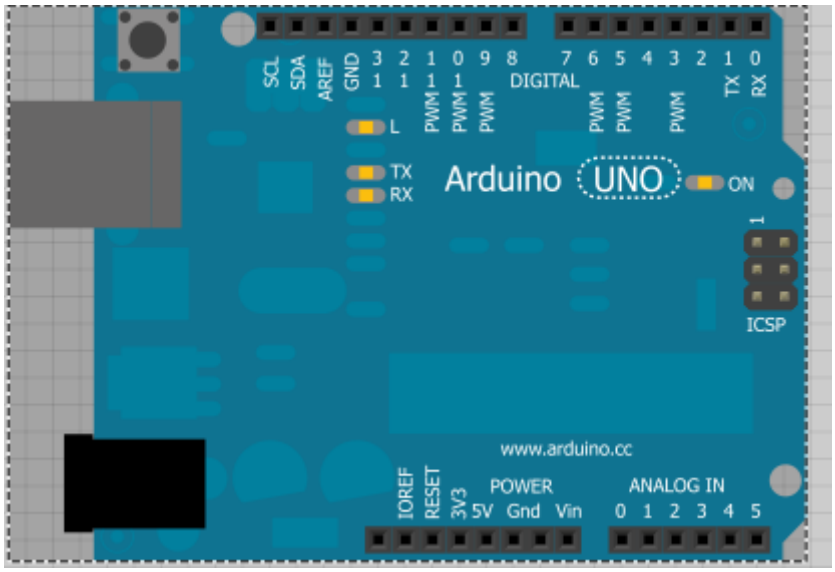
Programming output with the Arduino

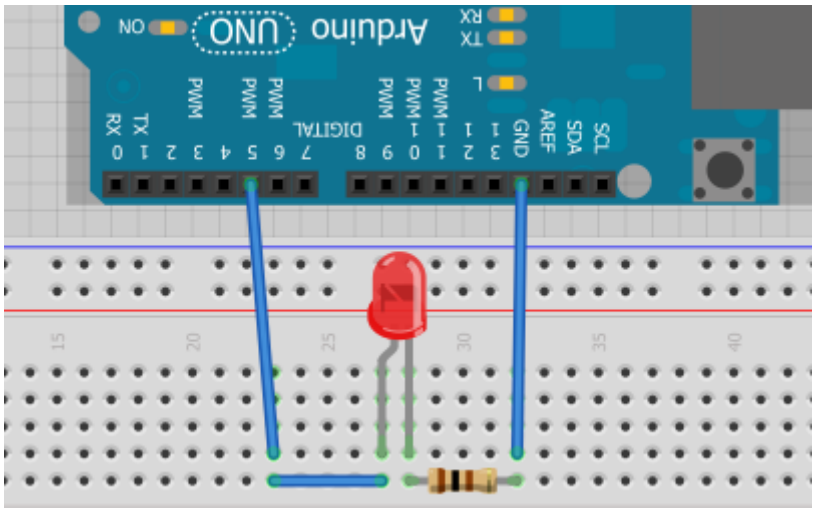
Along the top of the Arduino Uno are ports numbered 0 to 13. These can output a small voltage or sense the input of a voltage. Pin 13 is also permanently connected to a LED on the Arduino board that will turn on if that pin is set to output. These pins can be set to output or input but not both at the same time. Pins 3, 5, 6, 9, 10 and 11 are marked with a ~ and can be used for pulse width modulation



The lower row of pins marked 0 to 5 are used for analogue input but could be adapted as additional digital input connections.

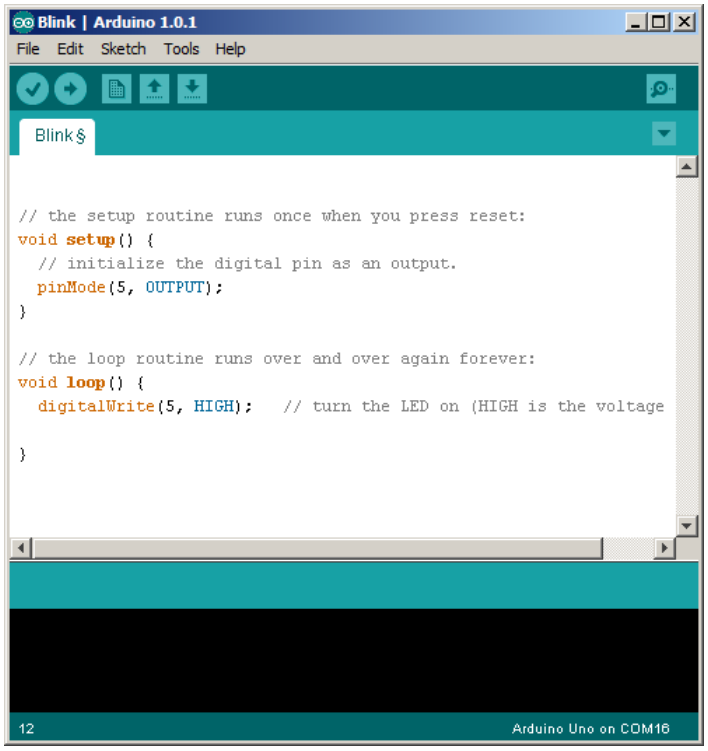
The Arduino is programmed through a variant of the Processing language. The code is similar to C.

Here pin 5 provides a current to the LED that returns through the GND pin.



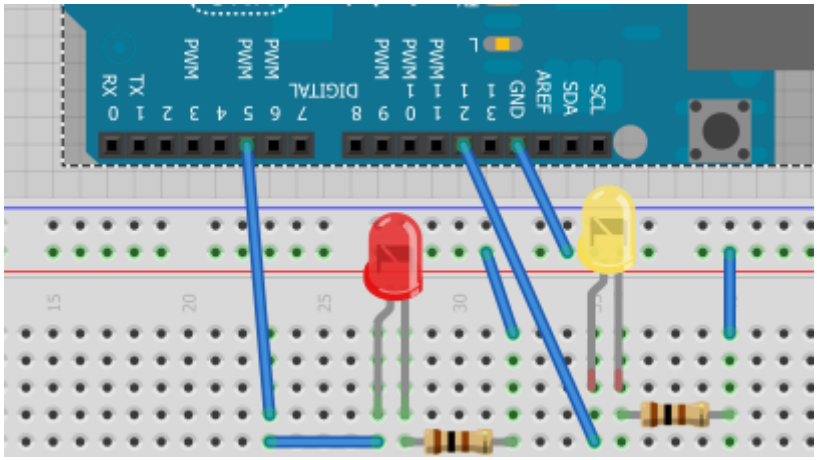
If this circuit is constructed and connected nothing will happen because no voltage flows from pin 5.

The Processing program needs to tell pin 5 to be an output pin and also to send current to it.



Click the arrow to run the program. A common problem is that the Arduino is not detected. This is often solved by selecting another port from the Tools menu.

2 LEDs are connected in this diagram. One is powered from pin 5, the other from pin 12. Note that the ground connection attaches to a single row of pin holes. The LEDs are sharing a common ground (cathode) but each needs a dedicated voltage in.



Here is some basic blinking code for a single LED. This can be modified to run waves of lights up and down LEDs or to create traffic light simulations.

```
void setup() {
  // initialize the digital pin as an output.
  pinMode(5, OUTPUT);
}
void loop() {
  digitalWrite(5, HIGH);   // turn the LED on
  delay(1000);               // wait for a second
  digitalWrite(5, LOW);    // turn the LED off
  delay(1000);               // wait for a second
}
```

Pulse Width Modulation

The light can only be turned on or off with this code. There is no partly on or almost off. If the delay is set to a very short time the LED turns on and off very quickly and appears to be constantly on but not quite as bright as it was before.

```
void setup() {
  // initialize the digital pin as an output.
  pinMode(5, OUTPUT);
}
void loop() {
  digitalWrite(5, HIGH);   // turn the LED on
  delay(10);               // wait for a second
  digitalWrite(5, LOW);    // turn the LED off
  delay(10);               // wait for a second
```

True Pulse Width Modulation programming only works on the pins marked with a ~ (3, 5. 6, 9, 10, 11). The fading code demonstrates this. There are 2 loops here that follow after each other. In the 1st loop the ledPin (pin 5) is instructed to analogWrite values from 0 to 255 in stages of 5. There are 51 steps each with a delay of 30 milliseconds. After 30 times 51 or about 1 ½ seconds the next loop begins which starts at 255 and counts down to 0. Then the code returns to the 1st loop.

Note that although the code uses the term analogWrite the output is purely digital. The controller is faking different light levels by turning the light on and off very quickly.

```
int ledPin = 5;    // LED connected to digital pin 5
void setup()  {
  // nothing happens in setup
}
void loop()  {
  // fade in from min to max in increments of 5 points:
  for(int fadeValue = 0 ; fadeValue <= 255; fadeValue +=5) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }

  // fade out from max to min in increments of 5 points:
  for(int fadeValue = 255 ; fadeValue >= 0; fadeValue -=5) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }
}
```