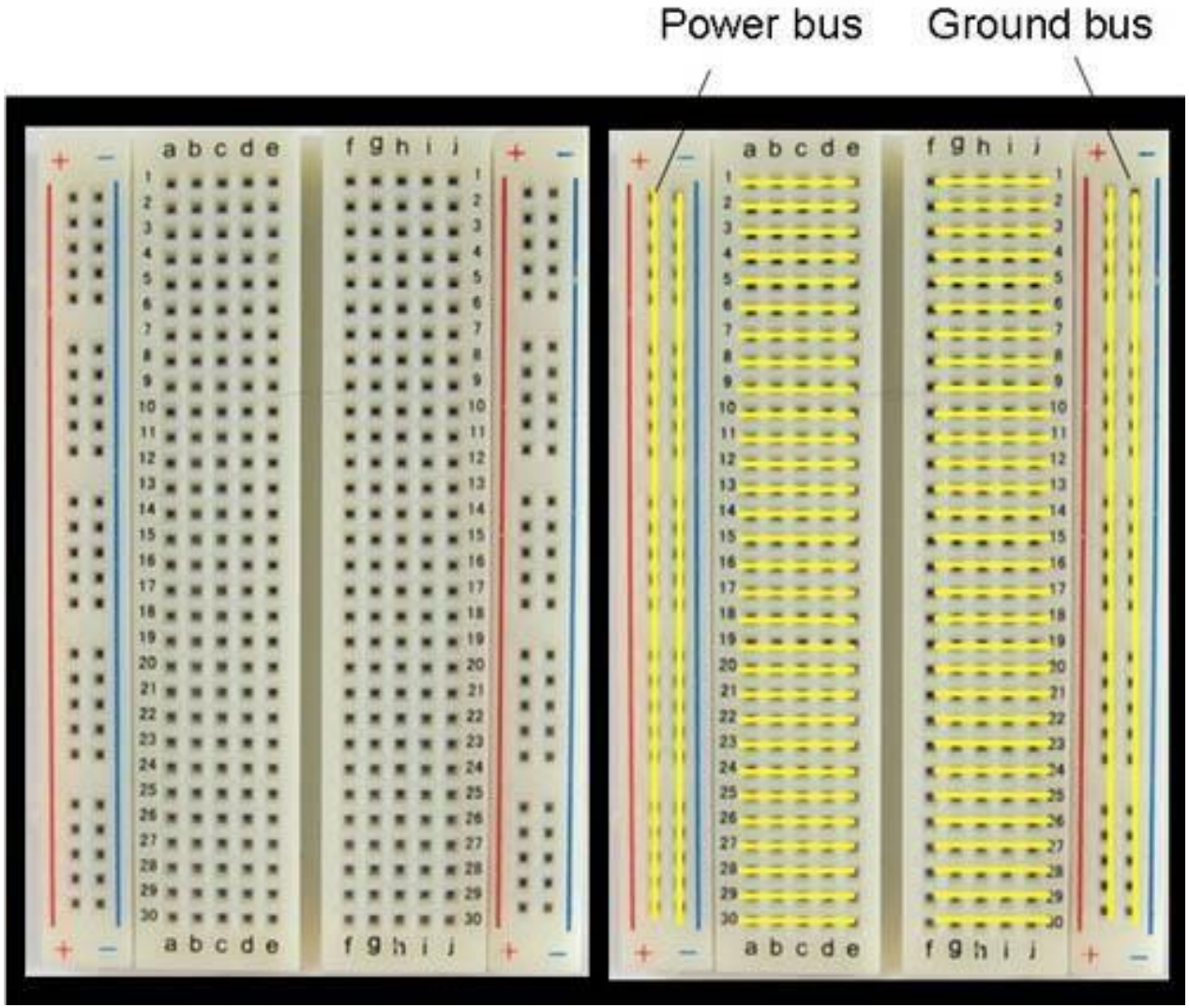
Module 3
Arduino Programming

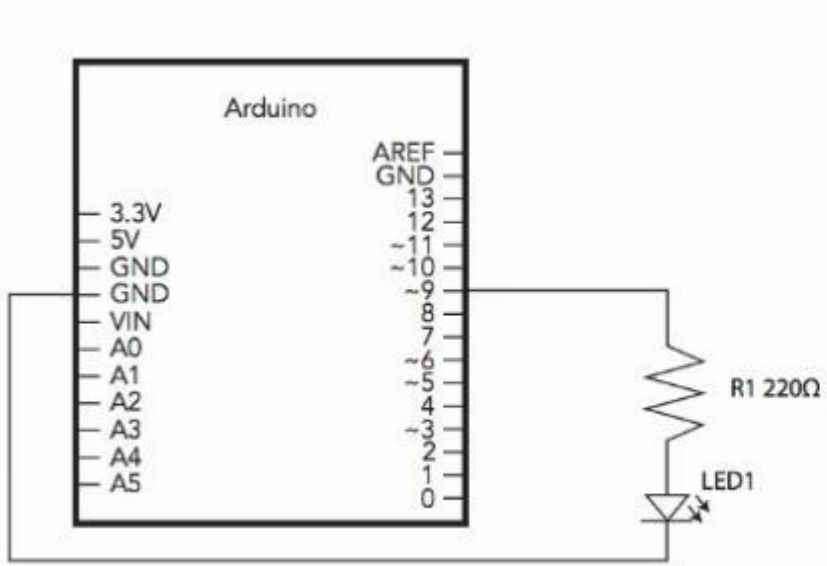
Electrical Components

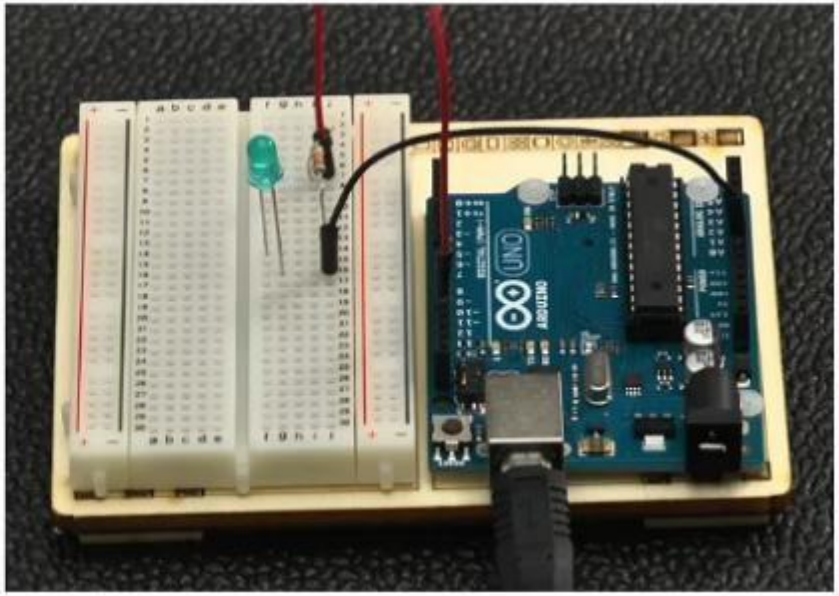
Solderless Breadboard



Solderless Breadboard

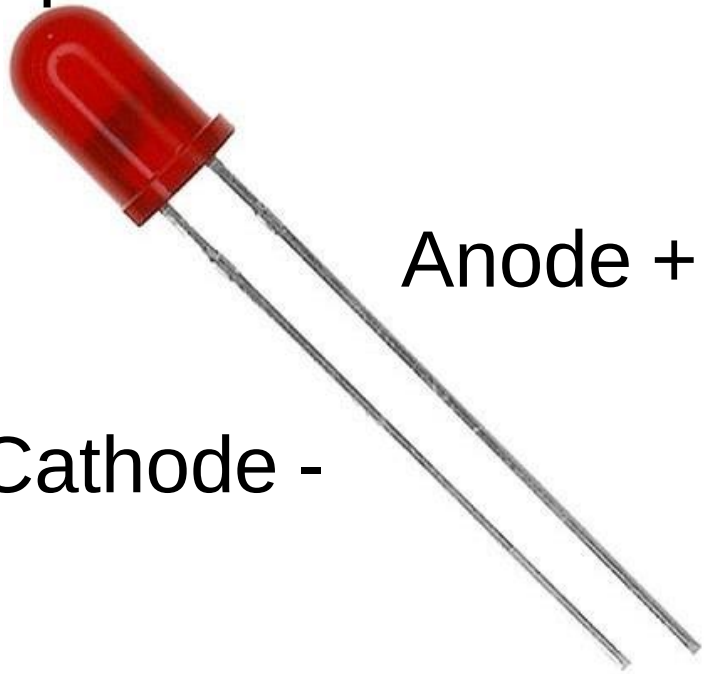
Prototyping without soldering





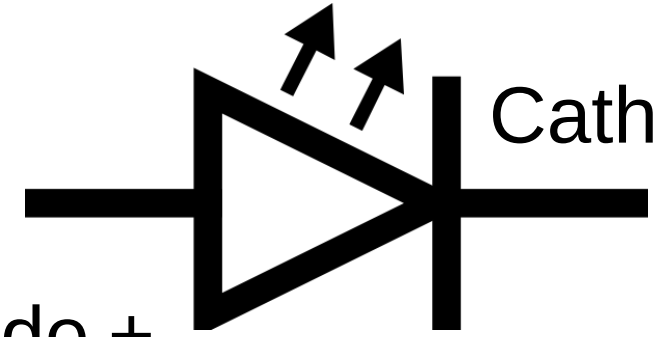
LED

- Light-emitting Diode
- Pasess current one way
- Emits Lights
- spec of current threshold : 20mA



Anode +

Cathode -

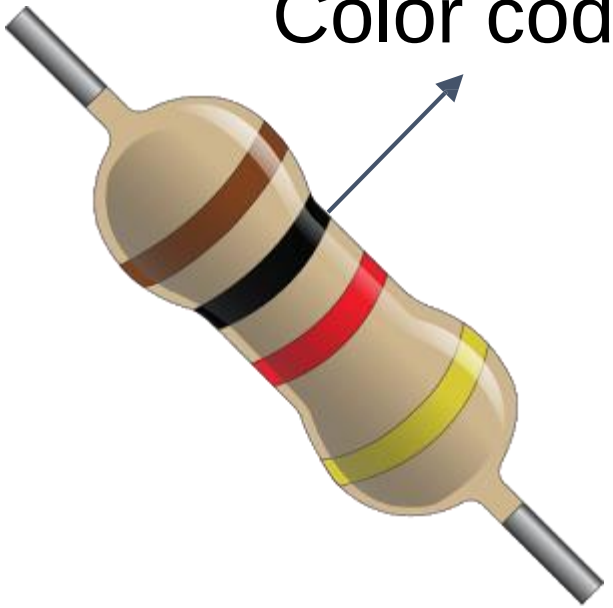


Cathode -

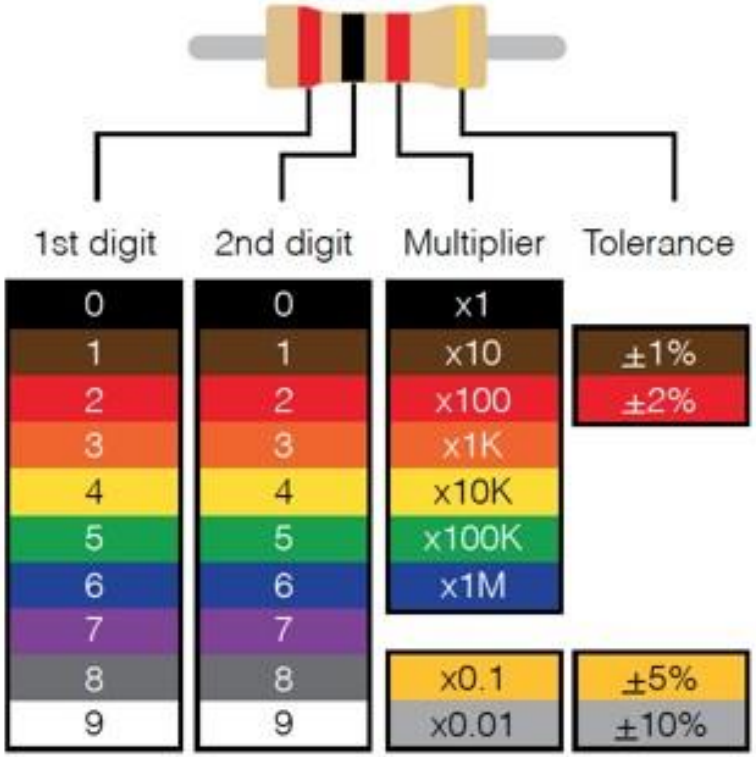
Anode +

Resistor

Color code represent the resistance value



Resistor Color Code



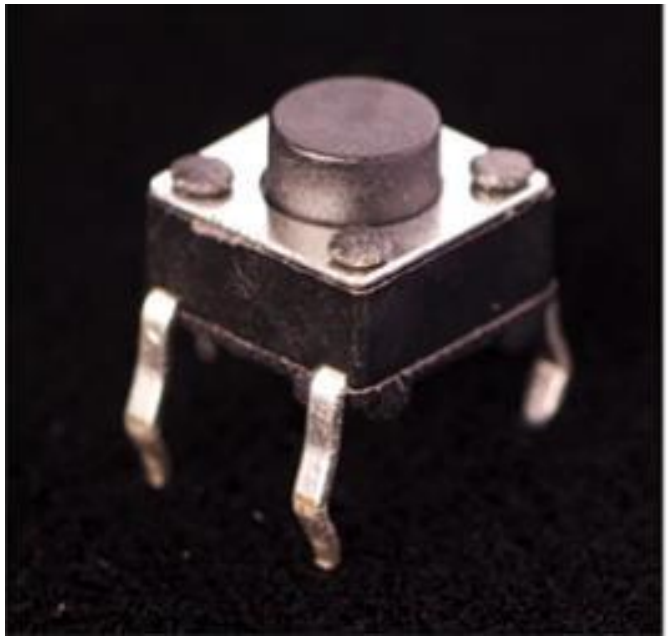
2, 2, x 10 = 220Ω

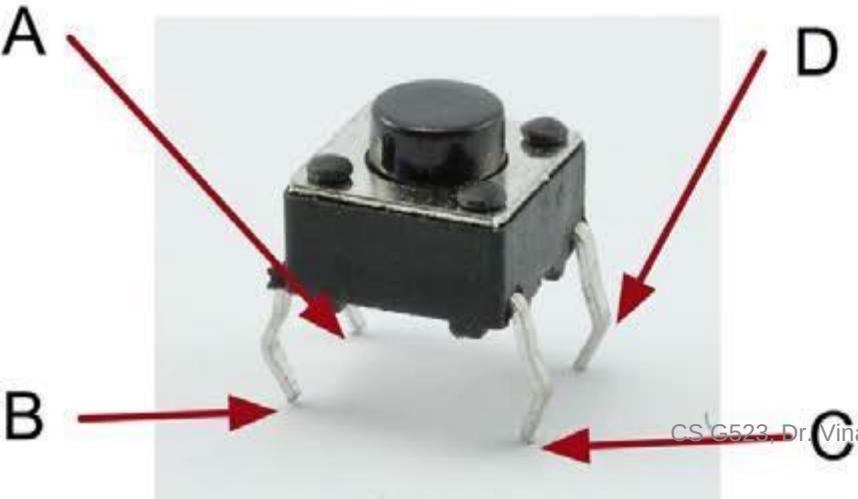
1, 1, x 1,000 = 11KΩ

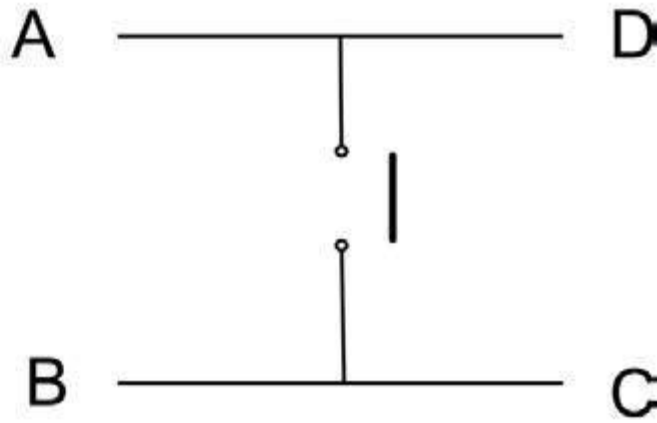
Push Button

Press to Turn On
Release to Turn Off

Generate Digital Input

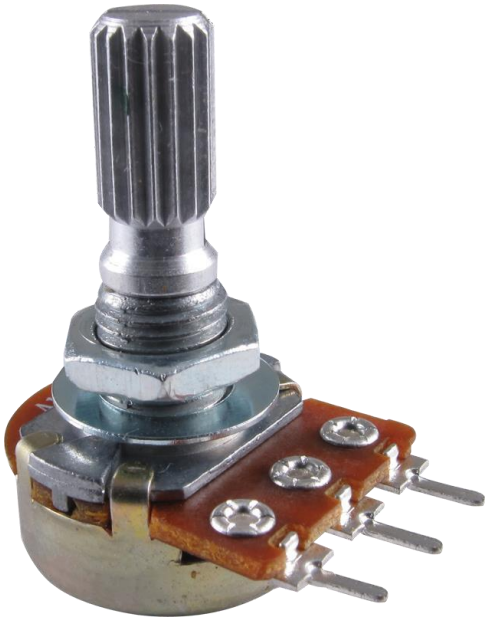






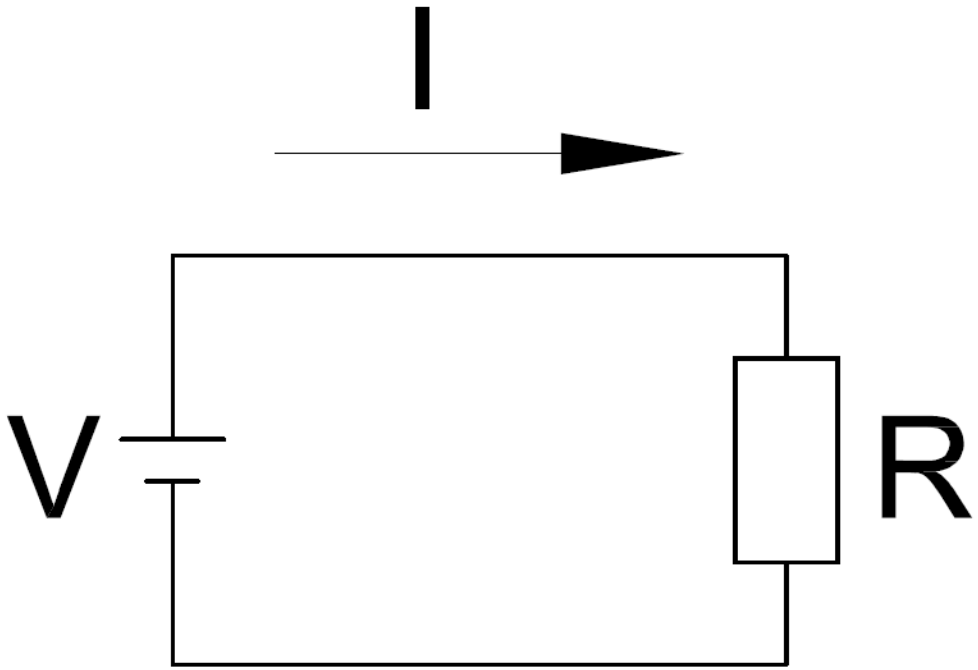
Potentiometer

Variable Resistor



Ohm's Law

Basic Concept: Ohm's Law

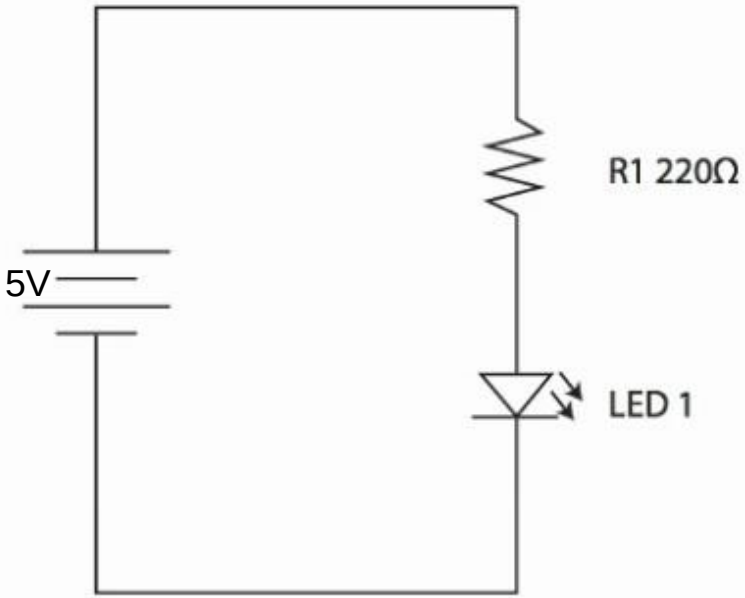


Ohm's Law

$$I = \frac{V}{R}$$

Electric current = Voltage / Resistance

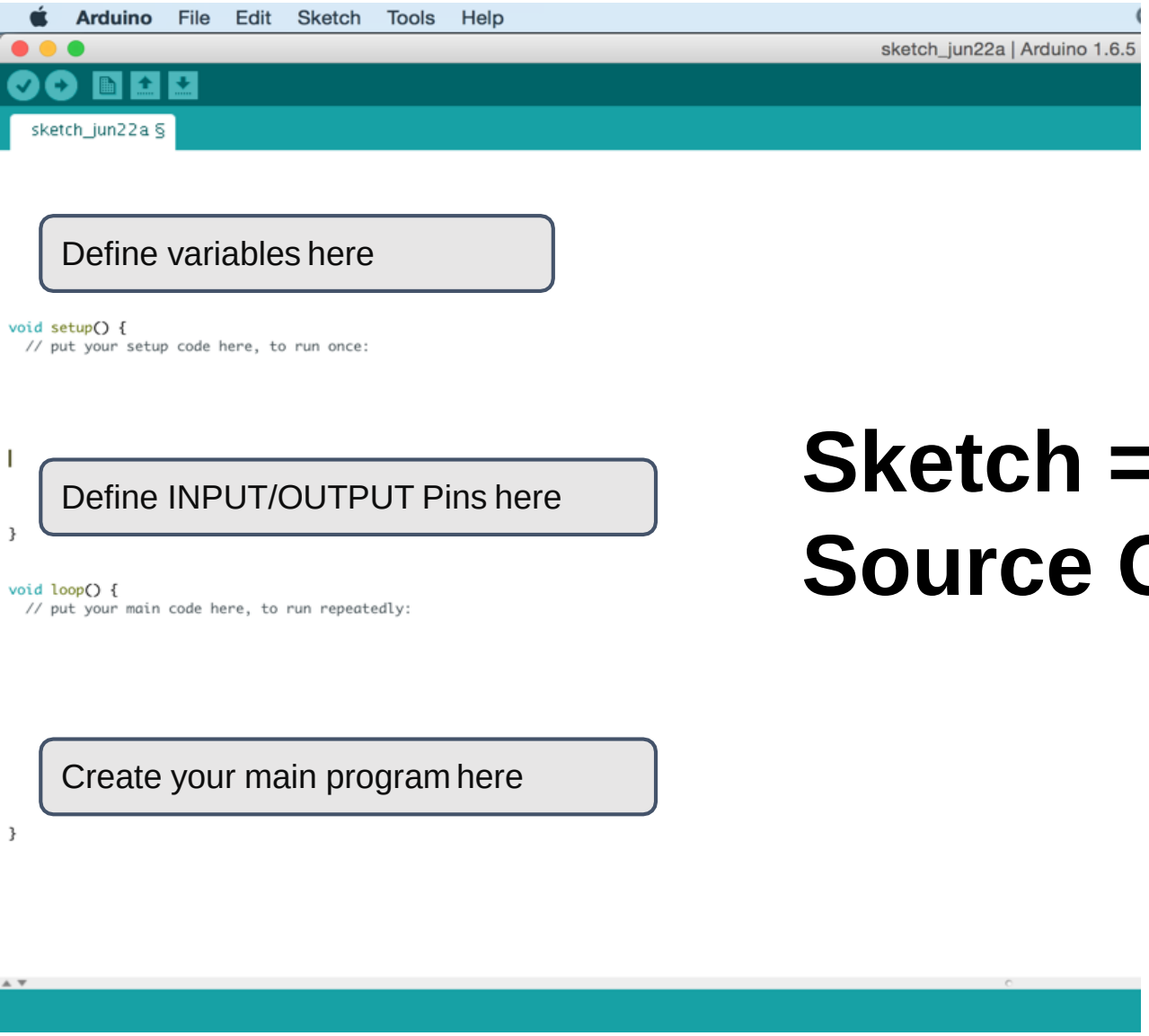
Applying Ohm's Law to LED



Voltage across LED is 2V
Voltage across R1 resistor is 5V
Maximum current for LED is 0.02A

R= 5V/0.02A= 250Ω

Arduino Program Structure



Sketch = Program = Source Code

Pin Setup

PinMode Syntax

pinMode(pin, INPUT/OUTPUT)

Digital Output

digitalWrite

Syntax:
digitalWrite(pin, HIGH/LOW)

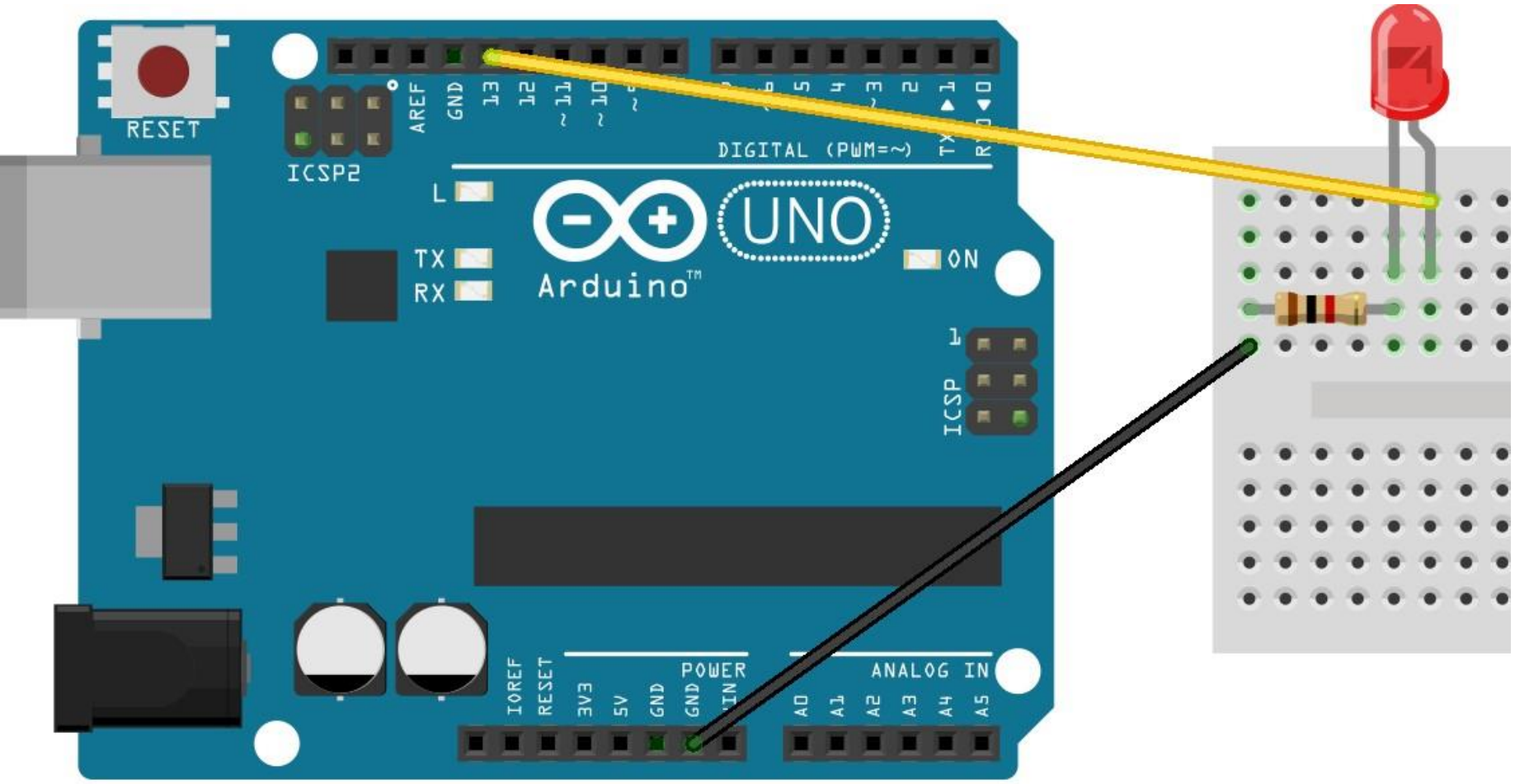
Eg
digitalWrite(9,HIGH)

delay

Pauses the program for the amount of time (in miliseconds) specified as parameter

Syntax
delay(ms)

Connection



Made

Exercise: pinMode, digitalWrite

1.Write a sketch to turn on one LED
Use digital pin 13 and use digitalWrite

Challenge:

2. Write a sketch to blink one LED

3.Write a sketch to alternate blinking 2 LEDs

Time for Exercise: 10 mins

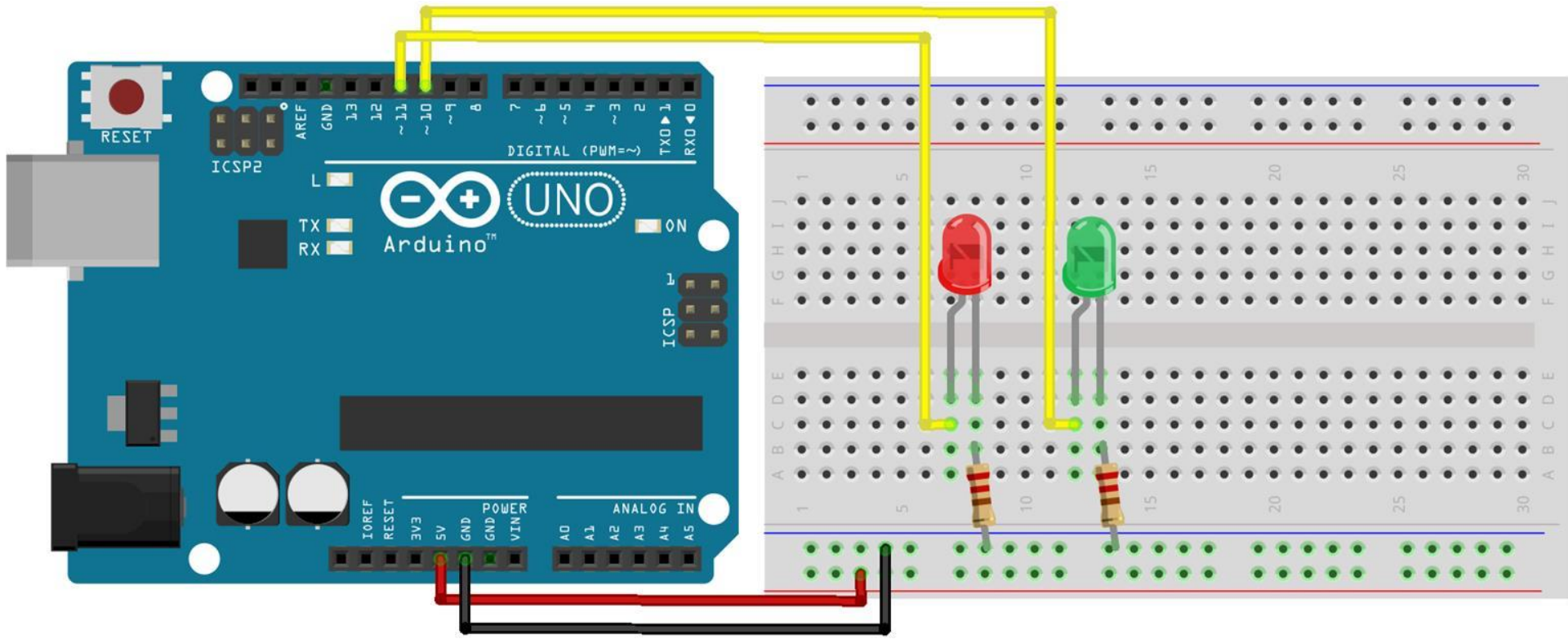
Turn on LED

```
// the setup function runs once when you
press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an
output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again
forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on
(HIGH is the voltage level)
}
```

Blink LED (Challenge)

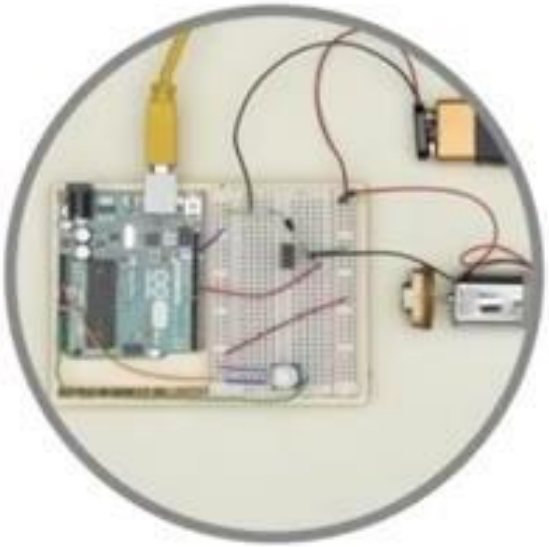
```
// the setup function runs once when you press
reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an
output.
  pinMode(12, OUTPUT);
}

// the loop function runs over and over again
forever
void loop() {
  digitalWrite(12, HIGH);   // turn the LED on
(HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(12, LOW);    // turn the LED off by
making the voltage LOW
delay(1000);                       // wait for a second
}
```

Two LED: Hint on Connection

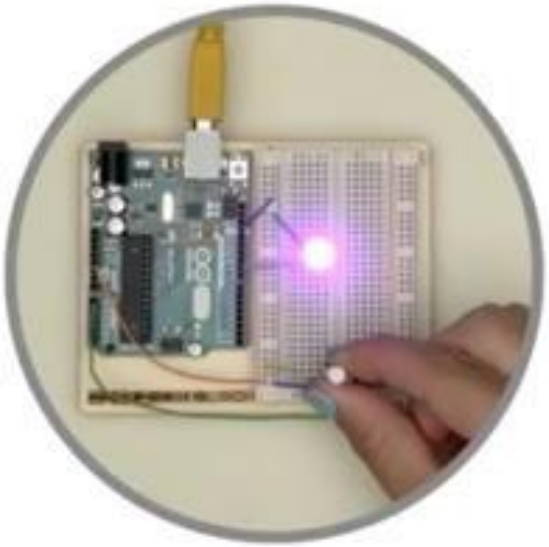


Analog Output

Analog Output Applications



Motor Speed

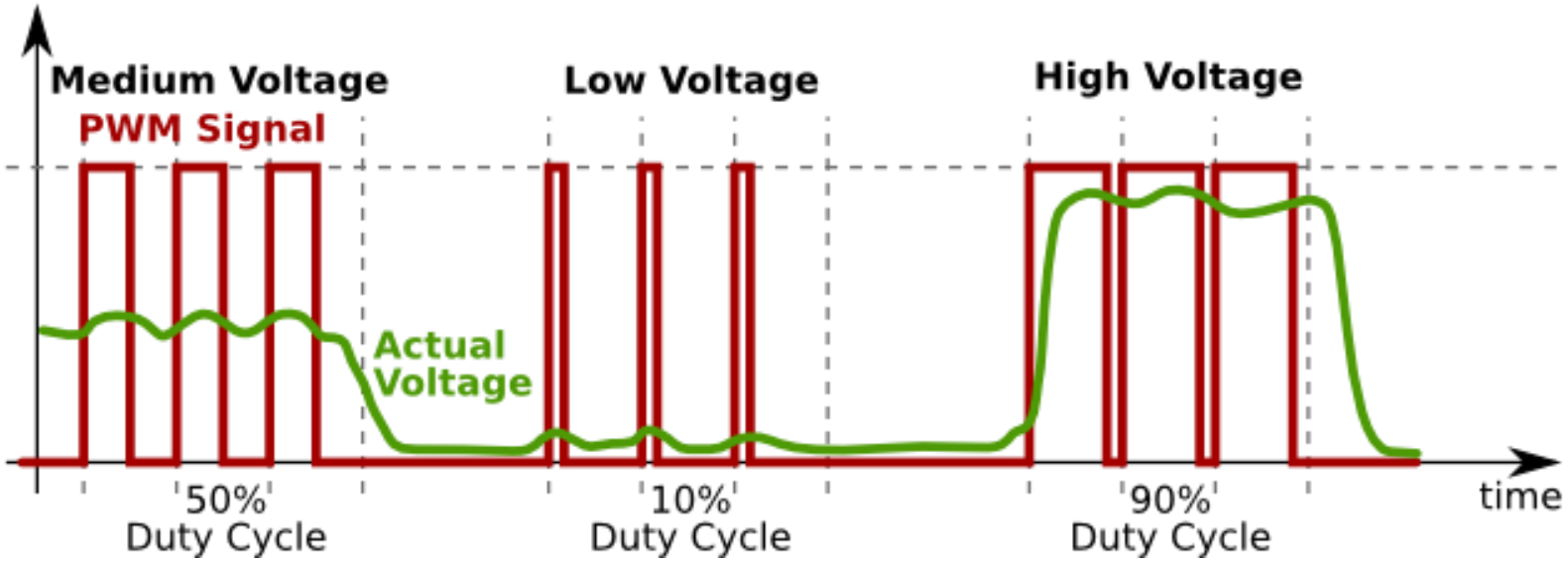


LED Brightness

Servo Angle

Pulse Width Modulation (PWM)

To create an analog signal, the microcontroller use a technique called PWM. By varying the pulse width or duty cycle, we can create an analog voltage.



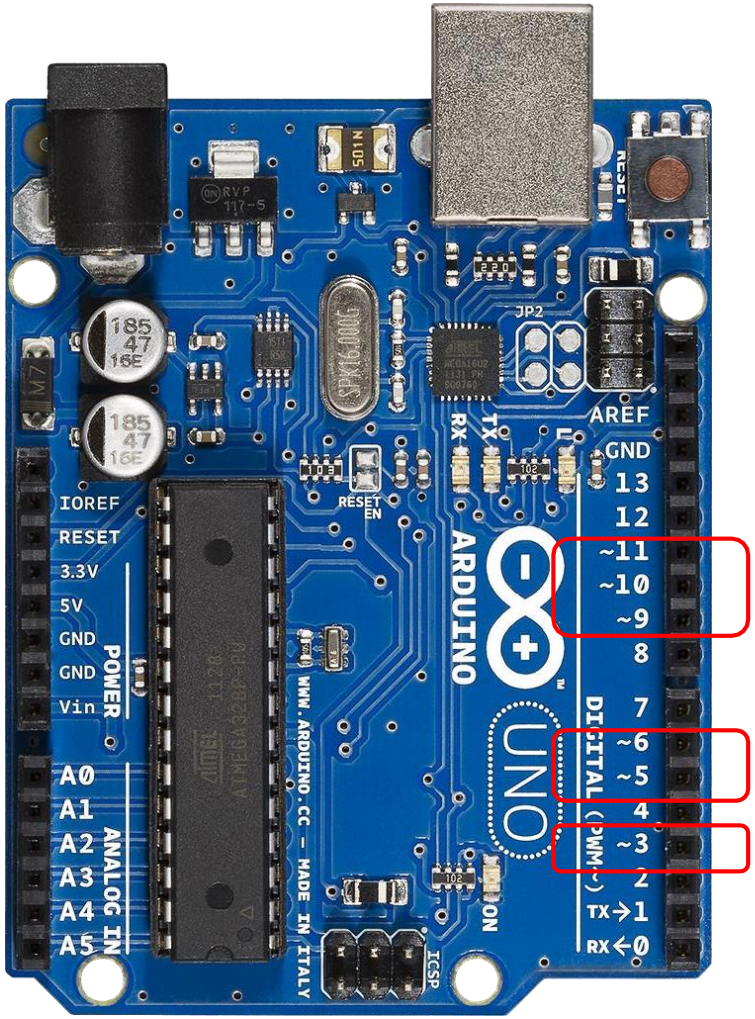
Analog Output

A few PINs on the Arduino Digital PINs allow to modify the output to mimic analog signal

Pin 3, 5, 6, 9, 10, 11

They are indicated by a "~" besides the pin no.

Analog Write Pin



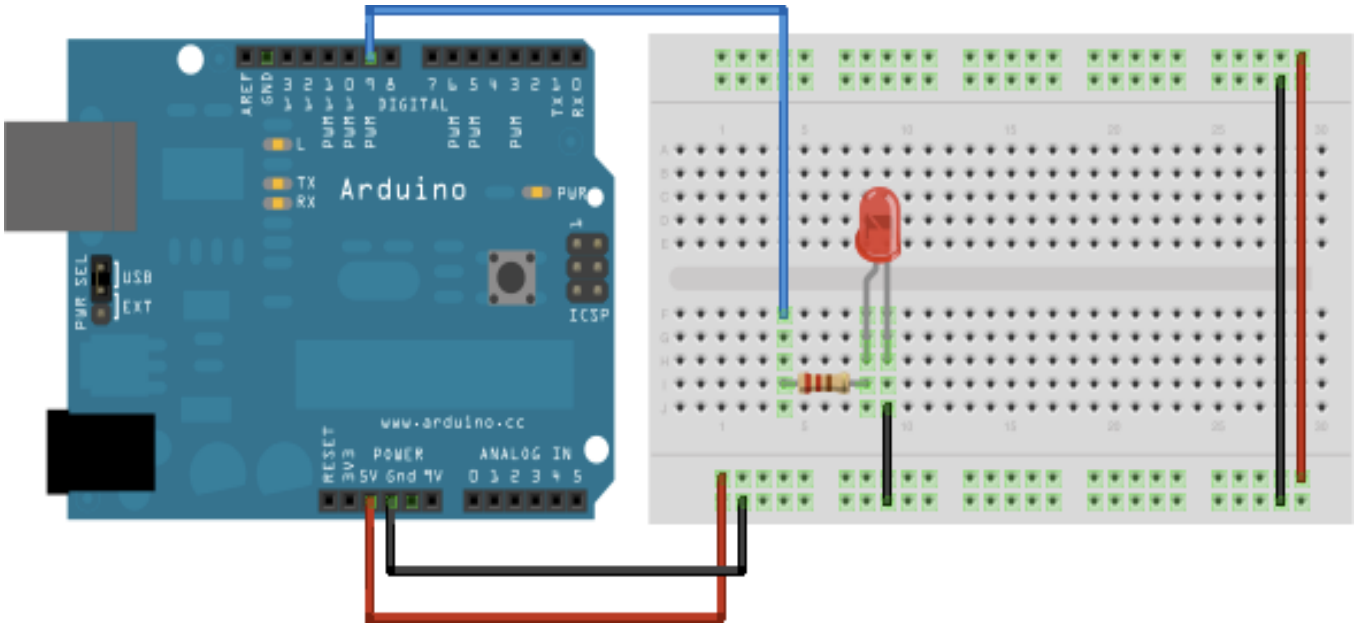
The analog write pins are those pins with ~ sign

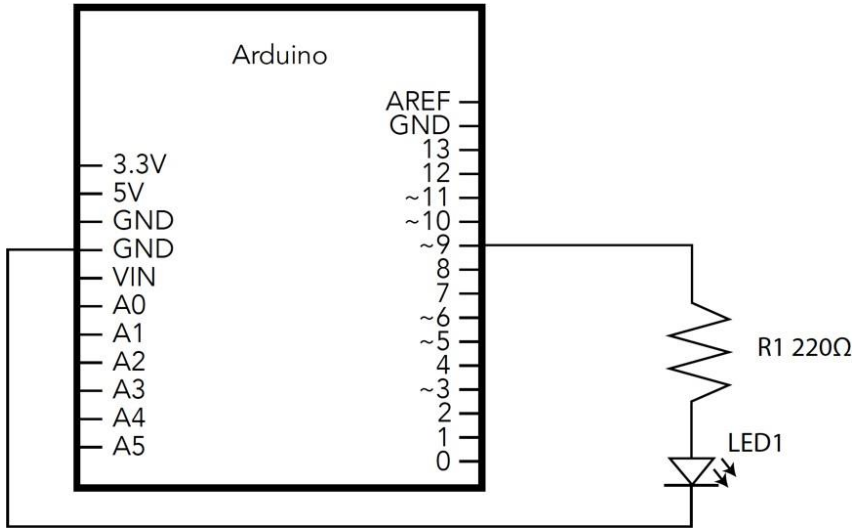
analogWrite Syntax

Syntax
analogWrite(PIN, value)

Eg
analogWrite(9, 125)

Connection


Arduino
DIGITAL
POWER
ANALOG IN
ICSP
PWR SEL
USB
EXT
www.arduino.cc


Arduino
3.3V
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
1
0
R1 220Ω
LED1

LED analog value

```
void setup() {
  // initialize digital pin LED_BUILTIN as an
output.
  pinMode(9, OUTPUT);
}

// the loop function runs over and over again
forever
void loop() {
  analogWrite(9, 50)    ;
}
```

Challenge: analogWrite

1.Write a simple sketch to change the brightness the LED light from low to high repeatedly

2.Write a sketch to fade the LED brightness from low to high, and high to low repeatedly.

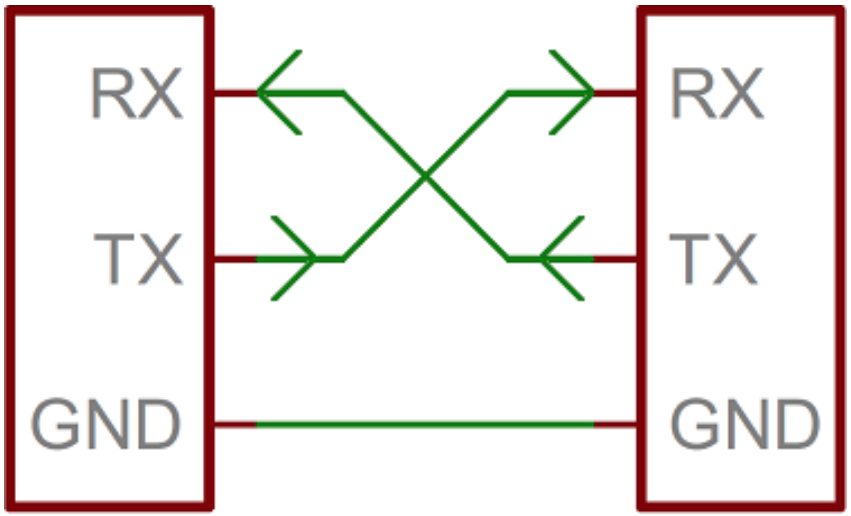
Time: 10 mins

Gradually increasing intensity (Challenge solution)

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(9, OUTPUT);
}

// the loop function runs over and over again
forever
void loop() {
  analogWrite(9, 0)   ;
  delay(1500);                    // wait for a second
  analogWrite(9, 50)  ;
  delay(1500);
  analogWrite(9, 150)  ;
  delay(1500);
  analogWrite(9, 255);
  delay(1500); // wait for a second
}
```

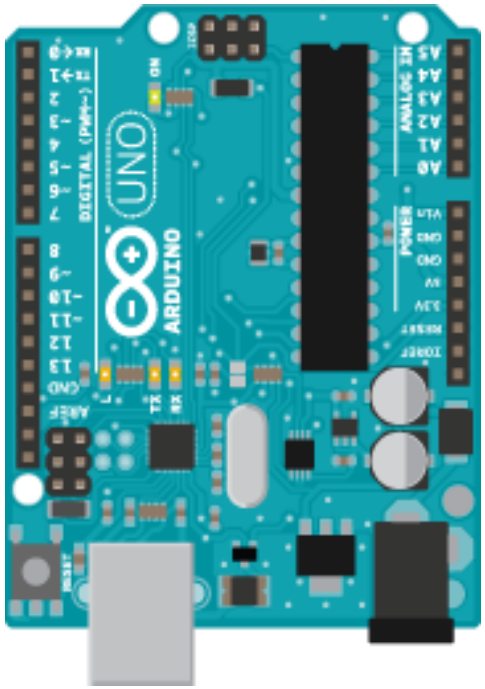
Serial Communication

A serial bus consists of just two wires - the transmitter TX wire for sending data and receiver RX wire for receiving data.

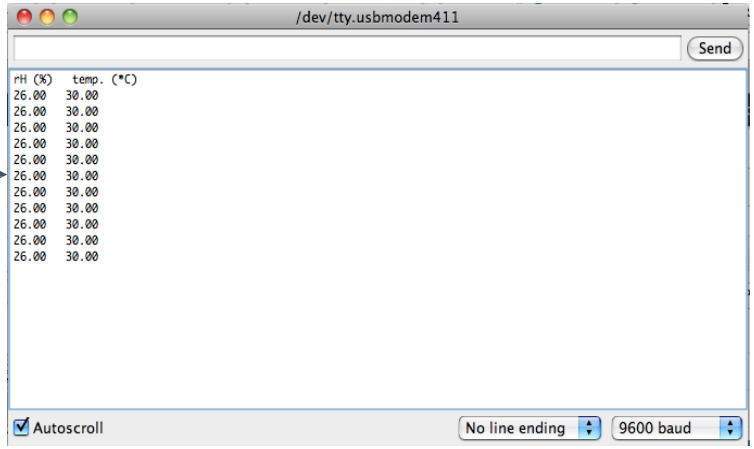


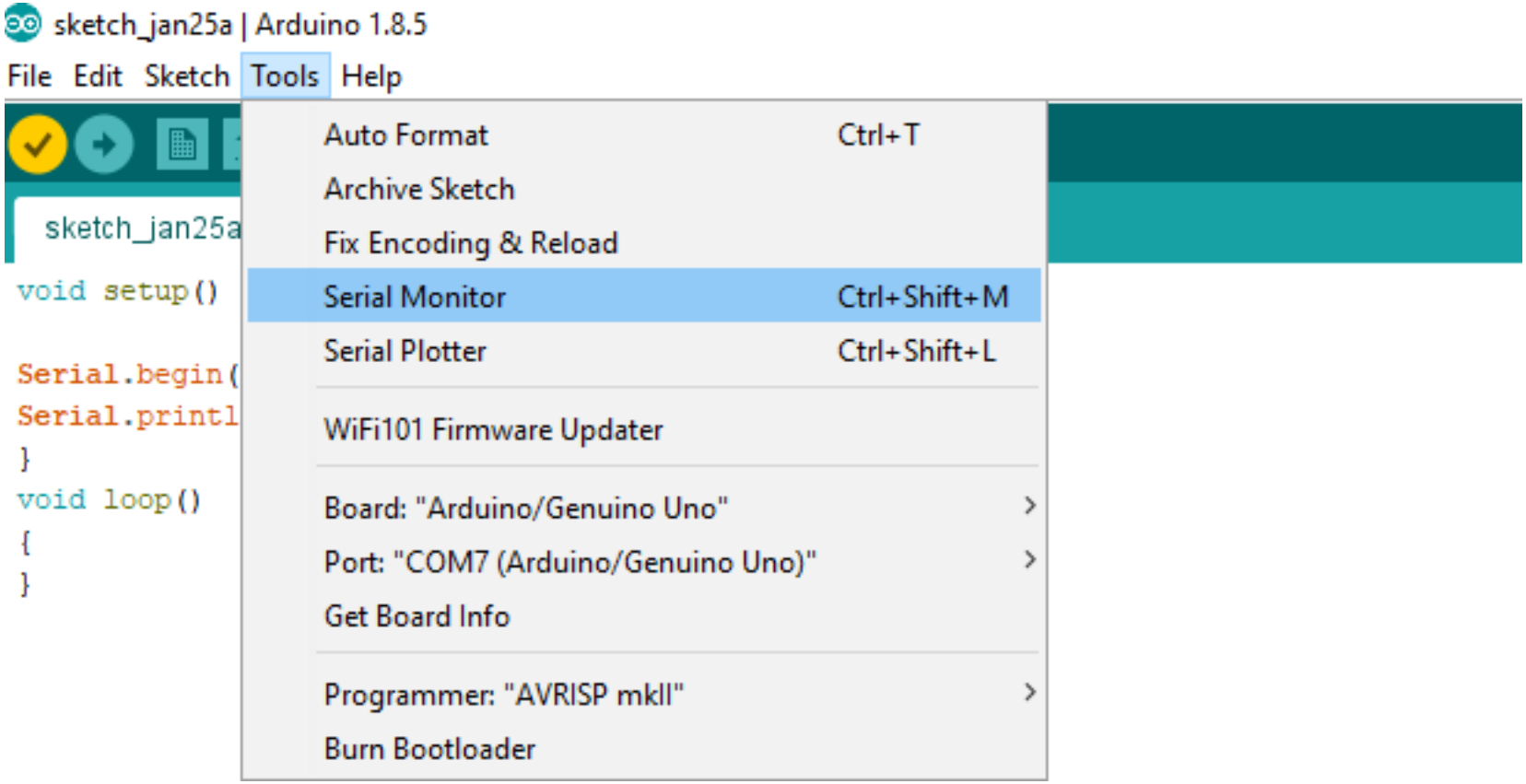
Serial Communication on Arduino

Serial.begin(baudrate)

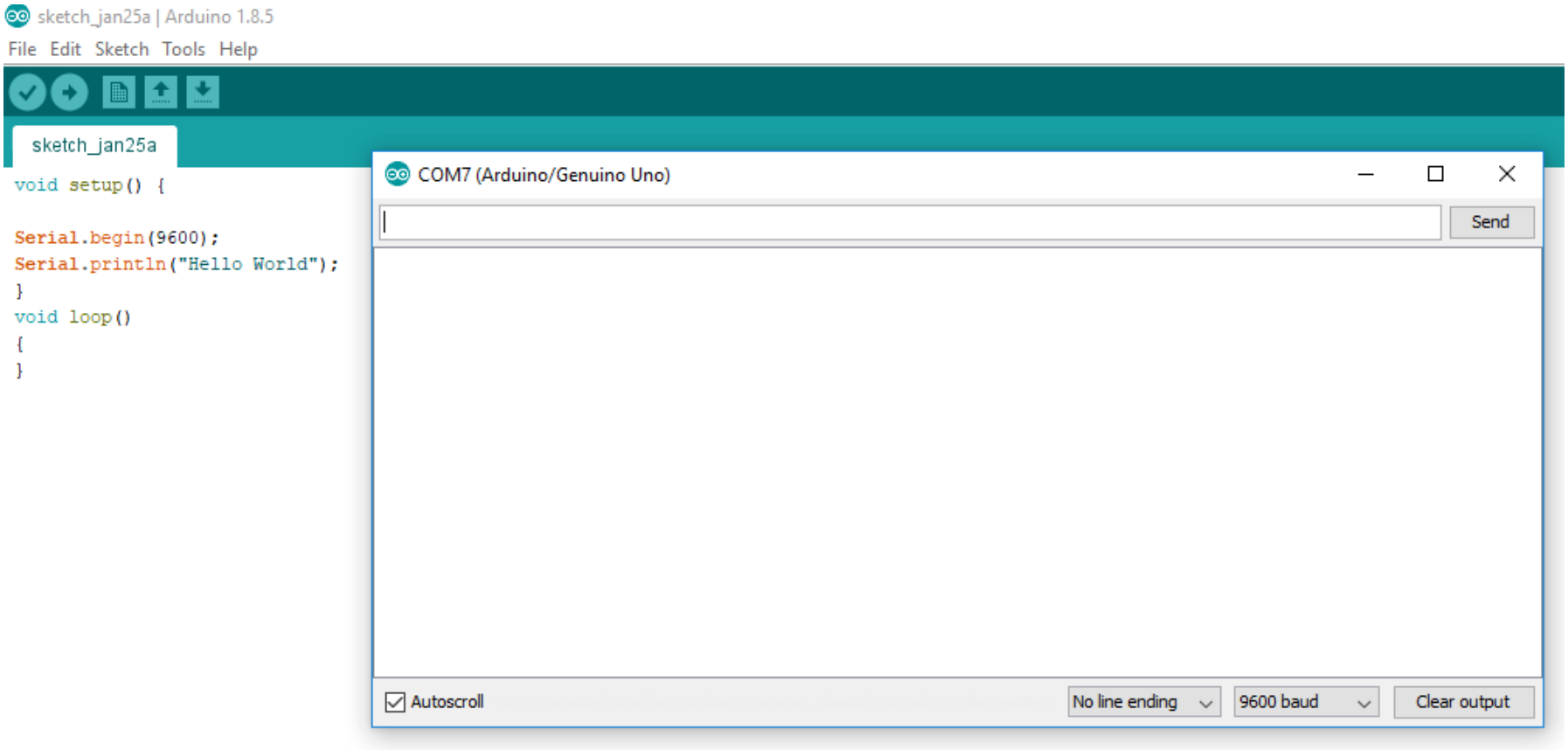


COM Port




sketch_jan25a | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_jan25a
void setup()
Serial.begin(
Serial.printl
}
void loop()
{
}
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM7 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader

Serial monitor



Print to Serial Monitor

Serial.print(value)
Serial.println(value)

Ex: Print to Serial Monitor

Print some text to the Serial Monitor. Eg

```
void setup() {

Serial.begin(9600); // set up Serial library at 9600 bps
Serial.println("Hello World");
}
void loop()
{
}
```

Print what's sent in serial monitor

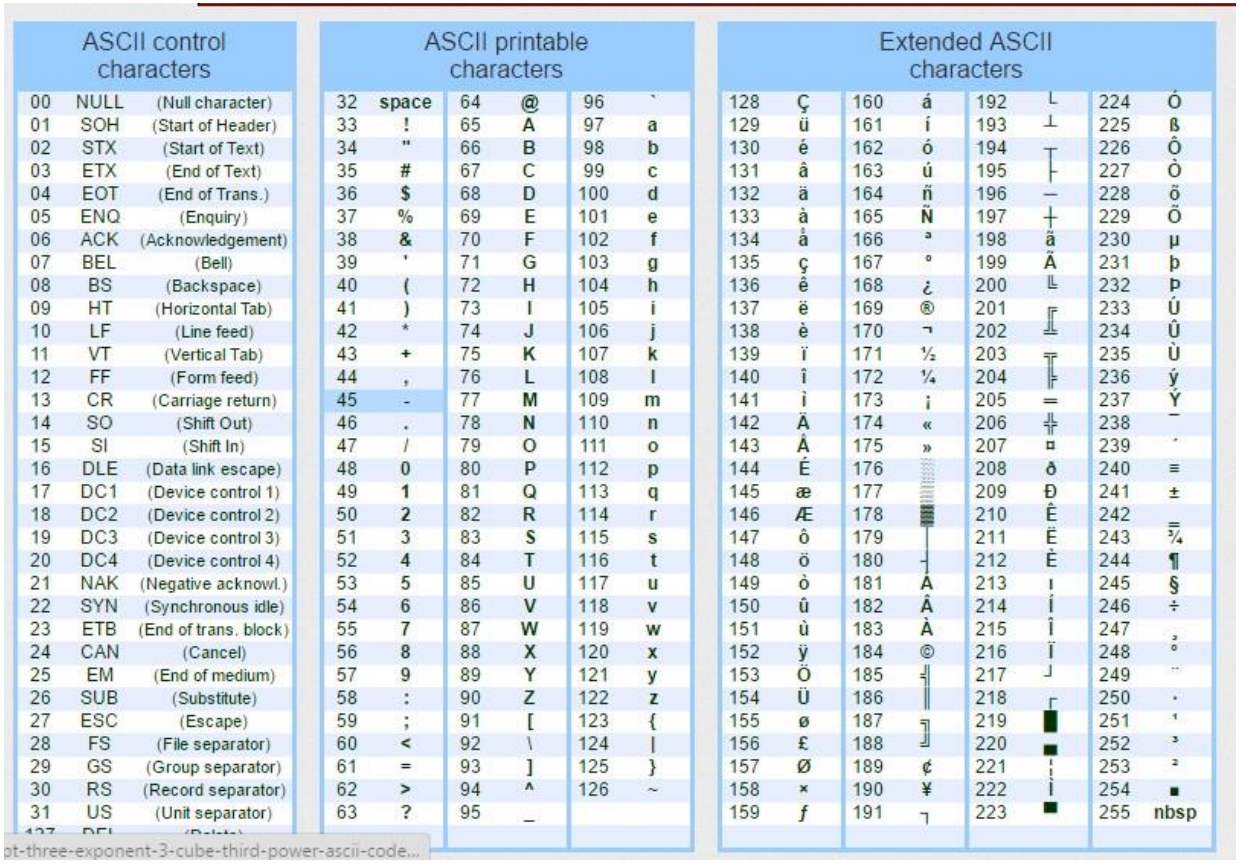
```
void setup() {
Serial.begin(9600); //set up serial library baud rate to 9600
}
void loop() {
if(Serial.available()) //if number of bytes (characters) available for
reading from serial port
{
Serial.print("      I received: "); //print I received
Serial.write(Serial.read()); //send what you read
}
}
```

Read Data from Serial Monitor

Serial.available()
Serial.read();

HINT : Serial Communication

Use the correct ascii input



ASCII control characters

00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)

ASCII printable characters

32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters

128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Serial Read Example: print ascii value

```
void setup() {
   Serial.begin(9600);
}

void loop() {
   while (Serial.available() == 0);

   int val = Serial.read() -'0';
   Serial.println(val);
}
```

Challenge!

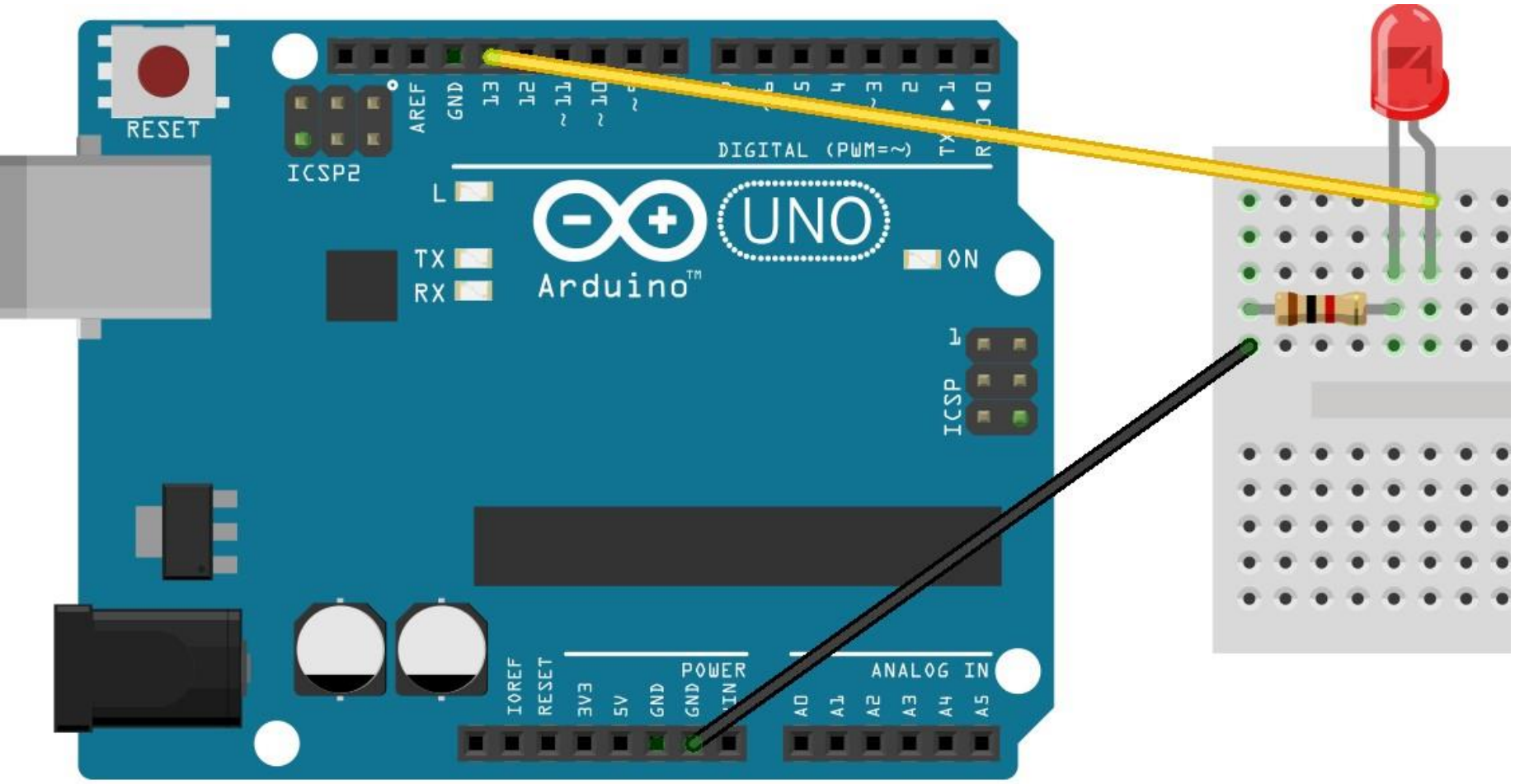
Challenge: Serial Communication

1. Use Serial Read to control the LED ON/OFF

If Serial Read value is 1, turn on LED
If Serial Read value is 0, turn off LED

Time: 10 mins

Connection



Made

Turn on/off based on inputs (Challenge)

```
void setup() {
Serial.begin(9600);
pinMode(13, OUTPUT);
}

void loop() {
while (Serial.available() == 0);
int val = Serial.read() -'0';
if(val==0)
{digitalWrite(13, LOW);}
else
{digitalWrite(13, HIGH); }
Serial.println(val);
}
```

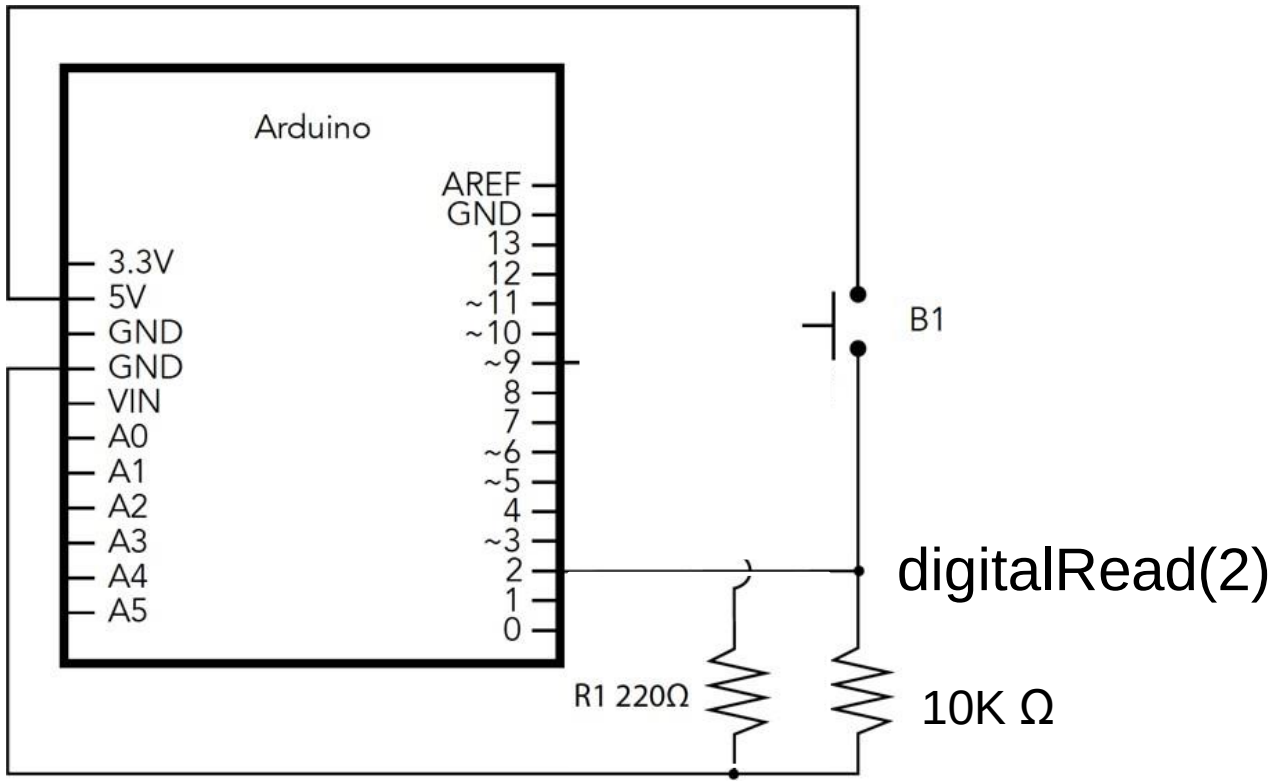
Digital Input

digitalRead Syntax

Syntax:
digitalRead(PIN)

Eg
digitalRead(2)

Connection


Arduino
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
1
0
3.3V
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
B1
digitalRead(2)
R1 220Ω
10K Ω

Exercise: digitalRead

1. Write a simple sketch to read the digital output of pin 2 and to display on serial monitor.

Setup : just a loose wire hanging from pin 2 and based on whether you connect to 5V or GND you get 1 or 0.

```
int ledPin = 2;
int val=0;    // variable to store the read value

void setup()
{
  pinMode(ledPin, INPUT);      // sets the digital pin 13 as
output
}
void loop()
{
val = digitalRead(ledPin);     // read the input pin
Serial.println(val);
}
```

Analog Input

AnalogRead Syntax

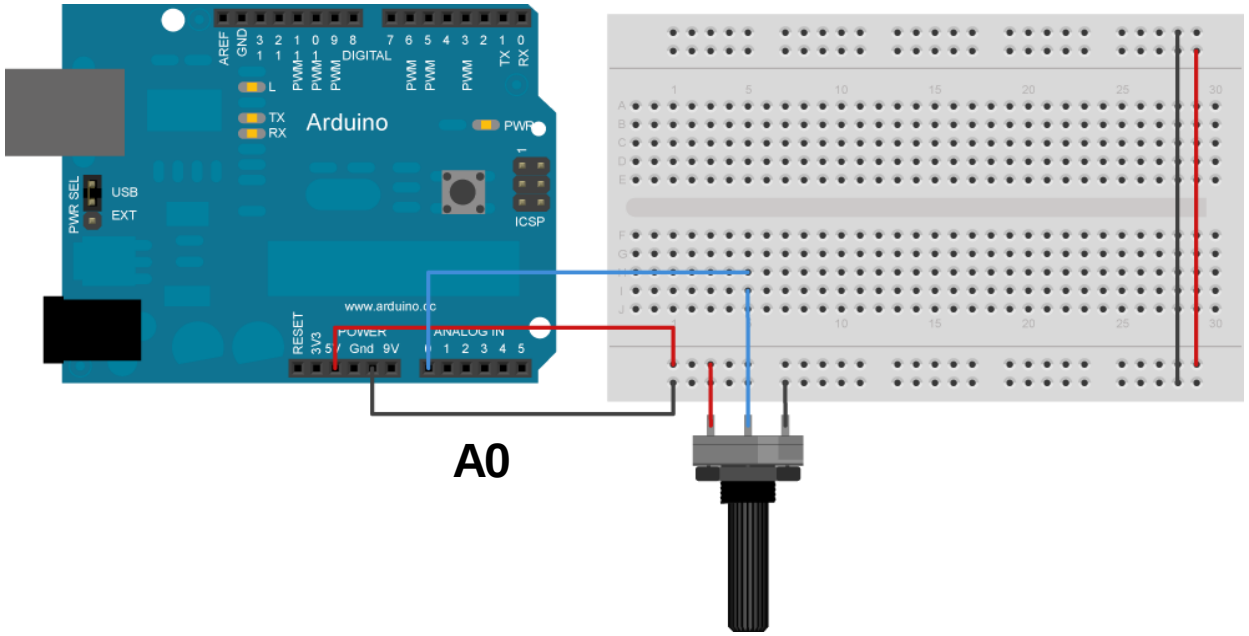
Syntax
analogRead(PIN)

Eg
analogRead(A0)

- 10-bit analog to digital converter
- map input voltages between 0 and 5 volts into integer values between 0 and 1023.

Exercise: analogRead

Write a simple sketch to read the analog output of a potentiometer/an analog pin and output to the serial monitor



Map

Syntax:
map(value, fromLow, fromHigh, toLow, toHigh)

Eg

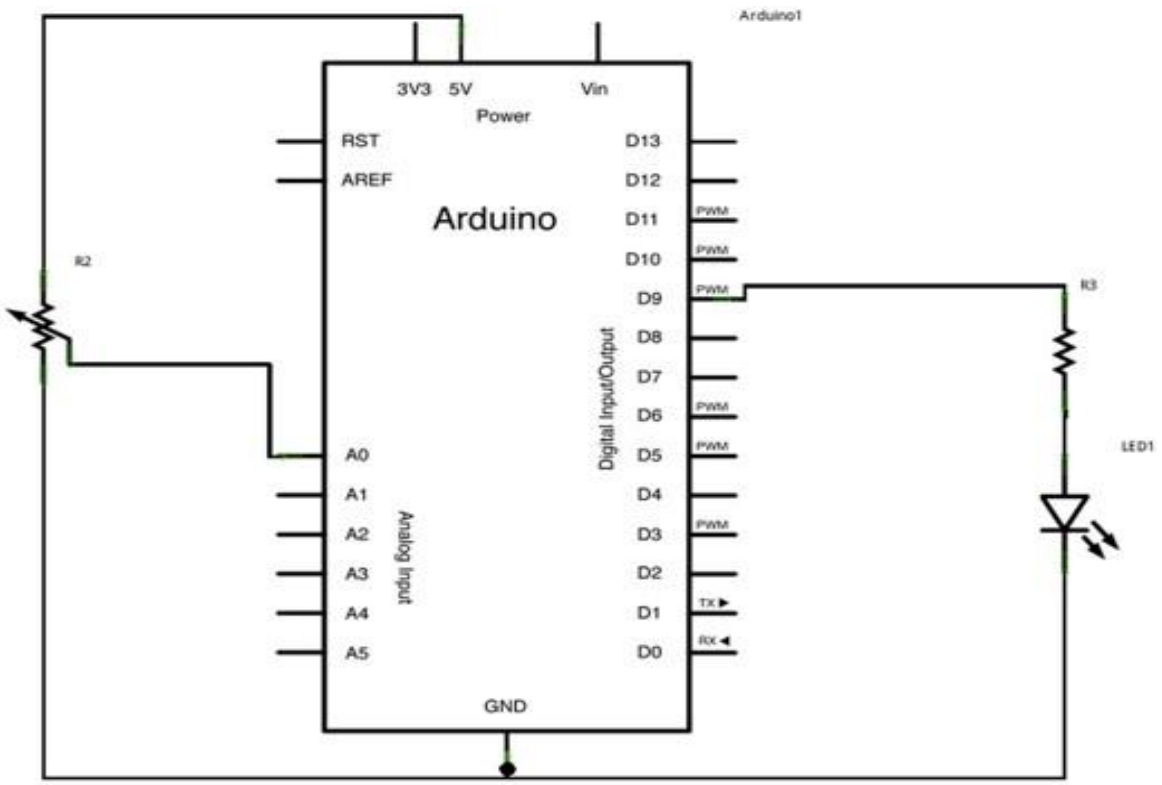
```
int val = analogRead(0);
val = map(val, 0, 1023, 0, 255);
analogWrite(9, val);
```

Challenge

Exercise: Output the brightness

1. Print out the brightness value of the varying brightness LED done in previously done exercise

Challenge 2: Analog Read/Write



Parts Needed:

- Arduino Uno
- Solderless breadboard
- 1 LED
- 1 220 resistor
- 1 Potentiometer
- Wires

- Connect the circuit shown on the left
- Code a sketch to vary the brightness of the LED using potentiometer.

Tone

Tone

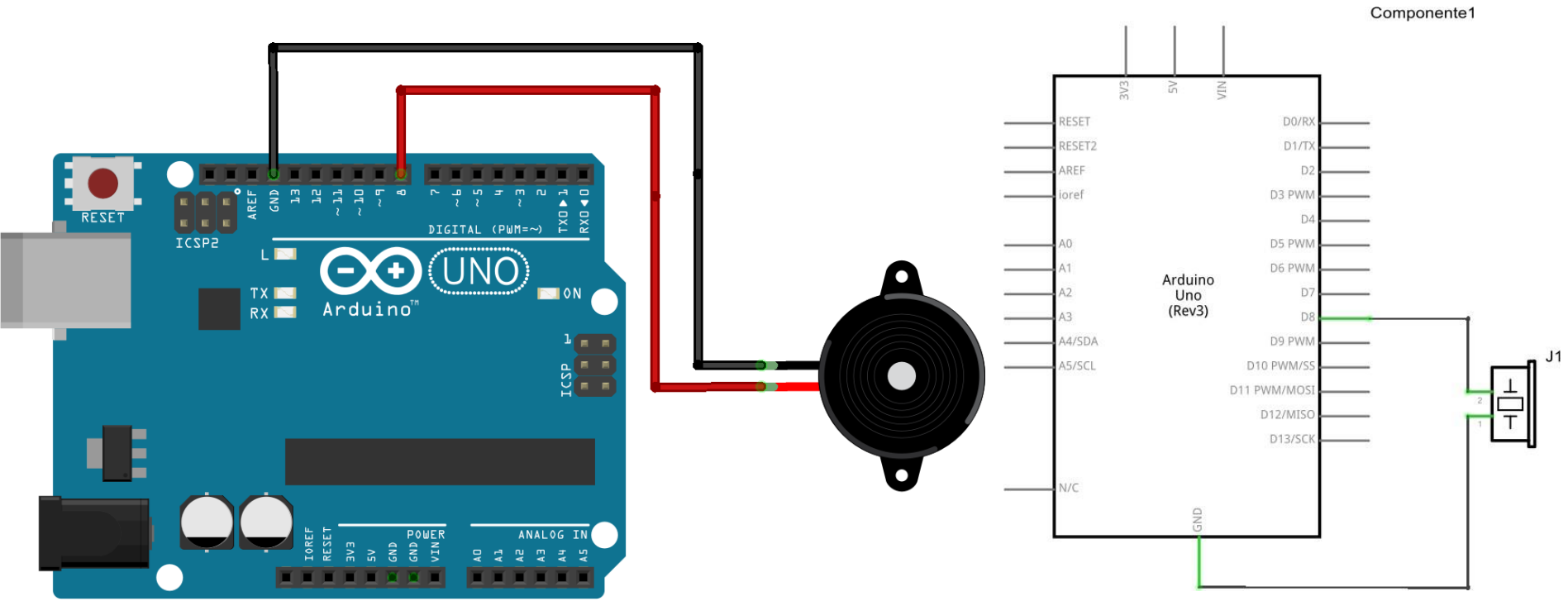
Generates a square wave of the specified frequency (and 50% duty cycle) on a pin.

Syntax:

tone(pin, frequency)
tone(pin, frequency, duration)
noTone(pin)

Ex: Tone

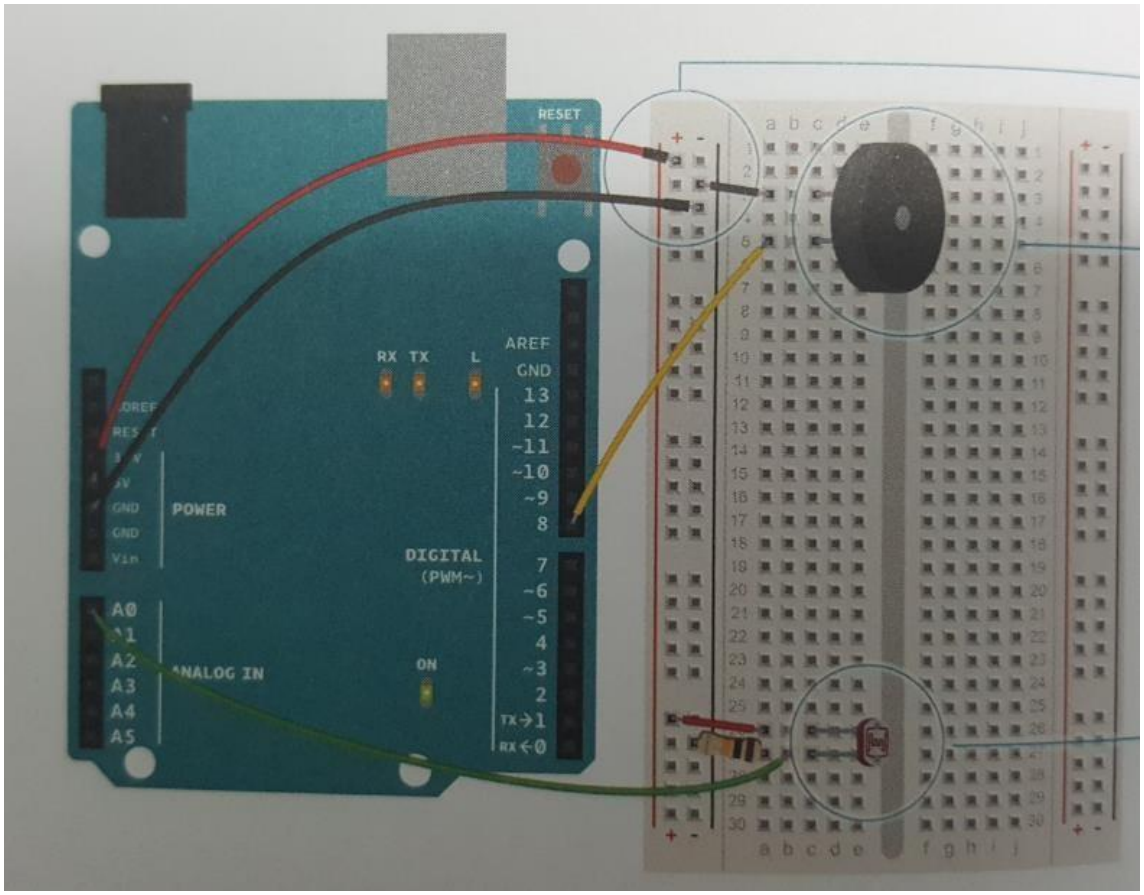
Connect like below and copy the code from [https://www.arduino.cc/en/Tutorial/ToneMelody?from=Tutorial.Tone and run the program](https://www.arduino.cc/en/Tutorial/ToneMelody?from=Tutorial.Tone)



Challenge

Add a Piezo and photoresistor sensor as below. Move your hands over the sensor, this will change the light that falls on the photoresistor’s face. This will change the voltage on the analog pin that will determine the frequency note to play

Circuit Diagram

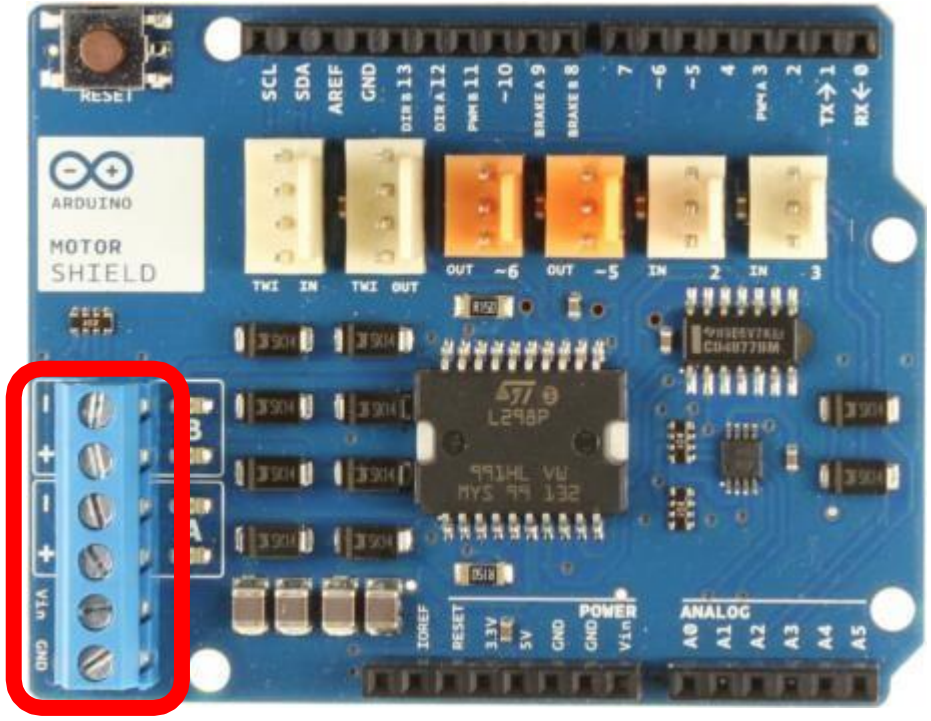


Hint

```
  while(millis() < 5000) {
    sensorValue = analogRead(A0);
    if (sensorValue > sensorHigh) {
      sensorHigh = sensorValue;
    }

    if (sensorValue < sensorLow){
      sensorLow = sensorValue;

    }
  }
  digitalWrite(ledPin,LOW);
}

void loop() {
  sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  int pitch =  map(sensorValue,sensorLow,sensorHigh, 50,4000);
  tone (8,pitch,20);
  delay(10);
}
```

Module 4
Arduino Shields

Motor Shield



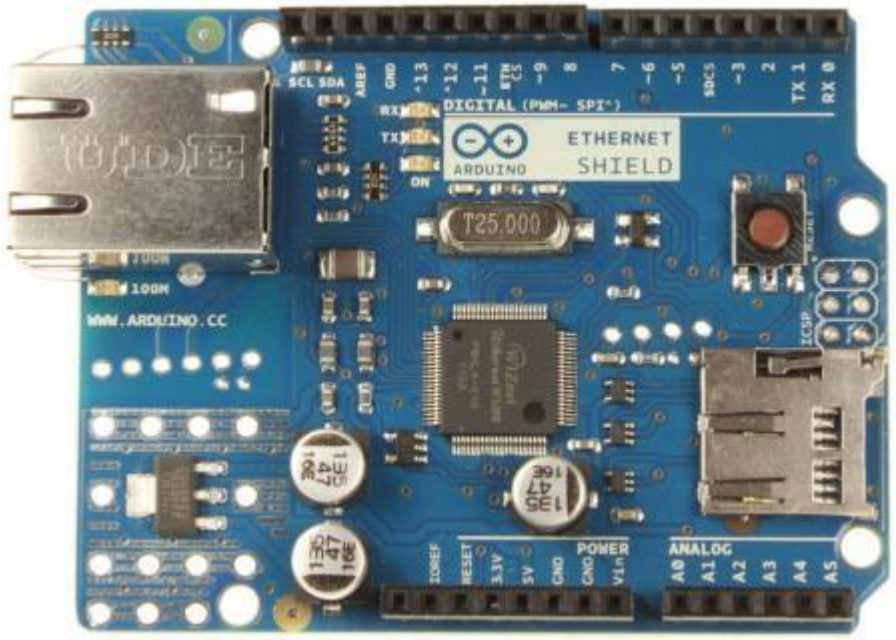
Connect to external power supply

Pin Used:
3,8,911,12,13,A0,A1

Able to drive two DC motors with Arduino UNO, controlling the speed and direction of each one independently.

Ref:
https://www.arduino.cc/en/Main/ArduinoMotorShieldR3

Ethernet Shield

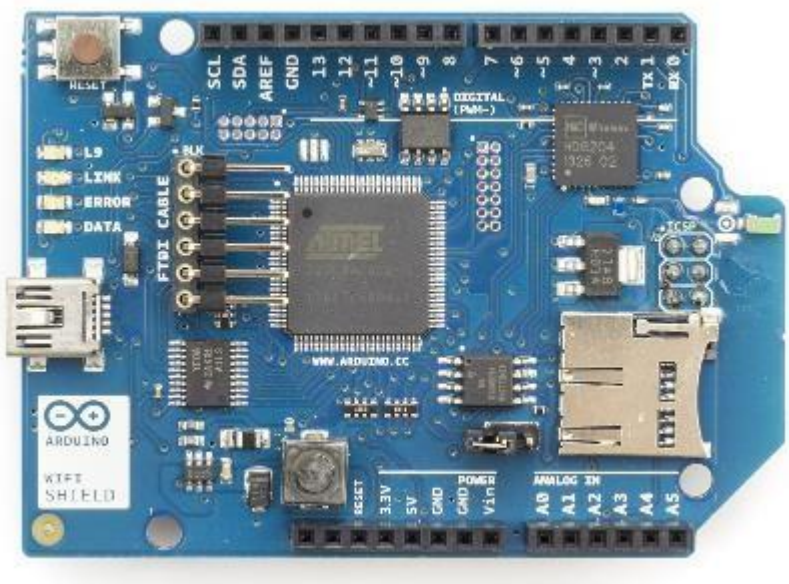


Pin Used: 4,10,11,12,13

The Arduino Ethernet Shield connects your Arduino to the internet
Eg monitor twitter hashtag or uploading sensor data.
The shield can save data to microSD card.

Ref: https://www.arduino.cc/en/Main/ArduinoEthernetShield

WiFi Shield



Pin Used: 4,7,10,11,12,13

The Arduino WiFi Shield connects your Arduino to the internet wirelessly

The shield can save data to microSD card.

Ref:
https://www.arduino.cc/en/Main/ArduinoWiFiShield

Module 5
Arduino Standard Libraries

EEPROM Library

- Arduino has built-in EEPROM which act as permanent storage like hard disk. The data will stay in the EEPROM when the board is turned off and may be retrieved later by another sketch.

```
#include <EEPROM.h>
```

Ref: https://www.arduino.cc/en/Reference/EEPROM

Ethernet Library

- This library allows an Arduino board to connect to the internet
- Allow Arduino to act as server or client.
- Arduino communicates with the shield using the SPI bus.

```
#include <SPI.h>
#include <Ethernet.h>
```

Ref: https://www.arduino.cc/en/Reference/Ethernet

Firmata Library

This library implements firmata protocol that allow you to control your Arduino from software on a computer.

#include <Firmata.h>

Ref: https://www.arduino.cc/en/Reference/Firmata

Liquid Crystal Library

This library allows an Arduino board to control LiquidCrystal displays (LCDs) based on the Hitachi HD44780 (or a compatible) chipset, which is found on most text-based LCDs.

#include <LiquidCrystal.h>

Ref: https://www.arduino.cc/en/Reference/LiquidCrystal

SD Library

The SD library allows for reading from and writing to SD or microSD cards, e.g. on the Arduino Ethernet Shield.
Use SPI for data transfer.

```
#include <SPI.h>
#include <SD.h>
```

https://www.arduino.cc/en/Reference/SD

Servo Library

The Servo library allows you to control up to 12 servo motors on UNO and 48 on Mega.
Most servo can be positioned from 0 to 180 degree.

#include <Servo.h>

https://www.arduino.cc/en/Reference/Servo

SPI Library

The Serial Peripheral Interface (SPI) is a synchronous serial data protocol used by microcontrollers for communicating with one or more peripheral devices quickly over short distances.
It can also be used for communication between two microcontrollers.

#include <SPI.h>
https://www.arduino.cc/en/Reference/SPI

SPI Communication

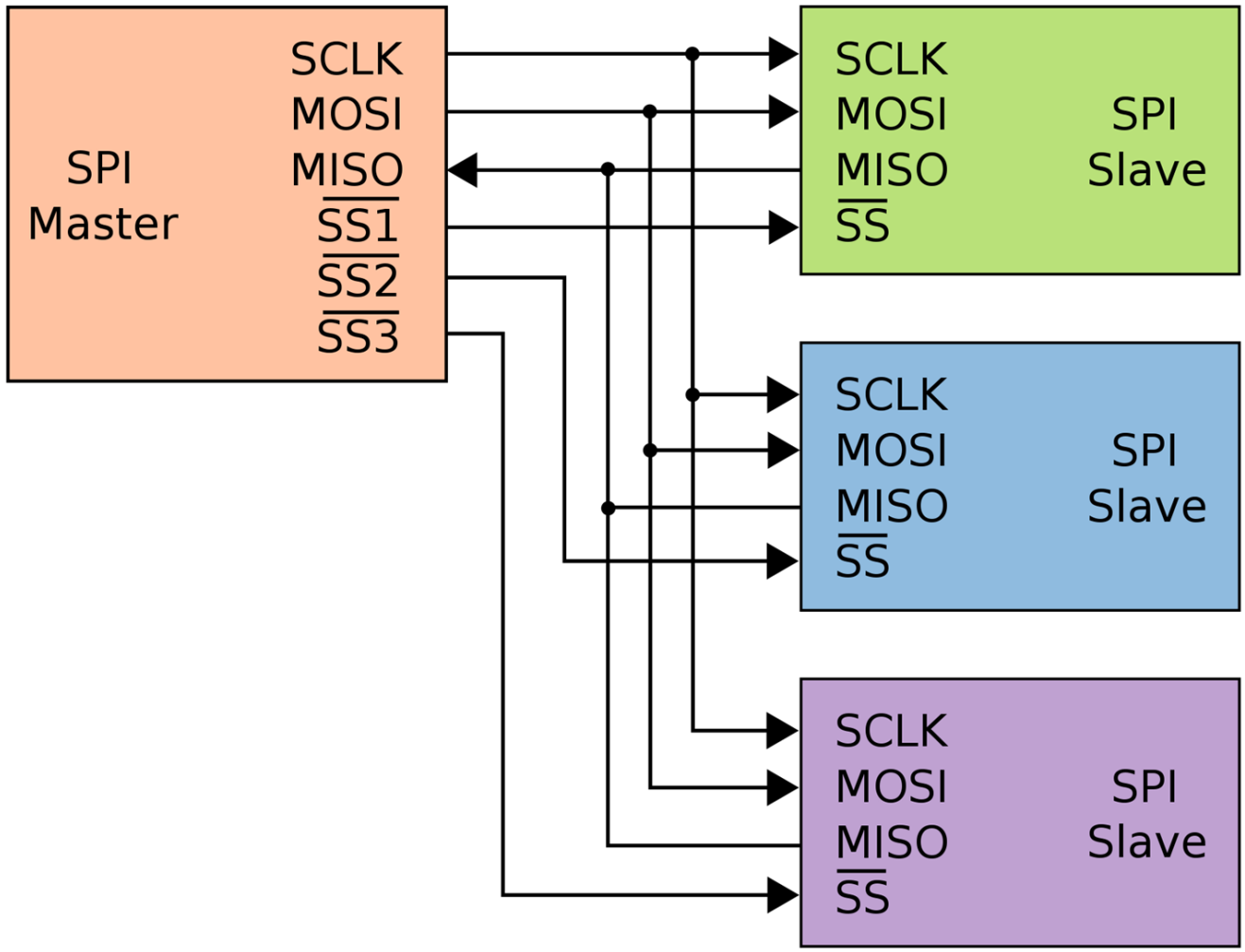
With an SPI connection there is always one master device controlling the peripheral devices. Typically there are three lines common to all the devices:

- MISO (Master In Slave Out) - The Slave line for sending data to the master,
- MOSI (Master Out Slave In) - The Master line for sending data to the peripherals,
- SCK (Serial Clock) - The clock pulses which synchronize data transmission generated by the master

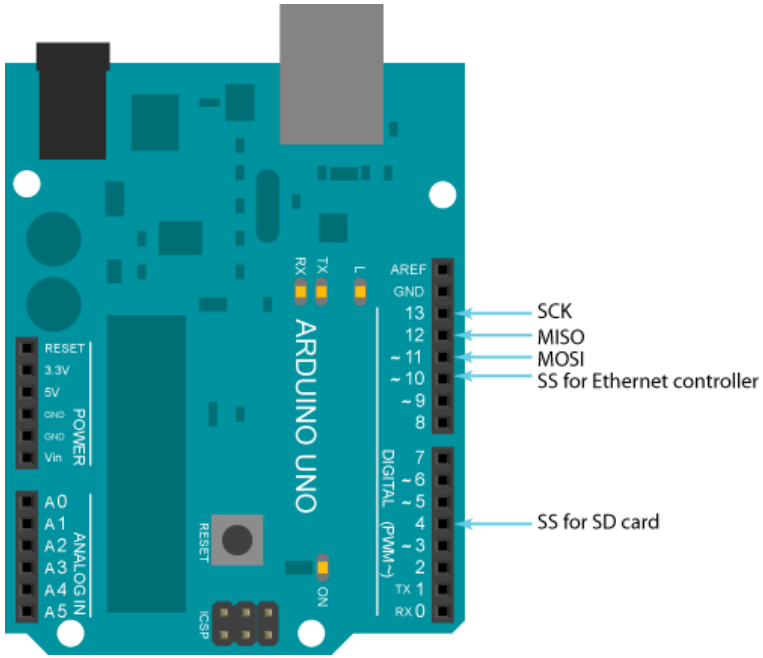
and one line specific for every device:

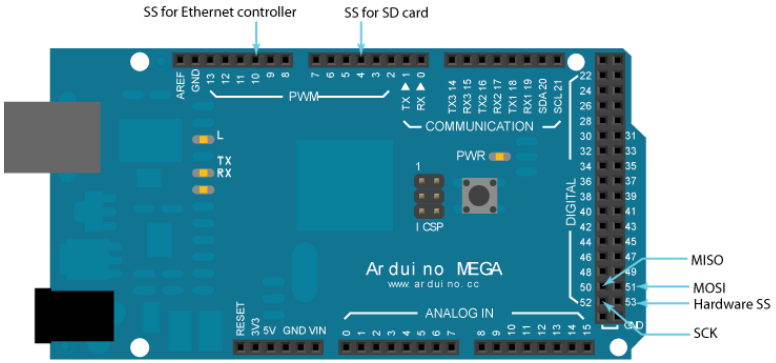
- SS (Slave Select) - the pin on each device that the master can use to enable and disable specific devices.

SPI Communication



Arduino SPI Pins





Arduino UNO/Mega SPI:
MOSI - pin 11/50
MISO - pin 12/51
CLK - pin 13/54
CS - pin 4/53

Pin 4 - SS for Ethernet Controller for UNO/Mega

SoftwareSerial Library

The Arduino hardware has built-in support for serial communication on pins 0 and 1
The softwareSerial Library allows you to use any digital pins to send and receive serial message instead of pin 0 and 1.

#include <SoftwareSerial.h>

https://www.arduino.cc/en/Reference/softwareSerial

Stepper Library

The Stepper library allows you to control unipolar or bipolar stepper motors.

#include <Stepper.h>

https://www.arduino.cc/en/Reference/Stepper

WiFi Library

The WiFi library is based on the Ethernet library. It allows you to wirelessly connect to internet.

```
#include <WiFi.h>
```

https://www.arduino.cc/en/Reference/WiFi

Wire Library

The Wire library allow your Arduino to communicate with I^2C/TWI interface.

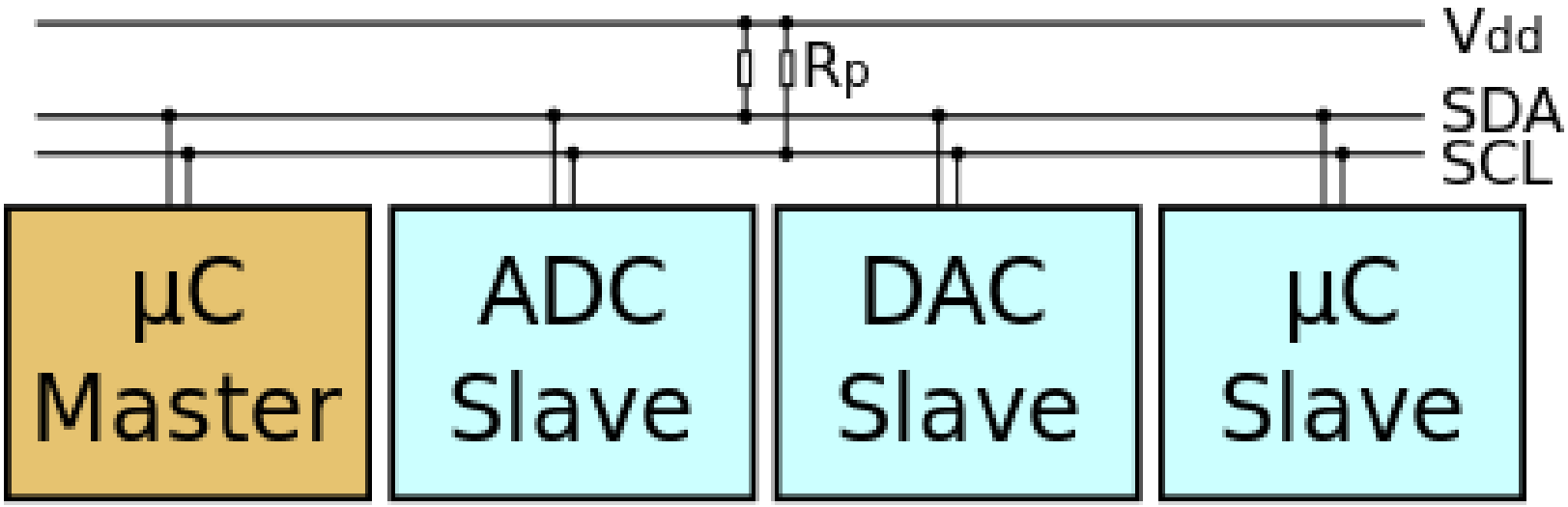
```
#include <Wire.h>
```

https://www.arduino.cc/en/Reference/Wire

I²C Communication

- Used only two bidirectional lines - Serial Data Line (SDA) and Serial Clock Line (SCL), pulled up with resistors.
- Typical voltages used are +5 V or +3.3 V
- Allowed up to 127 uniquely addressed devices
- Up to 400kHz data rates
- Data rate depends on the length of the wire, the type of cable, and the pull up resistor value

I²C Connection



- Bidirectional Bus
- Device pull down, resistor pull up
- Resistor value typically from 1.5k to 4K.
- More flexible than SPI, but slower
- The Slave address start from 01 to 7F.

Wire Library

Board	I2C / TWI pins
Uno, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	2 (SDA), 3 (SCL)
Due	20 (SDA), 21 (SCL), SDA1, SCL1

Installing Additional Library

- Libraries are usually distributed in zip files
- To install additional libraries, goto Sketch Menu -> Include Library -> Add Zip Library

https://www.arduino.cc/en/Reference/Libraries

Ex: Install Library

Download CapSense library from http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense and install on your Arduino IDE