Digital input

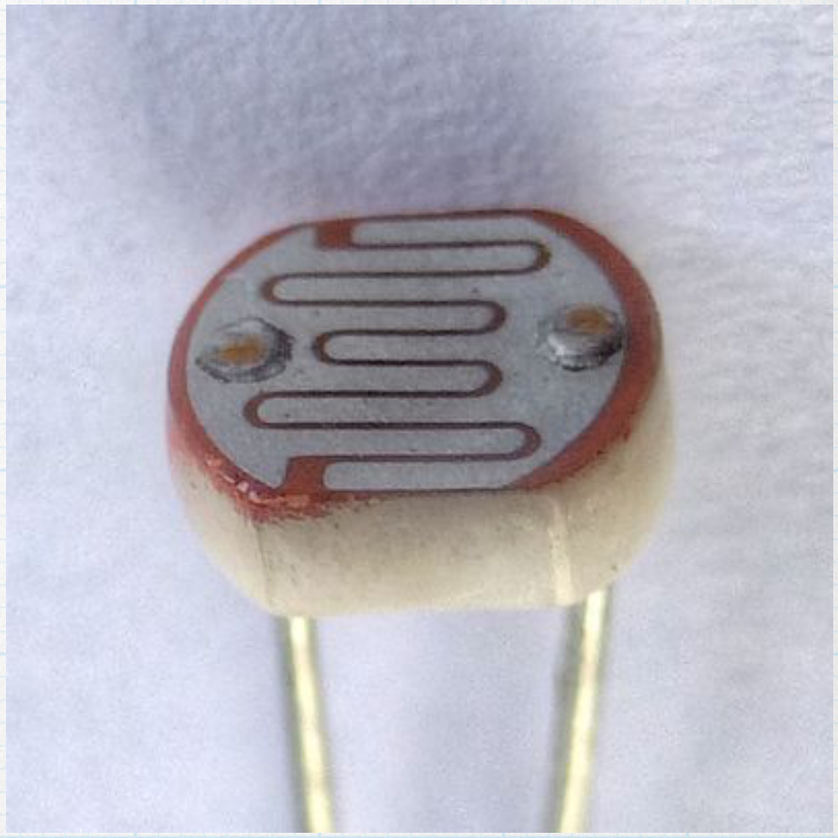
- The digital pins on the Arduino can be set to either read a digital input or write a digital output. We will discuss the two modes in separate documents.
- A digital input reads a voltage as being either HIGH or LOW. The dividing line between the two is $V_{CC}/2$. (2.5 V for $V_{CC} = 5$ V.) It works like a comparator. (See EE 230.)
- Two steps are required: first set the input/output mode to output with `pinMode( pin, INPUT )`, where `pin` is the number of the specific digital pin and `INPUT` specifies that pin will be reading an input. (`INPUT` is a standard keyword). This is usually in the setup function, but can be in the repeat function if needed.
- Then read the level: `x = digitalWrite( pin )`, where `x` is an integer variable and `pin` is the pin number. Usually in the loop function, but can be in either.

Example

```
//Read the input of digital pin 4. Use it to set the
//output of digital pin 6.

int inPin = 4;   //Sensor input
int outPin = 6;  //Control output
int x;

void setup()
{
  pinmode( inPin, INPUT );
  pinmode( outPin, OUTPUT );
}

void repeat()
{
  x = digitalRead( inPin );
  if (x == HIGH )
     digitalWrite( outPin, HIGH);
  else
      digitalWrite( outPin, LOW );
  delay( 1000 );
}

     Consider: digitWrite( outPin, digitalRead( inPIN ));
```

Example - photoresistive light sensor

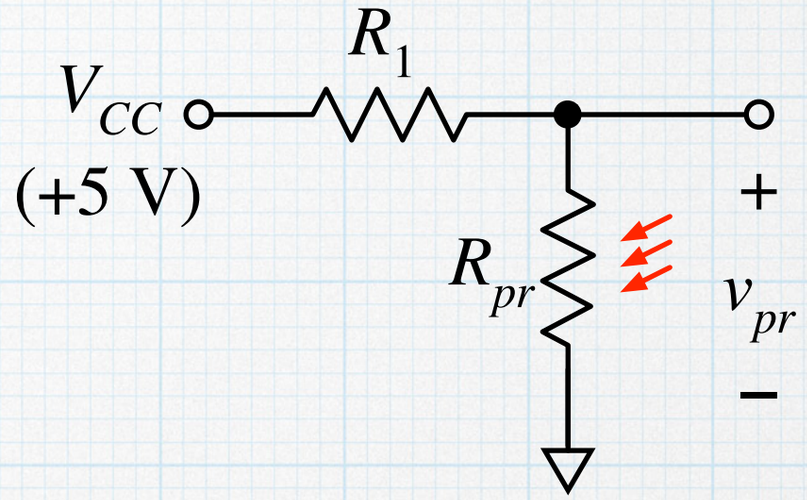
- Simple type of light detector.
- Resistor made with a light-sensitive semiconductor — cadmium sulfide (CdS) or similar.
- When exposed to ambient light, the semiconductor absorbs the incident photons, which create extra charge carriers — electrons and holes.



- As the carrier concentration increases, the resistance goes down. In essence, this is a light-dependent variable resistor.
- The change in resistance can be quite large.
- Well-suited for yes/no decisions: Is it day or night? Are the room lights on or off?

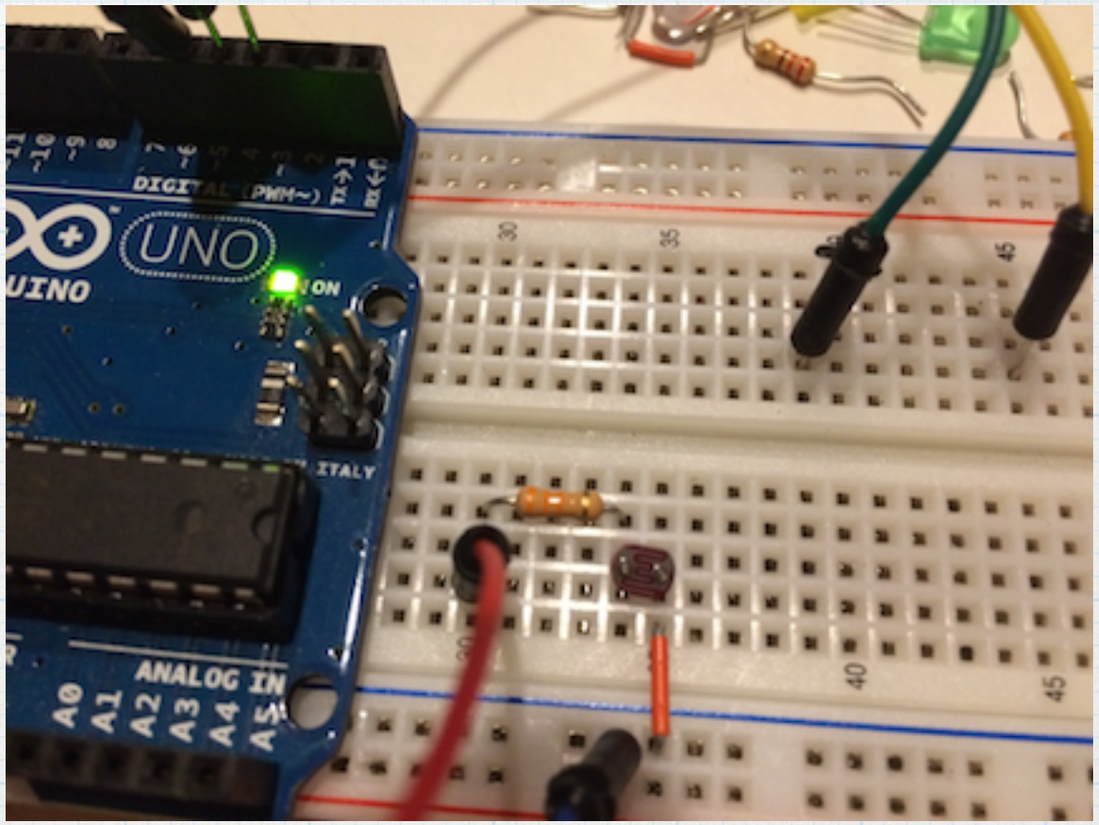
A simple application is to pair a photoresist with a fixed resistor to make voltage divider.

$$v_{pr} = \frac{R_{pr}}{R_{pr} + R_1} V_{CC}$$



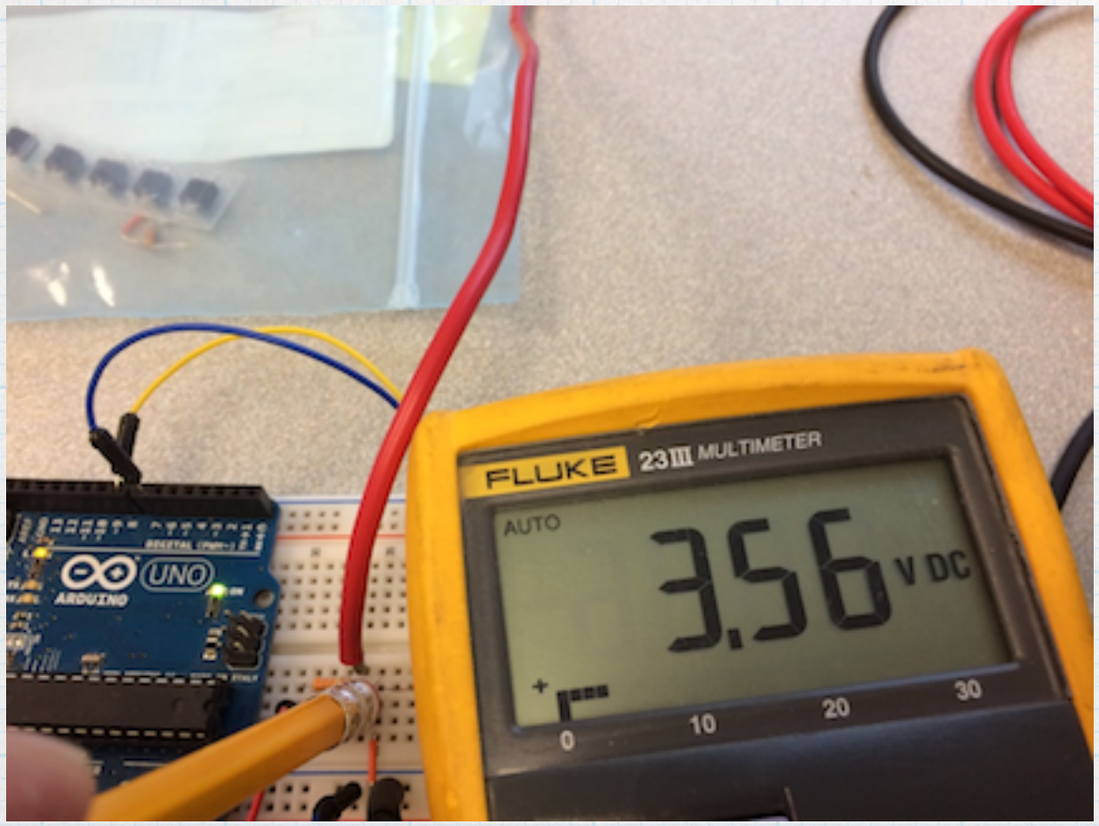
As the R_{pr} changes with variations of the incident light, v_{pr} change accordingly.

If the voltage change is big enough, a digital input on the Arduino can detect it.

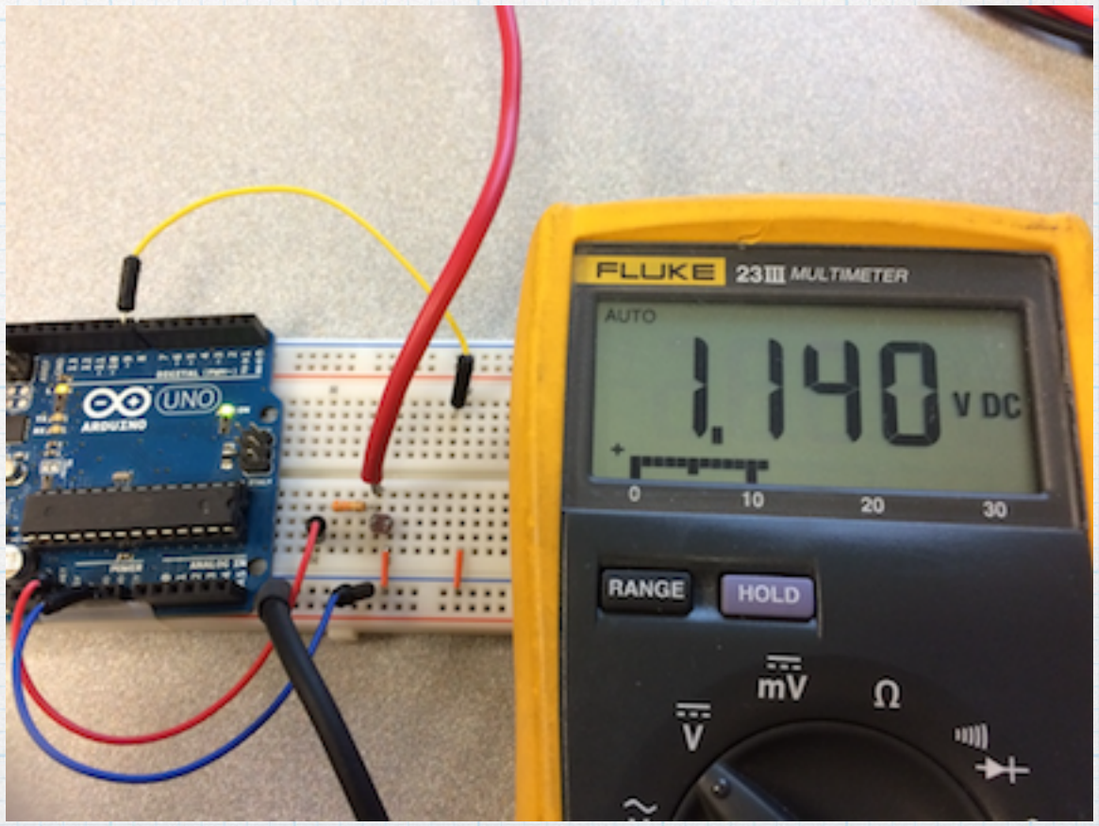


For the photoresistor used here (PDV-9002-1 from Luna optoelectronics, included in EE lab 230 kits), the dark resistance is >150 kΩ. In typical room light, it measures about 15 kΩ. Using it with a 33-kΩ fixed resistor to make a simple voltage divider with a 5-V supply, the dark and light voltages are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

dark



light

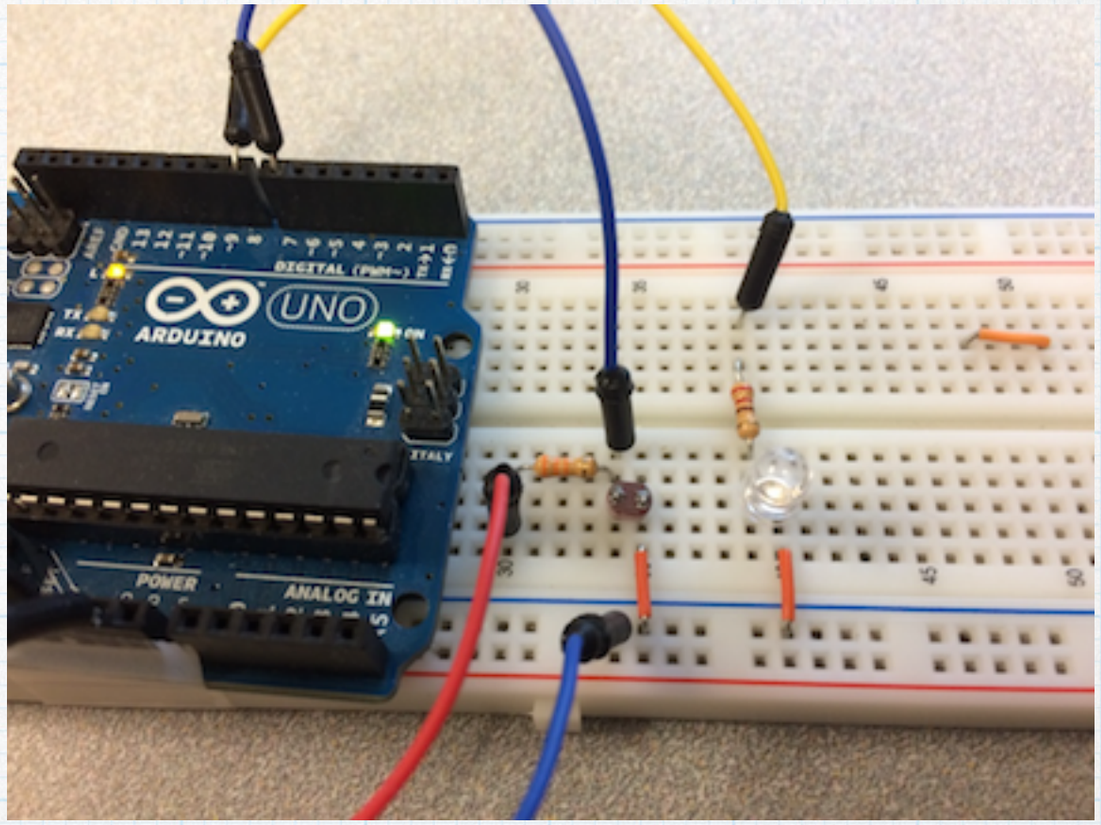


- Use the sensor input to initiate some action. When the room is dark, the sensor voltage will go high. In this case, we will simply turn on an LED.
- The photoresistor is connected to pin 7, which is configured as an input to read the voltage.
- The LED is connected to pin 8, configured as an output, with a 220-Ω limiting resistor in series.

light_sensor.ino

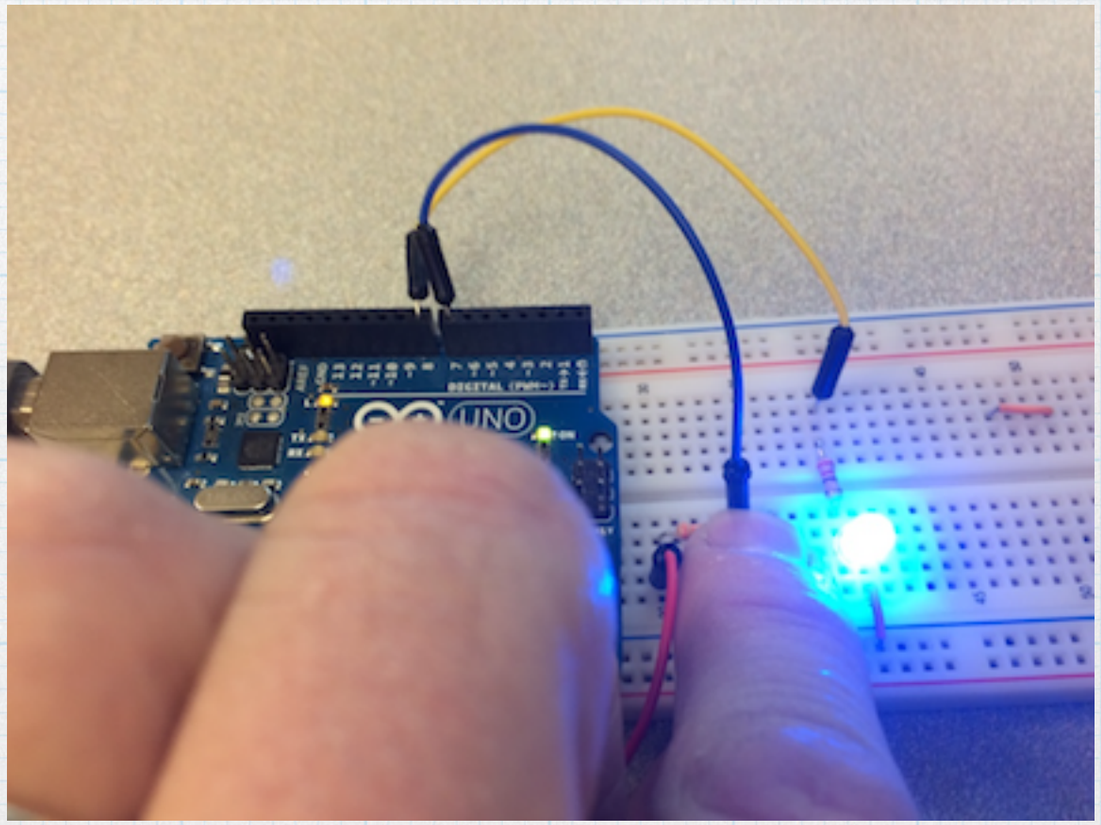
```
1   //Use a simple light sensor to control an LED
2   int sensorPin = 7;
3   int ledPin = 8;
4
5   void setup()
6   {
7     pinMode( sensorPin, INPUT );
8     pinMode( ledPin, OUTPUT );
9     digitalWrite( ledPin, LOW );
10  }
11
12  void loop()
13  {
14    pinValue = digitalRead( sensorPin );  //Read the sensor. LOW = lights on.
15
16    if( pinValue == HIGH )
17      digitalWrite( ledPin, HIGH );   //If the lights are off, turn on the LED.
18    else
19      digitalWrite( ledPin, LOW );    //Otherwise, turn it off.
20
21    delay( 100 );                   //Delay a bit between loops
22  }
```

Room light is incident on the sensor. The photoresistor voltage is “high”. The Arduino turns off the LED.



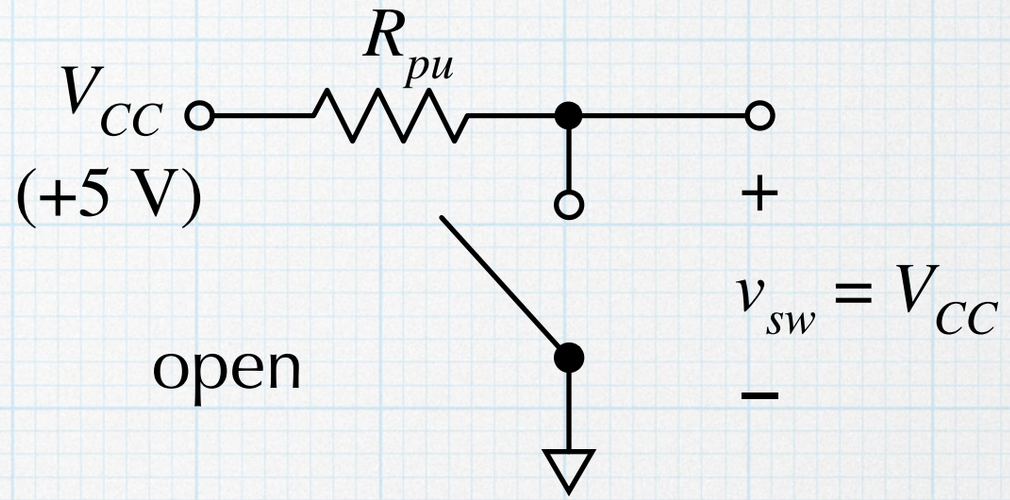
No light hitting the sensor, so the LED is turned on.

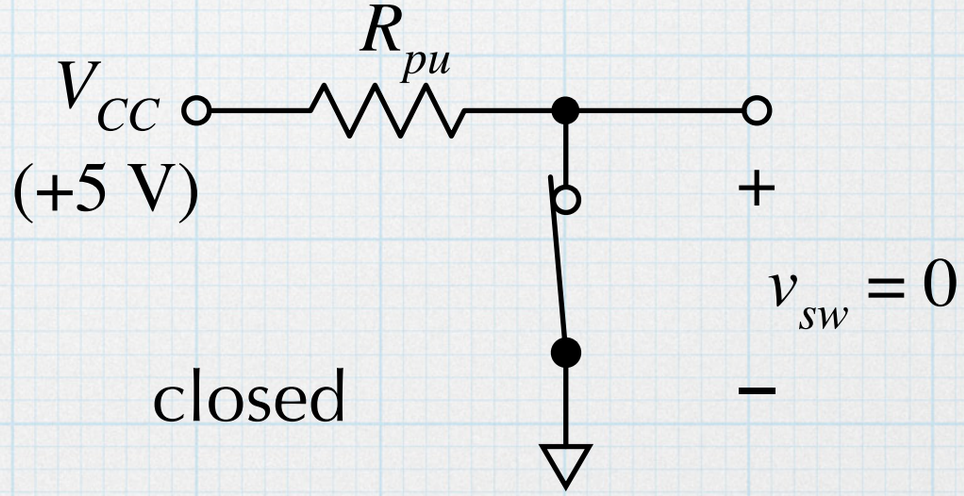
Of course, the micro-controller can initiate many different actions based on the sensor input.



Switches

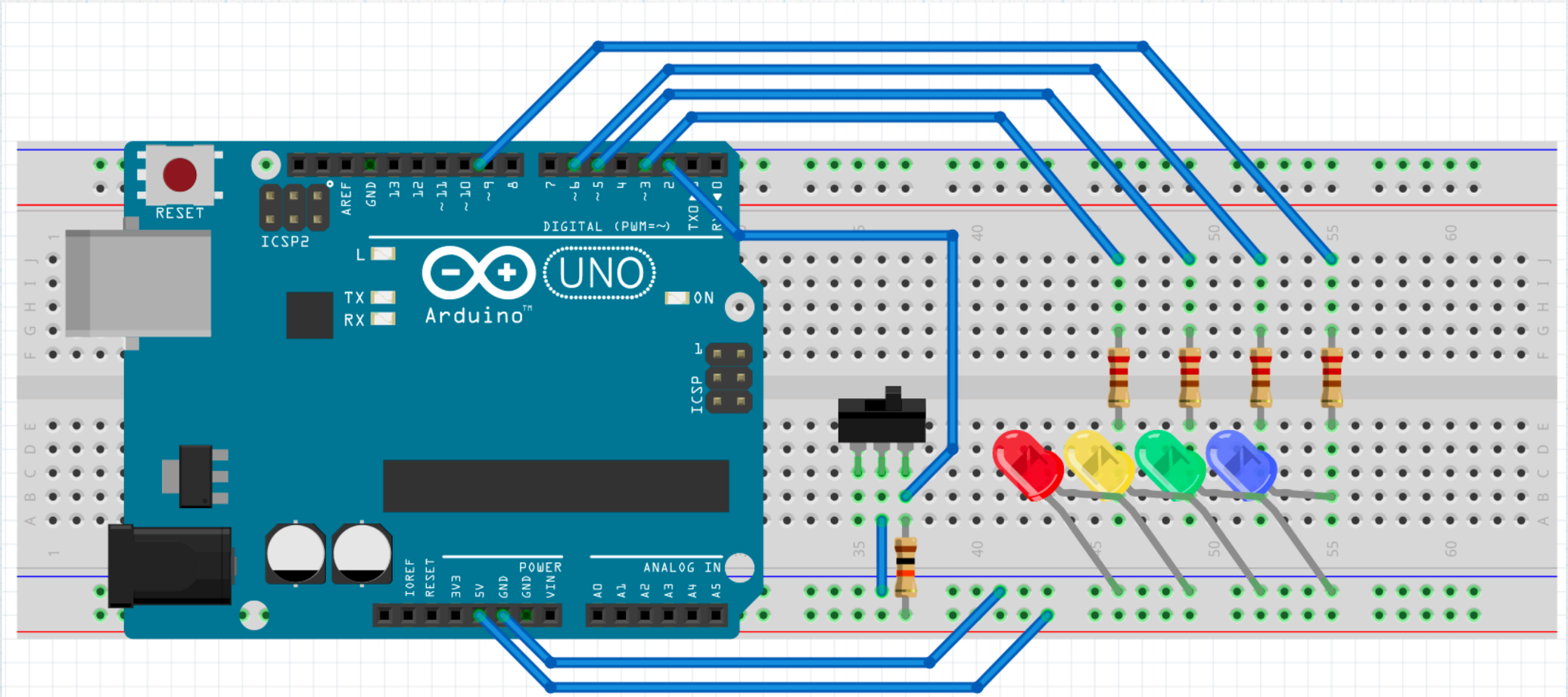
- Another common application for a digital input is read the state of mechanical switch.
- A switch can be combined with a "pull-up" resistor to create a voltage divider whose value can be varied between zero and V_S.
- The value of R_{pu} can be anything, but in order to limit the current drawn when the switch is closed, make it bigger. A value of 10 kΩ — leading to a current of 0.5 mA when the switch is closed is probably reasonable.





Example: Switch

We can use a switch to control the four LEDs from the digital output example. First use a toggle switch, which will serve to turn the blinking on or off. Use pin 2 as a digital input. When the switch is closed, the input will go low and the blinking will commence. When the is open the input is pulled high, and the blinking will stop.



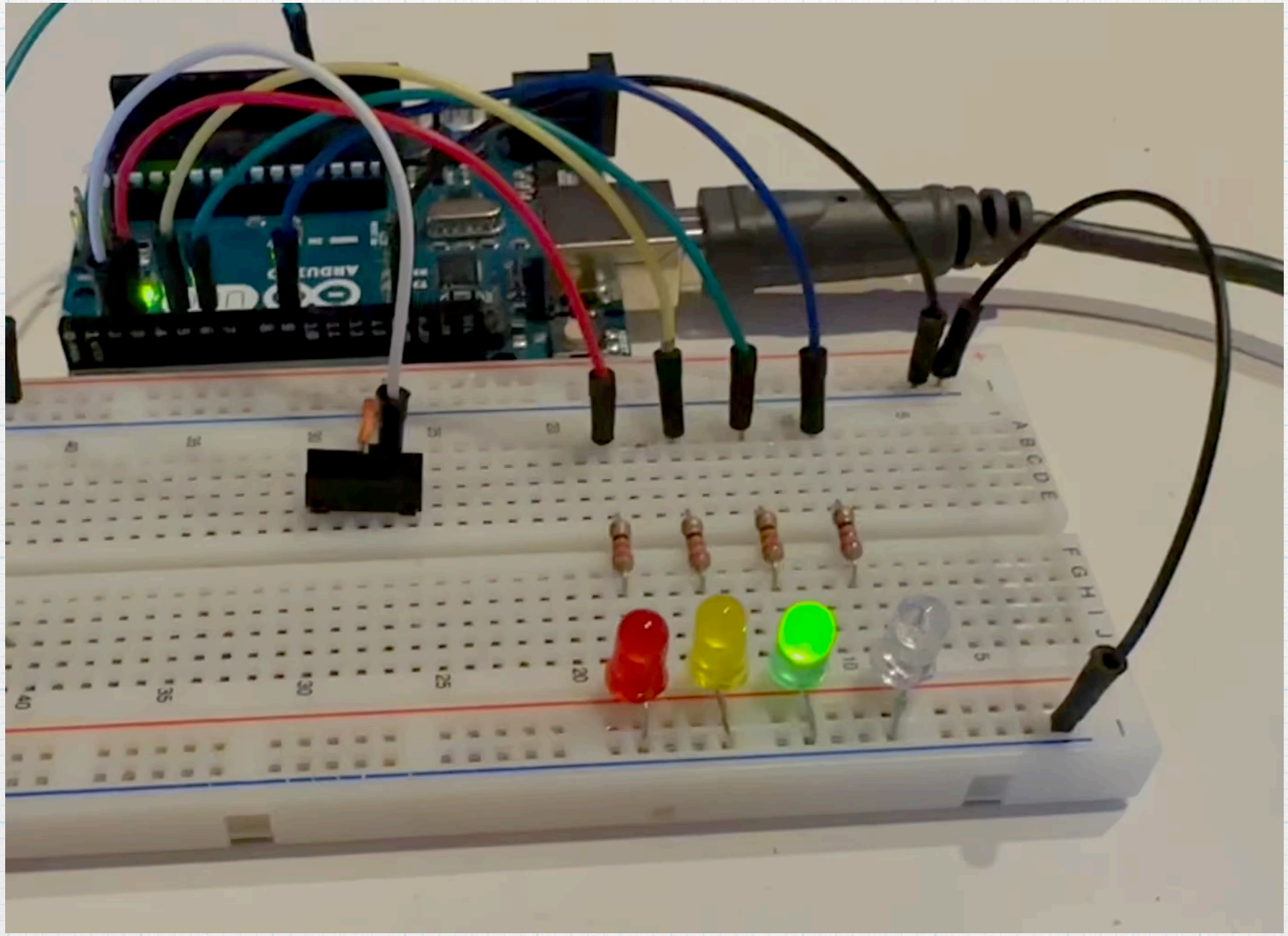
Code: 4 LEDs with switch

four_leds_switch.ino

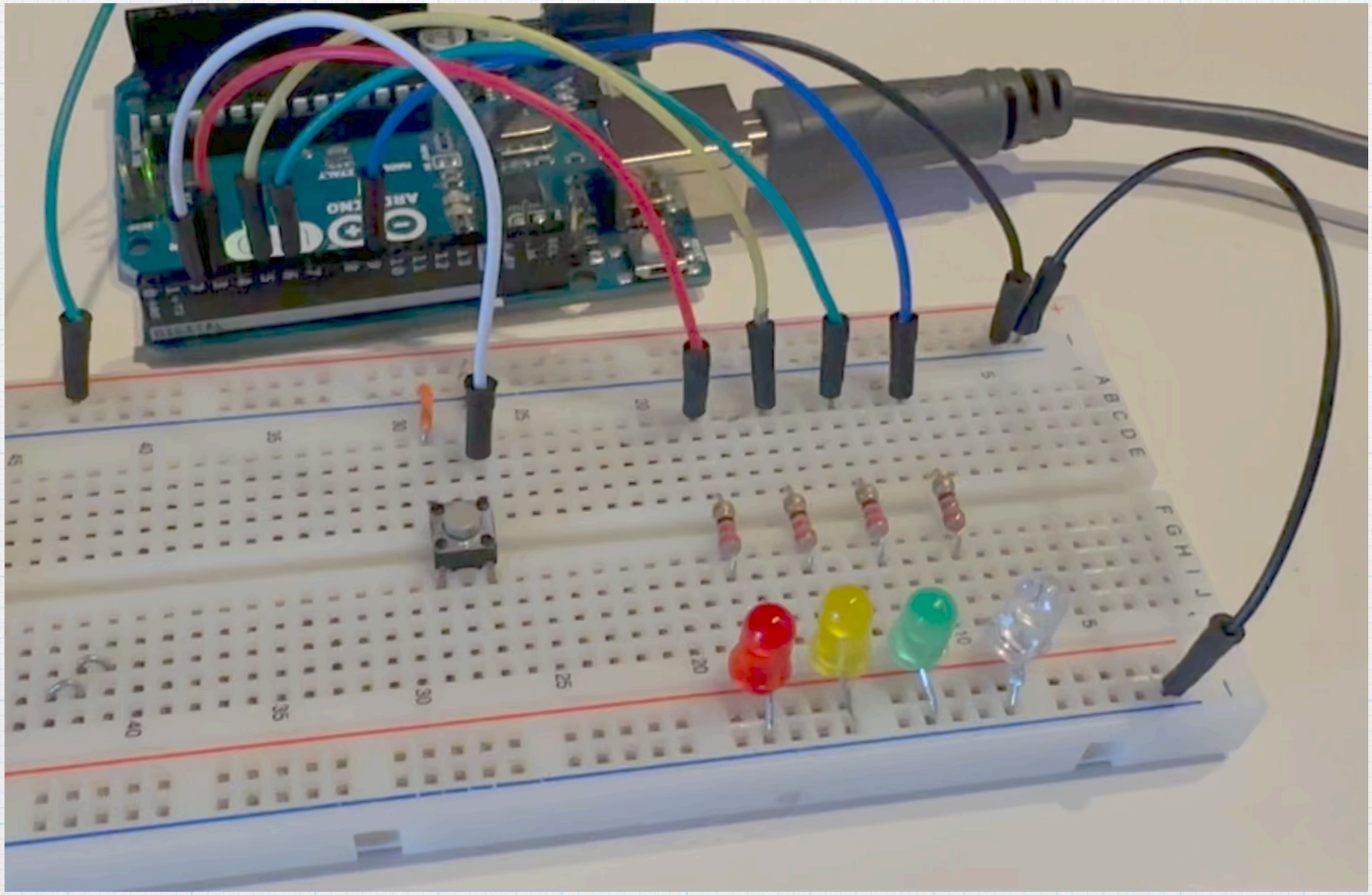
```
1   //Control the flashing LEDs with a switch
2   int i;
3   int pin[4];             //digital out pins
4   int switchPin = 2;      //input pin
5   int loopTime = 0;       //time delay between loops, in millisec
6   int onTime = 500;       //on time, in millisec
7   int offTime = 0;        //off time, in millisec
8   int switchState;
9
10  void setup()
11  {
12    pin[0] = 3;
13    pin[1] = 5;
14    pin[2] = 6;
15    pin[3] = 9;
16
17    for( i = 0; i <=3; i++ )
18      pinMode( pin[i], OUTPUT );
19
20    pinMode( switchPin, INPUT );
21  }
22
23  void loop()
24  {
25    switchState = digitalRead( switchPin );  //Check the switch.
26    if( switchState == LOW )                 //LOW = switch closed.
27    {
28      for( i = 0; i <= 3; i++ )
29      {
30        digitalWrite( pin[i], HIGH );
31        delay( onTime );
32        digitalWrite( pin[i], LOW );
33        delay( offTime );
34      }
35    }
36
37    delay( loopTime );
38  }
```

As always, code could be made more compact — use function calls directly to eliminate variables, etc.

Using a simple toggle switch. (Same switch as in the Altoids project.)

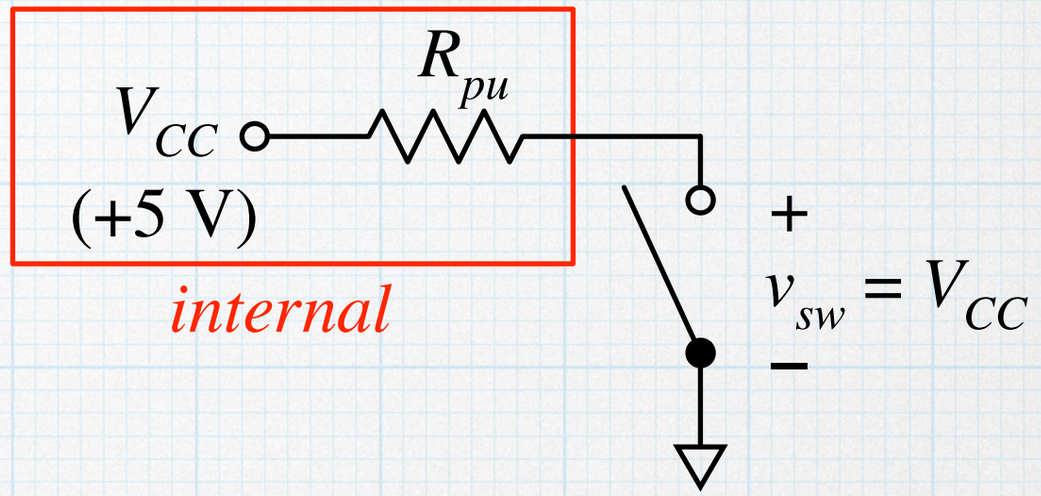


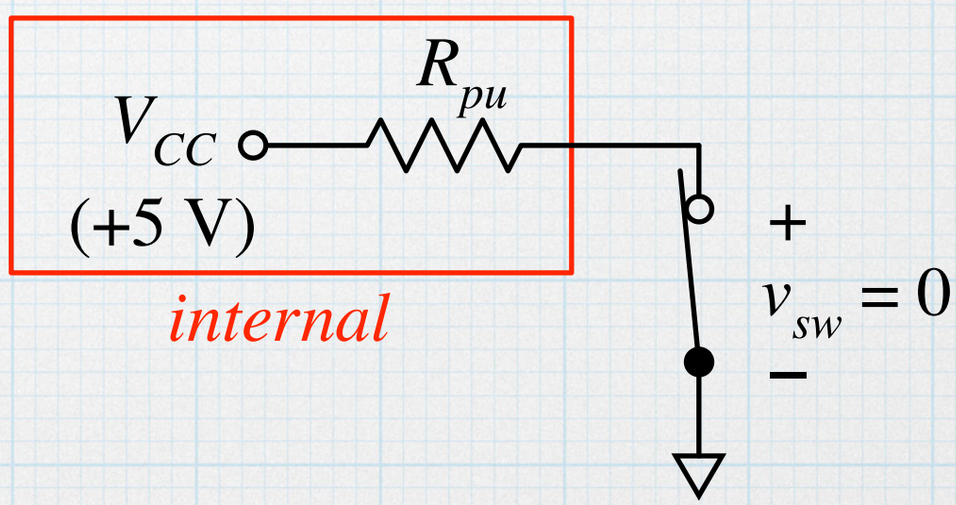
We can also us a momentary switch, which is closed only while we hold down the button. This is an “activate” switch. The code is exactly the the same.



Internal pull-up

- Reading switches is such a common activity that the Atmel chip includes internal pull-up resistors that can be used.
- This saves having to include a resistor and a connection to the power supply. (Hardware manufacturers will do anything to save a nickel on component costs.
- To use the internal pull-up, chand the pin mode command in setup: `pinmode( pin, INPUT_PULLUP )`. Then connect the switch between the pin and ground.
- The internal pull-up resistor value is around 10 kΩ. (Measure it, if you want.)





Setup function showing the use of the internal pull-up. The Fritzing diagram shows a momentary switch wired to use the internal pull-up. It couldn't be simpler.

The system functions exactly the same as before — push the button to make the lights blink.

```
10  void setup()
11  {
12    pin[0] = 3;
13    pin[1] = 5;
14    pin[2] = 6;
15    pin[3] = 9;
16
17    for( i = 0; i <=3; i++ )
18      pinMode( pin[i], OUTPUT );
19
20    pinMode( switchPin, INPUT_PULLUP );
21  }
```