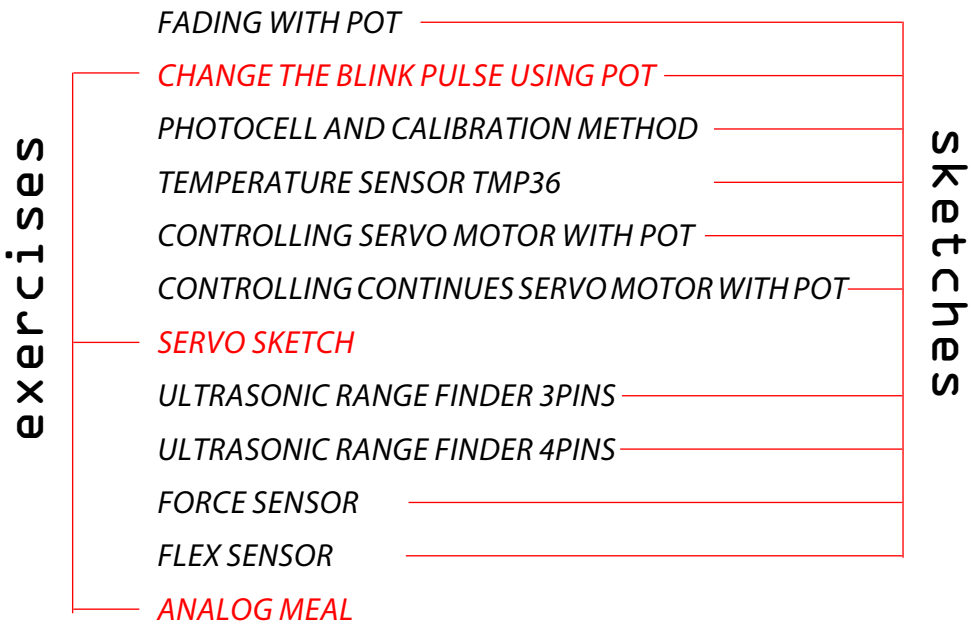
ANALOG

CIRCUITS

physical
computing
meals

02

MAIN PLATE


exercises
FADING WITH POT
CHANGE THE BLINK PULSE USING POT
PHOTOCELL AND CALIBRATION METHOD
TEMPERATURE SENSOR TMP36
CONTROLLING SERVO MOTOR WITH POT
CONTROLLING CONTINUES SERVO MOTOR WITH POT
SERVO SKETCH
ULTRASONIC RANGE FINDER 3PINS
ULTRASONIC RANGE FINDER 4PINS
FORCE SENSOR
FLEX SENSOR
ANALOG MEAL
sketches

Fading With Pot

http://arduino.cc/en/Tutorial/AnalogInput

A potentiometer is a simple knob that provides a variable resistance, which you can read into the Arduino board as an analog value. In this example, you'll connect a poterntiometer to one of the Arduino's analog inputs to control the rate at which the built-in LED on pin 13 blinks.

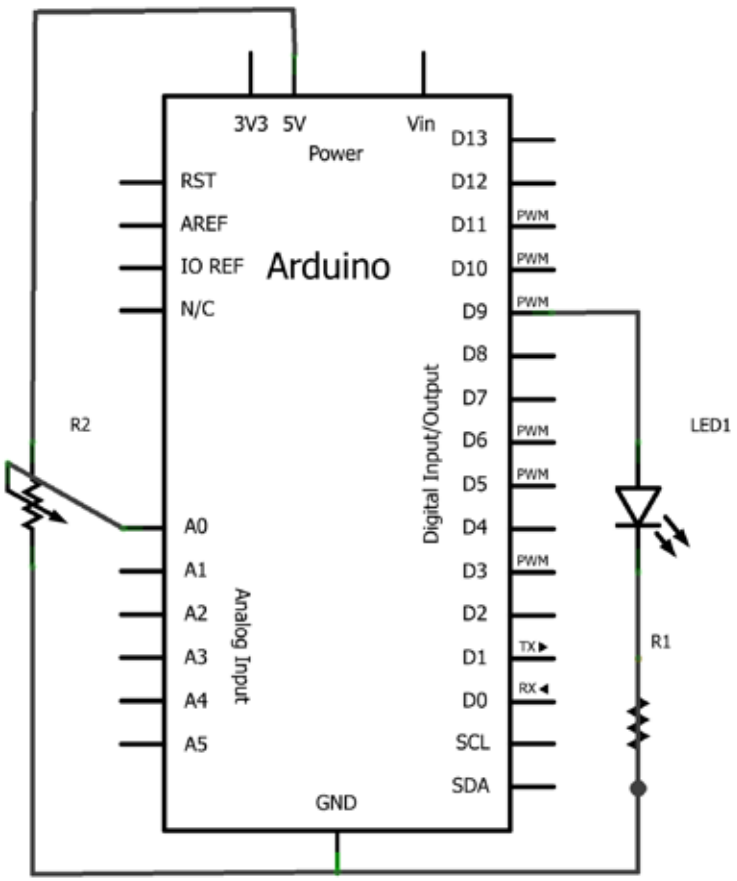
HARDWARE REQUIRED

Arduino Board
1 10K Potentiometer
1 LED
1 220Ω Resistor

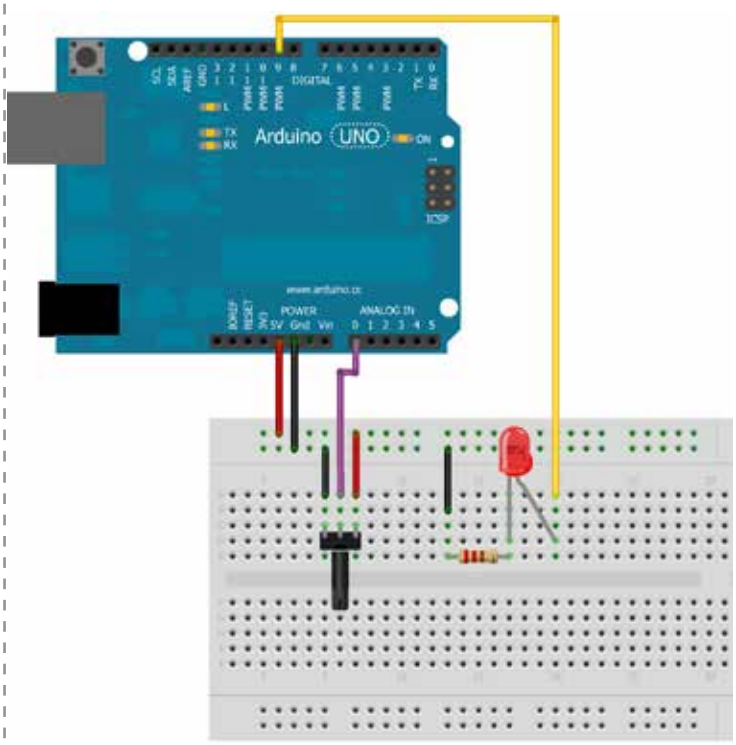
CIRCUIT

Connect three wires to the Arduino board. The first goes to ground from one of the outer pins of the potentiometer. The second goes from 5 volts to the other outer pin of the potentiometer. The third goes from analog input 0 to the middle pin of the potentiometer. For this example, it is possible to use the Arduino board's built in LED attached to pin 13. To use an additional LED, attach its longer leg (the positive leg, or anode), to digital pin 13, and it's shorter leg (the negative leg, or cathode) to the ground (gnd) pin next to pin 13. Because of the low amount of current coming from digital pin 13, it is not necessary to use a current limiting resistor in this particular case.

SCHEMATIC



IMAGE



CODE

```
/*
 Fading

 This example shows how to fade an LED using the analogWrite() function.
 The circuit:
 * LED attached from digital pin 9 to ground.

 Created 1 Nov 2008
 By David A. Mellis
 modified 30 Aug 2011
 By Tom Igoe
 http://arduino.cc/en/Tutorial/Fading

 This example code is in the public domain.
 */


int ledPin = 9;    // LED connected to digital pwm pin

void setup()  {
   pinMode(ledPin, OUTPUT);
}

void loop()  {

   // read the Value
   int sensorValue = analogRead(A0);

   // transform the value to 0-255 scale
   sensorValue = map(sensorValue, 0, 1023, 0, 255);

   // apply the value to the LED
   analogWrite(ledPin, sensorValue);

   // add some delay
   delay(10);
  }
```

Change the Blink Pulse Using Pot

EXERCISE

Use a potentiometer to control the blinking pulse of an LED

HARDWARE REQUIRED

Arduino Board
10K Potentiometer
One (1) LED
One 220Ω Resistor
Breadboard

SCHEMATIC

Photocell and Calibration Method

http://arduino.cc/en/Tutorial/Calibration
http://learn.adafruit.com/photocells

This example demonstrates one techinque for calibrating sensor input. The Arduino takes sensor readings for five seconds during the startup, and tracks the highest and lowest values it gets. These sensor readings during the first five seconds of the sketch execution define the minimum and maximum of expected values for the readings taken during the loop.

HARDWARE REQUIRED

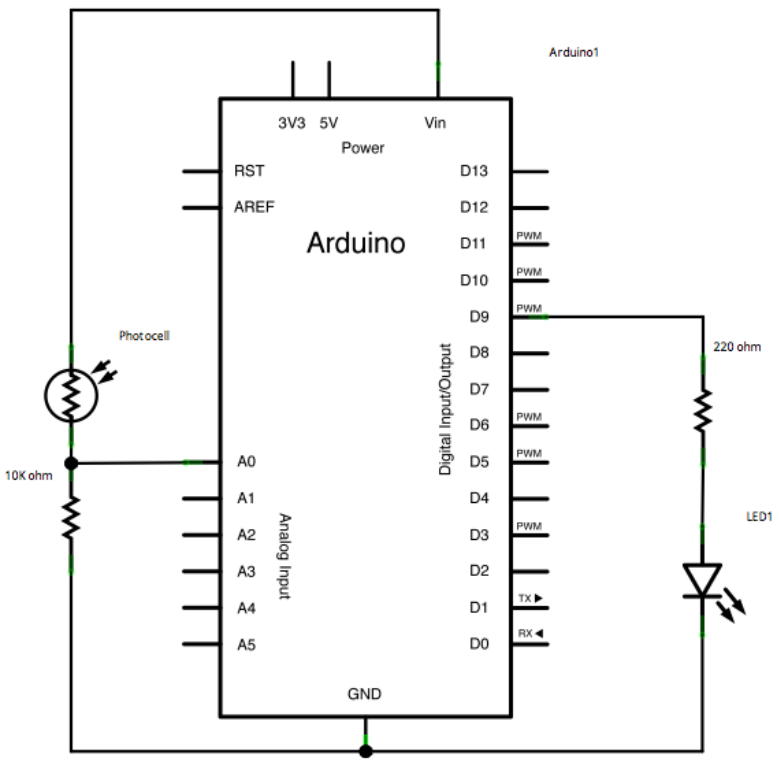
Arduino board
1 LED
1 Photocell (LDR) or any analog sensor
1 10K ohm resistor
1 220 ohm resistor
breadboard

CIRCUIT

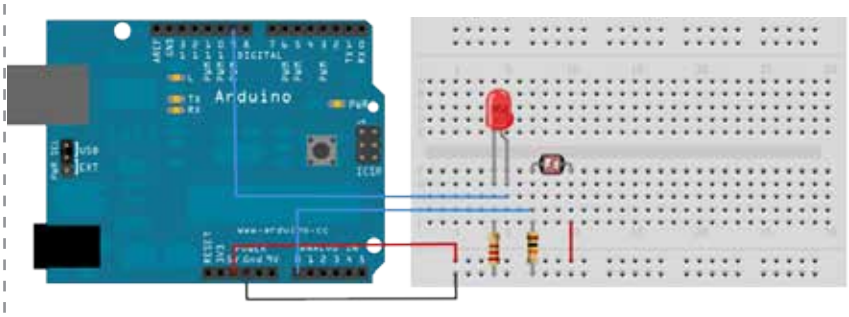
Analog sensor (e.g. potentiometer, light sensor) on analog input 2.
LED on digital pin 9.

Connect an LED to digital pin 9 with a 220 ohm current limiting resistor. Connect a photocell to 5V and then to analog pin 0 with a 10K ohm resistor as a reference to ground.

SCHEMATIC



IMAGE



CODE

```
// These constants won't change:
const int sensorPin = A0;    // pin that the sensor is attached to
const int ledPin = 9;        // pin that the LED is attached to

// variables:
int sensorValue = 0;         // the sensor value
int sensorMin = 1023;        // minimum sensor value
int sensorMax = 0;           // maximum sensor value


void setup() {
  Serial.begin(9600);
   pinMode(ledPin, OUTPUT);
  // turn on LED to signal the start of the calibration period:
  pinMode(13, OUTPUT); // ONBOARD LED pin 13
  digitalWrite(13, HIGH);

  // calibrate during the first five seconds
  while (millis() < 5000) {
    sensorValue = analogRead(sensorPin);

    // record the maximum sensor value
    if (sensorValue > sensorMax) {
      sensorMax = sensorValue;
    }

    // record the minimum sensor value
    if (sensorValue < sensorMin) {
      sensorMin = sensorValue;
    }
  }

  // signal the end of the calibration period
  digitalWrite(13, LOW);
}

void loop() {
  // read the sensor:
  sensorValue = analogRead(sensorPin);
   Serial.print(sensorValue);
   Serial.print(" ");

  // apply the calibration to the sensor reading
  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);

  // in case the sensor value is outside the range seen during calibration
  sensorValue = constrain(sensorValue, 0, 255);
  Serial.print(sensorValue);

  // fade the LED using the calibrated value:
  analogWrite(ledPin, sensorValue);
}
```

```
/*
 Calibration

 Demonstrates one technique for calibrating sensor input.  The
 sensor readings during the first five seconds of the sketch
 execution define the minimum and maximum of expected values
 attached to the sensor pin.

 The sensor minimum and maximum initial values may seem backwards.
 Initially, you set the minimum high and listen for anything
 lower, saving it as the new minimum. Likewise, you set the
 maximum low and listen for anything higher as the new maximum.

 The circuit:
 * Analog sensor (photocell will do) attached to analog input 0
 * LED attached from digital pin 9 to ground

 created 29 Oct 2008
 By David A Mellis
 modified 30 Aug 2011
 By Tom Igoe

 http://arduino.cc/en/Tutorial/Calibration

 This example code is in the public domain.

 */
```

Temperature Sensor TMP36

http://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor

hese sensors have little chips in them and while they're not that delicate, they do need to be handled properly. Be careful of static electricity when handling them and make sure the power supply is connected up correctly and is between 2.7 and 5.5V DC - so don't try to use a 9V battery!

They come in a "TO-92" package which means the chip is housed in a plastic hemi-cylinder with three legs. The legs can be bent easily to allow the sensor to be plugged into a breadboard. You can also solder to the pins to connect long wires. If you need to waterproof the sensor, you can see below for an Instructable for how to make an excellent case.

HARDWARE REQUIRED

Arduino board
TMP36
breadboard

CIRCUIT

This example code for Arduino shows a quick way to create a temperature sensor, it simply prints to the serial port what the current temperature is in both Celsius and Fahrenheit.

Unlike the FSR or photocell sensors we have looked at, the TMP36 and friends doesn't act like a resistor. Because of that, there is really only one way to read the temperature value from the sensor, and that is plugging the output pin directly into an Analog (ADC) input.

Remember that you can use anywhere between 2.7V and 5.5V as the power supply. For this example I'm showing it with a 5V supply but note that you can use this with a 3.3v supply just as easily. No matter what supply you use, the analog voltage reading will range from about 0V (ground) to about 1.75V.

If you're using a 5V Arduino, and connecting the sensor directly into an Analog pin, you can use these formulas to turn the 10-bit analog reading into a temperature:

Voltage at pin in milliVolts = (reading from ADC) * (5000/1024)
This formula converts the number 0-1023 from the ADC into 0-5000mV (= 5V)

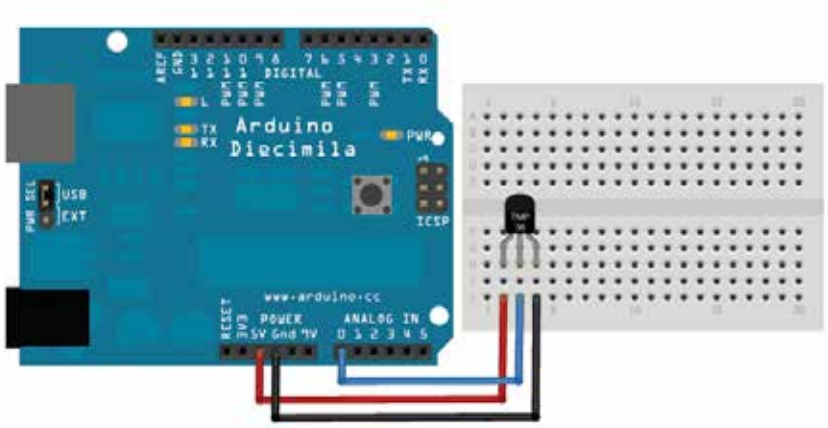
If you're using a 3.3V Arduino, you'll want to use this:

Voltage at pin in milliVolts = (reading from ADC) * (3300/1024)

This formula converts the number 0-1023 from the ADC into 0-3300mV (= 3.3V)

Then, to convert millivolts into temperature, use this formula:
Centigrade temperature = [(analog voltage in mV) - 500] / 10

IMAGE



CODE

```
//TMP36 Pin Variables
int sensorPin = 0; //the analog pin the TMP36's Vout (sense) pin is connected to
                   //the resolution is 10 mV / degree centigrade with a
                   //500 mV offset to allow for negative temperatures

/*
 * setup() - this function runs once when you turn your Arduino on
 * We initialize the serial connection with the computer
 */

void setup()
{
  Serial.begin(9600);  //Start the serial connection with the computer
                       //to view the result open the serial monitor
}

void loop()                     // run over and over again
{
 //getting the voltage reading from the temperature sensor
 int reading = analogRead(sensorPin);

 // converting that reading to voltage, for 3.3v arduino use 3.3
 float voltage = reading * 5.0;
 voltage /= 1024.0;

 // print out the voltage
 Serial.print(voltage); Serial.println(" volts");

 // now print out the temperature
 float temperatureC = (voltage - 0.5) * 100 ;
//converting from 10 mv per degree wit 500 mV offset
//to degrees ((volatge - 500mV) times 100)
 Serial.print(temperatureC); Serial.println(" degrees C");

 // now convert to Fahrenheit
 float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
 Serial.print(temperatureF); Serial.println(" degrees F");

 delay(1000);                                    //waiting a second
}
```

Controlling Servo Motor with Pot (Knob)

http://arduino.cc/en/Tutorial/Knob

Control the position of a RC (hobby) servo motor with your Arduino and a potentiometer. This example makes use of the Arduino servo library, by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

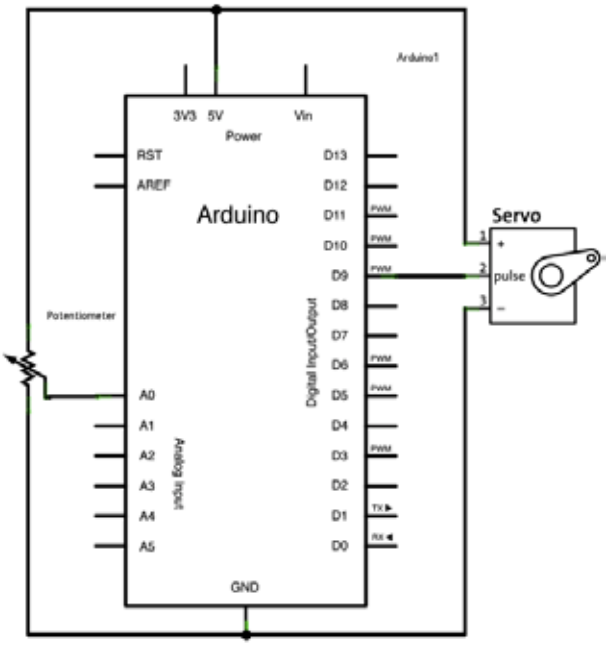
HARDWARE REQUIRED

Arduino Board
(1) Servo Motor 0-180º
(1) 10K Potentiometer

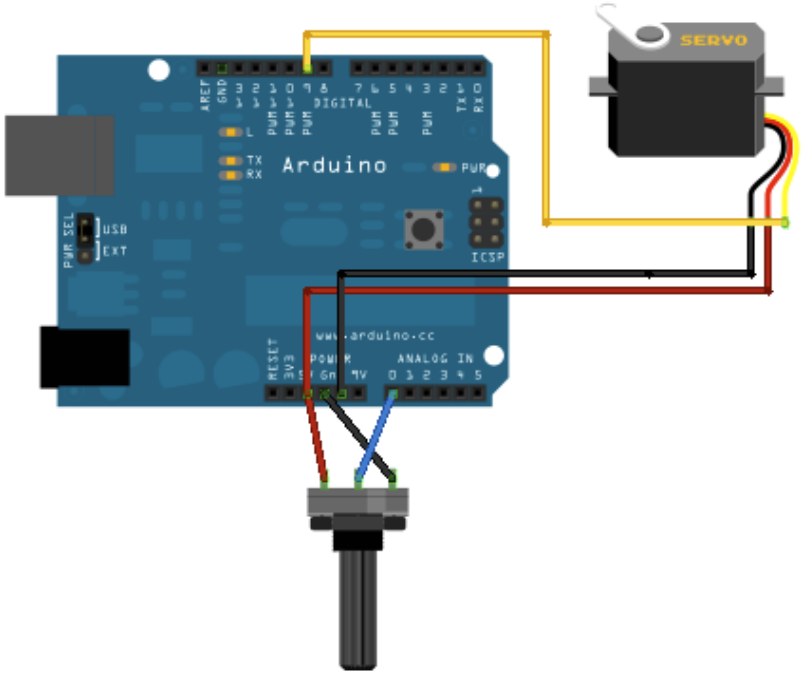
CIRCUIT

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. The signal pin is typically yellow or orange and should be connected to pin 9 on the Arduino board.
The potentiometer should be wired so that its two outer pins are connected to power (+5V) and ground, and its middle pin is connected to analog input 0 on the Arduino.

SCHEMATIC



IMAGE



CODE

```
// Controlling a servo position using a potentiometer (variable resistor)
// by Michal Rinott <http://people.interaction-ivrea.it/m.rinott>

#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin
int servoPin=9; // Servo connected on PWM  pin 9

void setup()
{
  myservo.attach(servoPin);  // attaches the servo on pin 9 to the servo object
}

void loop()
{
  val = analogRead(potpin);            // reads the value of the potentiometer
(value between 0 and 1023)
  val = map(val, 0, 1023, 0, 179);     // scale it to use it with the servo (value
between 0 and 180)
  myservo.write(val);                  // sets the servo position according to the
scaled value
  delay(15);                           // waits for the servo to get there
}
```

Controlling Continues Servo Motor with Pot

http://arduino.cc/en/Tutorial/Knob

Control a continues servo motor with your Arduino and a potentiometer. This example makes use of the Arduino servo library. This example it is based in Sweep example made by by BARRAGAN <http://barraganstudio.com>

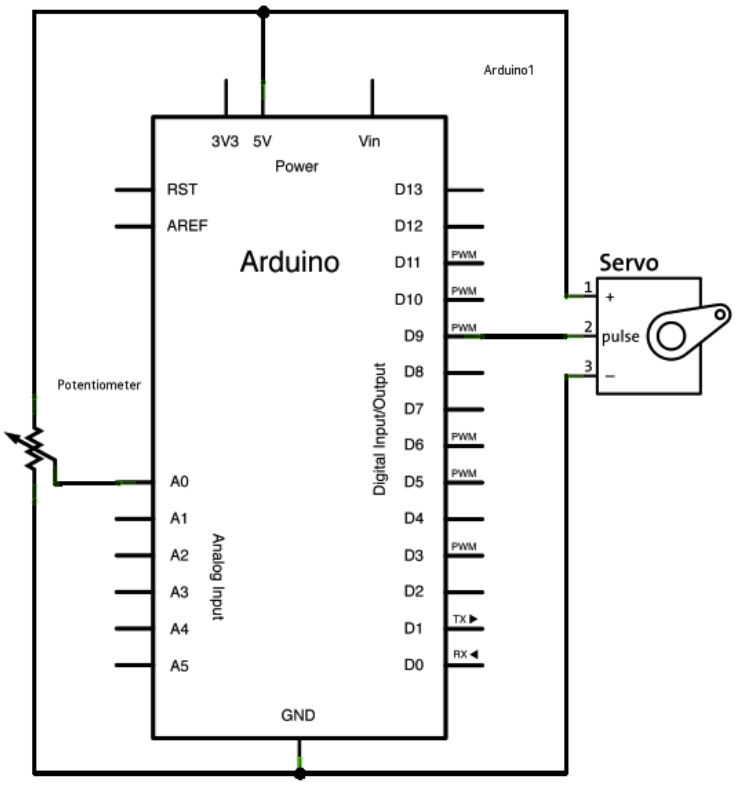
HARDWARE REQUIRED

Arduino Board
(1) Continues 360º Servo Motor
(1) Potentiometer

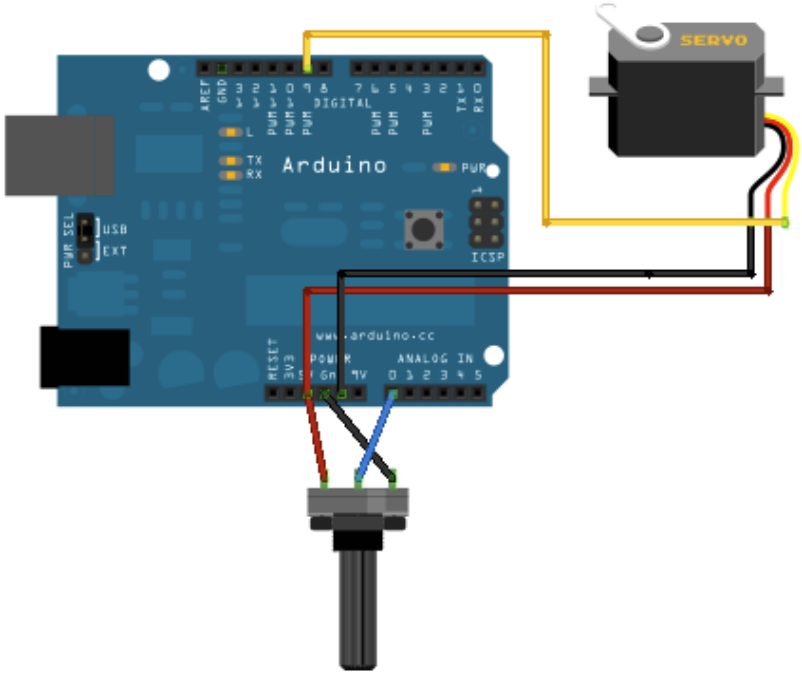
CIRCUIT

Servo motors have three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. The signal pin is typically yellow or orange and should be connected to pin 9 on the Arduino board.
The potentiometer should be wired so that its two outer pins are connected to power (+5V) and ground, and its middle pin is connected to analog input 0 on the Arduino.

SCHEMATIC



IMAGE



CODE

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
                // a maximum of eight servo objects can be created

int pos = 1;    // variable to store the servo position
int myDirection=1; // indicates the servo direction
int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup()
{
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
  Serial.begin(9600);
}


void loop()
{
  val = analogRead(potpin);            // reads the value of the potentiometer
(value between 0 and 1023)
  val = map(val, 0, 1023, 0, 5.5);     // scale it to use it with the servo (value
between 0 and 180)

  if (pos==0 || pos==180)

 // specify servo's direction
  myDirection=-myDirection;
  pos=pos+myDirection;

  // print to Serial the results
  Serial.print(val);
  Serial.print(" ");
  Serial.println(pos);

  // Use 90 to stop the servo
  myservo.write(180-pos); // apply position

  delay(val);
}
```

Servo Sketch

EXERCISE

Develop a project using an analog sensor (a poterntiometer or a photocell) and a servo motor. You can use optional LEDs.

HARDWARE REQUIRED

Arduino Board
10K Potentiometer
Potentiometer or Photocell
Servo Motor

SCHEMATIC

Ultrasonic Range Finder 3pins

http://arduino.cc/en/Tutorial/Ping

The Ping))) is an ultrasonic range finder from Parallax. It detects the distance of the closest object in front of the sensor (from 2 cm up to 3m). It works by sending out a burst of ultrasound and listening for the echo when it bounces off of an object. The Arduino board sends a short pulse to trigger the detection, then listens for a pulse on the same pin using the pulseIn() function. The duration of this second pulse is equal to the time taken by the ultrasound to travel to the object and back to the sensor. Using the speed of sound, this time can be converted to distance.

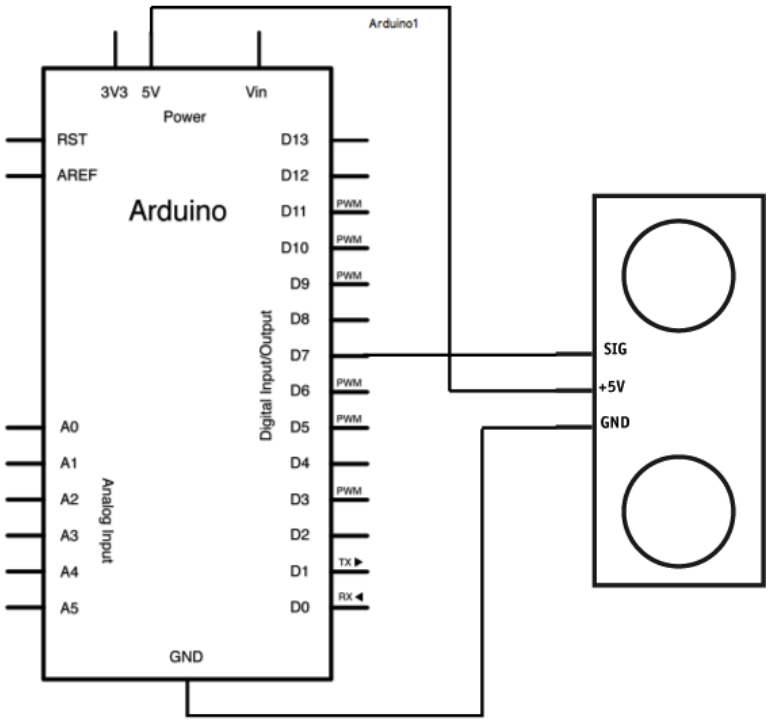
HARDWARE REQUIRED

Arduino Board
(1) Ping Ultrasonic Range Finder
hook-up wire

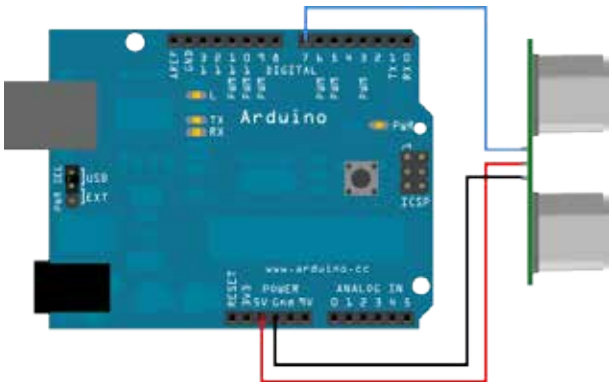
CIRCUIT

The 5V pin of the PING))) is connected to the 5V pin on the Arduino, the GND pin is connected to the GND pin, and the SIG (signal) pin is connected to digital pin 7 on the Arduino.

SCHEMATIC



IMAGE



CODE

```
// this constant won't change.  It's the pin number of the sensor's output:
const int pingPin = 7;

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
}

void loop()
{
  // establish variables for duration of the ping,
  // and the distance result in inches and centimeters: long duration, inches, cm;

  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);

  // The same pin is used to read the signal from the PING))): a HIGH
  // pulse whose duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);

  // convert the time into a distance
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);

  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();

  delay(100);
}

long microsecondsToInches(long microseconds)
{
  // According to Parallax's datasheet for the PING))), there are
  // 73.746 microseconds per inch (i.e. sound travels at 1130 feet per
  // second).  This gives the distance travelled by the ping, outbound
  // and return, so we divide by 2 to get the distance of the obstacle.
  // See: http://www.parallax.com/dl/docs/prod/acc/28015-PING-v1.3.pdf
  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds)
{
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the
  // object we take half of the distance travelled.
  return microseconds / 29 / 2;
}
```

```
/* Ping))) Sensor

  This sketch reads a PING))) ultrasonic rangefinder and returns the
  distance to the closest object in range. To do this, it sends a pulse
  to the sensor to initiate a reading, then listens for a pulse
  to return.  The length of the returning pulse is proportional to
  the distance of the object from the sensor.

  The circuit:
   * +V connection of the PING))) attached to +5V
   * GND connection of the PING))) attached to ground
   * SIG connection of the PING))) attached to digital pin 7

  http://www.arduino.cc/en/Tutorial/Ping

  created 3 Nov 2008
  by David A. Mellis
  modified 30 Aug 2011
  by Tom Igoe

  This example code is in the public domain.

 */
```

Ultrasonic Range Finder 4pins

http://learning.grobotronics.com/arduino-ultrasonic-tutorial.html
http://arduinobasics.blogspot.gr/2012/11/arduinobasics-hc-sr04-ultrasonic-sensor.html

Ultrasound is a high frequency sound (typically 40 KHz is used).
A short burst of sound waves (often only 8 cycles) is sent out the "Transmit" transducer (left, above). Then the "Receive" transducer listens for an echo. Thus, the principle of ultrasonic distance measurement is the same as with Radio-based radar.

HARDWARE REQUIRED

Ultrasonic Ranging Detector (SR04)
Arduino Board
Breadboard
Breadboard Jumper Cables

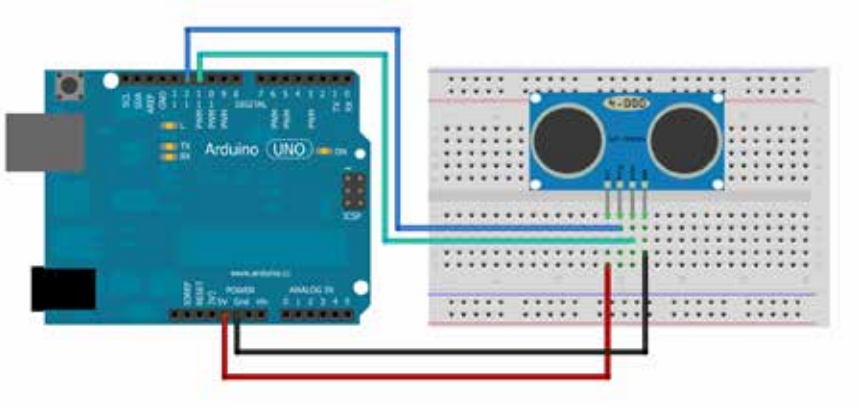
CIRCUIT

SCHEMATIC

- Works with many different ultrasonic sensor models: SR04, SRF05, SRF06, DYP-ME007 & Parallax PING)))™.
- Option to interface with all but the SRF06 sensor using only one Arduino pin.
- Doesn't lag for a full second if no ping echo is received like all other ultrasonic libraries.
- Ping sensors consistently and reliably at up to 30 times per second.
- Timer interrupt method for event-driven sketches.
- Built-in digital filter method ping_median() for easy error correction.
- Uses port registers when accessing pins for faster execution and smaller code size.
- Allows setting of a maximum distance where pings beyond that distance are read as no ping «clear».
- Ease of using multiple sensors (example sketch that pings 15 sensors).
- More accurate distance calculation (cm, inches & microseconds).
- Doesn't use pulseIn, which is slow and gives incorrect results with some ultrasonic sensor models.
- Actively developed with features being added and bugs/issues addressed.

NewPing Library Version 1.5
New in v1.5 - Released 8/15/2012:
Added ping_median() method which does a user specified number of pings (default=5) and returns the median ping in microseconds (out of range pings ignored). This is a very effective digital filter. Optimized for smaller compiled size (even smaller than sketches that don't use a library).

IMAGE



CODE

```
#include <NewPing.h>

// Arduino pin tied to trigger pin on the ultrasonic sensor.
#define TRIGGER_PIN  12
// Arduino pin tied to echo pin on the ultrasonic sensor.
#define ECHO_PIN     11
// Maximum distance we want to ping for (in centimeters). Maximum sensor dis-
tance is rated at 400-500cm.
#define MAX_DISTANCE 400

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
// NewPing setup of pins and maximum distance.

void setup() {
  Serial.begin(115200);
  // Open serial monitor at 115200 baud to see ping results.
}

void loop() {
  delay(50);
 // Wait 50ms between pings (about 20 pings/sec).
 // 29ms should be the shortest delay between pings.

  unsigned int uS = sonar.ping();
  // Send ping, get ping time in microseconds (uS).
  if (uS / US_ROUNDTRIP_CM==0) {
  // on spam data (=0) do nothing
  }
  else
  {
    Serial.print("Ping: ");
    Serial.print(uS / US_ROUNDTRIP_CM);
// Convert ping time to distance in cm and print result
// (0 = outside set distance range)
    Serial.println("cm");
  }
}
```

Force Sensor

http://itp.nyu.edu/physcomp/sensors/Reports/ForceSensorResistor
http://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr

The sensor allows one to detect and measure the change in the applied force and also the rate at which the force is changing. It could detect contact or touch. Identify force thresholds and trigger actions. Its force sensitivity is optimized for use in human touch control of electronic devices.The FSR sensors have wide usage in the commercial and industry arena. The FSR can be applied to various fields such as industry, medical science, robotics, automotive, recreational and body pressure equipment.

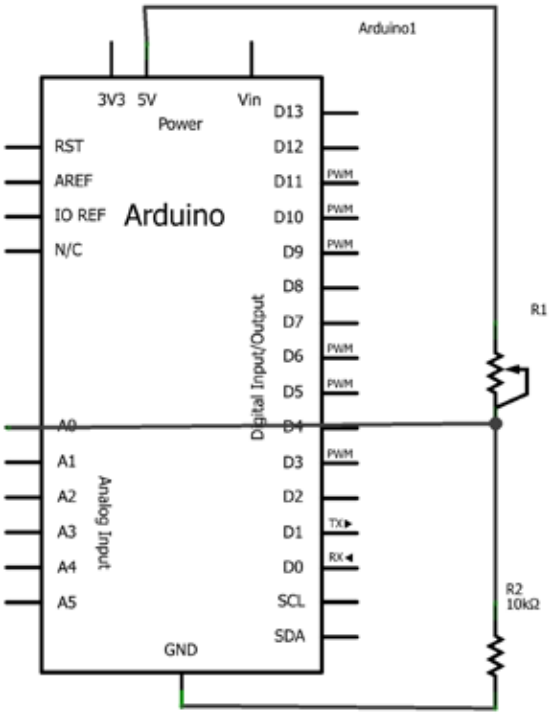
HARDWARE REQUIRED

Arduino Board
Breadboard
Pre sure Sensor (Force)
(1) 10K Resistor

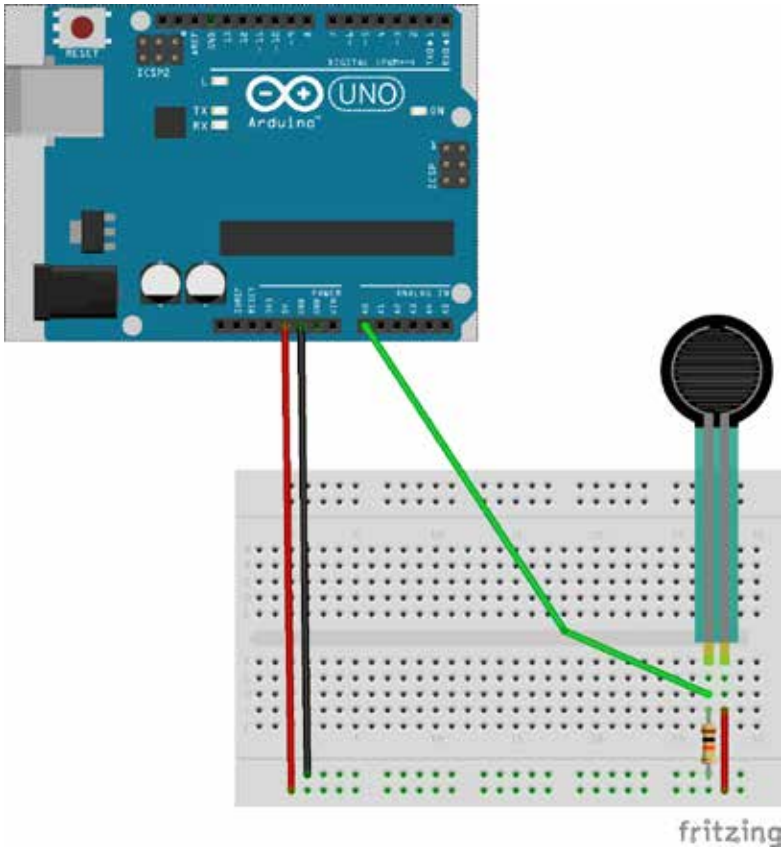
CIRCUIT

Parameter	Value
Size Range	Max : 20» x 24» / Min : 0.2» x 0.2»
Device Thickness	0.008 h x 0.050 h
Force Sensitivity Range	< 100g to > 10kg
Pressure Sensitivity Range	< 1.5psi to > 150psi
Part-to-Part Force Repeatability	+-15% to +-25% of established nominal resistance
Single Part Force Repeatable	+-2% to +-5% of established nominal resistance
Force Resolution Better than	0.5% full scale
Break Force	20g to 100g
Stand-Off Resistance	> 1M
Switch Characteristic	Essentially zero travel
Device Rise Time	1-2 msec
Lifetime	> 10 million actuation
Temperature Range	-30 degree Celsius to +70 deg C
Maximum Current	1mA/cm2 of applied force
Maximum Voltage	5V
Sensitivity to Noise	Passive device

SCHEMATIC



IMAGE



CODE

```
// Learn how to use Force Sensor form ITP tutorials
// http://itp.nyu.edu/physcomp/sensors/Reports/ForceSensorResistor

int sensorPin = A0;    // select the input pin for the potentiometer
int sensorValue = 0;  // variable to store the value coming from the sensor

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

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);
}
```

Force Sensor Resistor displays a decrease in resistance with an increase in the force applied to the active surface.
FSR is composed of 3 portions (Interlink Electronics Model No. 402).

Vent : The vent assures pressure equilibrium.
Spacer : The width of gap and fingers of the conductive grid.
Active Area : The area responds to force with decrease in resistance.
Tail : The area where the busing system terminates

At first, this sensor acts like a switch. It quickly goes from not being touched to feeling the force inflicted on it. From there on it continues to document the touch and it's particular force. It does reach a saturation point, where pressing harder is no longer detectable. It's a passive device since it starts out in a constant state that needs to be disrupted by the touch. It tracks the force applied to it instantly, I could not perceive a delay in the information sent. It's not affected by noise or vibration.

Flex Sensor

http://bildr.org/2012/11/flex-sensor-arduino/
http://arduinobasics.blogspot.gr/2011/05/arduino-uno-flex-sensor-and-leds.html

Flex sensor works similar to Force Sensor (FSR).

The flex sensor is one of those parts often overlooked by the advanced user. But what if you need to check if something bent? Like a finger, or a doll arm. (A lot of toy prototypes seem to have this need).

Anytime you need to detect a flex, or bend, a flex sensor is probably the part for you. They come in a few different sizes (small, large).

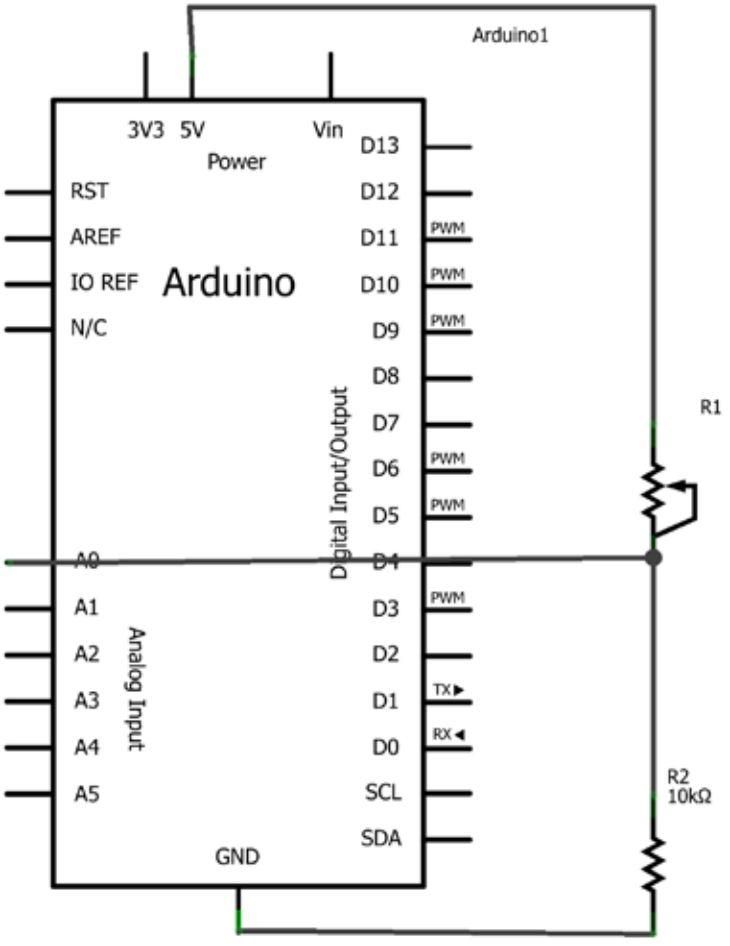
The flex sensor is basically a variable resistor that reacts to bends. Unbent it measures about 22KΩ, to 40KΩ when bend 180º. Note that the bend is only detected in one direction and the reading can be a bit shaky, so you will have best results detecting changes of at least 10º.

Also, make sure you don't bend the sensor at the base as it wont register as a change, and could break the leads. I always tape some thick board to the base of it to make it wont bend there.

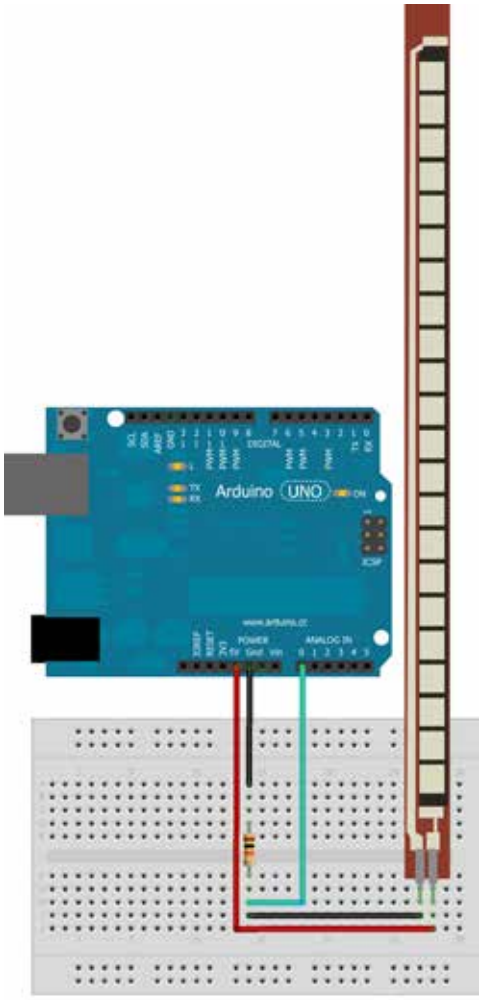
HARDWARE REQUIRED

Arduino Board
Breadboard
(1) Flex Sensor
(1) 10K Resistor

SCHEMATIC



IMAGE



CODE

```
// Learn how to use Force Sensor form ITP tutorials
// http://itp.nyu.edu/physcomp/sensors/Reports/ForceSensorResistor

int sensorPin = A0;    // select the input pin for the potentiometer
int sensorValue = 0;  // variable to store the value coming from the sensor

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

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue);
}
```

Analog Meal

EXERCISE

Develop a project using Digital and Analog features.
Please keep the concept concrete and the circuit simple.

HARDWARE REQUIRED

Arduino Board
breaboard
--

SCHEMATIC