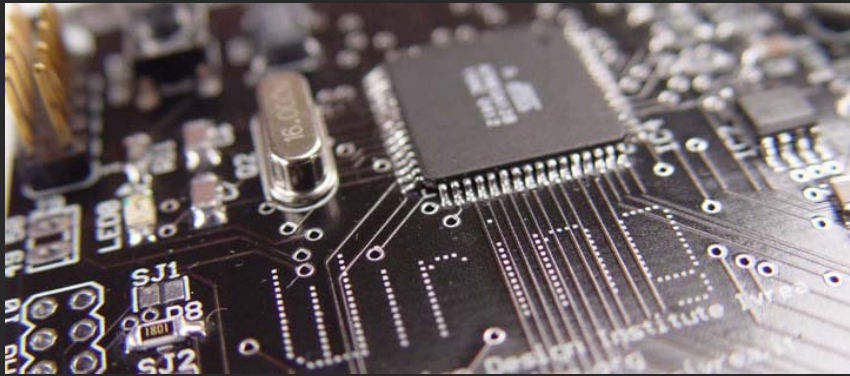


Wiring Lab

LCC 4730 / LCC 6318 / LCC 8803
Spring 2007

Wiring I/O Board



40 Digital I/O pins

8 Analog Inputs

6 Analog Outputs (PWM)

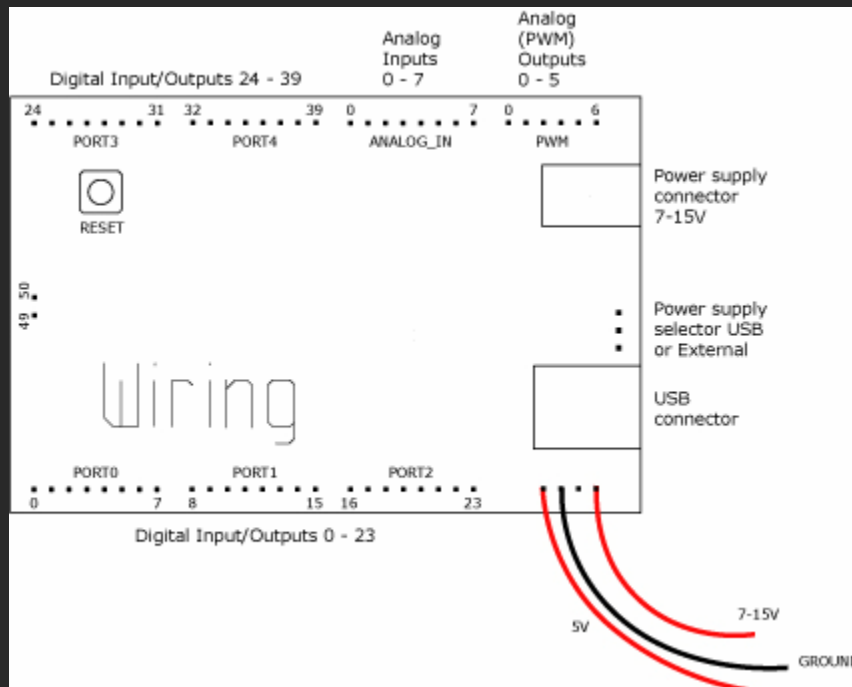
USB port

LED power indicator

Power supply: jumper selector

7-15V power adapter

USB connector



Power Adapter

VERY IMPORTANT NOTE:

SO YOU DON'T SHORT YOUR WIRING BOARD!!

Your power adapter came with different size connectors. Select the correct one for the Wiring board.

Make sure you plug the connector into the cable part of the power adapter with the correct orientation!!

You can test the orientation using a multimeter. The center **MUST** be positive (+5V).

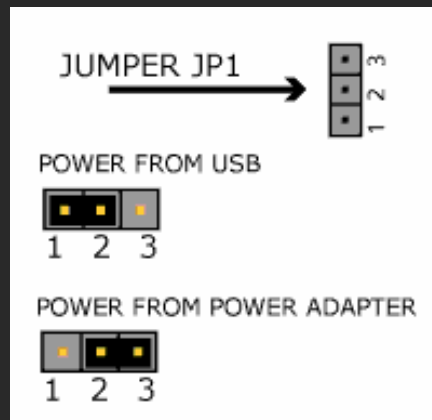
Once you have determined the correct orientation, use electrical tape to secure the power adapter connector to the cable. This will prevent you from accidentally shorting out your Wiring board later on! (there is diode protection, but just to be safe...)

Set your power adapter to 8.4V

Installation

Go to: <http://wiring.org.co/ioboard/installation.html>

1. Download and Install the [USB VCP drivers](#) for your platform (Windows, OSX, or Linux)
2. [Download](#) and install the Wiring development environment
3. Set the power jumper on the board accordingly
4. Plug the board to the USB cable and then to the PC, the green power LED on the board should come ON immediately
5. Launch the [Wiring](#) development environment



**For today: use
the USB power
setting**

Ex01 - Hello World!

Online reference:

<http://wiring.org.co/reference/>

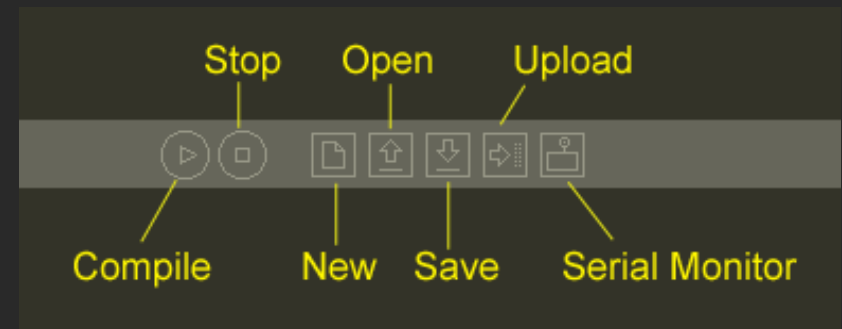
Flash the on-board LED:

- Pin 48 on board
- Digital write: HIGH (1) / LOW (0)
- Loop / delay

Code uploading:

* Make sure COM port is set

1. Compile
2. Reset (on board)
3. Upload (from Wiring)
4. Reset (on board) to run code



```
// EXERCISE 01
// -----
// Make the on-board LED blink

int ledPin = 48; // diagnostic LED on the Wiring I/O board

void setup() {
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop() {
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000);                // waits for a second
  digitalWrite(ledPin, LOW);  // sets the LED off
  delay(1000);
}
```

Ex02 – LED on/off

Goal:

Flash an LED via a digital output pin

Steps:

- 1. Build the circuit / connect to board**
- 2. Write code in Wiring environment, download to board**

Some Basics

The **current (I)** is the flow of charge and is measured in Ampere or Amps (A) *through* a point in the circuit.

In order for charge to flow it needs a force, supplied by potential difference or **voltage (V or E)**. This is measured in Volts (V) *across* two points in a circuit.

The charge flows from high potential energy to low potential energy in the circuit, i.e. from power to ground.

(Note: it is convenient to think of current flow from positive to negative, we call this *conventional current* and it corresponds to the flow of positively charged particles. The charge carriers in copper, electrons, actually flow from negative to positive as they are negatively charged particles)

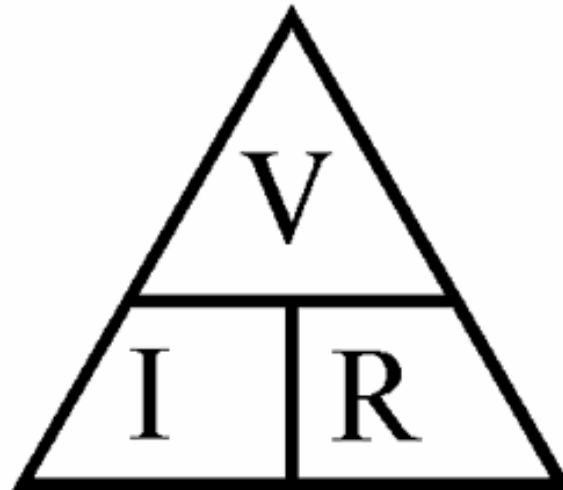
VCC = supply voltage, usually +5V

GND = ground, 0V

Resistance (R) measures how much a component opposes the passage of current in Ohms (Ω).

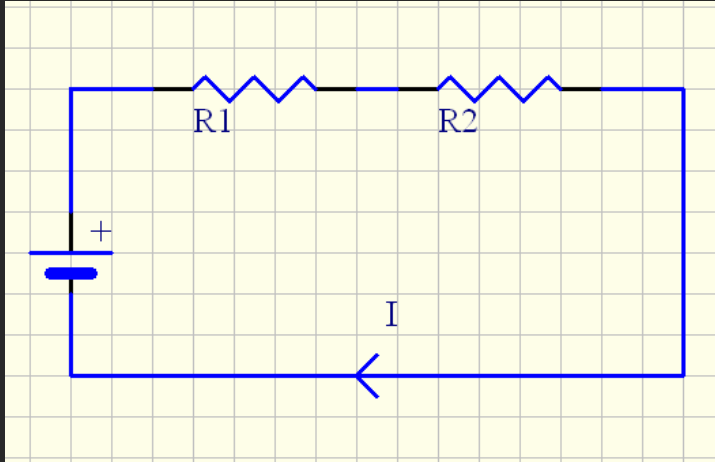
Ohm's Law

Ohm's Triangle



Cover the variable you want to find and perform the resulting calculation (*Multiplication/Division*) as indicated.

Series vs. Parallel

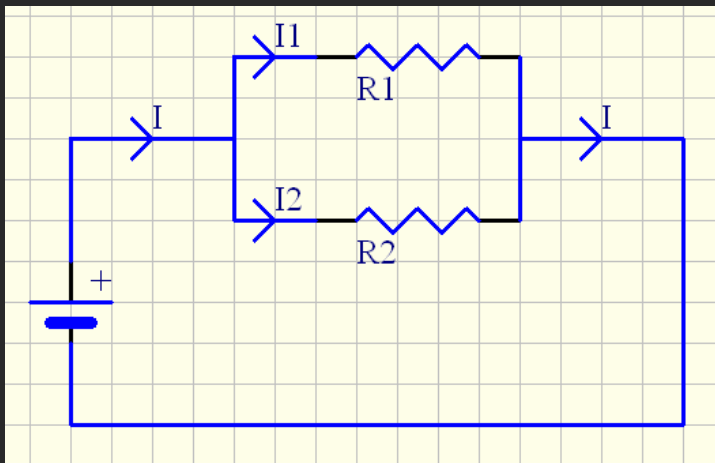


Components in series:

Current through each component is the same

Voltage drop across each component is different

$$R = R1 + R2$$



Components in parallel:

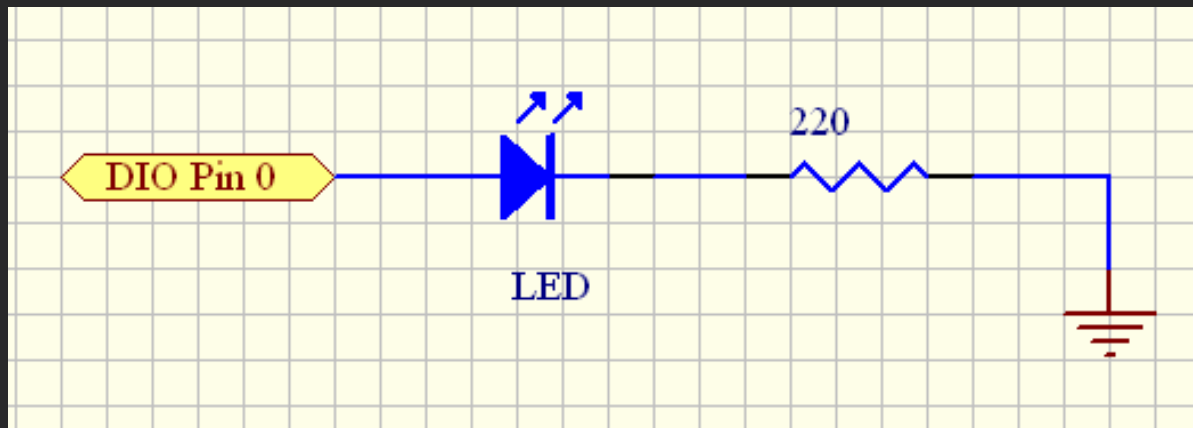
Current through each component is different

Voltage drop across each component is the same

$$R = 1 / (1/R1 + 1/R2)$$

$$R = R1R2 / (R1+R2)$$

Ex02 – Step 1 (circuit)



Typical voltage drop across LED $\sim 1.7V$

Typical current through LED $\sim 20mA$ (0.02A)

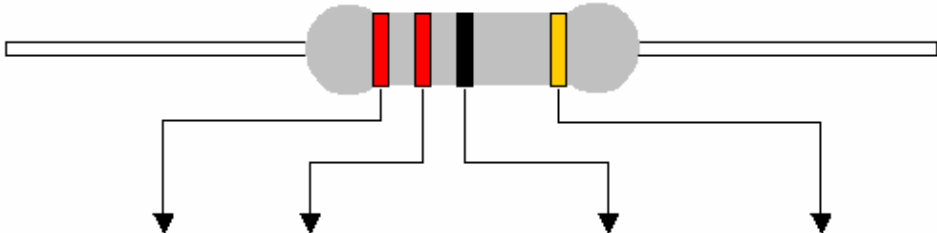
Supply voltage = 5V

Need to find a resistor that will take the extra $5 - 1.7 = 3.3V$


Ohm's Law: $R = V / I = 3.3 / 0.02 = 165 \Omega$

Choose $R = 220\Omega$ for typical LEDs

Resistor Color Coding



COLOR	1ST BAND	2ND BAND	3TH BAND	MULTIPLIER	TOLERANCE	
BLACK	0	0	0	1		
BROWN	1	1	1	10	± 1%	F
RED	2	2	2	100	± 2%	G
ORANGE	3	3	3	1K		
YELLOW	4	4	4	10K		
GREEN	5	5	5	100K	± 0.5%	D
BLUE	6	6	6	1M	± 0.25%	C
VIOLET	7	7	7	10M	± 0.10%	B
GREY	8	8	8		± 0.05%	A
WHITE	9	9	9			
GOLD				0.1	± 5%	J
SILVER				0.01	± 10%	K
PLAIN					± 20%	M



Quick Quiz:

What are the values of the following resistors?



(brown, black, orange)



(red, red, brown)

What is the color coding for a 1K resistor? And a 33K resistor?

Ex02 – Step 2 (code)

Modify your code from Ex01...

```
// EXERCISE 02
// -----
// Make an LED connected to one of the
// digital I/O pins blink

int ledPin = 0; // digital I/O pin 0 connected to LED

void setup() {
  pinMode(ledPin, OUTPUT); // sets the digital pin as output
}

void loop() {
  digitalWrite(ledPin, HIGH); // sets the LED on
  delay(1000); // waits for a second
  digitalWrite(ledPin, LOW); // sets the LED off
  delay(1000);
}
```

Ex03 – LED fading

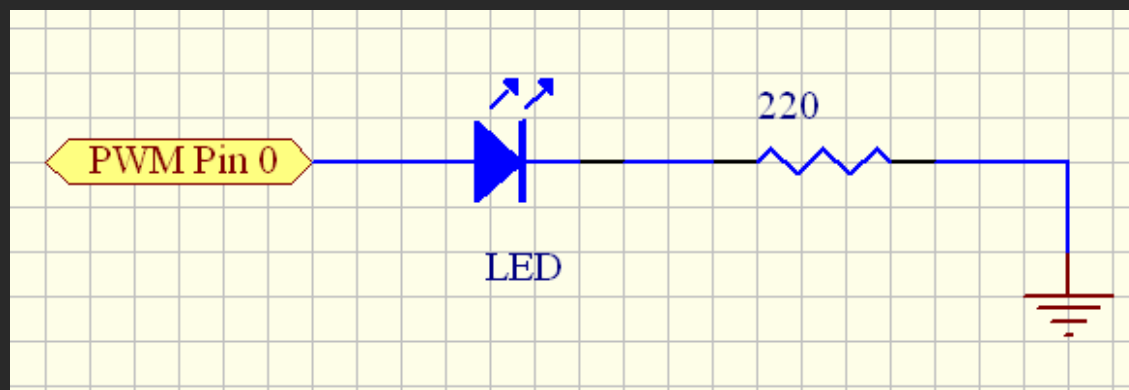
Goal:

Fade an LED via analog out (PWM)

Steps:

1. Build the circuit / connect to board
2. Write code in Wiring environment, download to board

Step 1: (circuit)



Ex03 – Step 2 (code)

```
// EXERCISE 03
// -----
// Fade an LED connected to an analog output pin

int ledPin = 0; // analog output pin 0
int value = 0; // fading value

void setup() {
  // nothing for setup
}

void loop() {
  for (value=0; value<=255; value+=5) { // fade in (from min to max)
    analogWrite(ledPin, value); // sets the value (range from 0 to 255)
    delay(30); // waits for 30 ms to see the fade-in effect
  }
  for (value=255; value>=0; value-=5) { // fade out (from max to min)
    analogWrite(ledPin, value); // sets the value (range from 0 to 255)
    delay(30); // waits for 30 ms to see the fade-out effect
  }
}
```

Ex04 – Pushbutton & LED

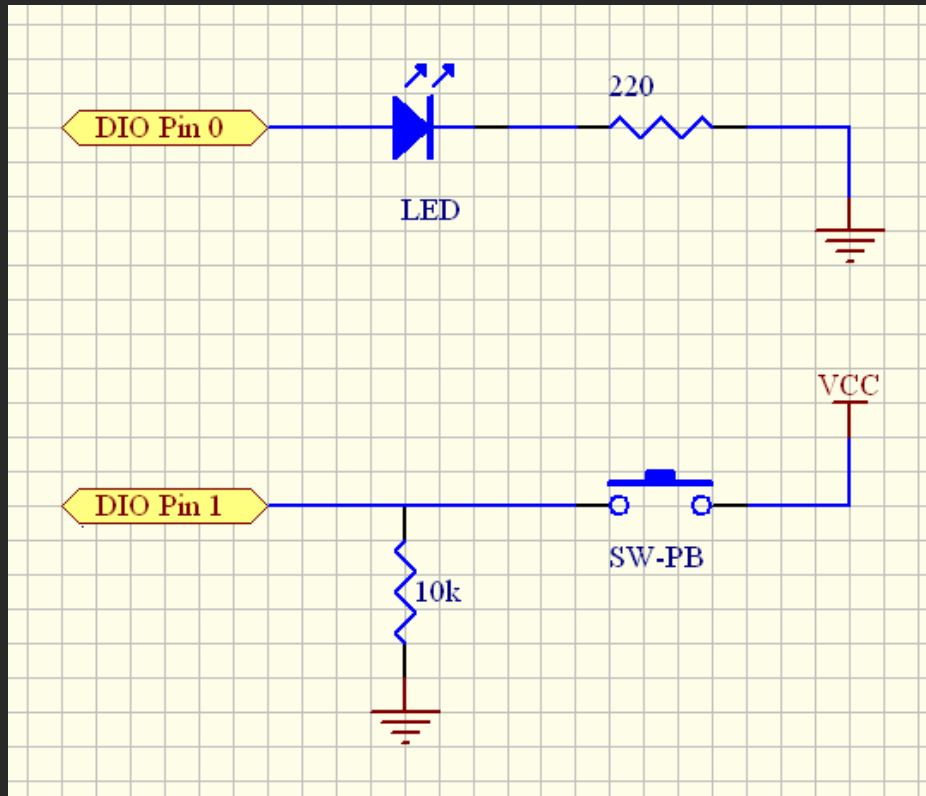
Goal:

Turn an LED on/off using a pushbutton switch

Steps:

1. Build the circuit / connect to board
 2. Write code in Wiring environment, download to board
- (Starting to look familiar?)

Ex04 – Step 1 (circuit)



Pull-down resistor:

When switch open, the pin is pulled to GND through a resistor.

Prevents the pin from floating.

Exact resistor value doesn't matter, but it needs to be big enough to limit the current flow between VCC and GND.

10k is a common value used in digital circuits.

You can also use a pull-up resistor to keep the pin HIGH instead of LOW.

Quick Quiz:

Draw a variation of this circuit that uses a pull-up resistor

(before continuing, draw a pull-up resistor version of the circuit...)

Ex04 – Step 2 (code)

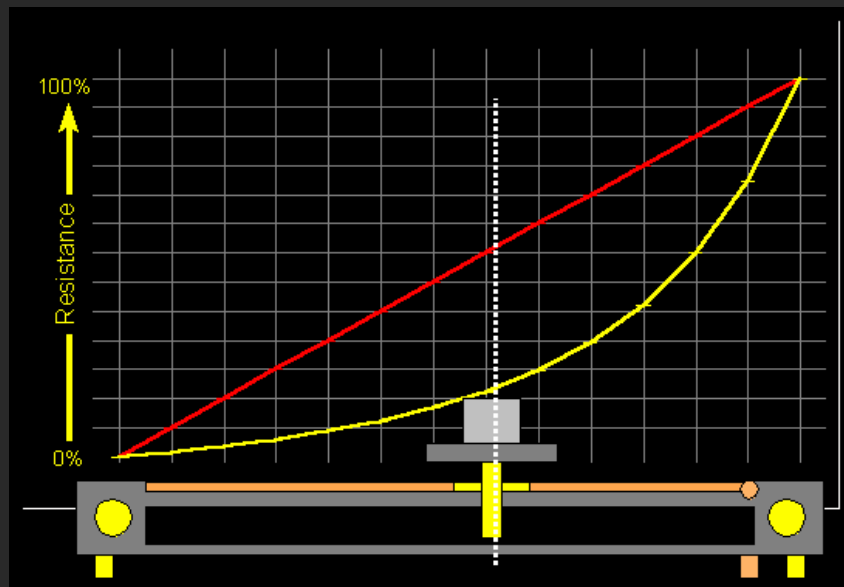
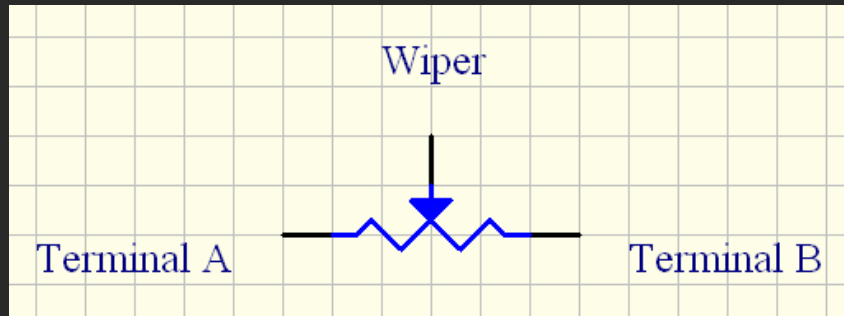
```
// EXERCISE 04
// -----
// Turn an LED on/off using a pushbutton switch

int ledPin = 0;    // digital I/O pin 0 connected to LED
int buttonPin = 1; // digital I/O pin 1 connected to button

void setup() {
  pinMode(ledPin, OUTPUT); // sets the pin mode of the LED pin to output
  pinMode(buttonPin, INPUT); // sets the pin mode of the button pin to input
}

void loop() {
  if (digitalRead(buttonPin) == LOW) {
    // if the button pin is low then turn off the LED
    digitalWrite(ledPin, LOW);
  } else {
    // otherwise if the button pin is high turn on the LED
    digitalWrite(ledPin, HIGH);
  }
}
```

Ex05 – Potentiometer Read



A potentiometer is a component with adjustable resistance:

Terminals A & B are connected to opposite ends of the resistive element

The wiper slides across the resistive element

Pots can have a linear taper (good for controlling DC voltage) or logarithmic taper (good for volume control)

Note: we have 5k pots

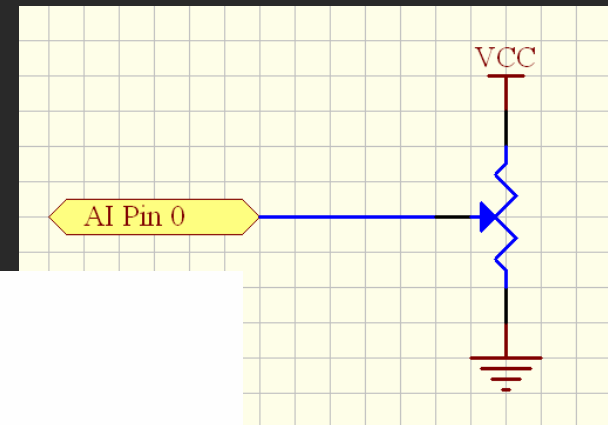
Ex05 – Circuit & Code

```
// EXERCISE 05
// -----
// Read in values from a potentiometer sensor and print
// them to the serial debugging window. Make sure to open
// the serial monitor to see the values.

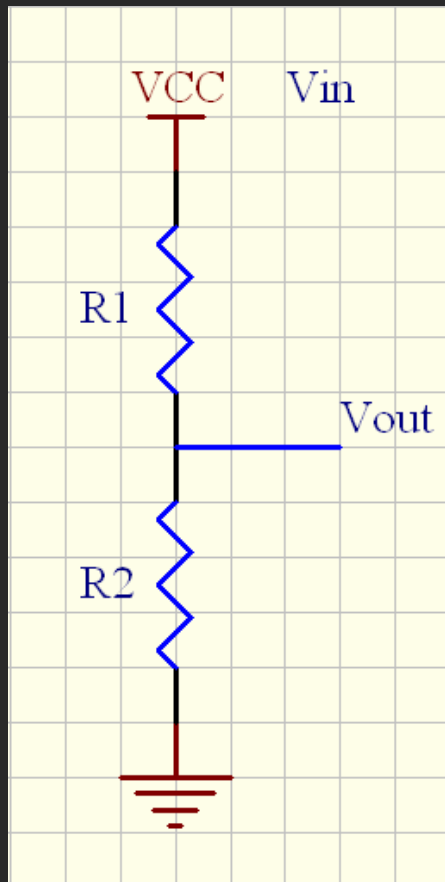
int sensorPin = 0; // analog input sensor pin
int val;          // sensor value

void setup() {
  Serial.begin(9600); // sets the serial port to 9600
  println(SERIAL); // sets the print command to use the serial port
}

void loop() {
  val = analogRead(sensorPin); // read analog input from sensorPin
  Serialprint("val: ", val); // prints the value read
  delay(100);                  // wait 100ms for next reading
}
```



Voltage Divider



(i) $V_{in} = I (R1 + R2)$

(ii) $V_{out} = I R2$

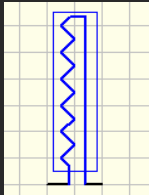
(i) $\Rightarrow I = V_{in} / (R1 + R2)$

(ii) $\Rightarrow V_{out} = V_{in} R2 / (R1 + R2)$

Suppose R1 is variable:

As R1 increases, Vout decreases

Resistive Sensors



Flex or bend sensor

Resistance increases with bend

$\sim 11\text{k} \rightarrow \sim 30\text{k}$ (straight \rightarrow bent)



Photo resistor (i.e. light sensor)

Resistance increases when covered

$\sim 2\text{k} \rightarrow \sim 60\text{k}$ (lots of light \rightarrow little light)



Force sensitive resistor (FSR, i.e. pressure sensor)

Resistance decreases with pressure

$\sim 4\text{k} \rightarrow \sim 0.5\text{k}$ (light touch \rightarrow lots of pressure)

Different sensors have different specifications

Read the datasheet or test using a multimeter

Ex06 – Resistive Sensors

Goal:

Read from a resistive sensor (bend, photo or pressure)

Steps:

1. Build the circuit / connect to board
2. Write code in Wiring environment, download to board

Ex06 – Circuits

Vout depends chosen value of R2

Photoresistor example:

R1 variable ($\sim 2k \rightarrow 60k$)

Choose R2=10k

For R1=2k, $V_{out} = 0.83 \times V_{in}$

For R1=60k, $V_{out} = 0.14 \times V_{in}$

Choose R2=1k

For R1=2k, $V_{out} = 0.33 \times V_{in}$

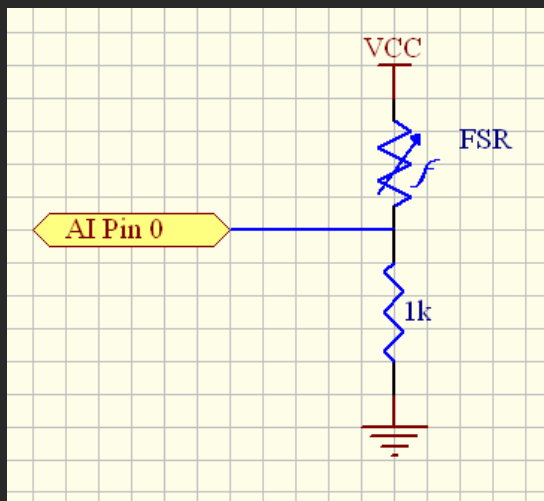
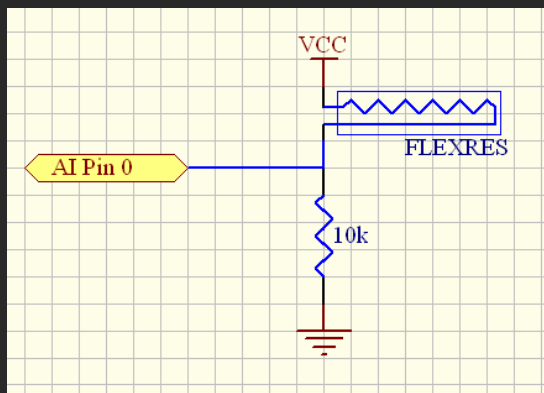
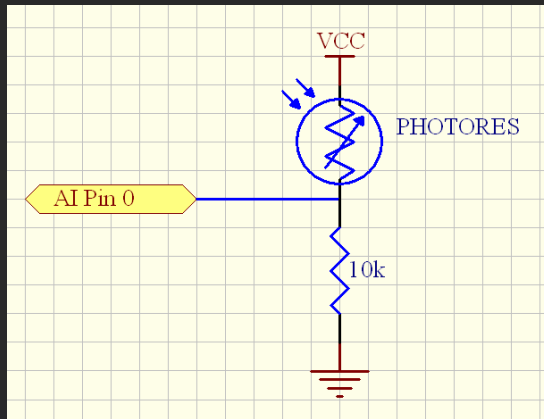
For R1=60k, $V_{out} = 0.01 \times V_{in}$

=> Vout decreases as R1 increases

=> Vout decreases as light decreases

R2=10k is a better choice

Quick Quiz: what happens to Vout with the bend and pressure sensors?



(before continuing, determine what happens with the bend and pressure sensors and choose an appropriate R2 value keeping in mind that we have 1k and 10k resistors...)

Resistor Selection

Bend sensor:

R1 variable (~11.5k -> 30k)

Choose R2=10k

For R1=11.5k, $V_{out} = 0.47 \times V_{in}$

For R1=30k, $V_{out} = 0.25 \times V_{in}$

Choose R2=1k

For R1=11.5k, $V_{out} = 0.08 \times V_{in}$

For R1=30k, $V_{out} = 0.03 \times V_{in}$

=> V_{out} decreases as R1 increases

=> V_{out} decreases as sensor bends

R2=10k is a better choice

Pressure sensor:

R1 variable (~4k -> 0.5k)

Choose R2=10k

For R1=4k, $V_{out} = 0.71 \times V_{in}$

For R1=0.5k, $V_{out} = 0.95 \times V_{in}$

Choose R2=1k

For R1=4k, $V_{out} = 0.2 \times V_{in}$

For R1=0.5k, $V_{out} = 0.67 \times V_{in}$

=> V_{out} increases as R1 decreases

=> V_{out} increases with pressure

R2=1k is a better choice

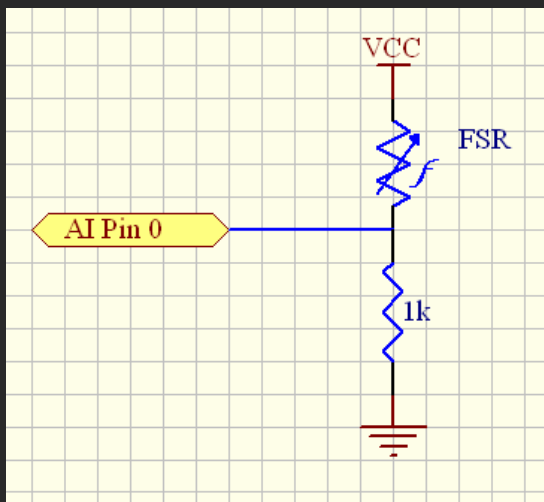
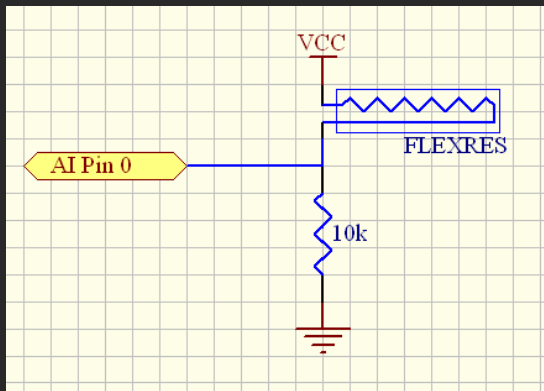
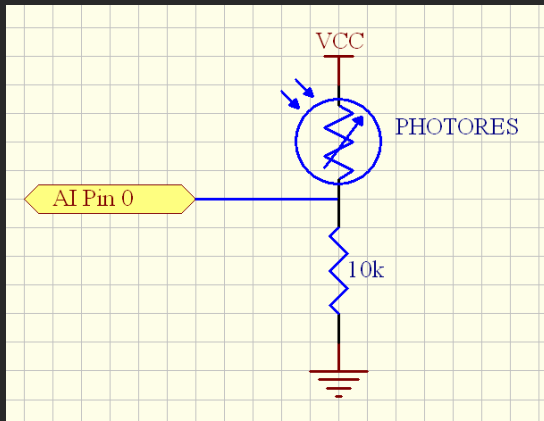
Ex06 – Code

```
// EXERCISE 06
// -----
// Read in values from a resistive sensor and print them
// to the serial debugging window. Make sure to open
// the serial monitor to see the values.

int sensorPin = 0; // analog input sensor pin
int val;          // sensor value

void setup() {
  Serial.begin(9600); // sets the serial port to 9600
  printMode(SERIAL); // sets the print command to use the serial port
}

void loop() {
  val = analogRead(sensorPin); // read analog input from sensorPin
  print("%d \n", val);         // prints the value read
  delay(100);                  // wait 100ms for next reading
}
```



Ex07 – LED fading revisited

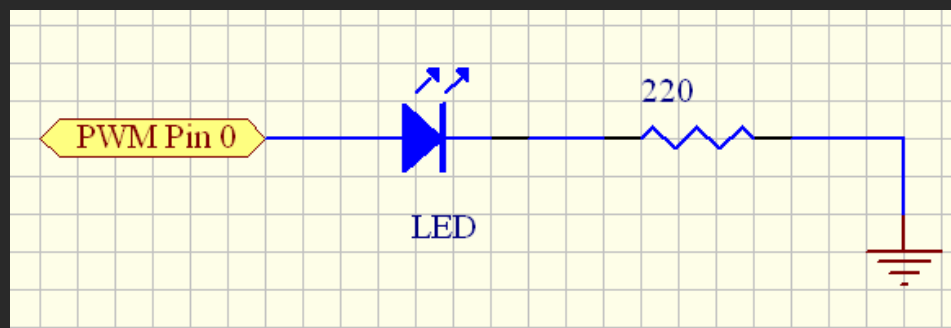
Goal:

Use one of the resistive sensors (light, bend, pressure) to fade an LED

Circuits:

Resistive sensor connected to analog input pin

LED connected to PWM pin



Scaling Functions

We need to scale and translate the input value to be within the correct output range:

Output on the PWM pin should be in the 0-255 range

Input from the sensor will be in some (MIN, MAX) interval that you can estimate using the code from Ex06 and watching the serial monitor

Suppose $x \in (\text{MIN}, \text{MAX})$ is the value read from the bend sensor, and y is the value we want to send to the PWM pin

We need to find a function $f : x \rightarrow y$ such that $y \in (0, 255)$

$$y / 255 = (x - \text{MIN}) / (\text{MAX} - \text{MIN})$$

$$\Rightarrow y = 255 (x - \text{MIN}) / (\text{MAX} - \text{MIN})$$

Quick Quiz: Implement this function in Wiring with $\text{MIN}=125$ and $\text{MAX}=500$ constants. What can you do to make sure the value is always within (0,255)?

**(before continuing, implement your
scaling function in wiring...)**

Ex07 - Code

Using your scaling function, write code that uses the sensor to fade an LED...

(before continuing, get your LED to fade using a resistive sensor and the scaling function you just wrote...)

```

// EXERCISE 07a
// -----
// Read in values from a bend sensor and use these to fade an LED

// minimum and maximum values of the bend sensor measured empirically
int MIN = 125;
int MAX = 500;

int sensorPin = 0; // analog input pin for the bend sensor
int ledPin = 0;    // digital I/O pin for the LED
int val, scaled;  // bend sensor values

void setup() {
  Serial.begin(9600); // sets the serial port to 9600
  printMode(SERIAL); // sets the print command to use the serial port
}

void loop() {
  val = analogRead(sensorPin); // read analog input from sensorPin
  print("Sensor Val: %d \t", val); // prints the value read
  scaled = scaleValue(val); // scales the value to (0,255)
  print("Scaled: %d \n", scaled); // prints the scaled value
  analogWrite(ledPin, scaled); // writes the value read to the LED pin
  delay(100); // wait 100ms for next reading
}

int scaleValue(int x) {
  // return the value of x scaled to the (0,255) interval
  return (int)constrain((255 * ((float)(x - MIN)/
                          (float)(MAX - MIN))),0,255);
}

```

Ex07a

**Bend
sensor**

```

// EXERCISE 07b
// -----
// Read in values from a photo sensor and use these to fade an LED

// minimum and maximum values of the photo sensor measured empirically
int MIN = 200;
int MAX = 850;

int sensorPin = 0; // analog input pin for the bend sensor
int ledPin = 0;    // digital I/O pin for the LED
int val, scaled;  // bend sensor values

void setup() {
  Serial.begin(9600); // sets the serial port to 9600
  printMode(SERIAL); // sets the print command to use the serial port
}

void loop() {
  val = analogRead(sensorPin); // read analog input from sensorPin
  print("Sensor Val: %d \t", val); // prints the value read
  scaled = scaleValue(val); // scales the value to (0,255)
  print("Scaled: %d \n", scaled); // prints the scaled value
  analogWrite(ledPin, scaled); // writes the value read to the LED pin
  delay(100); // wait 100ms for next reading
}

int scaleValue(int x) {
  // return the value of x scaled to the (0,255) interval
  return (int)constrain((255 * ((float)(x - MIN)/
                          (float)(MAX - MIN))),0,255);
}

```

Ex07a

Photo sensor

```

// EXERCISE 07c
// -----
// Read in values from a pressure sensor and use these to fade an LED

// minimum and maximum values of the pressure sensor measured empirically
int MIN = 10;
int MAX = 850;

int sensorPin = 0; // analog input pin for the bend sensor
int ledPin = 0;    // digital I/O pin for the LED
int val, scaled;  // bend sensor values

void setup() {
  Serial.begin(9600); // sets the serial port to 9600
  printMode(SERIAL); // sets the print command to use the serial port
}

void loop() {
  val = analogRead(sensorPin); // read analog input from sensorPin
  print("Sensor Val: %d \t", val); // prints the value read
  scaled = scaleValue(val); // scales the value to (0,255)
  print("Scaled: %d \n", scaled); // prints the scaled value
  analogWrite(ledPin, scaled); // writes the value read to the LED pin
  delay(100); // wait 100ms for next reading
}

int scaleValue(int x) {
  // return the value of x scaled to the (0,255) interval
  return (int)constrain((255 * ((float)(x - MIN)/
                          (float)(MAX - MIN))),0,255);
}

```

Ex07a

Pressure sensor

Ex08 – Serial Communication

Goal:

Read from a resistive sensor (bend, photo or pressure) and send these values to Processing via serial communication to control a graphical application

Steps:

1. Build the circuits (you've just done this!)
2. Write code in Wiring environment, download to board
3. Write code in Processing environment to read in values

Serial libraries in wiring and processing:

<http://wiring.org.co/reference/libraries/serial/>

<http://www.processing.org/reference/libraries/serial/>

Basic Concept

Wiring: the code running on the board needs to **write to** the serial port

Open serial port at desired baud rate...

```
Serial.begin(9600);
```

Set print mode to serial...

```
printMode(SERIAL);
```

In the loop, write the pin values to serial port...

```
Serial.write(scaled);
```

Processing: the code running on the PC needs to **read from** the serial port

Open the wiring board serial port with the correct baud rate...

```
sPort = new Serial(this, "COM2", 9600);
```

In the loop, read sensor values from the serial port...

```
val = sPort.read();
```

```

// EXERCISE 08a
// -----
// Send the bend sensor values to the serial port
// They can then be read in a Processing application to control
// graphical elements on screen

// minimum and maximum values of the bend sensor measured empirically
int MIN = 125;
int MAX = 500;

int sensorPin = 0; // analog input pin for the bend sensor
int val, scaled;  // bend sensor values

void setup() {
  Serial.begin(9600); // sets the serial port to 9600
  printMode(SERIAL); // sets the print command to use the serial port
}

void loop() {
  val = analogRead(sensorPin); // read analog input from sensorPin
  scaled = scaleValue(val);    // scales the value to (0,255)
  Serial.write(scaled);       // prints the scaled value
  delay(100);                 // wait 100ms for next reading
}

int scaleValue(int x) {
  // return the value of x scaled to the (0,255) interval
  return (int)constrain((255 * ((float)(x - MIN)/
                          (float)(MAX - MIN))),0,255);
}

```

Ex08a

**Wiring
code**

```
// EXERCISE 08b
// -----
// Read in values from a bend sensor via the serial port
// that is connected to the wiring board and use these
// values to control the background color of the screen

import processing.serial.*;

Serial sPort;
int val;

void setup() {
  println(Serial.list());           // check what serial ports we have
  sPort = new Serial(this, "COM2", 9600); // open the wiring board serial port
}

void draw() {
  while (sPort.available() > 0) {
    // while the serial port is still available read a byte
    // of data and use set the background color of the screen
    val = sPort.read();
    println(val);
    background(val);
  }
}
```

Ex08b Processing code

Actuators: Making Motion



Rotary motion: Motors

A coil is mounted on a spinning shaft between two magnets. Current is applied to the coil to create a magnetic field that is attracted to one of the magnets. The current flow is reversed at the half-cycle and the coil becomes attracted to other magnet causing it to spin.



Linear motion: Solenoids

Electromagnetic switch consisting of a coil wound around a moveable iron shaft. Current is applied to the coil to create a magnetic field that attracts the shaft. When the current is turned off, a spring causes the shaft to move out again.

In the next exercise we'll use a servo motor...

Motor Basics

Two main types to be concerned with:

1. DC motors:

Spin continually when given enough current. Reversing the current makes the motor spin the other way.

If you increase the voltage, the motor spins faster.

You can add gears to lower the speed and give the motor more torque (pulling force).

2. Stepper motors:

Contain a number of coils at fixed degrees around the shaft and a series of magnets on the shaft.

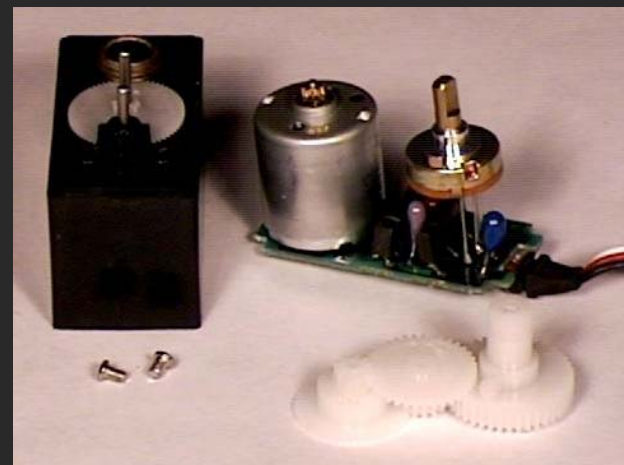
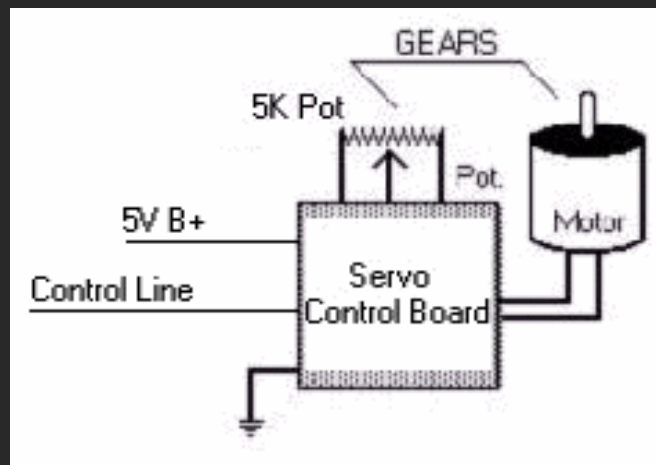
This allows the motor to rotate in discrete steps, so you can move it by a precise distance.

Servo Motors

Servo motors are DC motors with gear reduction, a position sensor and control electronics.

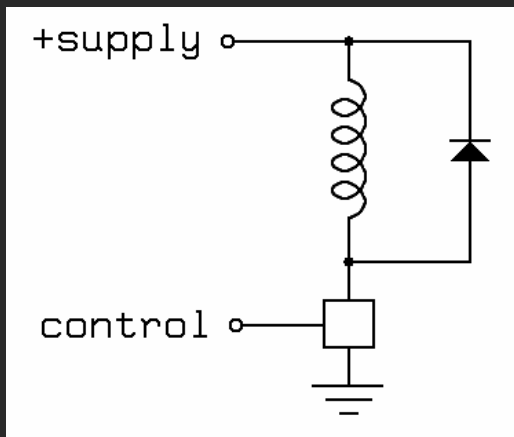
Allow precise positioning within a certain range of movement (typically 0-180 degrees).

Gives feedback about its position (typically using a potentiometer).



Inductive vs. Resistive Loads

Inductive loads:



Devices that induce a magnetic field by putting current through a wire, and then use the magnetic field to attract or repulse a magnetic body.

E.g. motors, solenoids

Produce back voltage, a small burst of current in the reverse direction when the magnetic field is turned off. Need diode protection in parallel with the component.

Resistive loads:

Devices that resist current and convert it to some other form of energy.

E.g. Lights, heaters

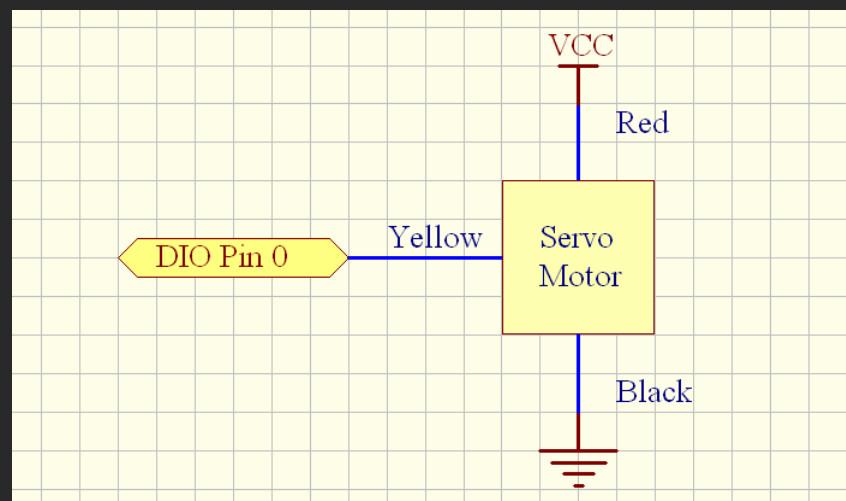
Ex09 – Servo Motors

Goal:

Make a servo motor step back and forth across its 180 range in increments of 1 degree

Steps:

1. Build the circuits, use external power supply
2. Write code in Wiring environment, download to board



Wiring Servo Library

<http://wiring.org.co/reference/libraries/servo/>

Creating a servo object (max 8 supported)

```
Servo serv;
```

Attaching and detaching the servo

```
serv.attach(0); // attaches servo to digital pin 0
```

```
serv.detach(); // detaches servo
```

```
boolean b = serv.attached(); // checks if servo is attached
```

Reading and writing servo positions

```
int deg = serv.read(); // reads servo position
```

```
serv.write(90); // sets servo position to 90 degrees
```

(before continuing, write code to drive your servo forwards and backwards in steps of 1 degree...)

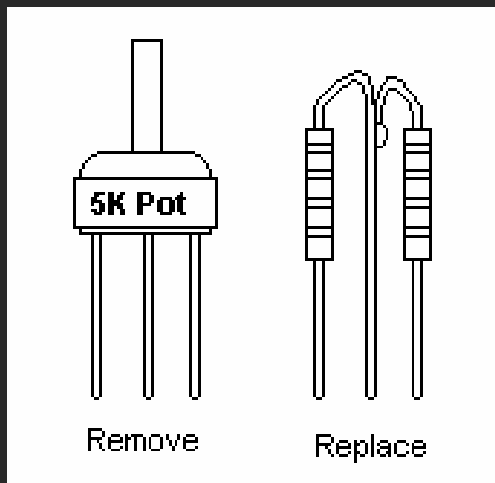
```
// EXERCISE 09
// -----
// Servo motor control

Servo serv;      // create servo object (max eight allowed)
int servoPin = 0; // servo on pin digital 0
int ang = 0;     // servo position

void setup() {
  serv.attach(servoPin); // attach the servo on digital pin 0
}

void loop() {
  // go from 0 to 180 degrees in steps of 1 degree
  for(ang = 0; ang < 180; ang += 1) {
    serv.write(ang); // set servo position
    delay(15);      // wait 15ms for servo to reach position
  }
  // go from 180 to 0 degrees in steps of 1 degree
  for(ang = 180; ang >= 1; ang -= 1) {
    serv.write(ang); // set servo position
    delay(15);      // wait 15ms for servo to reach position
  }
}
```

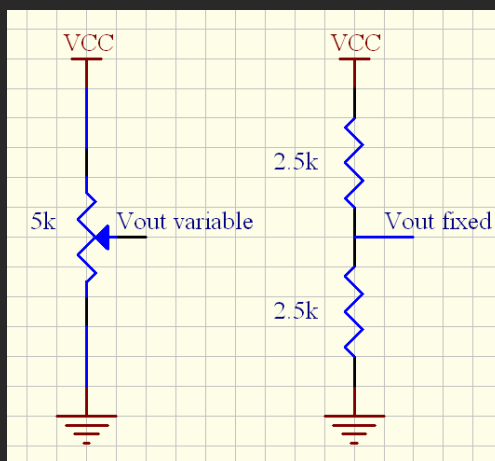
Servo Modification



You can trick a servo into spinning continuously with two small changes...

Replace the 5k potentiometer with a couple of 2.5k fixed resistors (somewhere $\sim 2.2k$ - $2.7k$ will work fine)

Trim off the small tab on the output gear (normally this tab prevents the motor from turning through a full circle)



Now we have a fixed instead variable output where the position sensor was, so the control electronics will think the servo is always centered at 90 degrees...

Send value $> \sim 90$ to spin forwards

Send value $< \sim 90$ to spin backwards

Spinning gets faster as $\text{abs}(\text{value}-90)$ increases

Demonstrations...

```

// EXERCISE 09a
// -----
// Changing the direction of spin of a modified servo

Servo serv;          // create servo object (max eight allowed)

int servoPin = 0;    // servo on digital pin 0
int buttonPin = 1;   // button on digital pin 1
int ledPin = 2;      // led on digital pin 2

int fwd = 180;       // fast forward
int rev = 0;         // fast reverse

void setup() {
  pinMode(buttonPin, INPUT); // set button pin to input
  pinMode(ledPin, OUTPUT);   // set the led pin to output
  serv.attach(servoPin);     // attach the servo on pin 0
}

void loop() {
  if (digitalRead(buttonPin) == LOW) {
    // reverse direction
    digitalWrite(ledPin, LOW);
    serv.write(rev);
  } else {
    // forward direction
    digitalWrite(ledPin, HIGH);
    serv.write(fwd);
  }
  delay(15);
}

```

Ex09a

Wiring code

```

// EXERCISE 09b
// -----
// Changing the spinning speed of a modified servo

int MIN = 125; // bend sensor max and min values
int MAX = 500;

int FWD_SLOW = 84;
int FWD_FAST = 180;

Servo serv;      // create servo object (max eight allowed)
int servoPin = 0; // servo on digital pin 0
int sensorPin = 0; // sensor on analog pin 0

int val, scaled;

void setup() {
  Serial.begin(9600); // sets the serial port to 9600
  printMode(SERIAL); // sets the print command to use the serial port
  serv.attach(servoPin); // attach the servo on pin 0
}

void loop() {
  val = analogRead(sensorPin);
  print("Sensor Val: %d \t", val); // prints the value read
  scaled = scaleValue(val,FWD_SLOW,FWD_FAST); // scales the value to (0,255)
  print("Scaled: %d \n", scaled); // prints the scaled value
  serv.write(scaled);
  delay(10);
}

int scaleValue(int x, int lo, int hi) {
  // return the value of x scaled to the (lo,hi) interval
  return (int)constrain(lo + (hi-lo)*((float)(x -MIN)/
                        (float)(MAX-MIN)),lo,hi);
}

```

Ex09b

**Wiring
code**

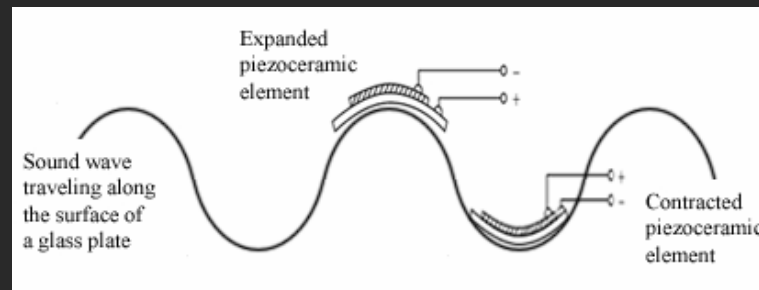
Piezo Buzzers

The piezoelectric effect is the ability of certain crystals to generate electrical energy in response to applied mechanical energy and vice versa.

Piezo buzzers consist of a piezoceramic disc adhered to a thin metal plate.

They generate sound by vibrating when an electrical signal is applied.

They generate a voltage in response to mechanical stress.



Note: output from a microcontroller is not strong enough to generate much sound, an amplifier would be needed...

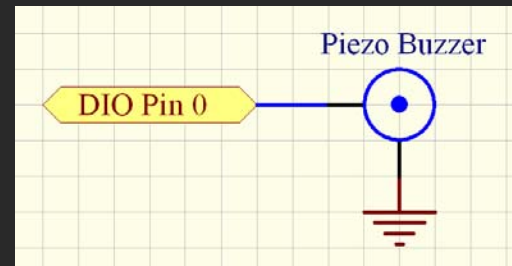
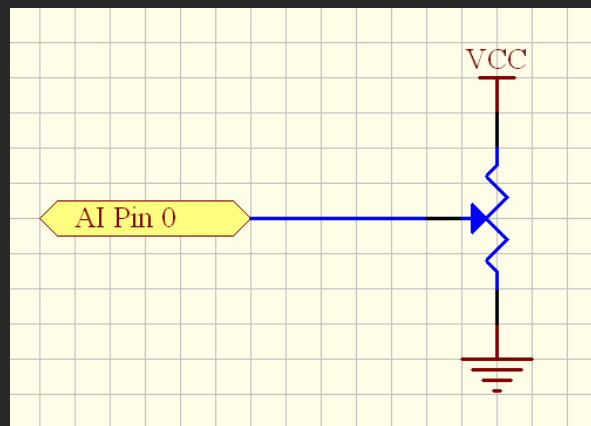
Ex10 – Piezo Buzzers

Goal:

Drive a piezo buzzer with a variable frequency that can be adjusted using a potentiometer.

Steps:

1. Build the circuits
2. Write code in Wiring environment, download to board



**(before continuing, write and test
your wiring code...)**

```

// EXERCISE 10
// -----
// Driving a piezo buzzer

int MIN = 0; int MAX = 1023;      // min and max pot values
int MIN_DEL = 1; int MAX_DEL = 20; // min and max delays

int piezoPin = 0; // digital pin 0 for piezo buzzer
int potPin = 0;   // analog input pin 0

int val, del;

void setup() {
  pinMode(piezoPin, OUTPUT); // sets the buzzer pin as output
}

void loop() {
  val = analogRead(potPin);           // read the pot value
  del = scaleValue(val,MIN_DEL,MAX_DEL); // scale to delay interval
  digitalWrite(piezoPin,HIGH);       // high pulse to buzzer
  delay(del);                         // delay
  digitalWrite(piezoPin,LOW);        // low pulse to buzzer
  delay(del);                         // delay
}

int scaleValue(int x, int lo, int hi) {
  // return the value of x scaled to the (lo,hi) interval
  return (int)constrain(lo + (hi-lo)*((float)(x -MIN)/
                        (float)(MAX-MIN)),lo,hi);
}

```

Ex10

Wiring code