+

Arduino → INPUTS

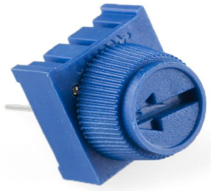
CCLab 2016 _ Week 5

What is an INPUT?

- An input device is a peripheral, component, or hardware equipment used to provide data and control signals to an information processing system such as a computer or information appliance.

+ Examples

Button



Potentiometer

Knob

Big Button

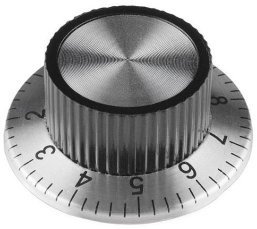
Keypad

Switch

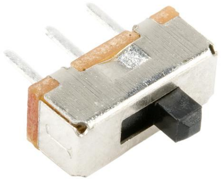
Switch

Arcade Button

5 way switch



Knob



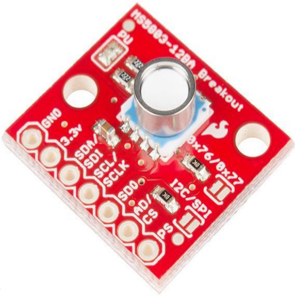
Switch

Potentiometer

Pulse

Gas

Motion

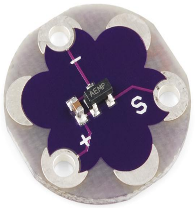


Pressure

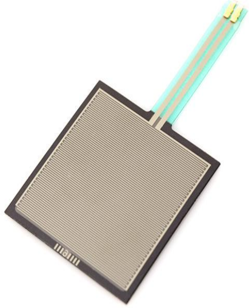
pH

Photocell

Range Finder



Temperature



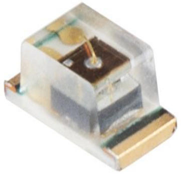
Force

Flex

Barometric
Pressure

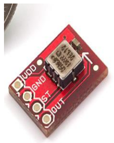
Color

Humidity



Light

Force



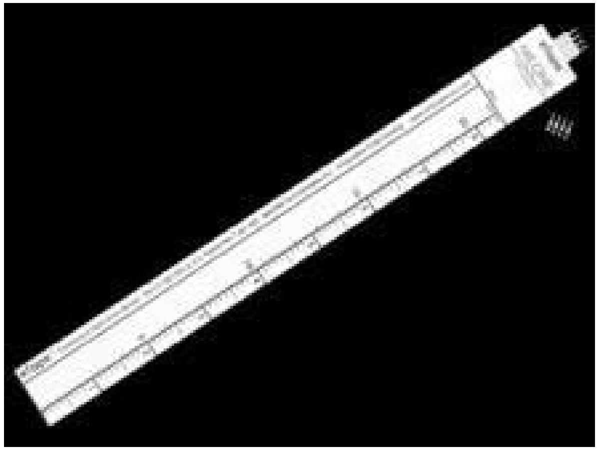
Single Axis
Accelerometer

INPUTS

GAME CONTROLLERS

SOIL TEMP / MOISTURE

TOUCHSCREEN



LIQUID LEVELS

LIQUID FLOW METERS

FINGERPRINT

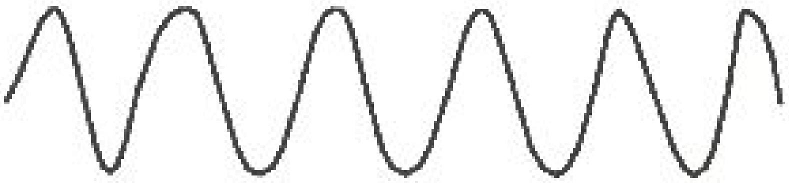
+

Digital _ Switch

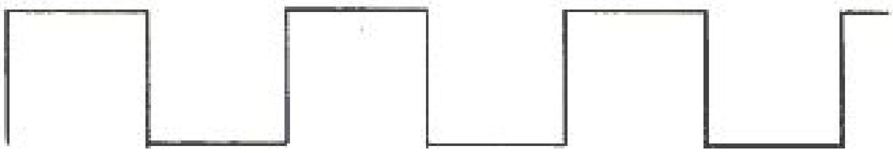
Analog-LDR

+

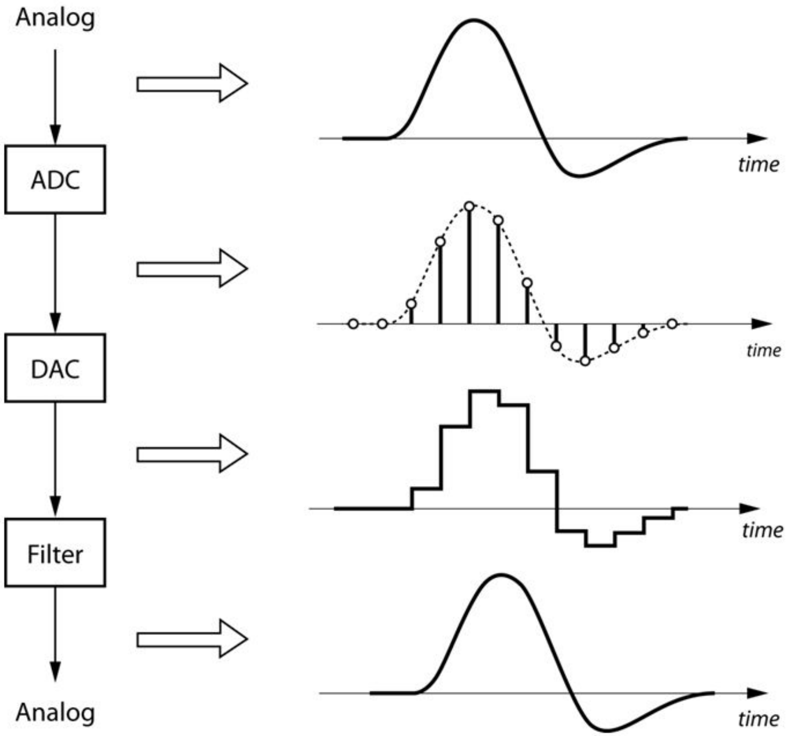
Analog vs. Digital Input



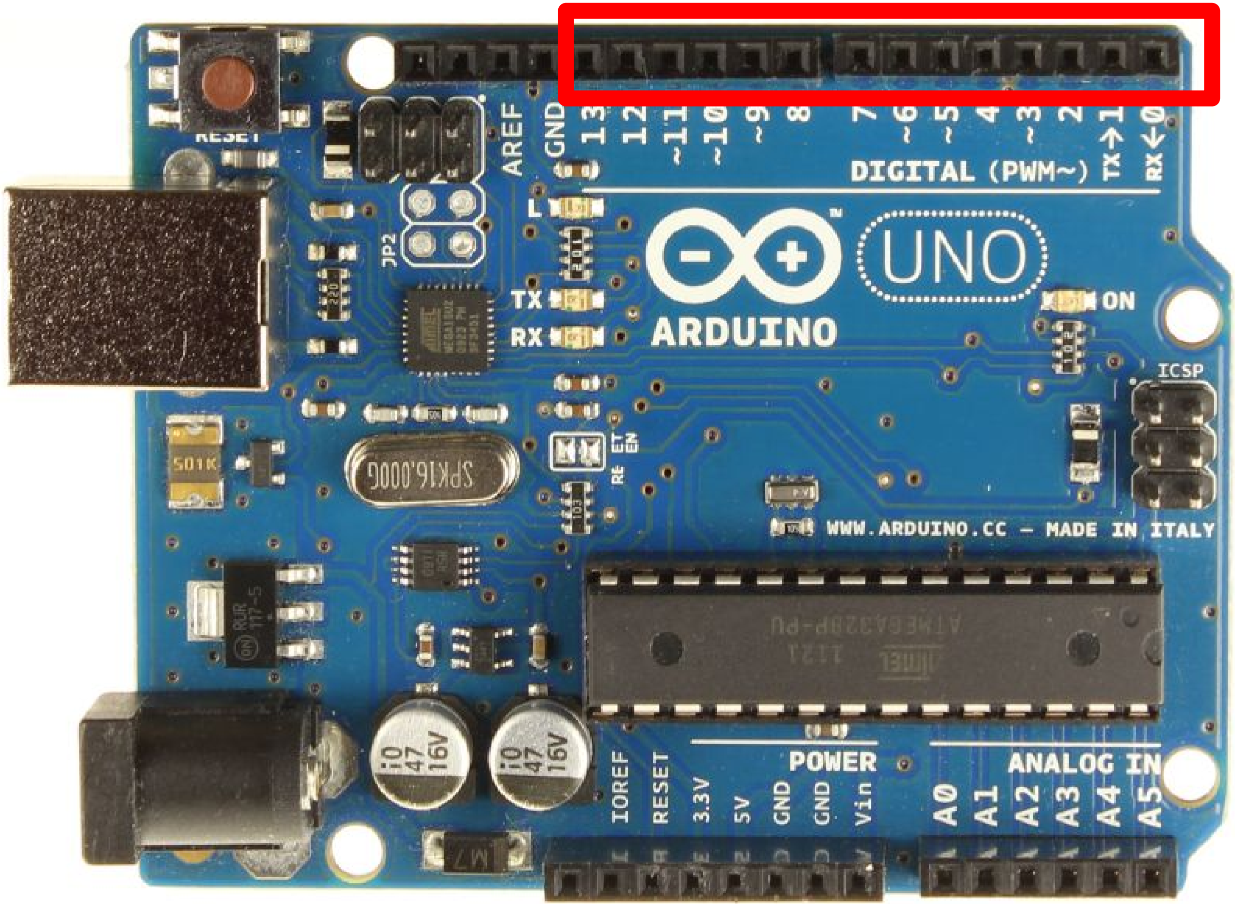
Analog Signal



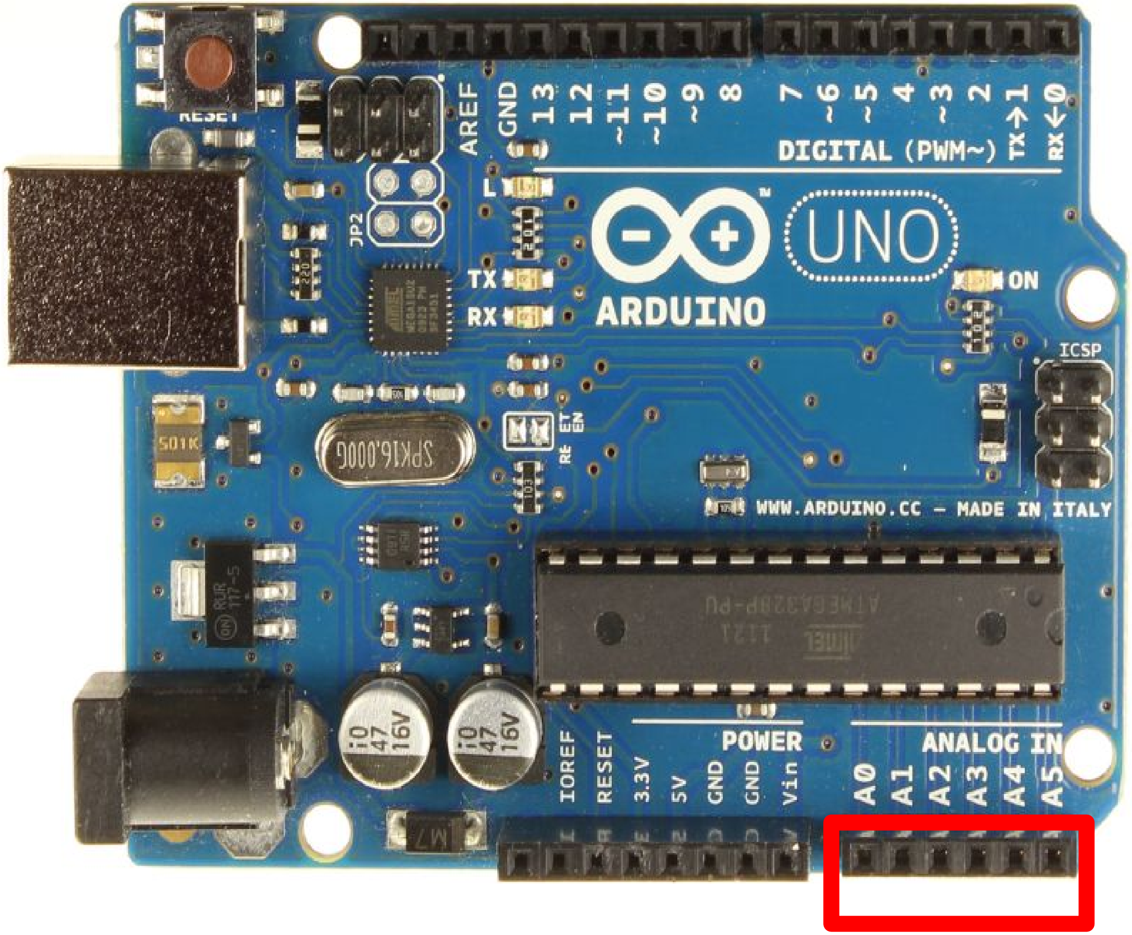
Digital Signal



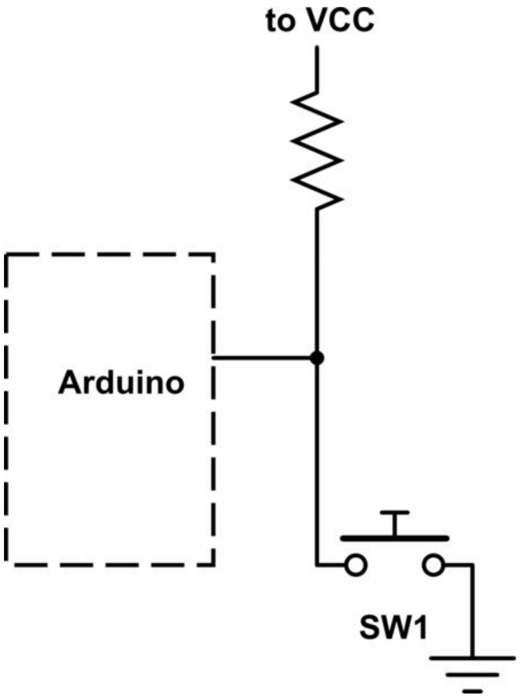
Digital Pins

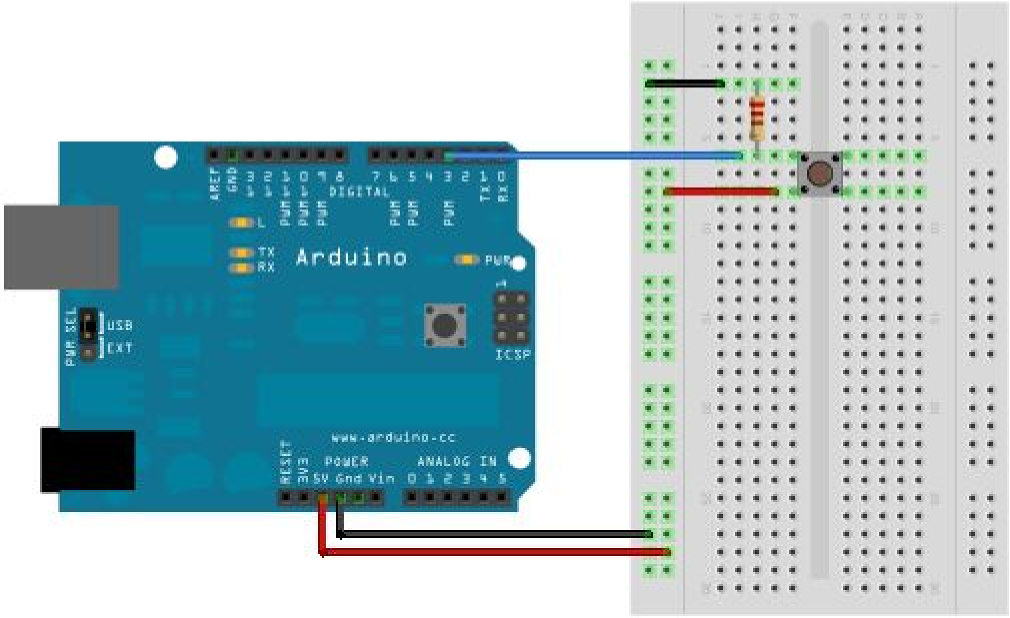


Analog Pins


RESET
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
JP2
L
TX
RX
ARDUINO
UNO
ON
ICSP
SPK16.000G
WWW.ARDUINO.CC – MADE IN ITALY
ATMEGA328P-PU
POWER
IOREF
RESET
3.3V
5V
GND
GND
Vin
ANALOG IN
A0
A1
A2
A3
A4
A5

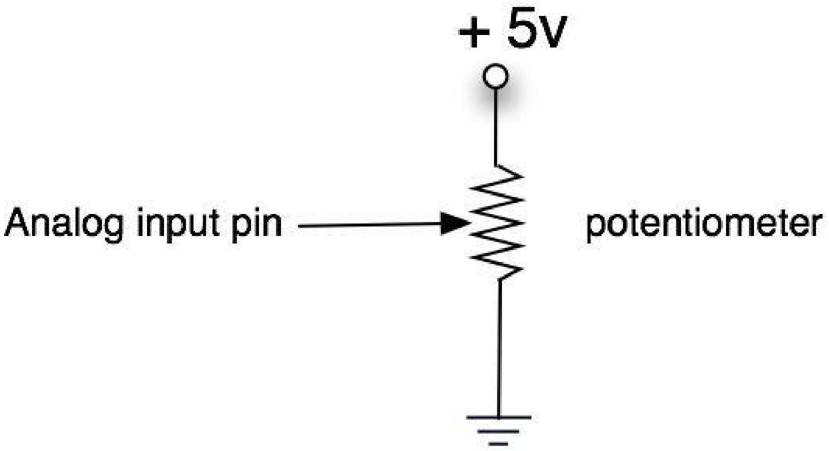
Digital Input - Hookup





Analog Input - Hookup

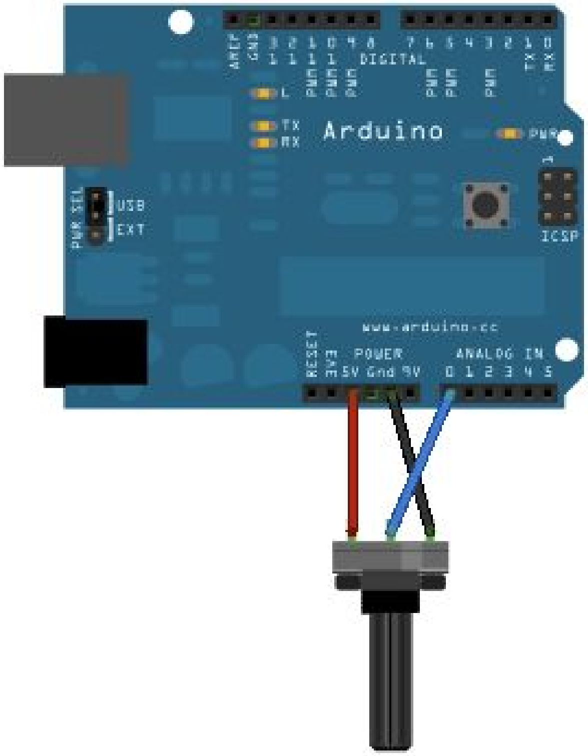
Potentiometer connected to the analog input of the arduino



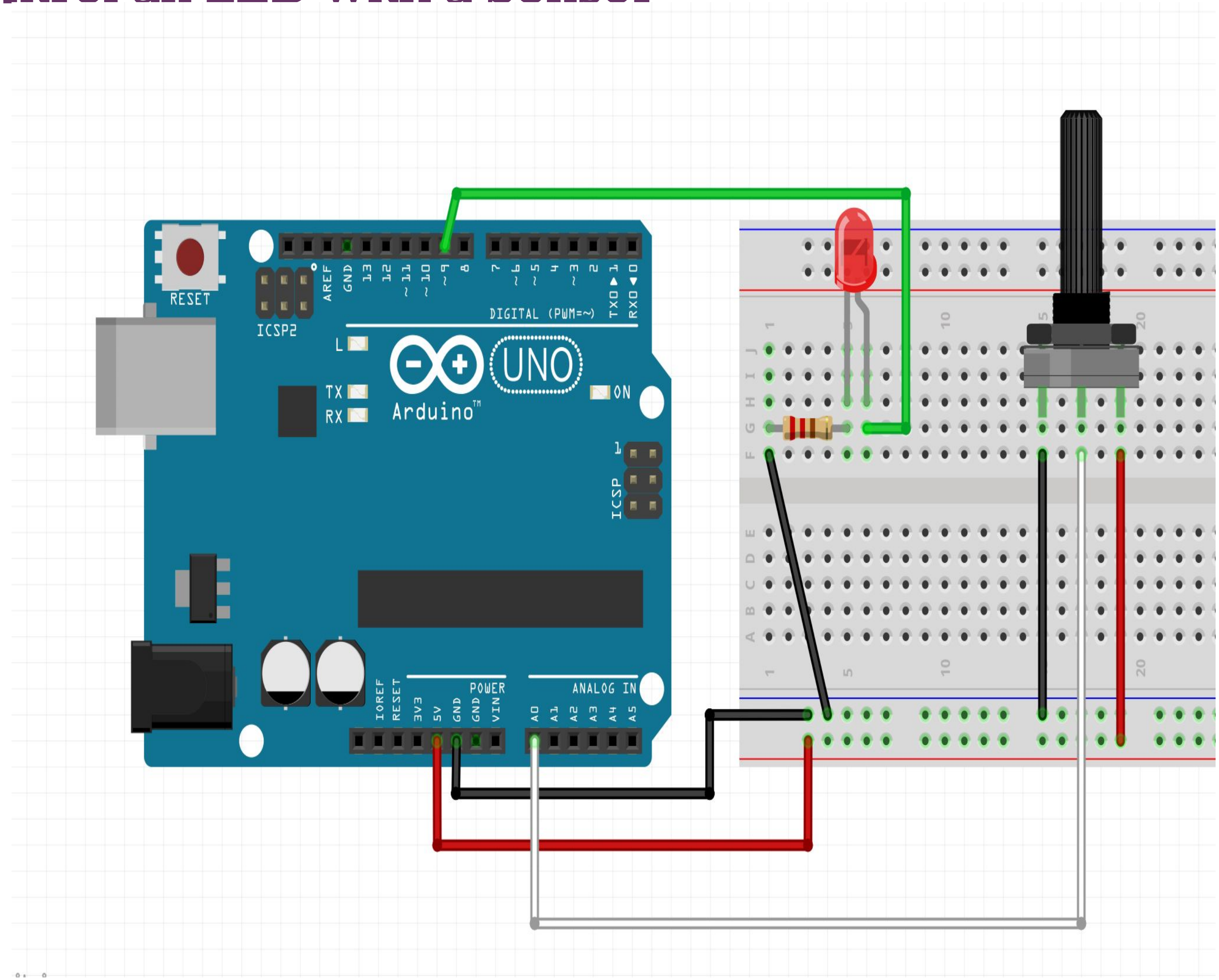
+ 5v

Analog input pin

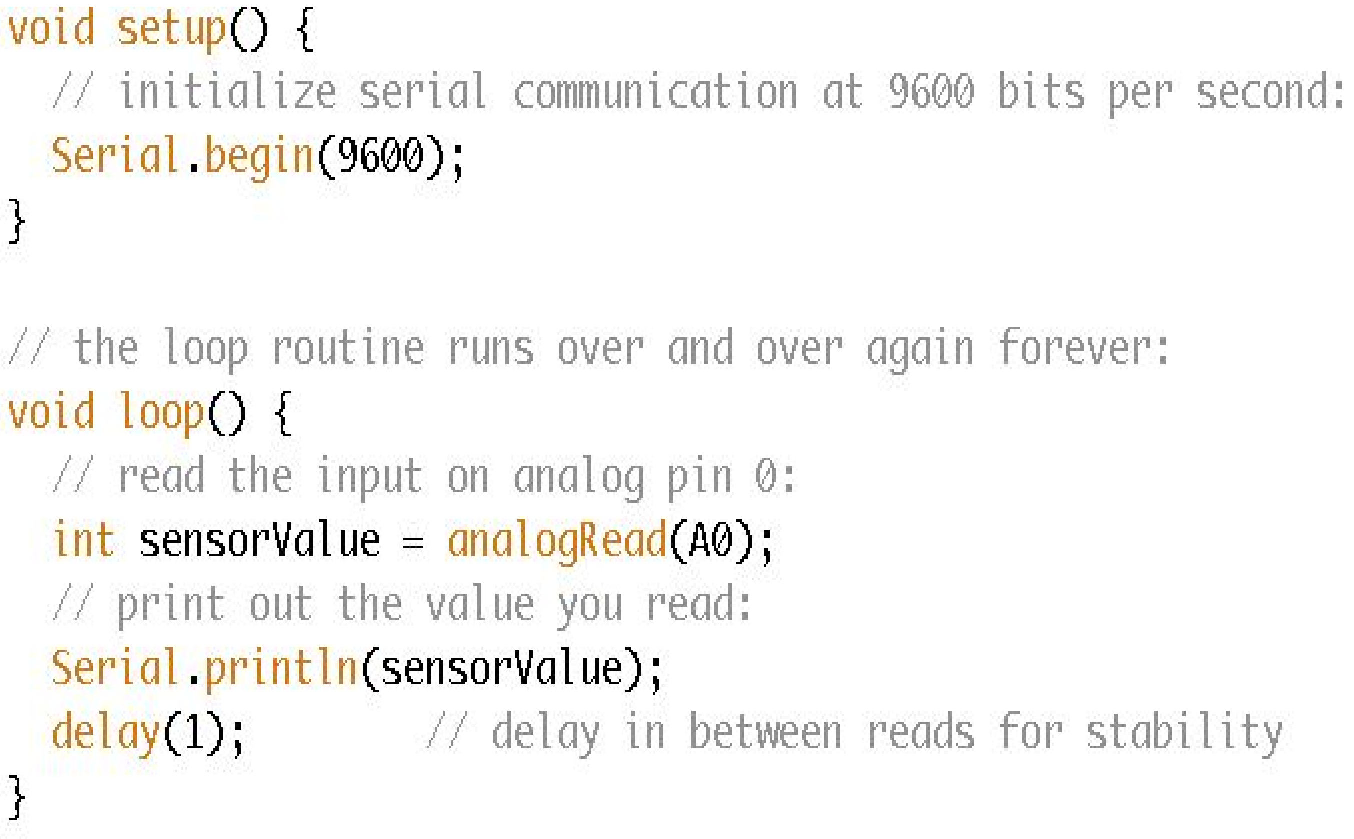
potentiometer



Control an LED with a Sensor

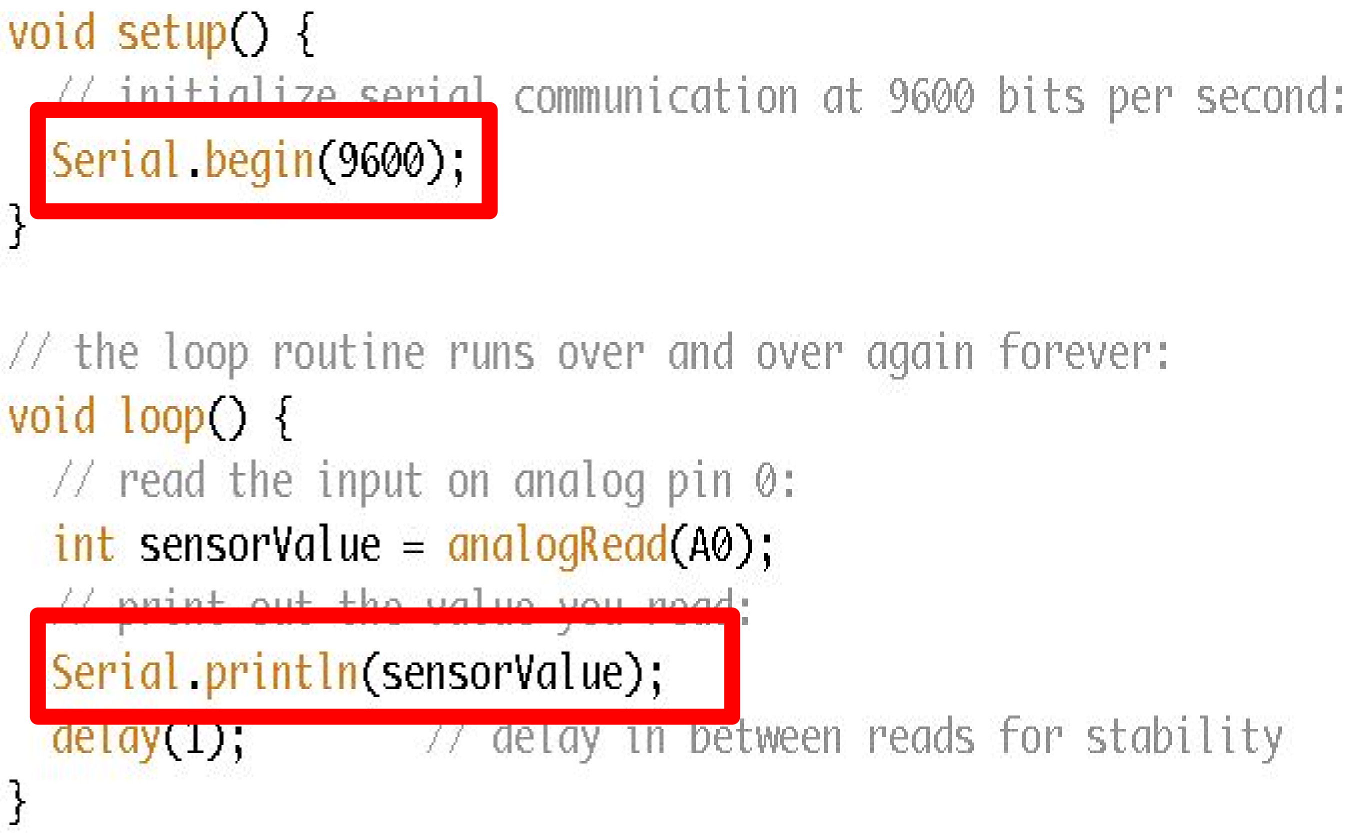


File > Examples > Basic > AnalogReadSerial



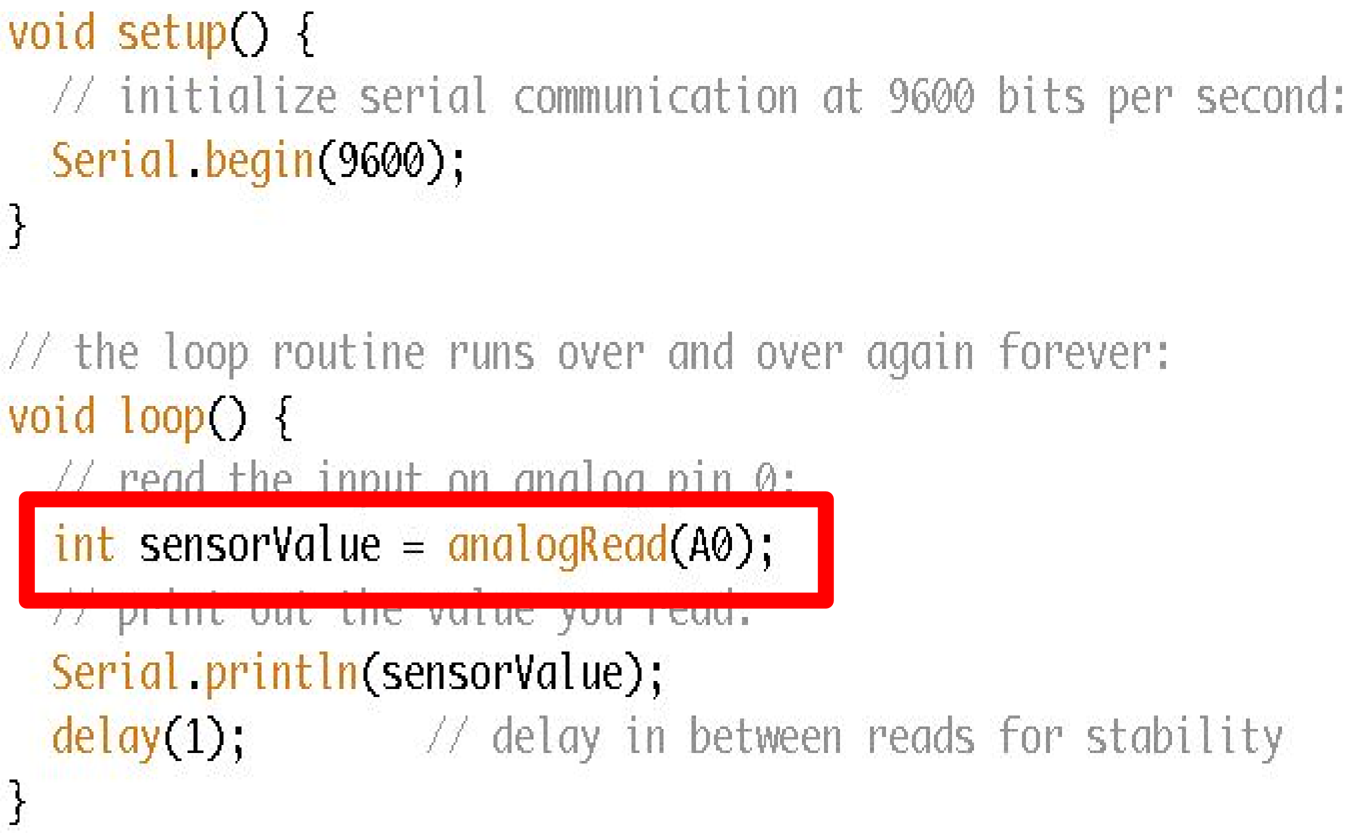
```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

File > Examples > Basic > AnalogReadSerial



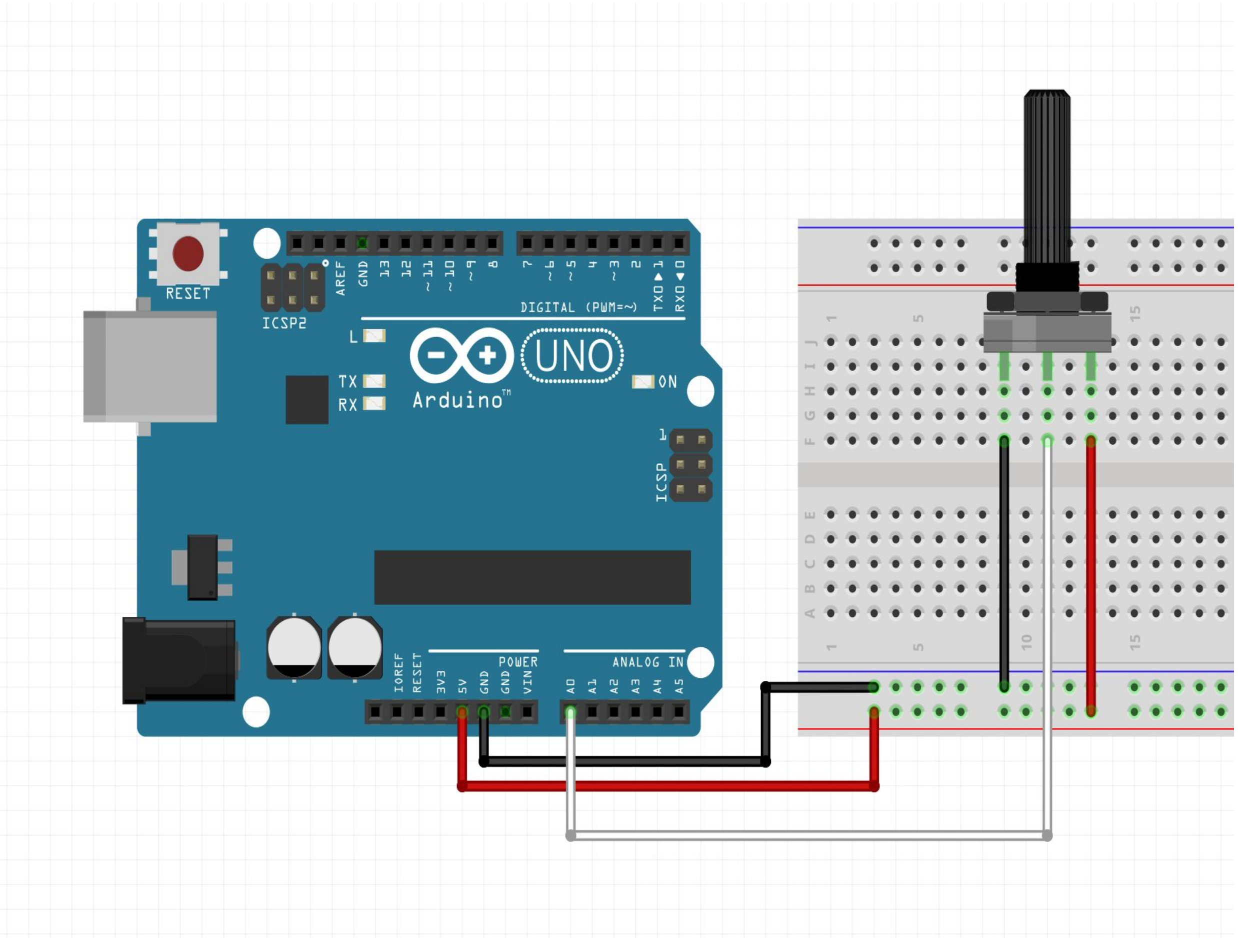
```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

File > Examples > Basic > AnalogReadSerial

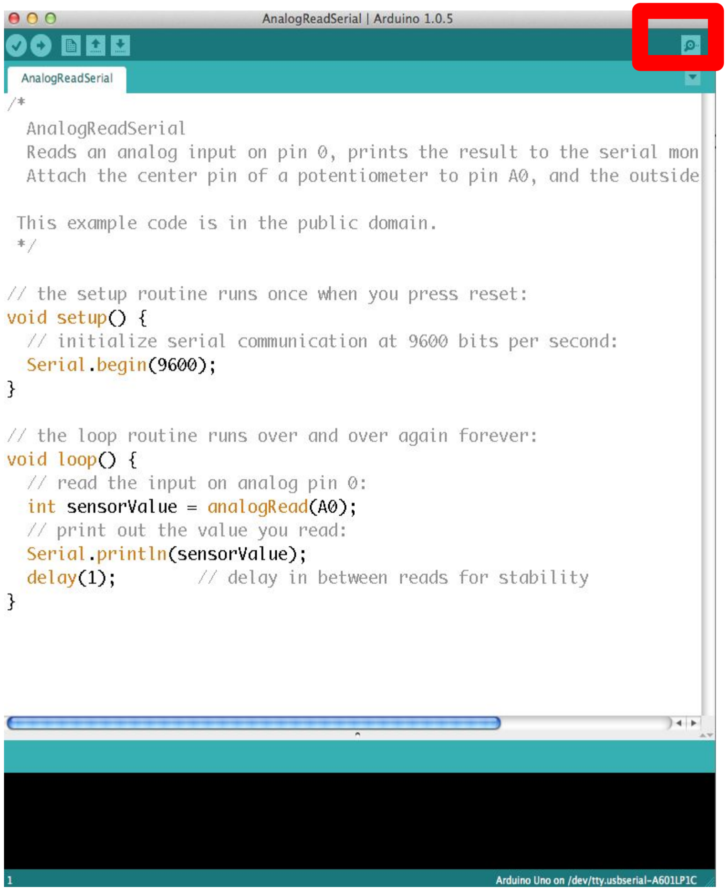


```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

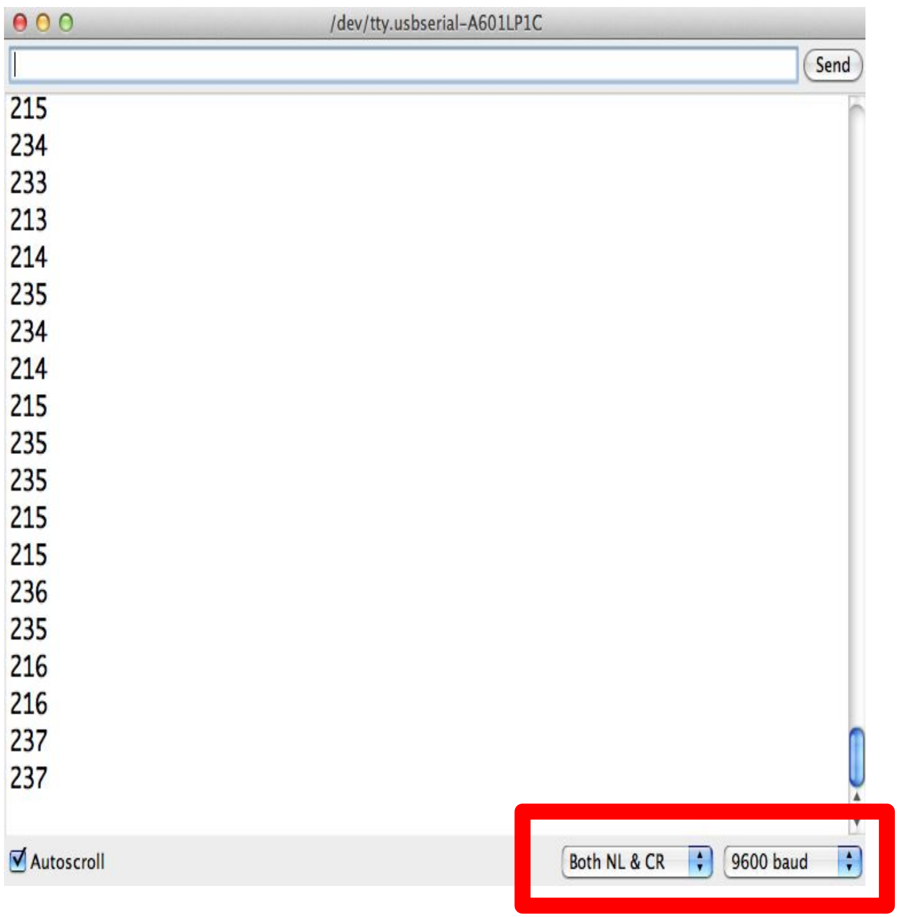
Connecting the Input



Serial Monitor


AnalogReadSerial | Arduino 1.0.5
AnalogReadSerial
1
Arduino Uno on /dev/tty.usbserial-A601LP1C

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial mon
  Attach the center pin of a potentiometer to pin A0, and the outside

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```



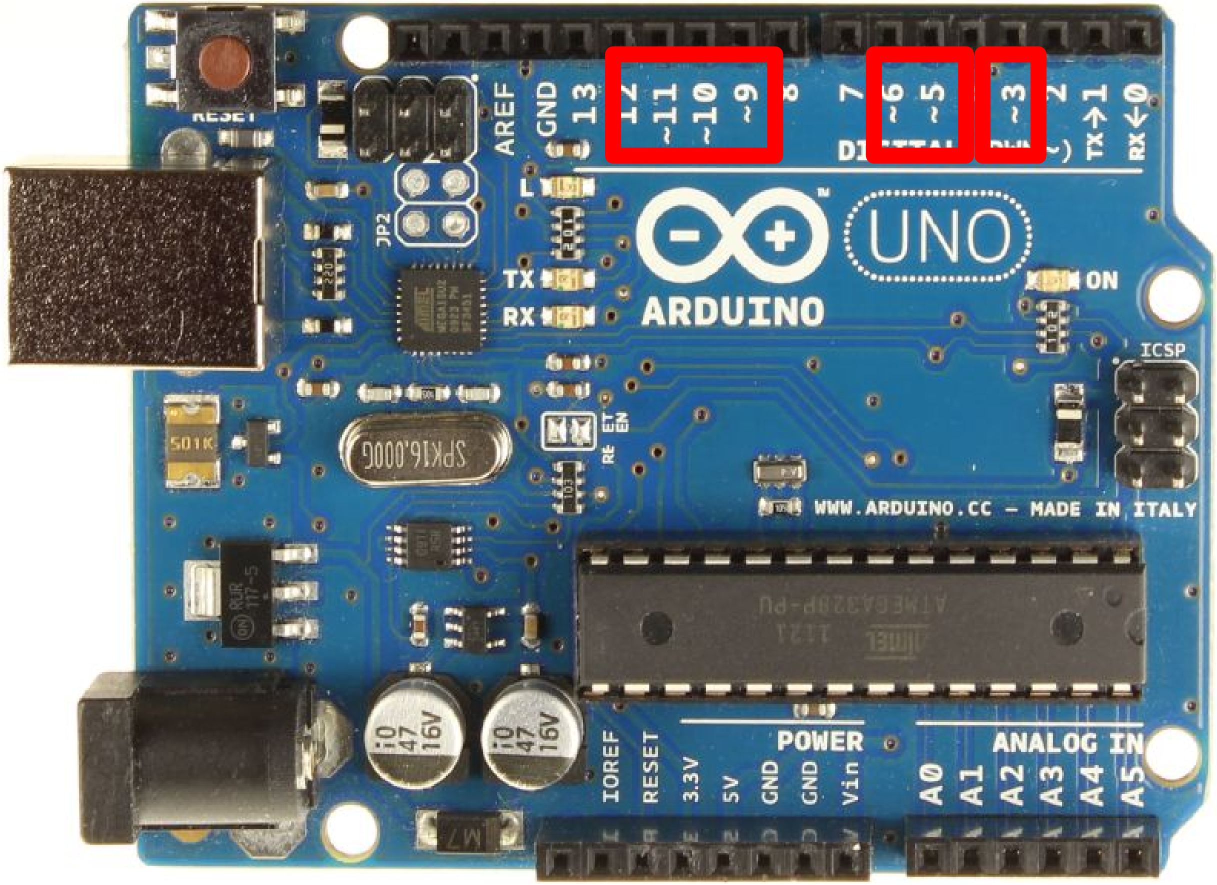
```
215
234
233
213
214
235
234
214
215
235
235
215
215
236
235
216
216
237
237
```

☑ Autoscroll

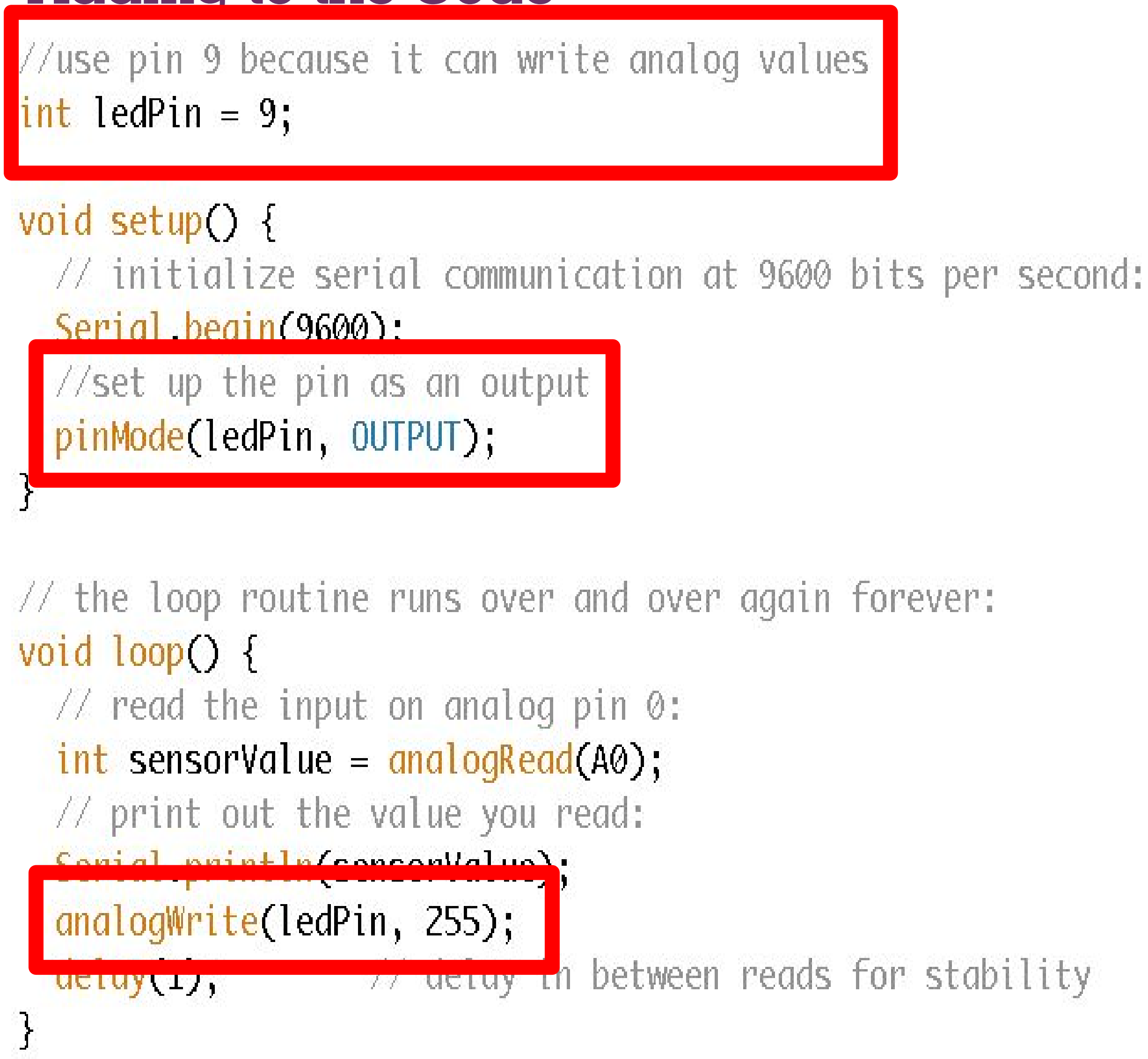
Both NL & CR

9600 baud

Analog Output

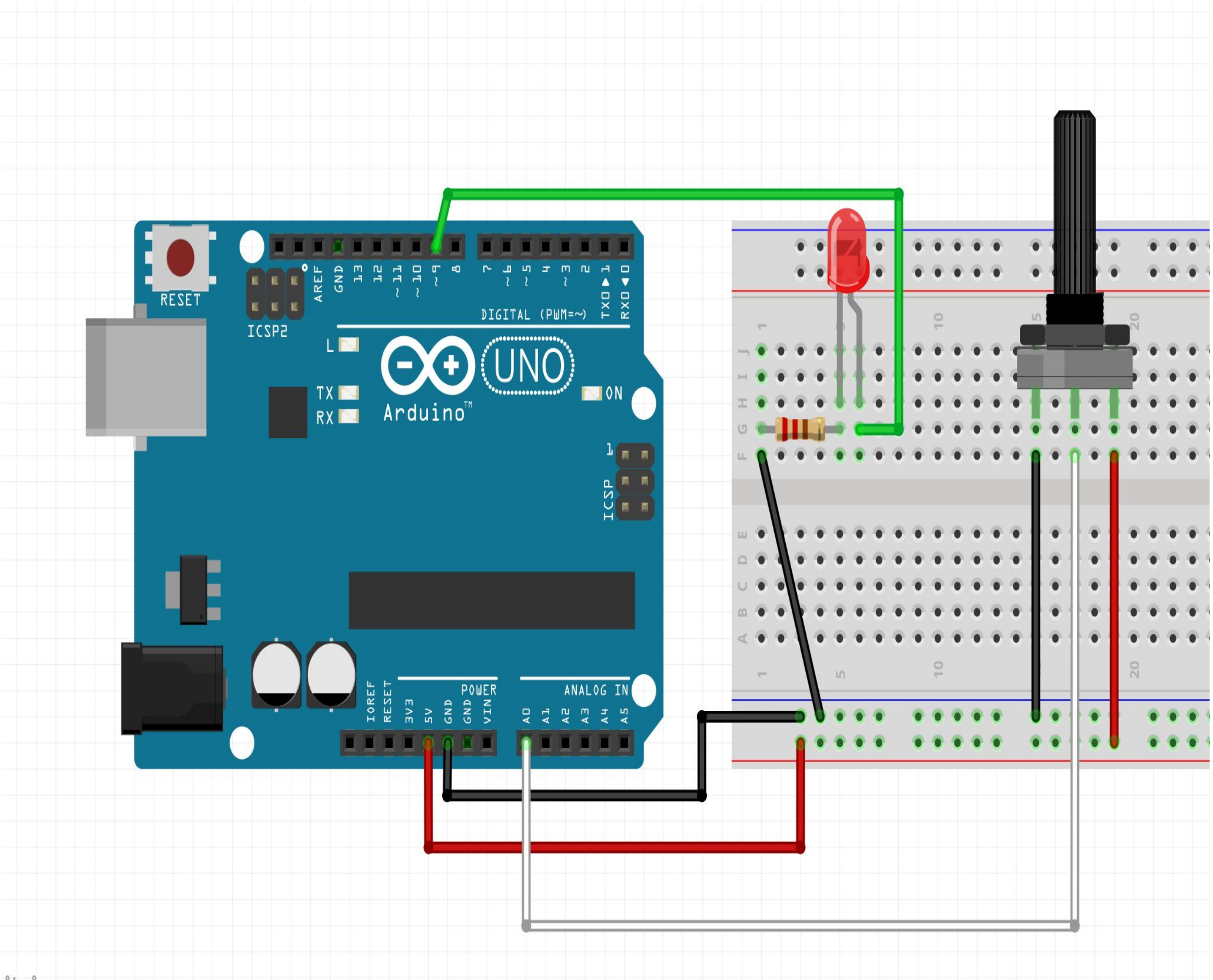


Adding to the Code

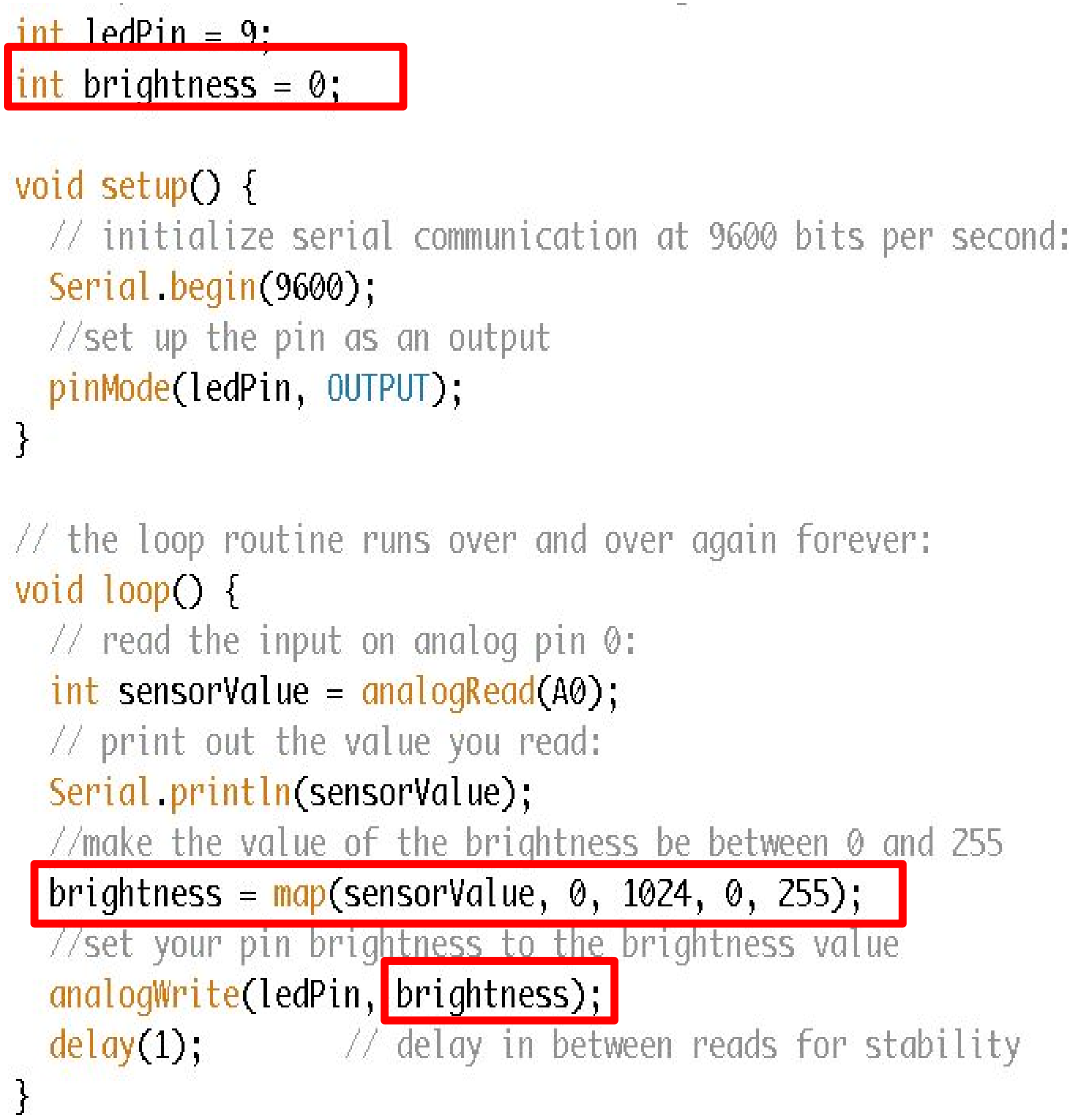


```
//use pin 9 because it can write analog values
int ledPin = 9;

void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  //set up the pin as an output
  pinMode(ledPin, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  analogWrite(ledPin, 255);
  delay(1);        // delay in between reads for stability
}
```

Connecting the LED



Adding to the Code

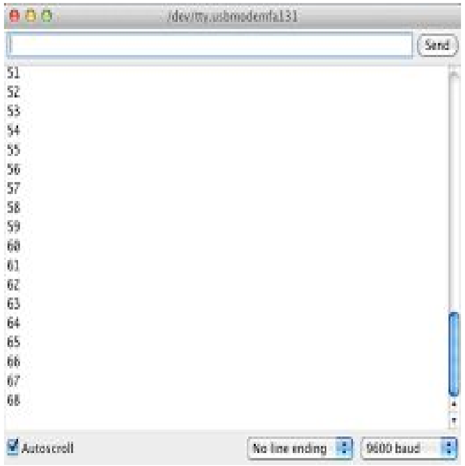


```
int ledPin = 9;
int brightness = 0;

void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  //set up the pin as an output
  pinMode(ledPin, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  //make the value of the brightness be between 0 and 255
  brightness = map(sensorValue, 0, 1024, 0, 255);
  //set your pin brightness to the brightness value
  analogWrite(ledPin, brightness);
  delay(1);        // delay in between reads for stability
}
```

PWM out - PWM in

PWM →

Serial →



```
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
```

☑ Autoscroll

No line ending

9600 baud

PWM - pulseIn()

The pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds.

```
byte PWM_PIN = 3;

int pwm_value;

void setup() {
  pinMode(PWM_PIN, INPUT);
  Serial.begin(115200);
}

void loop() {
  pwm_value = pulseIn(PWM_PIN, HIGH);
  Serial.println(pwm_value);
}
```

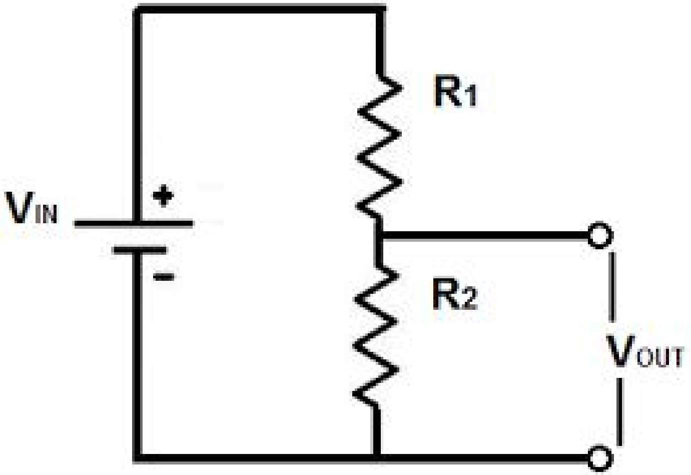
```
void setup() {
  pinMode(3,OUTPUT);
}

void loop() {
  analogWrite(3,255);
  delay (100);
  analogWrite(3,0);
  delay (100);

}
```

PWM Interrupt

Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3). These interrupts can be set to trigger on RISING or FALLING signal edges, or on low level. Once attached, when an interrupt is triggered, the specified interrupt service routine (ISR) will be called.

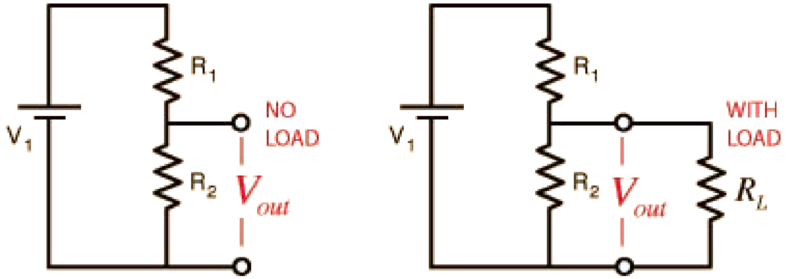
```
volatile int pwm_value = 0;
volatile int prev_time = 0;

void setup() {
  Serial.begin(9600);
  // when pin D2 goes high, call the rising
   function
  attachInterrupt(0, rising, RISING);
}
void loop() { }
void rising() {
  attachInterrupt(0, falling, FALLING);
  prev_time = micros();}

void falling() {
  attachInterrupt(0, rising, RISING);
  pwm_value = micros()-prev_time;
  Serial.println(pwm_value);}
```

Voltage Divider



OPEN CIRCUIT BEHAVIOR

BEHAVIOR UNDER LOAD



$$V_{out} = V_1 \frac{IR_2}{I(R_1+R_2)} = \frac{V_1R_2}{(R_1+R_2)}$$

OUTPUT VOLTAGE UNDER "NO LOAD" CONDITION (open circuit)

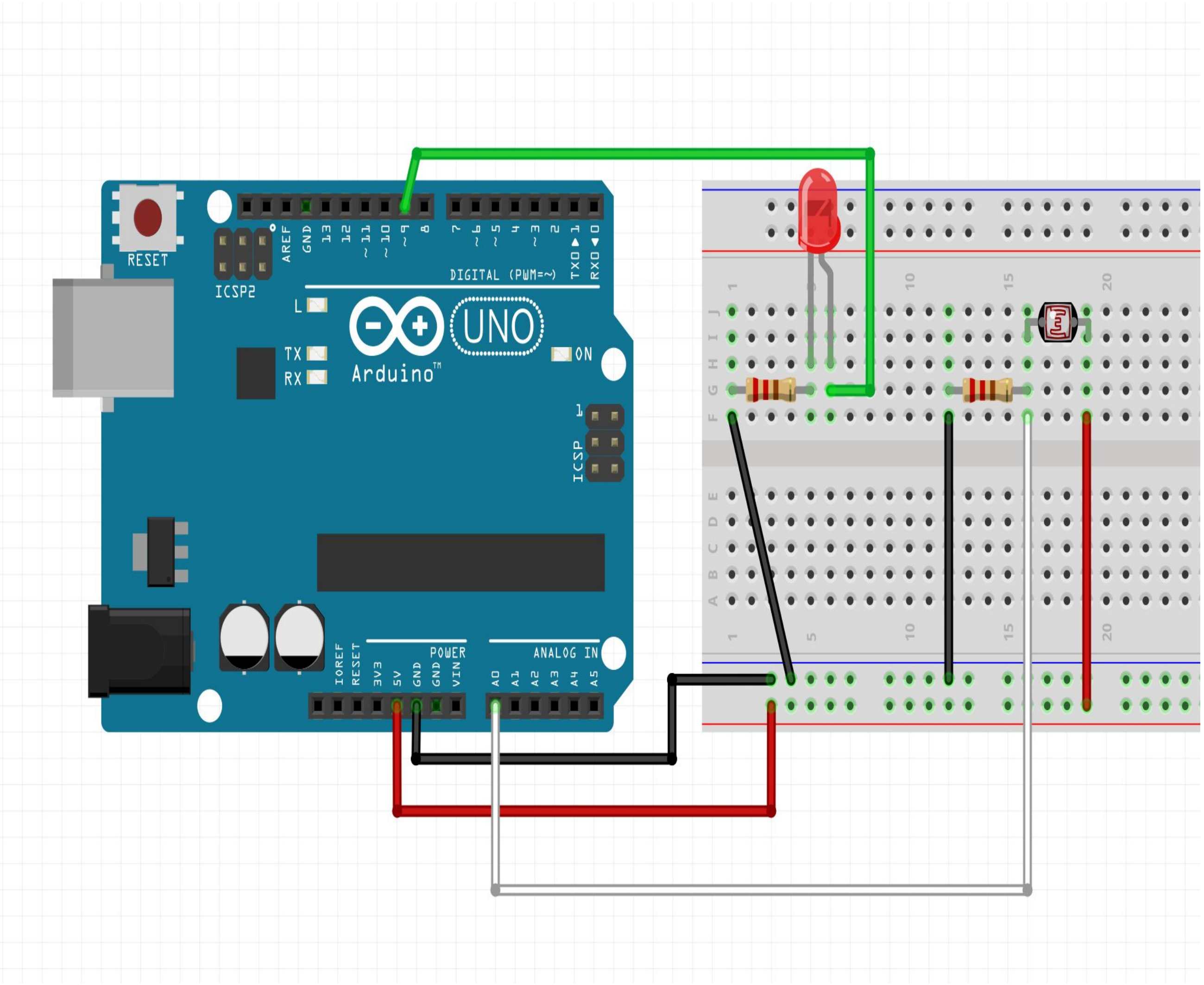
OUTPUT VOLTAGE UNDER LOAD

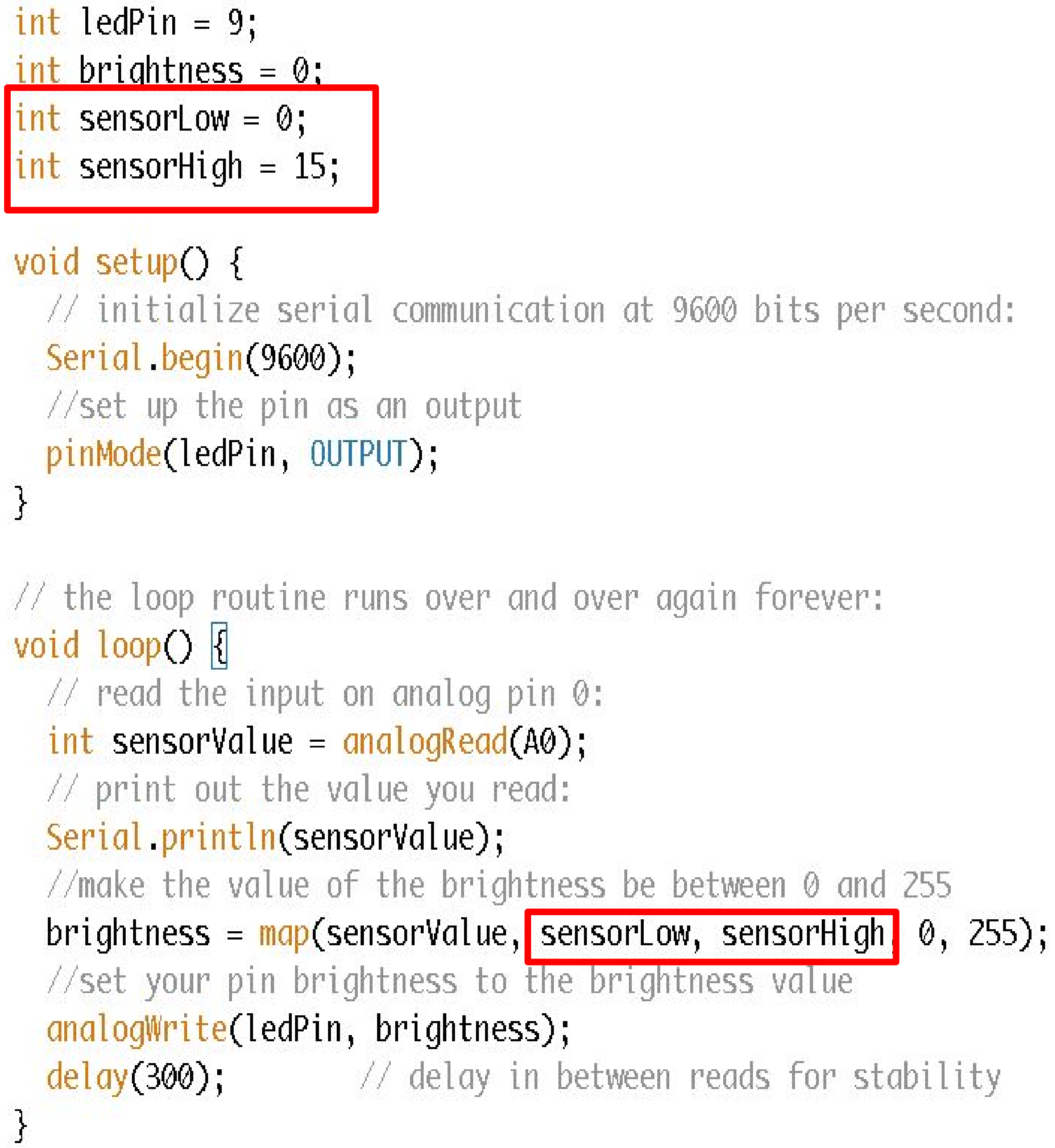
$$V_{out} = V_1 \frac{IR_2}{I(R_1+R_2)} = \frac{V_1(R_2 \| R_L)}{(R_1+R_2\| R_L)}$$

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1+R_2}$$

A voltage divider is a simple circuit which turns a large voltage into a smaller one.

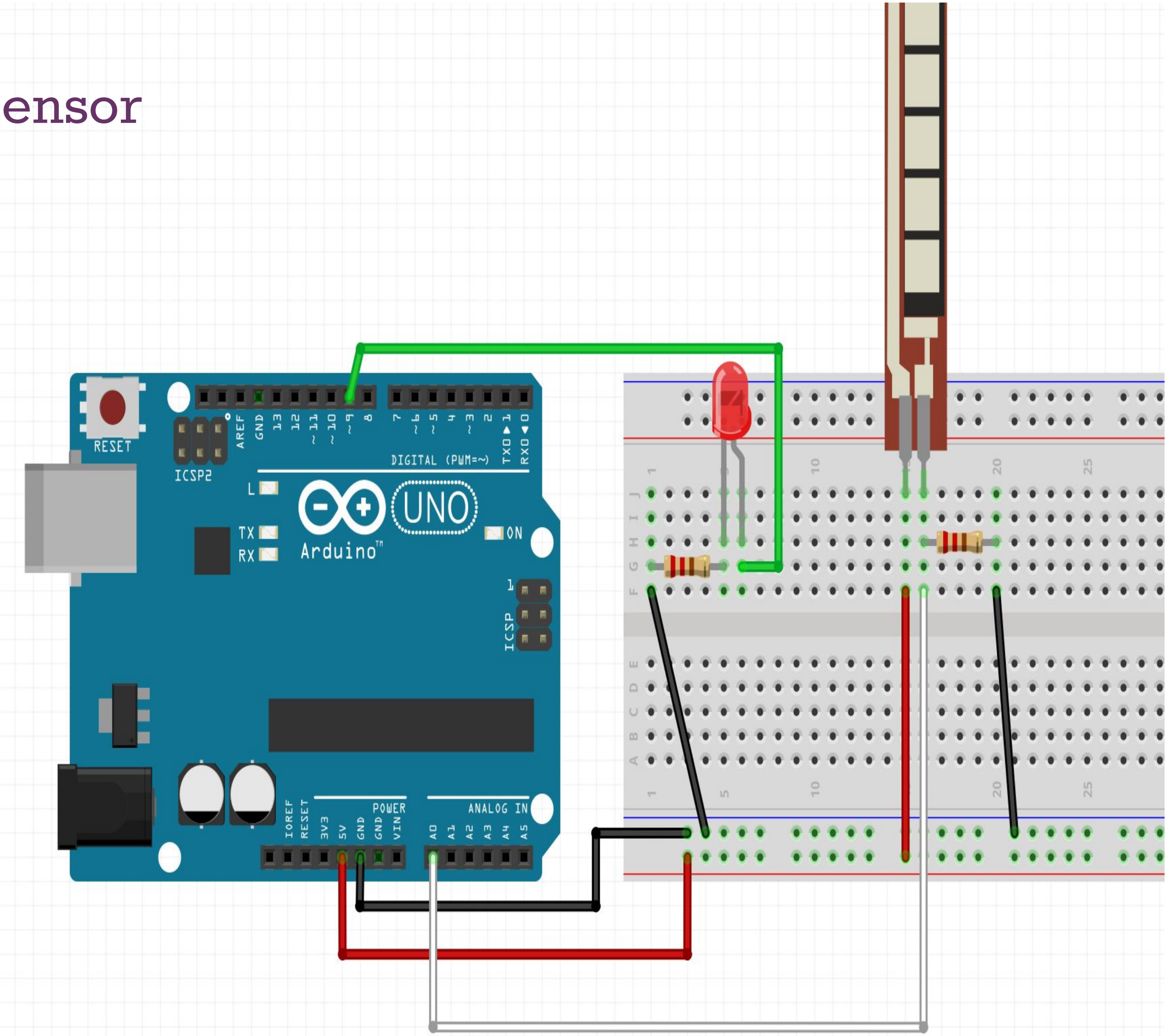
Photocell


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
1
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
1
5
10
15
20
J
I
H
G
F
E
D
C
B
A



```
int ledPin = 9;
int brightness = 0;
int sensorLow = 0;
int sensorHigh = 15;

void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  //set up the pin as an output
  pinMode(ledPin, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  //make the value of the brightness be between 0 and 255
  brightness = map(sensorValue, sensorLow, sensorHigh, 0, 255);
  //set your pin brightness to the brightness value
  analogWrite(ledPin, brightness);
  delay(300);        // delay in between reads for stability
}
```

Flex Sensor


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
1
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
J
I
H
G
F
E
D
C
B
A
1
5
10
20
25

Soldering!

Homework

Get the class code up and running.

Then, try it with a sensor we didn't cover in class.

Take a five second video.

Push code to git.

Make a blog post of a project idea for this sensor with video and git link.

Send me the blog post link.

Bring your project to class – we'll be sharing our demos!

Homework

- Take the project we did in class today and solder it.

NOTE: You can use any analog sensor you want.