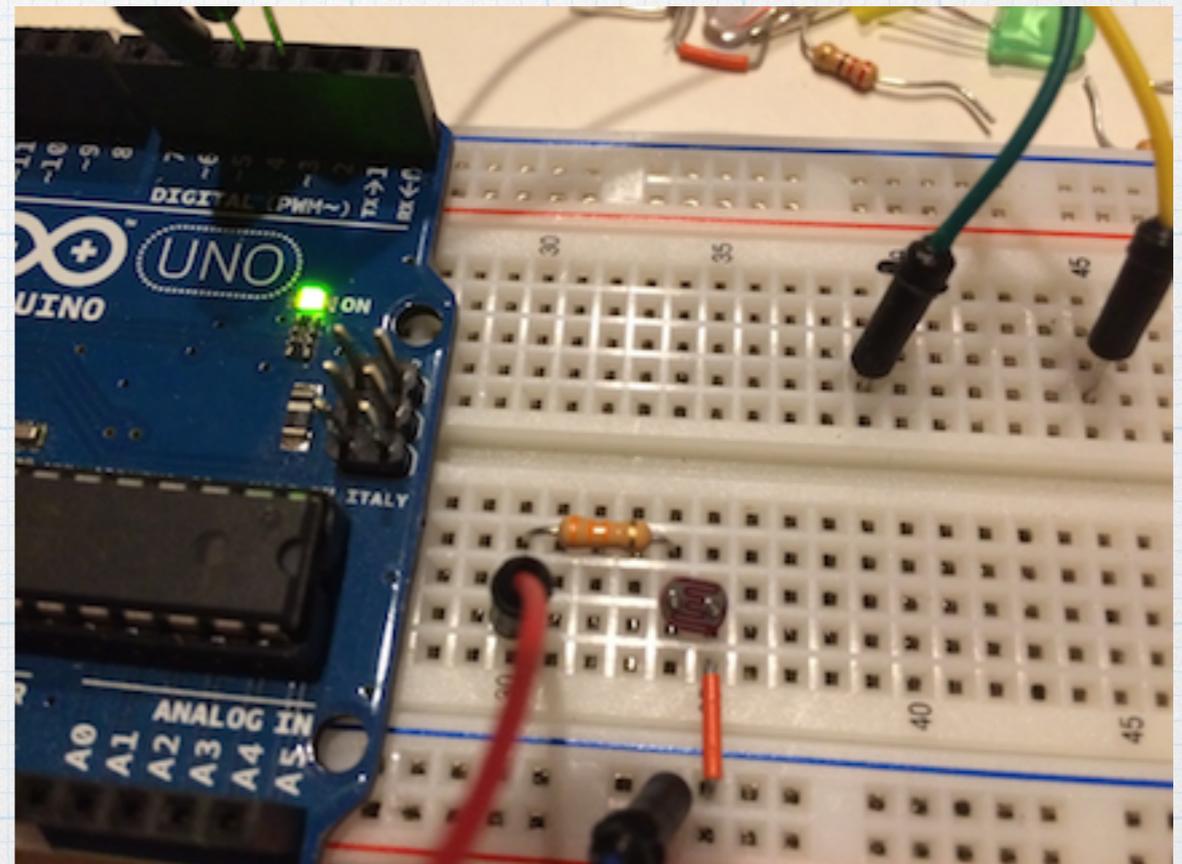
A photoresistor can exhibit a very large change in resistance value for a modest change in the ambient light. This makes it nicely suited for use in digital decision making: Is it day or night? Are the room lights on or off?

A photoresistor can be used with a fixed resistor in a simple voltage divider arrangement to provide a voltage that changes with incident light.

$$v_{pin} = \frac{R_{PR}}{R_{PR} + R_1} V_S$$

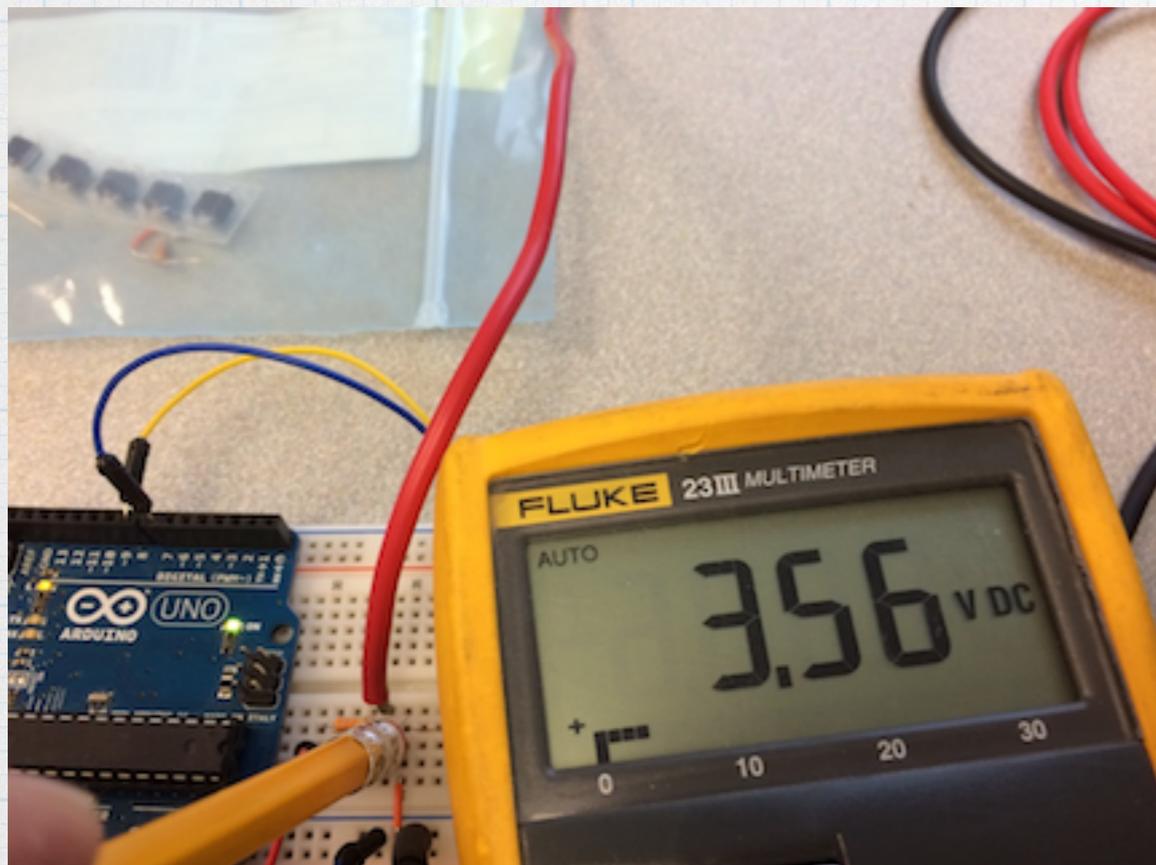


As the photoresistor resistance changes with variations of the incident light, the voltage across the photoresistor changes accordingly.

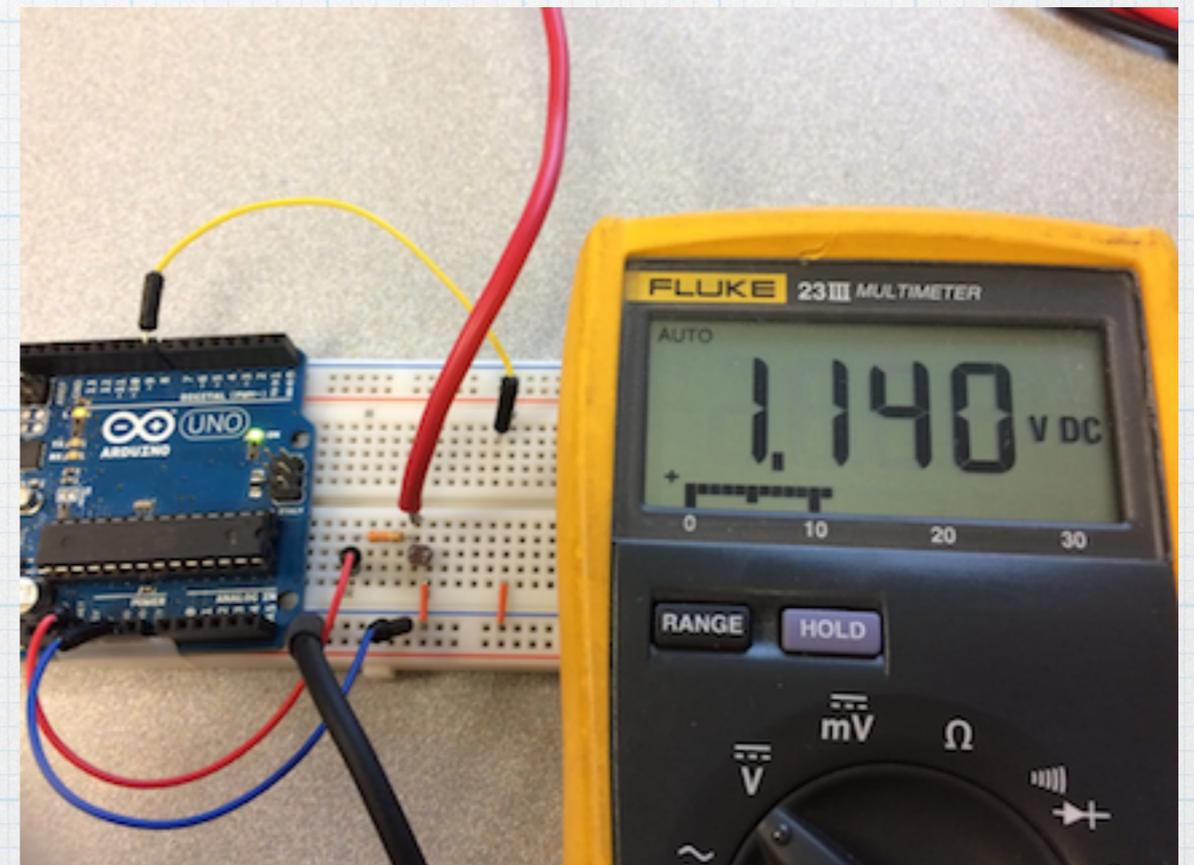


For the photoresistor used here (PDV-9002-1 from Luna optoelectronics, included in EE lab 230 kits). The dark resistance is $> 150 \text{ k}\Omega$. In typical room light, it measures about $15 \text{ k}\Omega$. Using it with a $33\text{-k}\Omega$ fixed resistor to make a simple voltage divider with a 5-V supply, the dark and light voltages are measured as shown below. The difference between dark and light will be easily discernible using a digital measurement.

dark



light



We can use the sensor input to initiate some action. In this case, we will simply turn on an LED when it is dark.

The photoresistor voltage is connected to pin 7, which is configured as an input. The LED is connected to pin 8 — configured as an output — with a 220- Ω limiting resistor.

The setup is straight-forward. In the loop, the voltage on the sensor is checked each time through the loop. If the voltage is high (meaning that the room is dark), the LED is turned on. Otherwise it is set to be off.

digital_in_1 §

```
int pinValue;
int inPin = 7;
int ledPin = 8;

void setup() {

    pinMode( inPin, INPUT );
    pinMode( ledPin, OUTPUT );
    digitalWrite( ledPin, LOW );
}

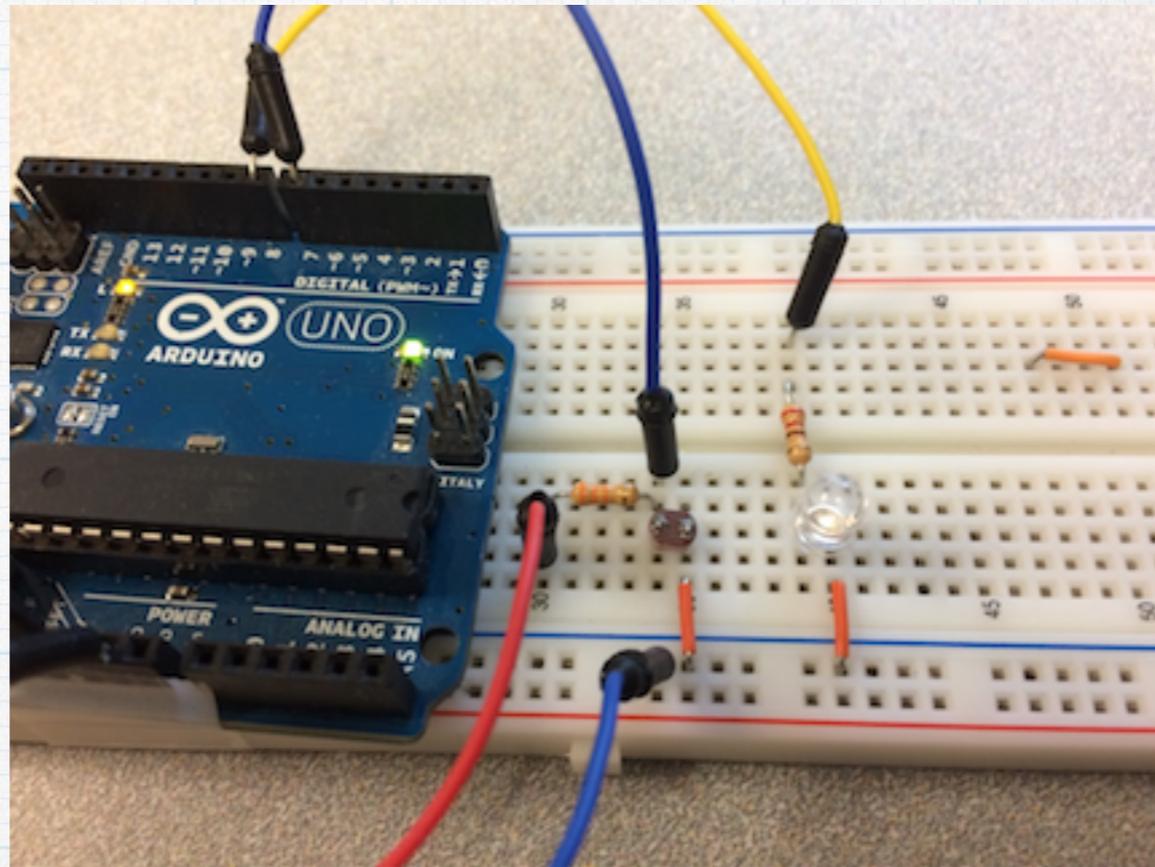
void loop() {

    pinValue = digitalRead( inPin );

    if( pinValue == HIGH )
        digitalWrite( ledPin, HIGH );
    else
        digitalWrite( ledPin, LOW );

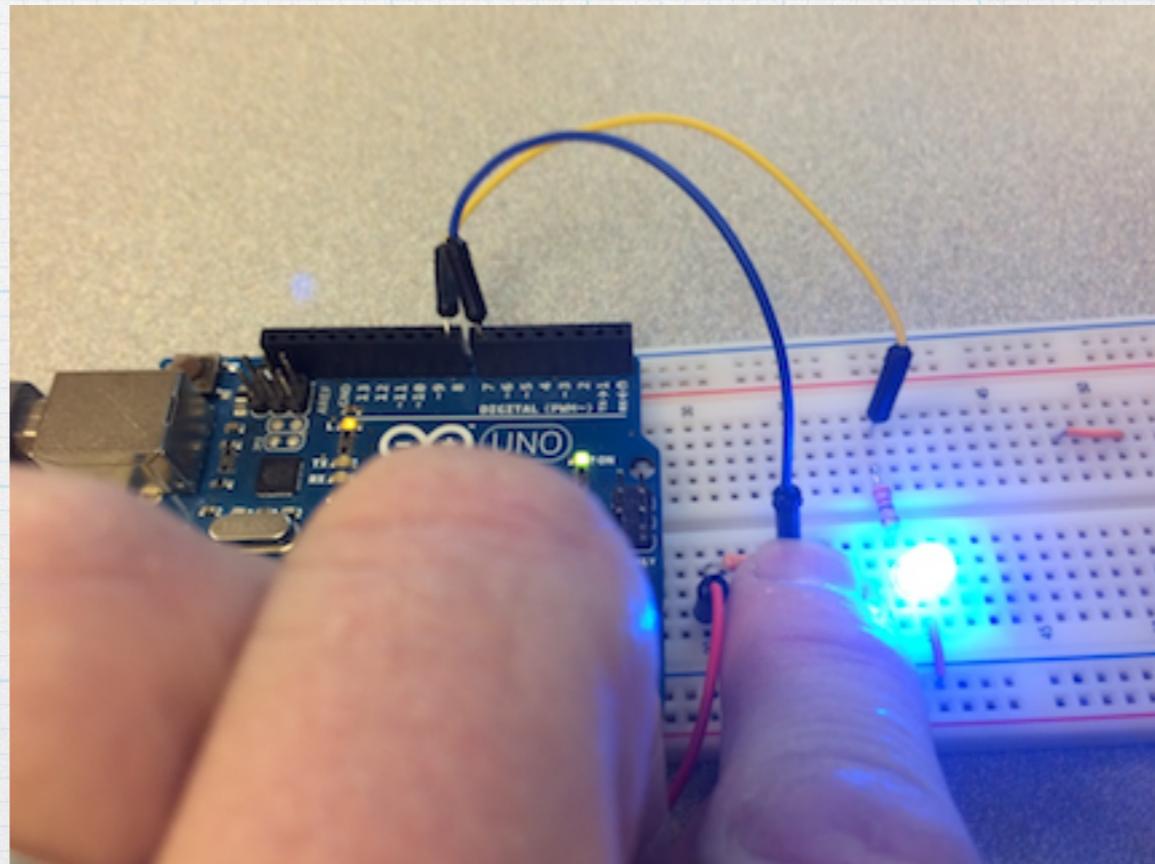
    delay( 100 );
}
```

Room light is incident on the sensor, so the LED is turned off.



No light hitting the sensor, so the LED is turned on.

Of course, the micro-controller can initiate many different actions based on the sensor input.



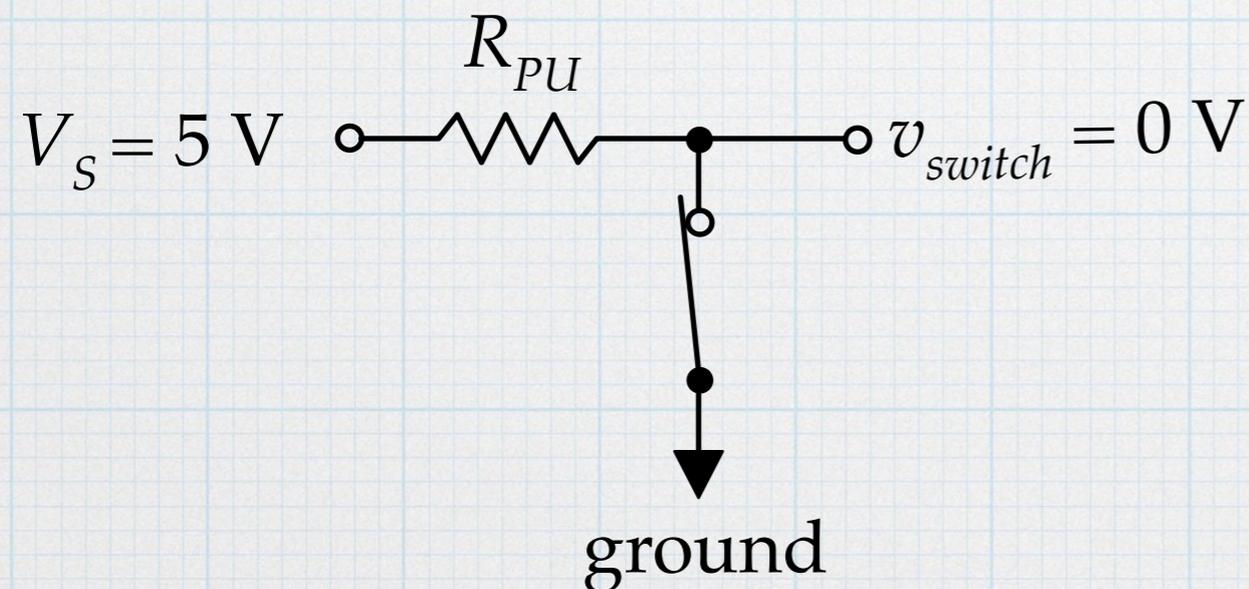
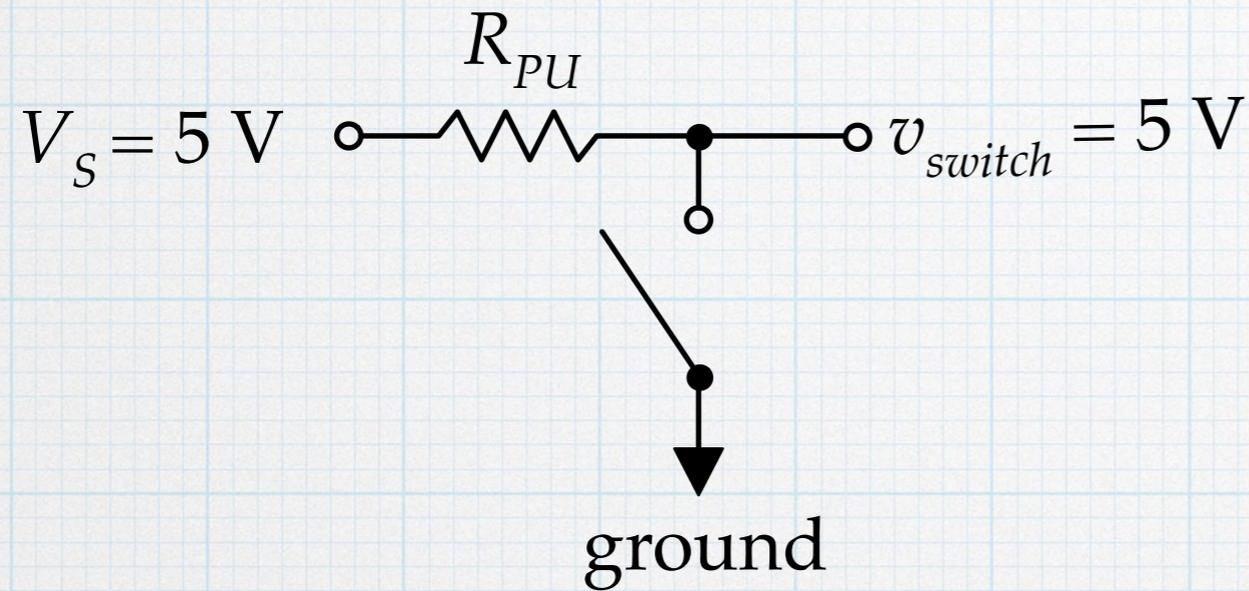
Here is a slight variation on the previous program. It is not necessary to set the value of the output pin every time through the loop. The only time when needs to be set is when the input changes. By using an extra variable, we can keep track of the current input state and watch for when the input changes. Once a change at the input has been detected, then the output can be changed.

This program functions similar to the previous one, but it changes the output only when the sensor value changes. The variable `currentSensorValue` has the current light on/off state. Each time through the loop, the sensor value is checked — `checkSensorValue` — and if this is different than `currentSensorValue`, the output is flipped, and `currentSensorValue` is updated.

```
digital_in_2 §  
  
int currentSensorValue;  
int checkSensorValue;  
int inPin = 7;  
int ledPin = 8;  
  
void setup() {  
  
    pinMode( inPin, INPUT );  
    pinMode( ledPin, OUTPUT );  
    currentSensorValue = digitalRead( inPin );  
    digitalWrite( ledPin, currentSensorValue );  
}  
  
void loop() {  
  
    checkSensorValue = digitalRead( inPin );  
  
    if( checkSensorValue != currentSensorValue ){  
        currentSensorValue = checkSensorValue;  
        digitalWrite( ledPin, currentSensorValue );  
    }  
    delay( 100 );  
}
```

Switches

A simple mechanical switch can be combined with a "pull-up" resistor to create a voltage divider whose value can be varied between zero and V_S .



The value of R_{PU} can be anything, but in order to limit the current drawn when the switch is closed, make it bigger. A value of $10\text{ k}\Omega$ — leading to a current of 0.5 mA is probably reasonable.

The program using the switch is similar to the previous one with the sensor. The program keeps track of the current state of the switch. Each time through the loop, the switch voltage is measured and compared to the current stored value. When a change in switch voltage is detected (because the switch has either opened or closed), the LED is turned on or off in response and the current switch state is updated.

Recall that the switch is voltage is HIGH when the switch is open and LOW when the switch is closed.

switch_1

```
int currentSwitchValue;
int checkSwitchValue;
int switchPin = 7;
int ledPin = 8;

void setup() {

    pinMode( switchPin, INPUT );
    pinMode( ledPin, OUTPUT );
    currentSwitchValue = digitalRead( switchPin );
    digitalWrite( ledPin, !currentSwitchValue );
}

void loop() {

    checkSwitchValue = digitalRead( switchPin );

    if( checkSwitchValue != currentSwitchValue ){

        currentSwitchValue = checkSwitchValue;
        digitalWrite( ledPin, !currentSwitchValue );
    }
}
```

De-bouncing

In the loop:

1. check the switch value.
2. Is it different than the current value? If not, then do nothing.
3. If yes, then wait a bit for any switch bounce to subside. (debounce time)
4. Check the switch value again. If it is the same as initial measurement, then this is a valid switch change. Take the desired action - in this case, change the state of the LED

switch_2

```
int currentSwitchValue;
int checkSwitchValue;
int switchPin = 7;
int ledPin = 8;
int debounceTime = 50;

void setup() {

  pinMode( switchPin, INPUT );
  pinMode( ledPin, OUTPUT );
  currentSwitchValue = digitalRead( switchPin );
  digitalWrite( ledPin, !currentSwitchValue );
}

void loop() {

  checkSwitchValue = digitalRead( switchPin );

  if( checkSwitchValue != currentSwitchValue ){

    delay( debounceTime );
    if( digitalRead( switchPin) == checkSwitchValue ){

      currentSwitchValue = checkSwitchValue;
      digitalWrite( ledPin, !currentSwitchValue );
    }
  }
}
```