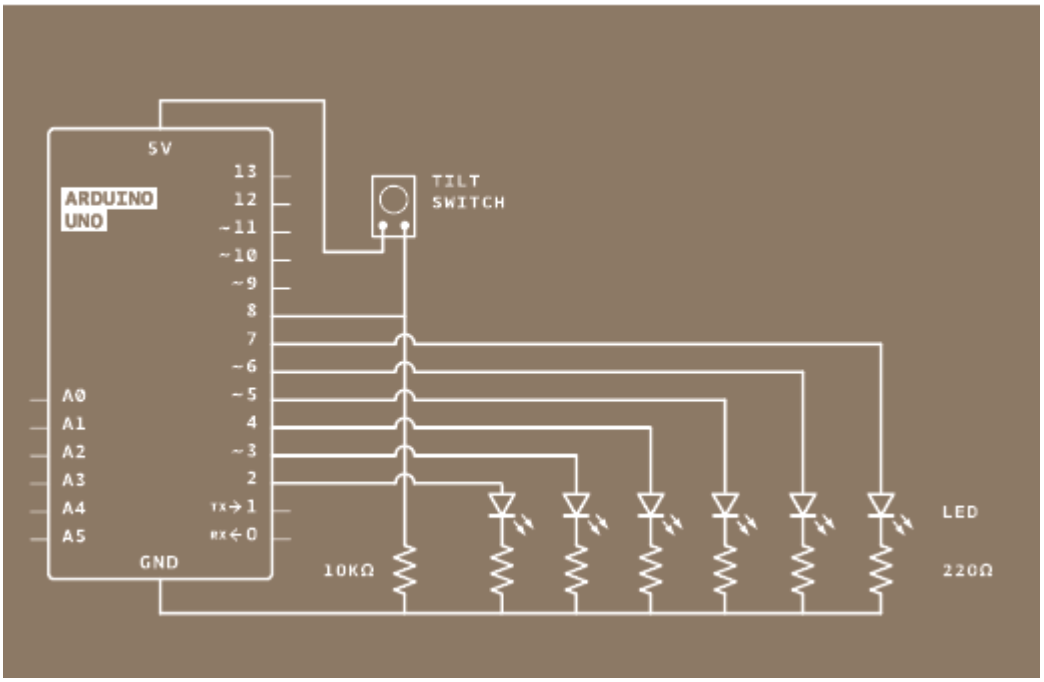
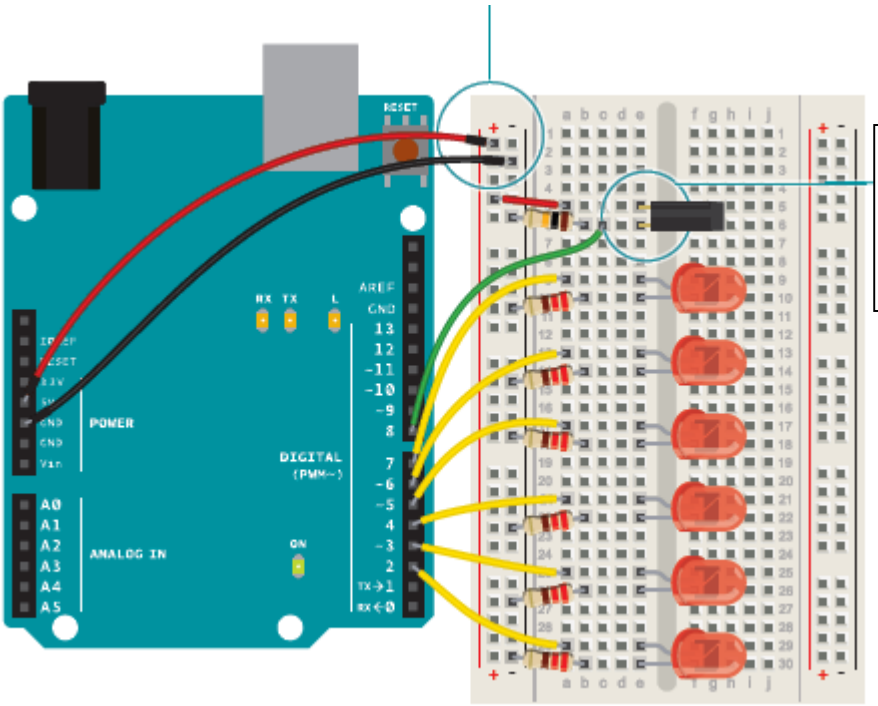
8.digital 6 min hourglass

The **millis()** function keeps track of the of the time your Arduino has been running in milliseconds. The variable is unsigned long, so it can keep memory of about 50 days.
In this project you will build a 6 min digital hourglass.
The tilt switch works just like a regular switch in that it is an on/off sensor. You'll use it here as a digital input. What makes tilt switches unique is that they detect orientation. Typically they have a small cavity inside the housing that has a metal ball. When tilted in the proper way, the ball rolls to one side of the cavity and connects the two leads that are in your breadboard, closing the switch.
With six LEDs, your hourglass will run for 6 min, just as its name implies.





Connect one lead of the tilt switch to 5V. Connect the other to a 10-kilohm resistor to ground. Connect the junction where they meet to digital pin 8.

```
// named constant for the switch pin
const int switchPin = 8;

unsigned long previousTime = 0; // store the last time an LED was updated
int switchState = 0; // the current switch state
int prevSwitchState = 0; // the previous switch state
int led = 2; // a variable to refer to the LEDs

// 60000 = 1 minutes in milliseconds
long interval = 60000; // interval at which to light the next LED

void setup() {
  // set the LED pins as outputs
  for (int x = 2; x < 8; x++) {
    pinMode(x, OUTPUT);
  }
  // set the tilt switch pin as input
  pinMode(switchPin, INPUT);
}

void loop() {
  // store the time since the Arduino started running in a variable
  unsigned long currentTime = millis();

  // compare the current time to the previous time an LED turned on
  // if it is greater than your interval, run the if statement
  if (currentTime - previousTime > interval) {
    // save the current time as the last time you changed an LED
    previousTime = currentTime;
    // Turn the LED on
    digitalWrite(led, HIGH);
    // increment the led variable
    // in 10 minutes the next LED will light up
    led++;

    if (led == 7) {
      // the hour is up
    }
  }

  // read the switch value
  switchState = digitalRead(switchPin);

  // if the switch has changed
  if (switchState != prevSwitchState) {
    // turn all the LEDs low
    for (int x = 2; x < 8; x++) {
      digitalWrite(x, LOW);
    }

    // reset the LED variable to the first one
    led = 2;

    //reset the timer
    previousTime = currentTime;
  }
  // set the previous switch state to the current state
  prevSwitchState = switchState;
}
```

You're going to need a number of global variables in your program to get this all working. To start, create a constant named switchPin. This will be the name of the pin your tilt switch is on.
Create a variable of type unsigned long. This will hold the time an LED was last changed.
Create a variable for the switch state, and another to hold the previous switch state. You'll use these two to compare the switch's position from one loop to the next.
Create a variable named led. This will be used to count which LED is the next one to be turned on. Start out with pin 2. The last variable you're creating is going to be the interval between each LED turning on. This will be be a long datatype.
In 1 minute (the time between each LED turning on) 60,000 milliseconds pass. If you want the delay between lights to be longer or shorter, this is the number you change.
In your setup(), you need to declare the LED pins 2-7 as outputs. A for() loop declares all six as OUTPUT with just 3 lines of code. You also need to declare switchPin as an INPUT.

When the loop() starts, you're going to get the amount of time the Arduino has been running with millis() and store it in a local variable named currentTime.
Using an if() statement, you'll check to see if enough time has passed to turn on an LED. Subtract the currentTime from the previousTime and check to see if it is greater than the interval variable. If 60,000 milliseconds have passed, you'll set the variable previousTime to the value of currentTime

previousTime indicates the last time an LED was turned on. Once you've set previousTime, turn on the LED, and increment the led variable. The next time you pass the time interval, the next LED will light up. Add one more if statement in the program to check if the LED on pin 7 is turned on. Don't do anything with this yet. You'll decide what happens at the end of the hour later. Now that you've checked the time, you'll want to see if the switch has changed its state. Read the switch value into the switchState variable.
With an if() statement, check to see if the switch is in a different position than it was previously. The **!=** evaluation checks to see if switchState does not equal prevSwitchState. If they are different, turn the LEDs off, return the led variable to the first pin, and reset the timer for the LEDs by setting previousTime to currentTime.
At the end of the loop(), save the switch state in prevSwitchState , so you can compare it to the value you get for switchState in the next loop().