RMIT INDUSTRIAL DESIGN

INTERACTIVE READER

RMIT Industrial Design Elective

Contributors

Dr. Scott Mitchell, RMIT
Sebastiaan Pijnappel, RMIT
Imran Shamsul, RMIT

Last updated: 05 March 2014

TABLE OF CONTENTS

LEGEND

COLOR CODING

Throughout the reader you'll find colored squares and pieces of text highlighted in various colors. Each of these is a click-able link taking you to on line tutorials and elaborations on the material, or a cross-reference taking you to another page of this reader. The various colors indicate the following:

More information on basic functionality and underlying technological principles of components or circuits.

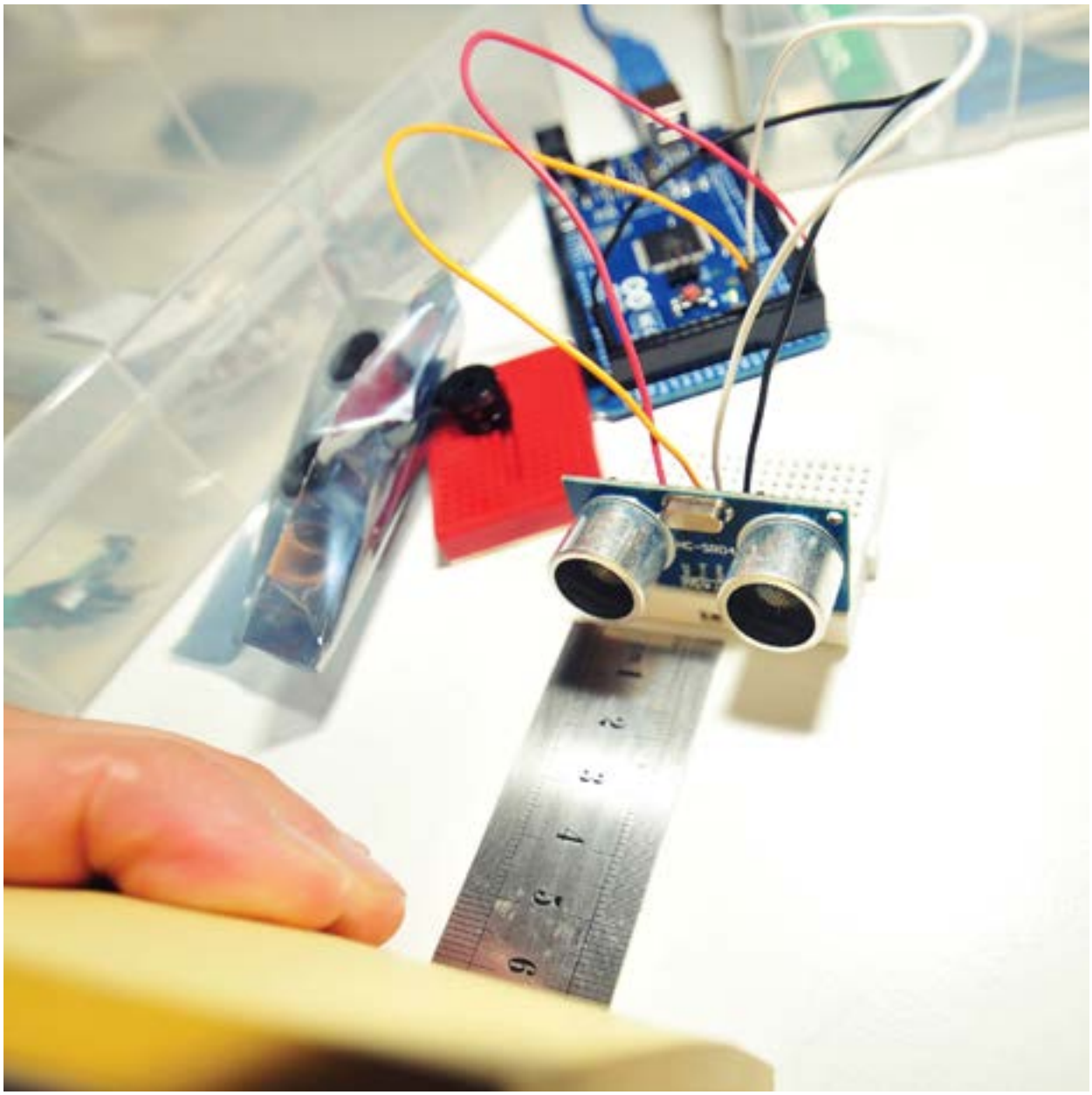
Examples or overviews of common applications of a component or circuits in everyday products that you might be designing one day.

Examples of how a component is commonly used in combination with other electronics to achieve more complex functionality. This includes tutorials to help you build some yourself.

Examples and walkthroughs of how to use a component with an Arduino. This includes code examples that you can copy and play with!

On line forums discussing issues regarding implementation components. This is where the DIY community helps each other out, so it should be your first stop if you can't figure things out yourself.

1. INPUT CASE EXAMPLES



Shamsul, Imran 2012

1.1 POTENTIOMETER

Wikipedia 2012

SUMMARY

A potentiometer, or potmeter as they are often called, is a simple knob that provides a variable resistance, which we can read into the Arduino board as an analog value. Potentiometers are commonly used to control electrical devices, for example as a volume knob on audio equipment. Potentiometers operated by a mechanism can be used as position transducers, for example in a joystick. Potentiometers are rarely used to directly control significant power (more than a watt), since the power dissipated in the potentiometer would be comparable to the power in the controlled load.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia
- Potmeter taken apart

Common applications relevant to industrial design.

- Joystick

Examples of interfacing with other electronics, incl. tutorials.

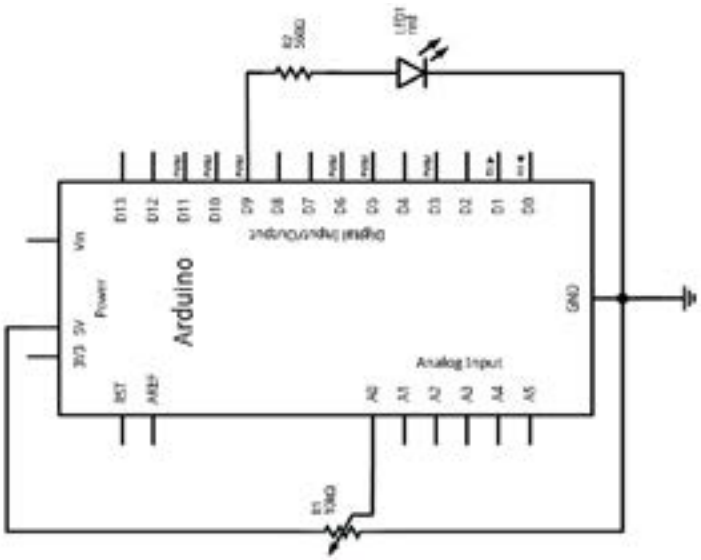
- Voltage divider

Arduino software examples, incl. elaboration.

- Arduino website

Forums covering issues regarding implementation.

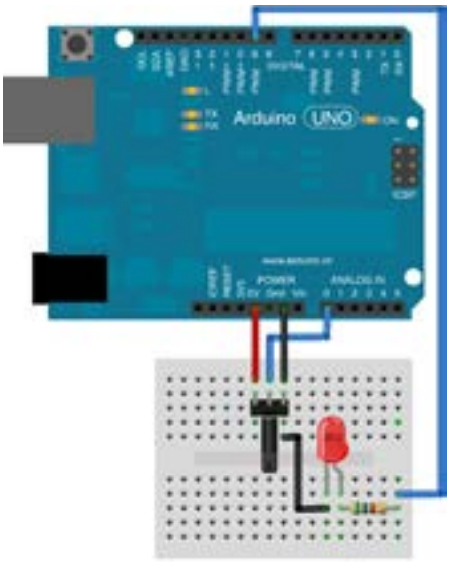
- Arduino forum



CIRCUIT DIAGRAM

The diagram shows a potmeter configured to make LED blink according to the resistance set. Use a 10KOhm resistor for R1, and a 560Ohm one for R2. See the next page for the Arduino code required for this setup.

In this example, the potmeter is used as a so called voltage divider. This allows it to regulate the voltage it puts out to the Arduino. For elaboration click here.



FRITZING DIAGRAM

This fritzing diagram shows that the LED ground is connected to the ground of the potentiometer. This saves leads and space on a circuit.

FRITZING DIAGRAM

This fritzing diagram shows that the LED ground is connected to the ground of the potentiometer. This saves leads and space on a circuit.

CODE EXAMPLE

This code, in combination with the circuit shown in the diagrams on the previous page, turn on and off a light emitting diode(LED) connected to digital pin 9. The amount of time the LED will be on and off depends on the value obtained by analogRead(). By turning the shaft of the potentiometer, we change the amount of resistance on either side of the wiper which is connected to the center pin of the potentiometer. This changes the relative "closeness" of that pin to 5 volts and ground, giving us a different analog input. When the shaft is turned all the way in one direction, the voltage on pin 2 is 0V, and we read 0. When the shaft is turned all the way in the other direction, the voltage on pin 2 is 5V, and we read 1023. Anywhere in between these extremes analogRead() returns a number between 0 and 1023 that is proportional to the amount of voltage being applied to the pin.

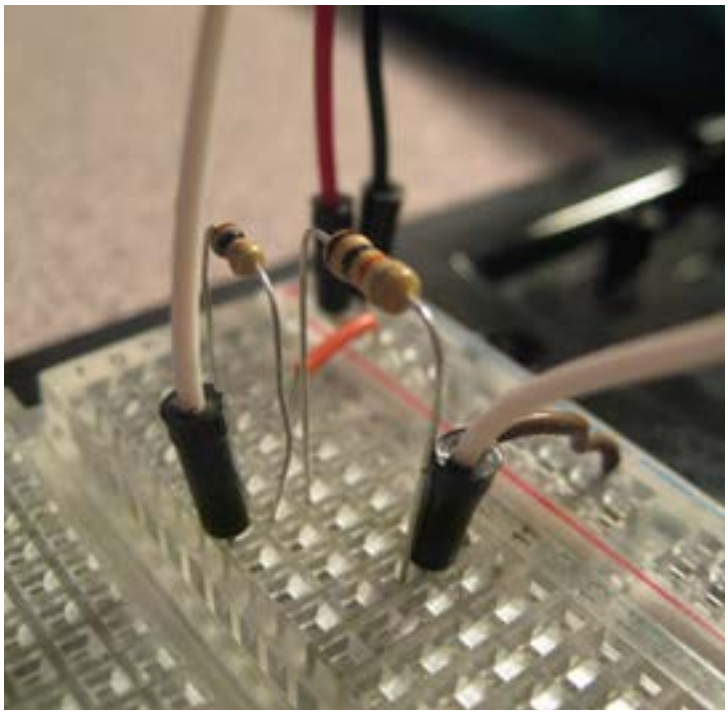
For elaboration on this example click here.

```
int potPin = 0;      // select the input pin for the potentiometer
int ledPin = 9;      // select the pin for the LED
int val = 0;     // variable to store the value coming from the sensor

void setup() {
    pinMode(ledPin, OUTPUT);  // declare the ledPin as an OUTPUT
}

void loop() {
    val = analogRead(potPin);    // read the value from the sensor
    digitalWrite(ledPin, HIGH);  // turn the ledPin on
    delay(val);                  // stop the program for some time
    digitalWrite(ledPin, LOW);   // turn the ledPin off
    delay(val);                  // stop the program for some time
}
```

RELATED CHAPTERS

1.2 Voltage divider circuit.

1.2 VOLTAGE DIVIDER



Arduinology 2012

SUMMARY

A voltage divider (VD) is a simple circuit that produces an output voltage (Vout) that is a fraction of its input voltage (Vin). An example of a voltage divider consists of two resistors in series or a potentiometer (see previous chapter). The output voltage is determined by the ratio of the values of both resistors, or the values on either side of the wiper of the potentiometer. Voltage dividers are commonly used to provide a reference voltage anywhere between 0 and the supply voltage. Note that they can supply only small amounts of current. If larger currents are drawn, the voltage drops.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia
- Voltage divider calculator
- YouTube: VD circuit

Common applications relevant to industrial design.

- Sparkfun

Examples of interfacing with other electronics, incl. tutorials.

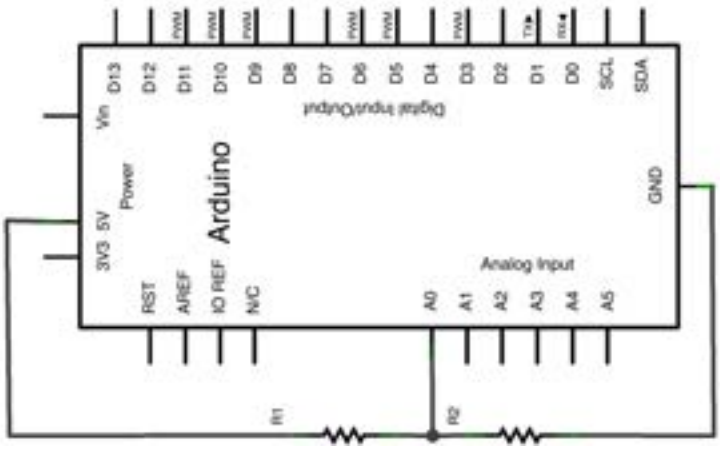
- Sparkfun: tutorial
- Interactive tutorial

Arduino software examples, incl. elaboration.

- Fritzing VD project
- Project: battery monitor

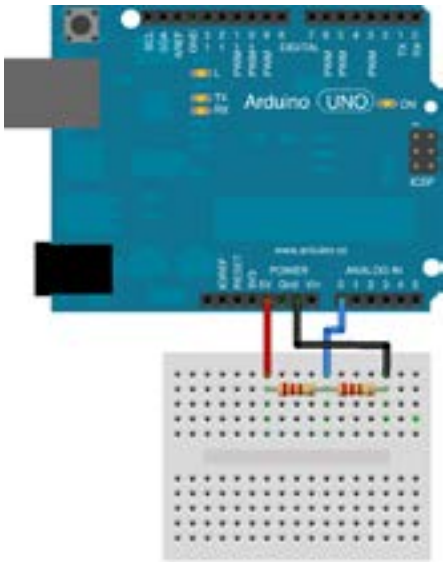
Forums covering issues regarding implementation.

- Tossen Robotics forum
- Arduino forum



CIRCUIT DIAGRAM

This circuit diagram shows a voltage divider wired so that the regulated output voltage can be read by the analog input of the Arduino. Feel free to play around different resistor values to get different outputs.



FRITZING DIAGRAM

CODE EXAMPLE

Here's a very basic bit of Arduino code that will read the output Voltage of the voltage divider and display it with a refresh rate of 4 times per second. The voltage is read as a number between 0 and 1023, which is linearly correlated with voltage values between 0V and 5V. Feel free to play around different resistor values to get different outputs.

This same example can be used for the photoresistor, thermistor and force sensitive resistor, which are explained in the next three chapters. Each of these components is a resistor of some sort, which can replace R1 in the voltage divider.

```
int FSR_Pin = A0; //analog pin 0

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

void loop(){
    int FSRReading = analogRead(FSR_Pin);
    Serial.println(FSRReading);
    delay(250); //just here to slow down the output for easier reading
}
```

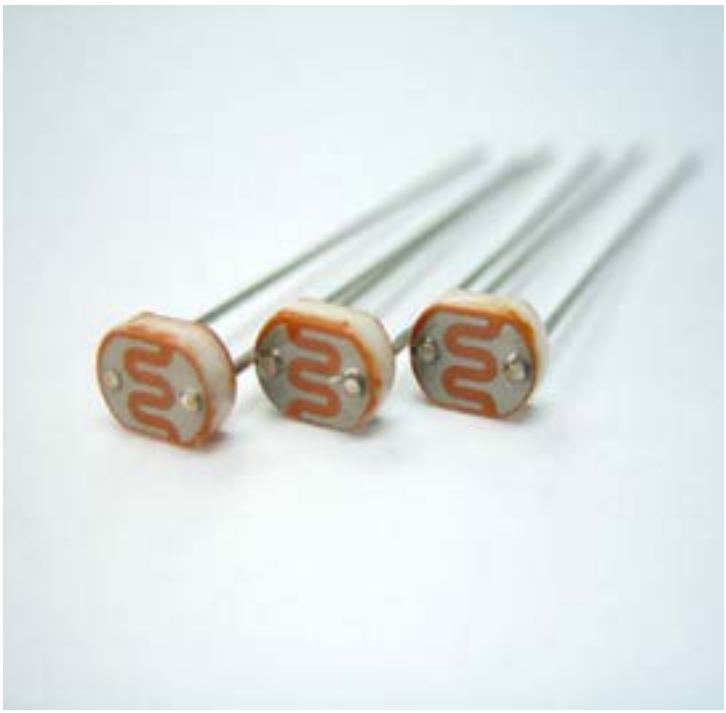
RELATED CHAPTERS

1.1 Potentiometer

1.3 Photoresistor

1.5 Force Sensistive Resistor

1.3 PHOTORESISTOR



CQbiz 2012

SUMMARY

A photoresistor, or Light Dependent Resistor (LDR), is a resistor whose resistance decreases with increasing light intensity. Photoresistors come in many different types, varying in size, sensitivity, the amount of power it can handle (indicated in watts), build quality, etc. Inexpensive types, such as the ones shown in the photo, can be found in many consumer items such as camera light meters, street lights, clock radios, alarm devices, outdoor clocks and solar street lamps, etc. In many cases they are used to turn a device on or off depending on environmental lighting conditions.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia
- Resistor guide
- Tech. student

Common applications relevant to industrial design.

- Light meter

Examples of interfacing with other electronics, incl. tutorials.

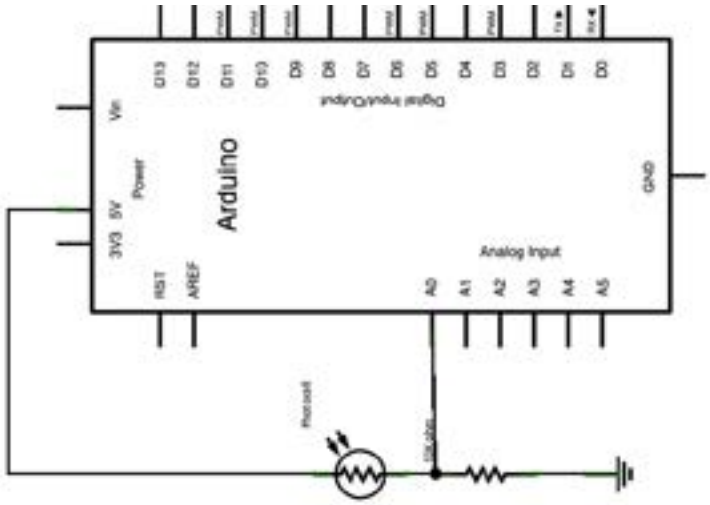
- Adafruit: learn
- Example projects

Arduino software examples, incl. elaboration.

- Simple LDR func. test
- LDR to measure light levels

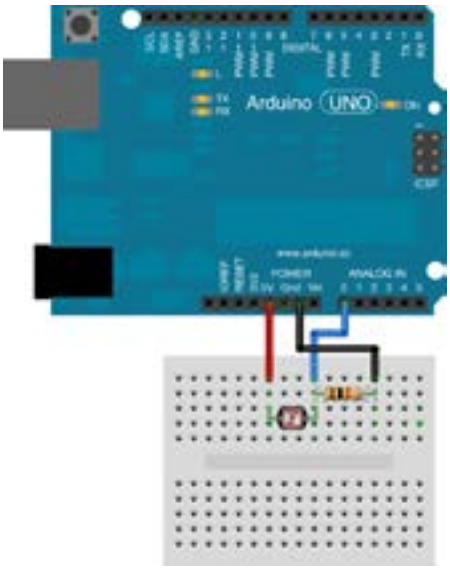
Forums covering issues regarding implementation.

- Arduino forum



CIRCUIT DIAGRAM

This circuit shows the LDR configured in a voltage divider in order for the light intensity on the sensor to influence the output voltage read by the analog Arduino input. Note that this exactly the same circuit as shown in the voltage divider chapter, but with one of the regular resistors swapped for an LDR. Use 10KOhm for the remaining resistor.



FRITZING DIAGRAM

CODE EXAMPLE

Although one could simply use the code example from the voltage divider chapter to display the varying voltage output, we'll go one step further and actually get the Arduino to tell us in words whether it's dark, dim, medium or bright. The code does almost the same thing as the voltage divider example, but in addition it also associates different ranges of sensor values with different words that you define. The current words describe the amount of light on the sensor, but feel free to make it say anything you want. For elaboration on this example click here.

```
const int sensorMin = 0;      // sensor minimum, discovered through experiment
const int sensorMax = 600;    // sensor maximum, discovered through experiment

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

void loop() {
    // read the sensor:
    int sensorReading = analogRead(A0);
    // map the sensor range to a range of four options:
    int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

    // do something different depending on the
    // range value:
    switch (range) {
    case 0:    // your hand is on the sensor
        Serial.println("dark");
        break;
    case 1:    // your hand is close to the sensor
        Serial.println("dim");
        break;
    case 2:    // your hand is a few inches from the sensor
        Serial.println("medium");
        break;
    case 3:    // your hand is nowhere near the sensor
        Serial.println("bright");
        break;
    }
    delay(1);        // delay in between reads for stability
}
```

RELATED CHAPTERS

1.2 Voltage Divider

1.4 THERMISTOR

Shenzen Winsun Technology 2012

SUMMARY

Similar to the light dependent resistors discussed in the previous chapter, thermistors are simple electronic components of which the resistance varies significantly (i.e. much more so than in standard resistors) in correlation to changes in an environmental variable; in this case temperature. Thermistors are often used as temperature sensors, self-regulating heating elements or inrush current limiters. Inrush current limiters are found in power supplies of everyday electrical devices such as laptops, mobile phones, etc. to protect other electrical components from sudden increases in current. A high current heats up the thermistor, increasing it's resistance and thus limiting the current through it.

Compared to other temperature sensors (such as resistance temperature detectors) thermistors usually achieve a higher precision within a limited temperature range, typically −90 °C to 130 °C. The material used in thermistors is generally a ceramic or polymer.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia
- YouTube: measuring temp.
- Adafruit: overview

Common applications relevant to industrial design.

- Fridge thermistor
- Replacing fridge thermistor

Examples of interfacing with other electronics, incl. tutorials.

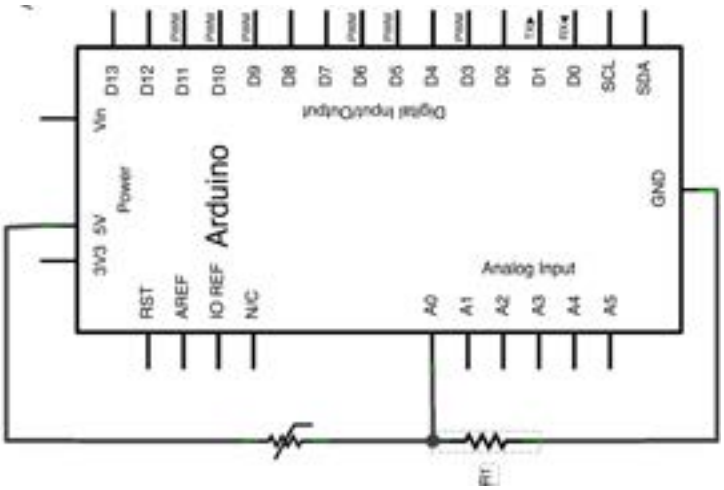
- Digital lab

Arduino software examples, incl. elaboration.

- YouTube: Analog Inputs
- Using a thermistor
- Binary thermo-meter

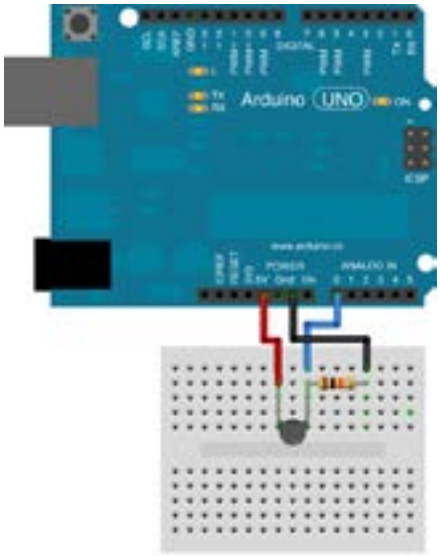
Forums covering issues regarding implementation.

- Arduino forum
- Printrbot forum



CIRCUIT DIAGRAM

Again, this circuit uses same basic voltage divider principle used in previous chapters to produce varying voltage outputs, in this case in relation to temperature “felt“ by the thermistor. Use 10KOhm for R1 (if you’re using a 10KOhm thermistor).



FRITZING DIAGRAM

CODE EXAMPLE

As with any of the resistors, you can simply use the Arduino example provided in the voltage divider chapter to start interfacing the thermistor with the Arduino. For a more sophisticated example, translating sensor voltage values to actual temperatures, click here.

RELATED CHAPTERS

1.2 Voltage Divider

1.5 FORCE SENSITIVE RESISTOR

Transformativedesign 2012

SUMMARY

Similar the light dependent resistors and thermistors, the resistance of a force sensitive resistor (or FSR) changes significantly in correlation to changes in an environmental variable, in this case force applied to the sensor's surface. The sensing film consists of both electrically conducting and non-conducting particles suspended in matrix. Applying a force to the surface of a the sensing film causes particles to touch the conducting electrodes, changing the resistance of the film.

Force-sensing resistors are commonly used to create pressure-sensing buttons and have applications in many fields, including musical instruments, car occupancy sensors, and portable electronics. They are inexpensive and thin, but rather inaccurate in comparison to other force-sensing solutions.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia
- Adafruit

Common applications relevant to industrial design.

- Electric drum
- How the e-drum works
- Tactile sensing

Examples of interfacing with other electronics, incl. tutorials.

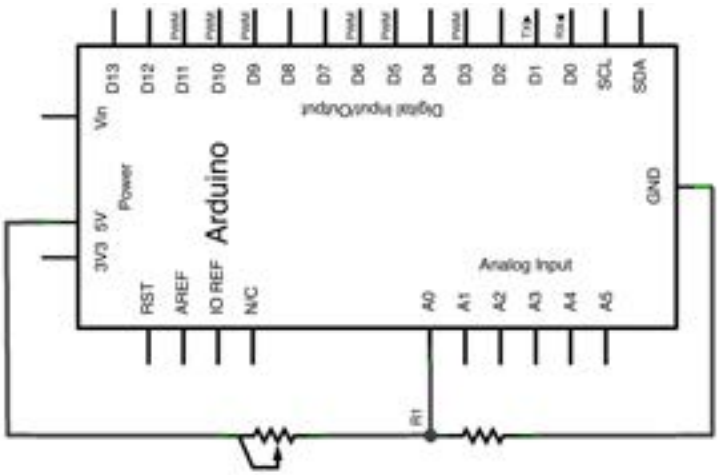
- DIY drum trigger pads
- FSR music applied
- MIDI shoes

Arduino software examples, incl. elaboration.

- Do-it-yourself FSR
- FSR tutorial
- FSR + Arduino

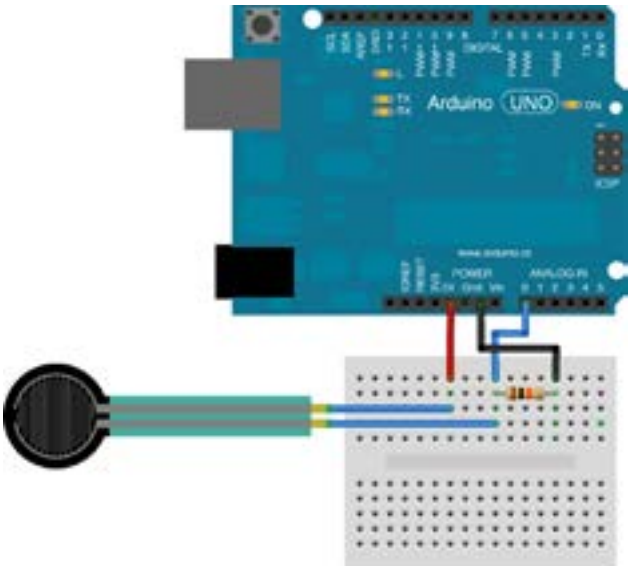
Forums covering issues regarding implementation.

- Arduino forum
- Forum Q: FSR punch force



CIRCUIT DIAGRAM

This circuit shows the force sensitive resistor configured in a voltage divider in order for the force applied to the sensor to influence the output voltage read by the analog Arduino input. Note that this exactly the same circuit as in the voltage divider chapter, but with one of the regular resistors swapped for the force sensitive resistor. Use 10KOhm for the remaining resistor.



FRITZING DIAGRAM

CODE EXAMPLE

As with any of the resistors, you can simply use the Arduino example provided in the voltage divider chapter to start interfacing the thermistor with the Arduino. For a more sophisticated examples, including translating voltage values to actual force in Newton, click here.

RELATED CHAPTERS

1.2 Voltage Divider

1.6 ULTRASONIC DISTANCE SENSOR

Sadoniatech 2012

SUMMARY

Ultrasonic distance sensors are a common solution for sensing the presence and distance of an object that is within range of the sensor. They generate high frequency sound waves and evaluate the echo which is received back by the sensor. The distance is calculated from the time interval between sending the signal and receiving the echo. They are used in ultrasonic people detection, assisting in autonomous UAV navigation, and in automated manufacturing processes.

Ultrasonic distance sensors for prototyping purposes can be bought pre-assembled on a circuit board (for example here or here) as shown in the photo. They plug straight into an Arduino. Their range can be set through the code.

Infrared distance sensors are another common distance sensing solution. They typically have a narrower range, and don't work well with dark colored or transparent objects. AustralianRobotics.com stock them.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia

Common applications relevant to industrial design.

- Parking sensors
- iGlasses
- The Miniguide

Examples of interfacing with other electronics, incl. tutorials.

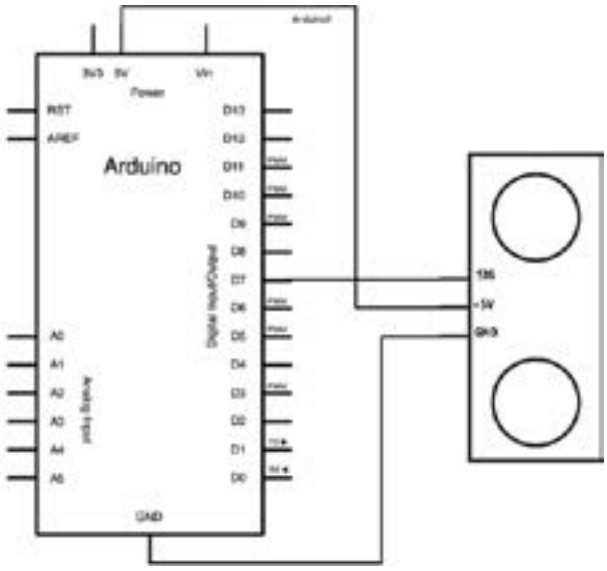
- VIA

Arduino software examples, incl. elaboration.

- Ping tutorial
- Garage parking assistant

Forums covering issues regarding implementation.

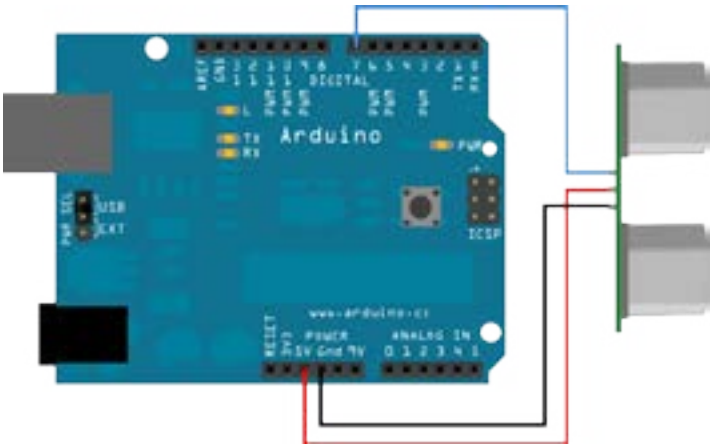
- Arduino forum



Arduino.cc 2012

CIRCUIT DIAGRAM

The specific ultrasonic distance sensor shown and used in this chapter is the Ping Ultrasonic Rangefinder from Parallax. It is built to conveniently interface with the Arduino, and it is featured in many on line examples and tutorials. It's signal pin is used to trigger the sensor, as well as to read the sensor data from. See the code example on the next page for elaboration.



Arduino.cc 2012

FRITZING DIAGRAM

CODE EXAMPLE

This code example displays the distance to an object placed in front of the sensor. 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. For elaboration on this example click here.

```
// 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) {
    // 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.
    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 traveled.
    return microseconds / 29 / 2;
}
```

RELATED CHAPTERS

1.8 Piezoelectric Sensor

1.7 ACCELEROMETER

Sparkfun 2012

SUMMARY

Accelerometers measure acceleration forces. These forces may be static, like the constant force of gravity pulling at your feet, or dynamic, caused by movement. When measuring static acceleration due to gravity, the sensor reading changes according to the angle at which the sensor is titled. When in movement, the acceleration and direction of movement can be derived from the sensor readings. These readings are any voltage value between 0 and the supply voltage. Most accelerometers measure along two (X, Y) or three (X, Y, Z) axes.

Accelerometers are ubiquitous these days, as they are found in many smart phones, tablet computers and laptops to (amongst other things) enable movement and orientation of the device as a mode of user input. Other uses are: performance measurement in cars, or aiding in aircraft / missile navigation systems.

Accelerometers are not to be confused with gyroscopes, which cannot measure stationary orientation.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia
- Dimension Engineering
- Sparkfun Learn

Common applications relevant to industrial design.

- Crash test dummy
- Smart phones
- Wii remote

Examples of interfacing with other electronics, incl. tutorials.

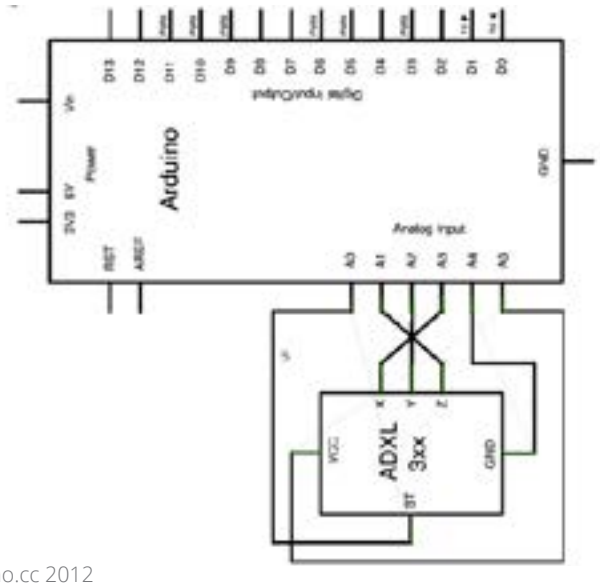
- Tilt detector
- Wii controller + Robot
- Gyro camera

Arduino software examples, incl. elaboration.

- Make: Arduino 101
- Tutorial: ADXL3xx Accel.

Forums covering issues regarding implementation.

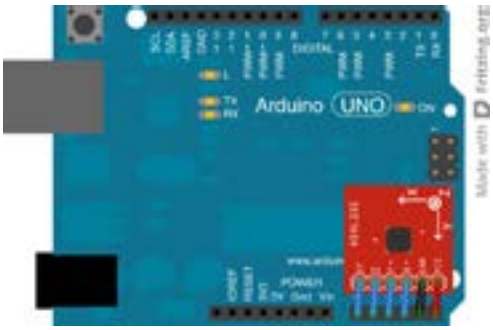
- Arduino forum



Arduino.cc 2012

CIRCUIT DIAGRAM

Accelerometers can be bought pre-soldered to a PCB. This saves you the burden of soldering the required auxiliary components. The PCB with the accelerometer on it can simply be connected to an Arduino as shown to the left. Note that this example uses analog pins as power and ground. These pins are then defined as such in the Arduino code (see next page). For elaboration click here.



FRITZING DIAGRAM

This Fritzing diagram of the above circuit shows wires. However, as distance between pins is standardized and is thus identical for the accelerometer and the Arduino, you can also plug the sensor directly on top of the Arduino's analog pins. Make sure you solder the connectors onto the bottom of the sensor, as otherwise all pins will be in reverse.

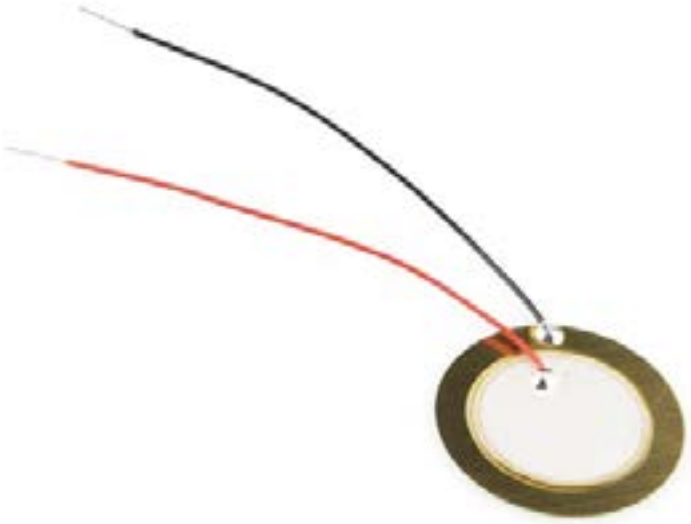
CODE EXAMPLE

This code reads the sensor values of all three axes (x, y, z) of the accelerometer, and displays them on the serial monitor on your screen. Short 100 millisecond delays after each cycle of reading and displaying the sensor values determine the refresh rate. For elaboration on this example click here.

```
const int groundpin = 18;             // analog input pin 4 -- ground
const int powerpin = 19;              // analog input pin 5 -- voltage
const int xpin = A3;                  // x-axis of the accelerometer
const int ypin = A2;                  // y-axis
const int zpin = A1;                  // z-axis (only on 3-axis models)

void setup() {// initialize the serial communications:
    Serial.begin(9600);

    // Provide ground and power by using the analog inputs as normal
    // digital pins.  This makes it possible to directly connect the
    // breakout board to the Arduino.  If you use the normal 5V and
    // GND pins on the Arduino, you can remove these lines.
    pinMode(groundpin, OUTPUT);
    pinMode(powerpin, OUTPUT);
    digitalWrite(groundpin, LOW);
    digitalWrite(powerpin, HIGH);
}

void loop() {
    // print the sensor values:
    Serial.print(analogRead(xpin));
    // print a tab between values: Serial.print("\t");
    Serial.print(analogRead(ypin));
    // print a tab between values:
    Serial.print("\t");
    Serial.print(analogRead(zpin));
    Serial.println();
    // delay before next reading:
    delay(100);
}
```

RELATED CHAPTERS

1.8 PIEZOELECTRIC SENSOR



MindKits 2012

SUMMARY

Piezoelectric sensors use the piezoelectric effect to measure vibration, shock or flex. Put simply, this effect causes a piezoelectric material to generate a charge upon (even very small) mechanical deformation. The charge is proportional to the amount of deformation. In reverse, piezoelectric material deforms when subjected to electrical charge.

Compared to many other sensors that can pick up deformation caused by pressure, acceleration, etc., piezoelectric sensors are extremely sensitive. This makes them perfect for high-precision applications, such as automatic focusing of optical assemblies. However, they are also used as touch sensors, as microphones (e.g. as guitar pickups), etc.

Important to note is that a piezoelectric sensor will not measure static deformation. For example, if you were to bend it, it will exhibit an increase in charge while bending, but drop back to 0 once static.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia
- General theory

Common applications relevant to industrial design.

- Piezo guitar pick-up
- Piezo sensor applications
- type a title

Examples of interfacing with other electronics, incl. tutorials.

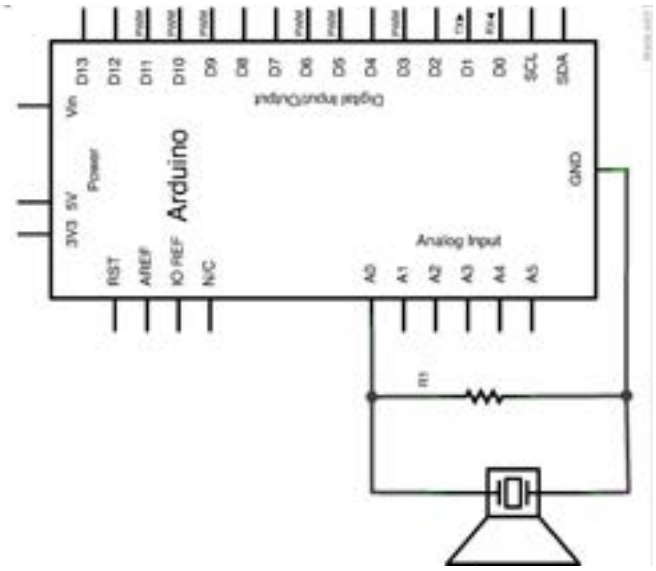
- Solar-powered Cricket

Arduino software examples, incl. elaboration.

- Knock sensor

Forums covering issues regarding implementation.

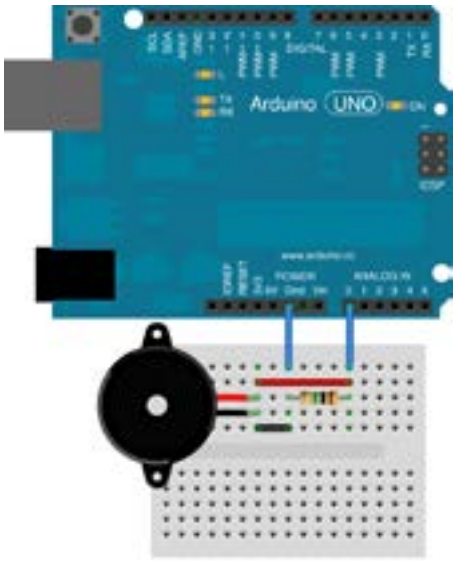
- Arduino forum



CIRCUIT DIAGRAM

As explained on the last page, piezoelectric elements generate small charged when deformed. In the circuit on the left these charges are fed into the Arduino's analog input to be measured. Use a 1MOhm resistor for R1. For elaboration click here.

Note that the diagram shows a speaker-like component. This is called a "piezo speaker". This component can produce sound, but can also be used as a vibration / deformation sensor in the same way as the basic piezoelectric element shown on the previous page can. You can use either of the two.



FRITZING DIAGRAM

CODE EXAMPLE

This sketch reads a piezo element to detect a knocking sound. It reads an analog pin and compares the result to a set threshold. If the result is greater than the threshold, it writes “knock” to the serial port, and toggles the LED on pin 13. For the full example click here.

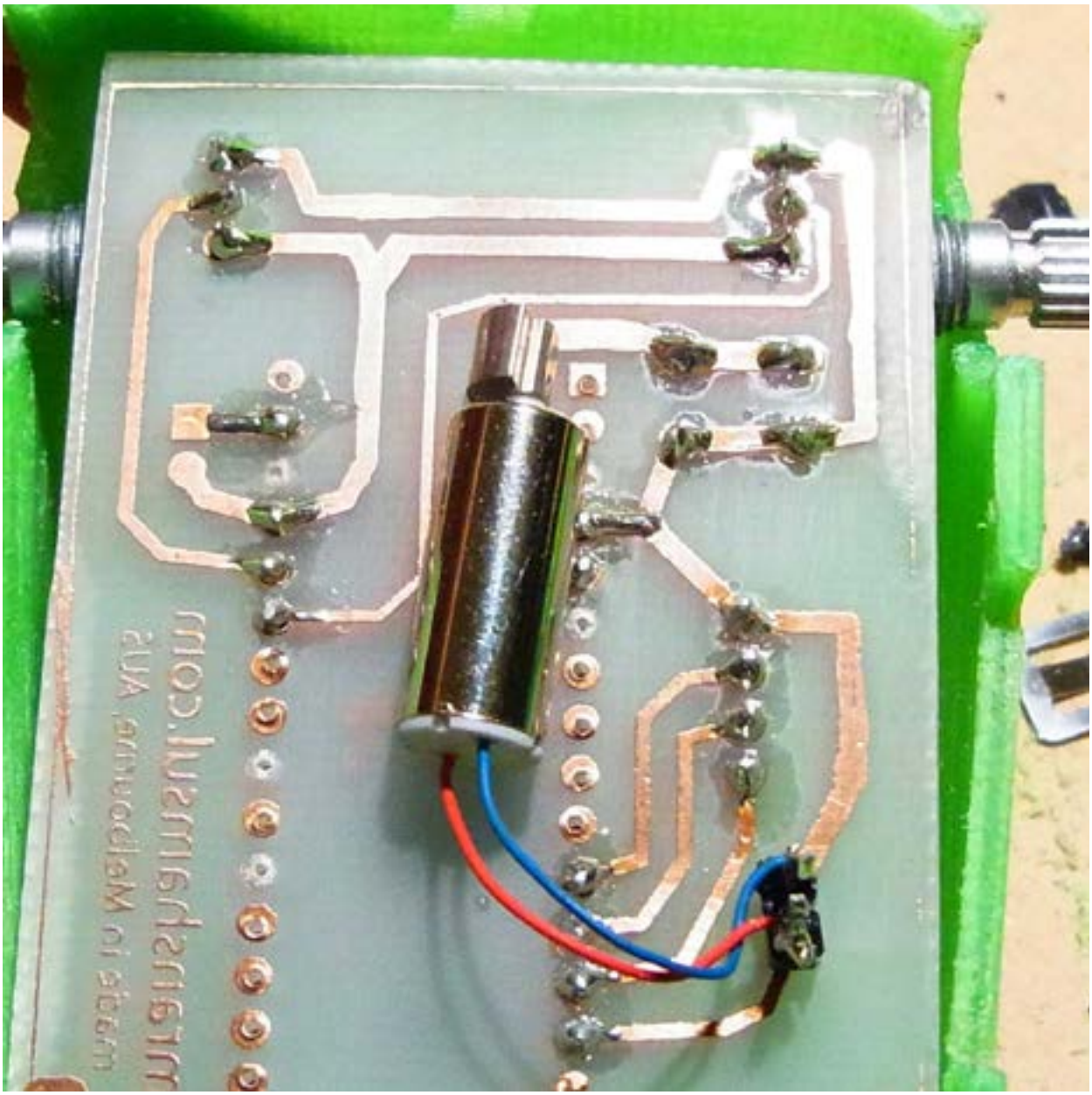
```
const int ledPin = 13;      // led connected to digital pin 13
const int knockSensor = A0; // the piezo is connected to analog pin 0
const int threshold = 100;  // threshold value to decide when the detectedsound is a knock or not

// these variables will change:
int sensorReading = 0;      // variable to store the value read from the sensor pin
int ledState = LOW;         // variable used to store the last LED status, to toggle the light

void setup() {
    pinMode(ledPin, OUTPUT); // declare the ledPin as as OUTPUT
    Serial.begin(9600);       // use the serial port
}

void loop() {  // read the sensor and store it in the variable sensorReading:
    sensorReading = analogRead(knockSensor);

    // if the sensor reading is greater than the threshold:
    if (sensorReading >= threshold) {
    // toggle the status of the ledPin:
    ledState = !ledState;
    // update the LED pin itself:
    digitalWrite(ledPin, ledState);
    // send the string “Knock!” back to the computer, followed by newline
    Serial.println(“Knock!”);
}

delay(100);  // delay to avoid overloading the serial port buffer
}
```

RELATED CHAPTERS

1.6 Ultrasonic Distance Sensor

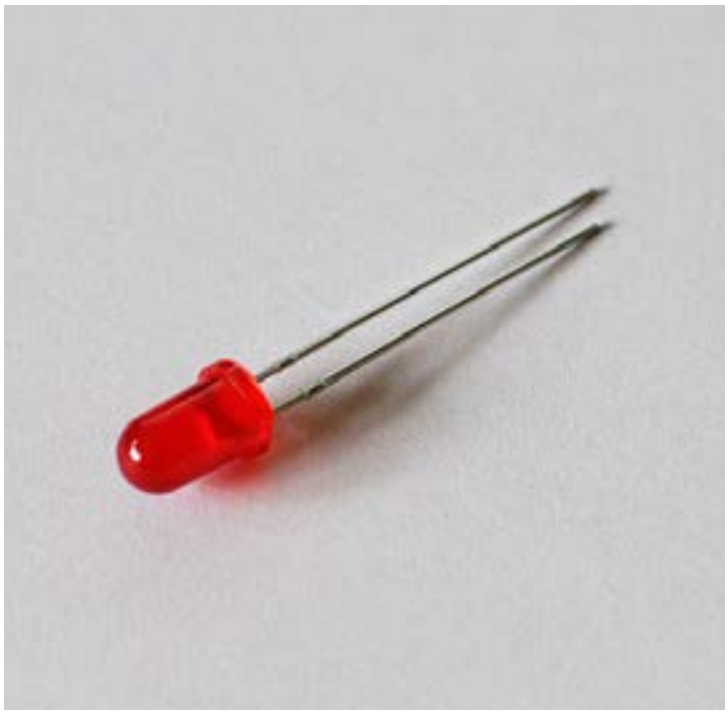
2.5 Piezo Buzzer Sound

2. OUTPUT CASE EXAMPLES



Shamsul, Imran 2012

2.1 SINGLE LED



Shamsul, Imran. 2014.

SUMMARY

A light-emitting diode (LED) is a two-terminal semiconductor, forward biased diode that relies on current flow from positive (anode) to negative (cathode) to create electroluminescent light. LED's require a resistor in series before it to limit the current flowing into it. This prevents burnout of the semiconducting material in the LED. Instructions on how to do that can be found here.

LED's are very practical electronic components as they are small in size, fast switching, lower in energy consumption, and are relatively more physically robust than incandescent light sources. As a result they are used in a diverse range applications including automotive/ aviation/advertising/general/traffic signals lighting; backlighting for televisions and mobile phones; and infra-red transmitters in remote controls.

They can be found in the colour red, orange, amber, yellow, green, blue and white. Bi/tri-colour LEDs are also available at a greater cost than single colour.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia
- Simple LED info
- More details on LEDs

Common applications relevant to industrial design.

- Samsung LCD Panel
- Philips LED Lamps
- Car Head lights

Examples of interfacing with other electronics, incl. tutorials.

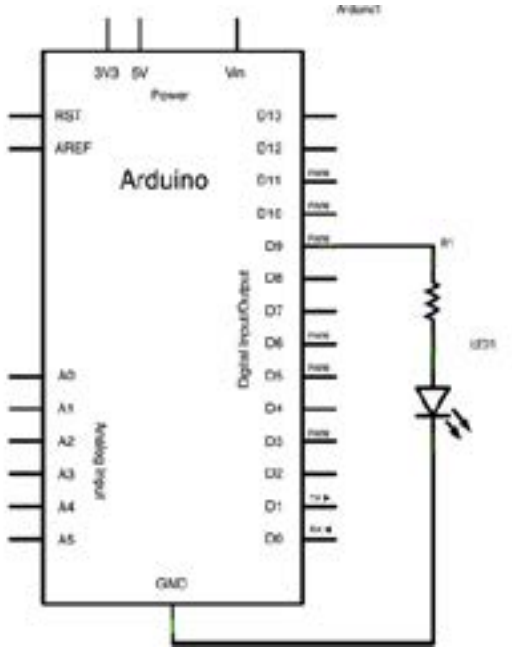
- Resistor Value Calculator
- Adafruit learn: LED
- Adafruit: Adjusting Brightness

Arduino software examples, incl. elaboration.

- LED Fade tutorial
- LED Blink Tutorial
- LED Cube Tutorial

Forums covering issues regarding implementation.

- Arduino Forum
- Candle Power Forums
- Budget Light Forum

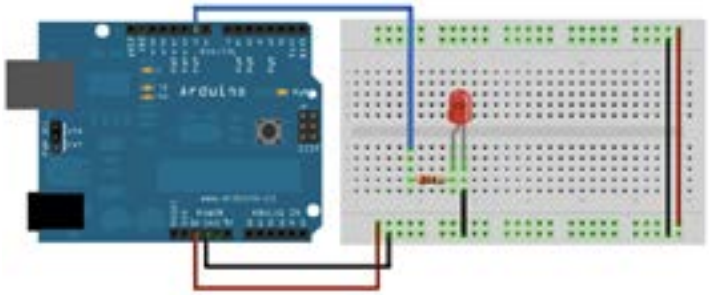


CIRCUIT DIAGRAM

This circuit diagram shows that you put a resistor in series with an LED between the digital pin no. 9 and the ground. It is important that you use a digital pin with pulse width modulation or PWM. This means you can declare any of the pins with PWM. If you'd like to change which pin to plug the resistor and LED combination into. Review the code for this on the next page.

For elaboration check out this web page.

Image source: http://arduino.cc/en/Tutorial/Fade



FRITZING DIAGRAM

This fritzing diagram shows that you use a breadboard to connect the digital pin 9 to the resistor and then the resistor to the LED. The circuit is grounded to the ground rail on the bread board. It is not necessary to connect the 5V pin to the bread board as shown in the diagram as the power comes directly from a digital pin with PWM. Review the code for this on the next page.

For elaboration check out this web page.

Image source: http://arduino.cc/en/Tutorial/Fade

CODE EXAMPLE

This code fades the LED from off (0) to on (255) back to off (0) at the end of the loop. This is done by controlling the voltage amount being pushed through the digital pin with pulse-width modulation and the analogWrite function.

For elaboration check out this web page.

```
/*
 Fade

 This example shows how to fade an LED on pin 9
 using the analogWrite() function.

 This example code is in the public domain.
 */

int led = 9;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup()  {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

RELATED CHAPTERS

2.2 Bipolar Junction Transistors

2.2 BIPOLAR JUNCTION TRANSISTORS

NPN Transistor - Sparkfun.com. 2014.

SUMMARY

A Bipolar Junction Transistor (BJT) is a sandwich of three semiconducting materials, with the outer layers made of the same material and the inner made from a different material. Much like the LED [link], a BJT is a combination of to P-N junctions. P-type semiconductors have positively charged holes while N-type semiconductors have negatively charged electrons. They can be found as either a PNP or NPN transistor and are used as either a switch or amplifier. The regions of a BJT are called 'emitter', 'base' and 'collector'. A BJT is not symmetrically doped, meaning it is not advised to reverse the polarity so to reverse the function, as the emitter is designed to inject and the collector is lightly doped to handle larger voltage swings. NPN and PNP transistors are similar in function but have differing polarities and subsequently actions. NPN transistors switch on as voltage is increased to the base and PNP transistors switch off as voltage is increased to the base. BJTs can also be found in integrated circuits when large numbers of them are combined and in solid state electronic amplifiers.

NPN vs PNP / Wikipedia

LINKS / REFERENCES

Functionality and principles of operation.

- Intro to bipolar transistors
- MAKE: The Transistor
- Transistor

Common applications relevant to industrial design.

- Solid state amplifiers
- Integrated circuits
- BJT uses

Examples of interfacing with other electronics, incl. tutorials.

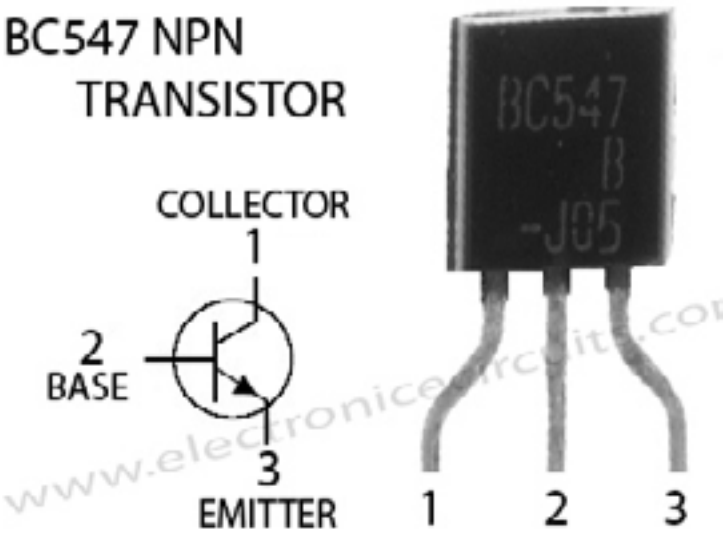
- How to use a BJT
- PNP vs. NPN transistors
- Motor and transistors

Arduino software examples, incl. elaboration.

- Multiplex with transistors
- NPN transistors
- PNP transistors

Forums covering issues regarding implementation.

- Arduino forum
- Questions about transistors



TRANSISTOR DIAGRAM

When looking at the flat side of the NPN transistor the following numbers means:

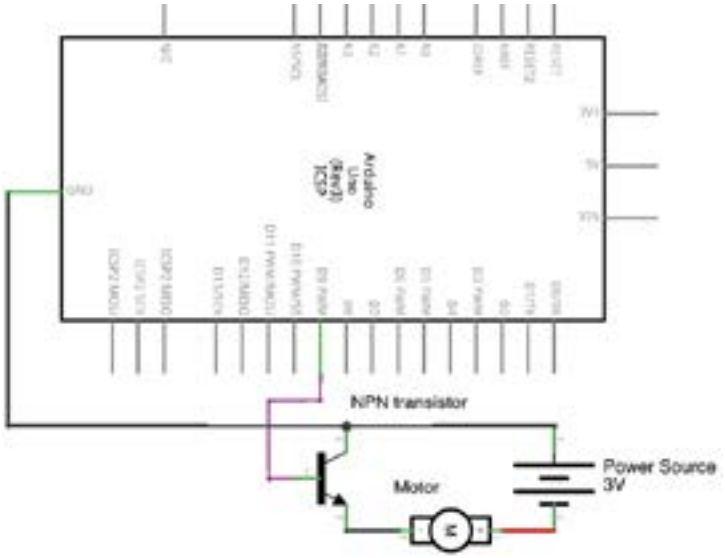
1. Collector - wired to the ground of the self powered motor.

2. Base - connects to the digital pin on the arduino.

3. Emitter - wired to the arduino ground (Gnd) pin.

For elaboration check out the tutorial web page.

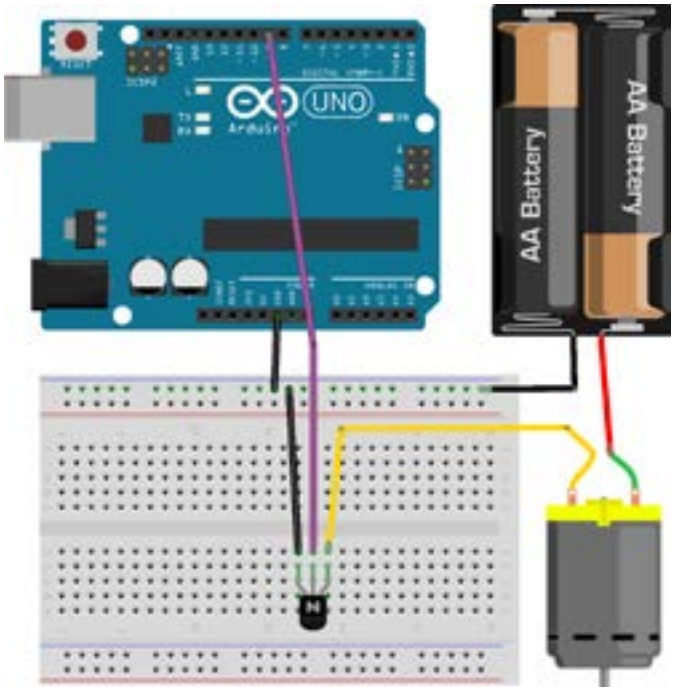
Image source: Quarkstream



CIRCUIT DIAGRAM

This circuit diagram shows that you have to connect the ground of the battery and the emitter pin of the transistor to the ground pin of the arduino. The positive of the power source must be connected to the positive of the motor as it needs to be self powered. Connect the collector pin of the transistor to the negative of the motor. The base of the transistor must be connected to an arduino digital pin, in this case, digital pin 9 (purple). Review the code for this on the next page.

For elaboration check out the tutorial web page.



FRITZING DIAGRAM

This diagram shows that the middle pin of the NPN transistor [Base] is connected to the digital output pin no. 9 (purple lead). When a voltage is applied to a base it allows current to pass from collector to emitter switching on the device. The battery positive must be connected to the motor, and the ground connected to the ground pin, meaning the battery and transistor have a common ground. This can be seen by the ground rail being highlighted in green. Review the code for this on the next page.

For elaboration check out the tutorial web page.

CODE EXAMPLE

Use this code to pulse the motor at one second intervals. Use pin 9 on the Arduino. Alternatively if you'd like to use a different pin on your arduino change the number at the statement: "int transistorPin = #;" If you'd like to change the delay between intervals adjust the value in the "delay();" syntax.

The motor is switched on by the "digitalWrite(transistorPin, HIGH);" line. 'HIGH' means there is 5V being passed through the pin, effectively switching on the motor. When the value is LOW there is 0V being sent to the pin.

For elaboration on this example click here.

Click on the links to learn more about digital pins, the digitalWrite syntax and constants.

```
int transistorPin = 9;

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

void loop()
{
 digitalWrite(transistorPin, HIGH);
 delay(1000);
 digitalWrite(transistorPin, LOW);
 delay(1000);

}
```

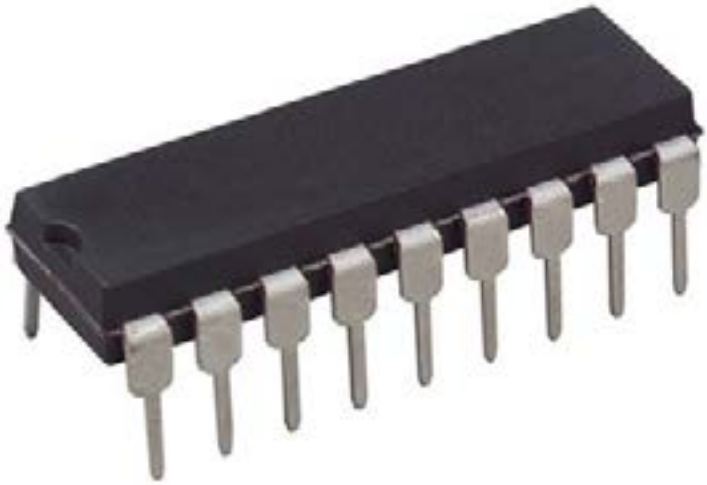
RELATED CHAPTERS

2.1 Single LED

2.3 Transistor Array

2.4 Servo motor

2.3 TRANSISTOR ARRAY



Major Brands ULN2803A Darlington Transistor NPN. Jameco Electronics 2014

SUMMARY

A transistor array can also be called an integrated circuit (IC), as it a group of transistors within a single semiconductor. It is arranged in such a way that they are connected to each other so to function independently or in a network. They are commonly known as a 'Darlington Array' as they consist of multiple Darlington Pairs (two bipolar transistors [link[), allowing them to manage higher levels of current and voltage depending on type and application. Darlington pairs and array's are commonly used to control DC-motor's speed.

LINKS / REFERENCES

Functionality and principles of operation.

- What is a transistor array?
- Wikipedia: transistor array
- What are integrated circuits?

Common applications relevant to industrial design.

- Transistor radio
- Darlington pair speed controller

Examples of interfacing with other electronics, incl. tutorials.

- UNL2803 Darlington array
- Driving a stepper motor
- Driving an 8x8 LED display

Arduino software examples, incl. elaboration.

- Stepper motor knob
- Expanding Arduino I/Os

Forums covering issues regarding implementation.

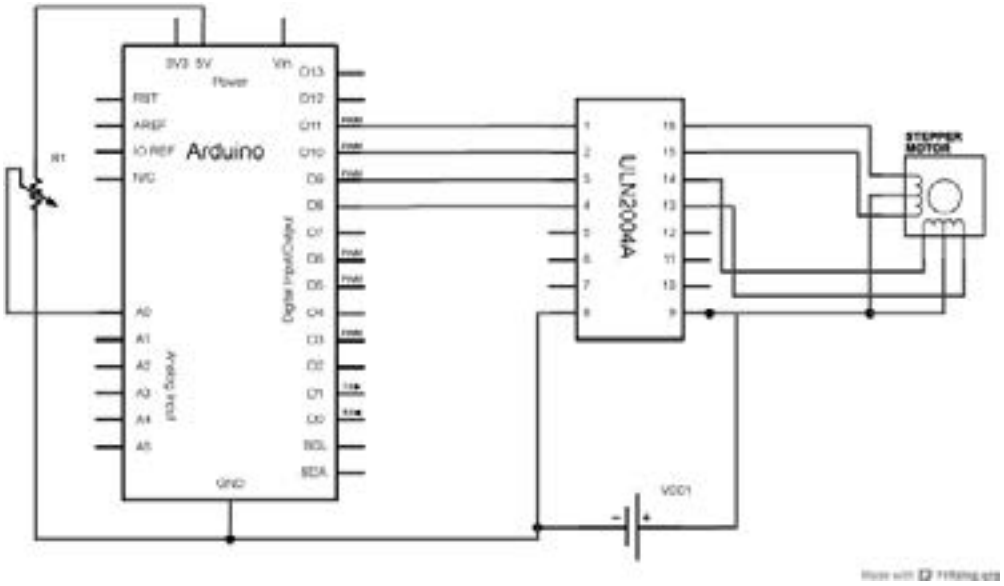
- Arduino forum
- Transistor arrays
- eLABZ forum

CIRCUIT DIAGRAM

This circuit diagram is for a unipolar stepper motor that uses a darlington array to connect it to the arduino. You will notice that the positive leads of the motor are connected together with the power source. This is run through the darlington array. Digital pins 8 to 11 are connected to the motor through the darlington array also. Allowing for better power management protecting the arduino as the motor draws more current than it can manage. Review the code for this on the next page.

For the full tutorial click here.

Image source: http://arduino.cc/en/Tutorial/MotorKnob



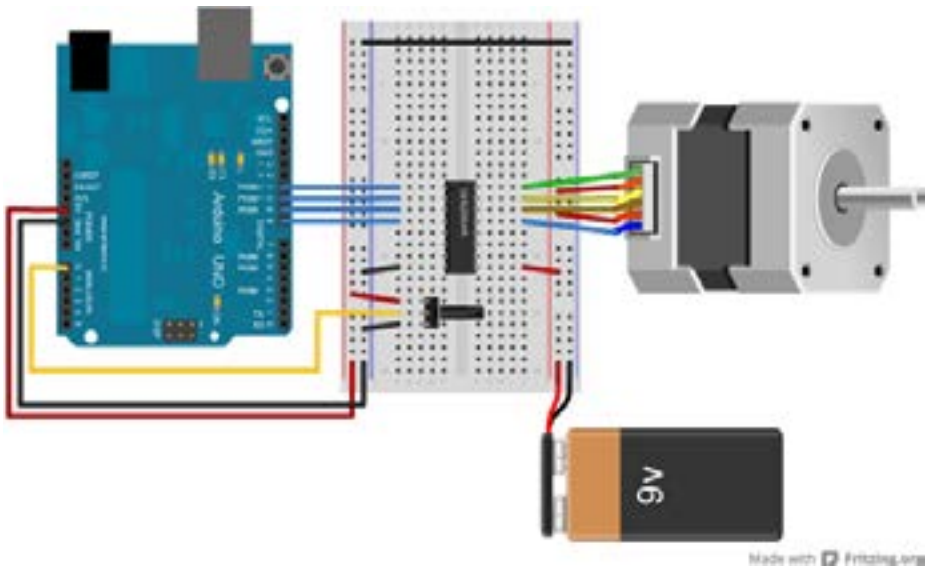
FRITZING DIAGRAM

This fritzing diagram shows that the unipolar stepper motor is connected to the arduino via a darlington array on a breadboard. It is a lot easier to visualise the layout of all the components. Pay attention to how the battery is connected to the motor, darlington array and arduino. It is important to find the relevant data sheet on the darlington array you decide to use for this tutorial.

Review the code for this on the next page.

For the full tutorial click here.

Image source: http://arduino.cc/en/Tutorial/MotorKnob



CODE EXAMPLE

Arduino stepper library must be installed first to use this code, find it here. This code can be used on either unipolar or bipolar stepper motors, note that only the unipolar motor uses a darlington array.

This code uses a potentiometer to adjust the position of the stepper motor by using the analogRead function.

For the full tutorial click here.

```
// MotorKnob
// A stepper motor follows the turns of a potentiometer
//  (or other sensor) on analog input 0.
// http://www.arduino.cc/en/Reference/Stepper
// This example code is in the public domain.

#include <Stepper.h> //Stepper.h library

#define STEPS 100  // change this to the number of steps on your motor

// create an instance of the stepper class, specifying
// the number of steps of the motor and the pins it's
// attached to
Stepper stepper(STEPS, 8, 9, 10, 11);

int previous = 0; // the previous reading from the analog input

void setup()
{
  stepper.setSpeed(30);  // set the speed of the motor to 30 RPMs
}

void loop()
{
  int val = analogRead(0); // get the sensor value

  // move a number of steps equal to the change in the
  // sensor reading
  stepper.step(val - previous);

  previous = val; // remember the previous value of the sensor
}
```

RELATED CHAPTERS

2.4 Hoby Servomotor

2.2 Bipolar junction transistors

2.4 SERVOMOTOR

Hobby Servo Motor. Sparkfun.com. *2014*

SUMMARY

A hobby servomotor, or Servo, is a self-contained device that can be considered a rotary actuator. It has control of angular position, velocity and acceleration, allowing it to push parts of a machine.

It contains a circuit board, a small direct current motor that has a high RPM (rotations per minute) and a gearing system that increases torque and reduces the rotational speed of the output shaft. A potentiometer [link] is used to detect the position, this information is in turn sent to the control board of the servomotor. The gearing system can be made of different materials to suit the speed and load that the servomotor has to deal with.

While hobby servomotors are used in remote control vehicles and toys, larger, more powerful and precise servomotors exist for use in robotics, cnc machinery and automated manufacturing.

In the case of Arduino, a hobby servomotor is controlled using the PWM (pulse width modulation) function, meaning that a servomotor must be connected to a digital pin via its signal lead.

LINKS / REFERENCES

Functionality and principles of operation.

- Wikipedia: R.C. Servo
- Robotics Servo Motors
- Servo teardown

Common applications relevant to industrial design.

- Servo-motor uses
- Wikipedia: Servo-motor

Examples of interfacing with other electronics, incl. tutorials.

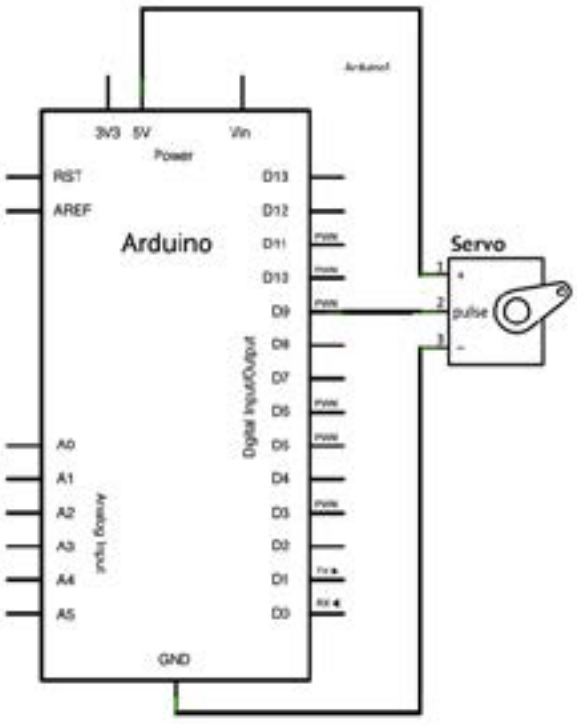
- Arduino Tutorial: Knob
- 5 Minute Tutorial: Servo

Arduino software examples, incl. elaboration.

- Arduino servo sweep
- Arduino servo library
- Simple servo control

Forums covering issues regarding implementation.

- Arduino Forum
- Servo Motor Position

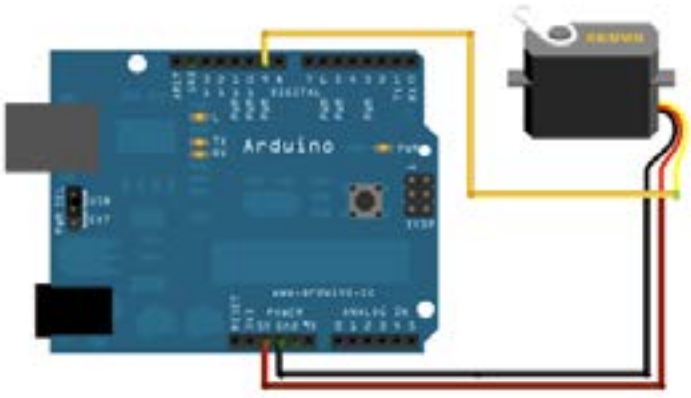


CIRCUIT DIAGRAM

This diagram shows that you connect the 5V pin to the positive input of the servo (1), the digital pin 9 to the pulse input (2) and the negative of the servo to the ground pin of the arduino. You will notice that the power does not come from the digital pin as it is used to send/receive information on servo positioning. Review the code for this on the next page.

For the full instructions visit this tutorial.

Image source: http://arduino.cc/en/Tutorial/sweep



FRITZING DIAGRAM

This fritzing diagram shows how to connect the cables from the hobby servo to the arduino. The yellow cable is the communication cable between the arduino and servo. Black is the ground lead and red is the positive lead where power is connected too. Review the code for this on the next page.

For the full instructions visit this tutorial.

Image source: http://arduino.cc/en/Tutorial/sweep

CODE EXAMPLE

This code sweeps the hobby servo within a 180 degree range. The comments in the code are quite helpful in explaining what each element does. There is the declaration before the setup() which calls a specific library, “Servo.h” - for more on libraries click here and for the servo library click here.

For the full instructions visit this tutorial.

```
// Sweep
// by BARRAGAN <http://barraganstudio.com>
// This example code is in the public domain.

#include <Servo.h>

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

int pos = 0;    // variable to store the servo position

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


void loop()
{
  for(pos = 0; pos < 180; pos += 1)  // goes from 0 degrees to 180 degrees
  {                                  // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable ‘pos’
    delay(15);                       // waits 15ms for the servo to reach the position
  }
  for(pos = 180; pos>=1; pos-=1)     // goes from 180 degrees to 0 degrees
  {
    myservo.write(pos);              // tell servo to go to position in variable ‘pos’
    delay(15);                       // waits 15ms for the servo to reach the position
  }
}
```

RELATED CHAPTERS

2.2 Bipolar Junction transistors

2.3 Transistor Array

2.5 PIEZOELECTRIC BUZZER SOUND

5V Piezo Buzzer. Jameco Electronics. *2014.*

SUMMARY

When used in Arduino applications a piezo speaker is better known as a 'piezo buzzer'. It consists of a ceramic layer glued to a metal plate and is driven by a built-in oscillating circuit. The application of an alternating electric field that stretches or compresses the metal-ceramic to produce sound depending on the frequency of the electrical signal.

LINKS / REFERENCES

Functionality and principles of operation.

- Piezo-electric sound
- TDK glossary
- How does it work?

Common applications relevant to industrial design.

- Piezo-electric speaker
- Casio watch teardown
- Digital clock

Examples of interfacing with other electronics, incl. tutorials.

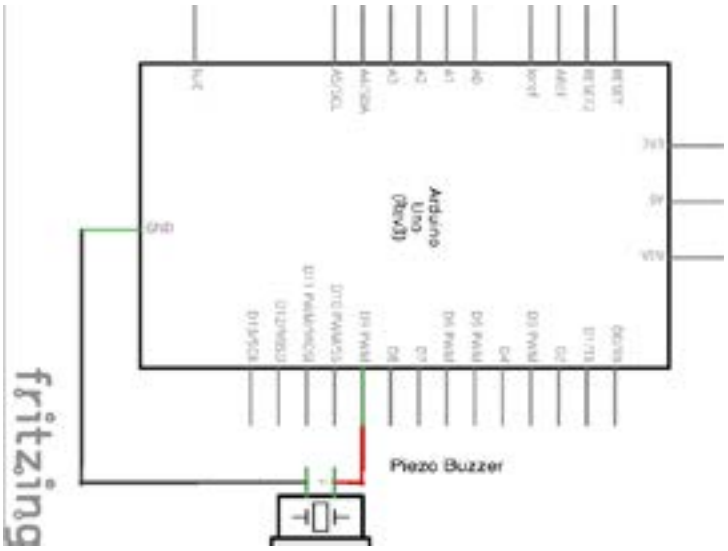
- VIA: vision aid
- Piezo buzzer inverter
- Electronic slate demo

Arduino software examples, incl. elaboration.

- Tutorial: play melody
- Custom piezo
- Play Mario Bros theme

Forums covering issues regarding implementation.

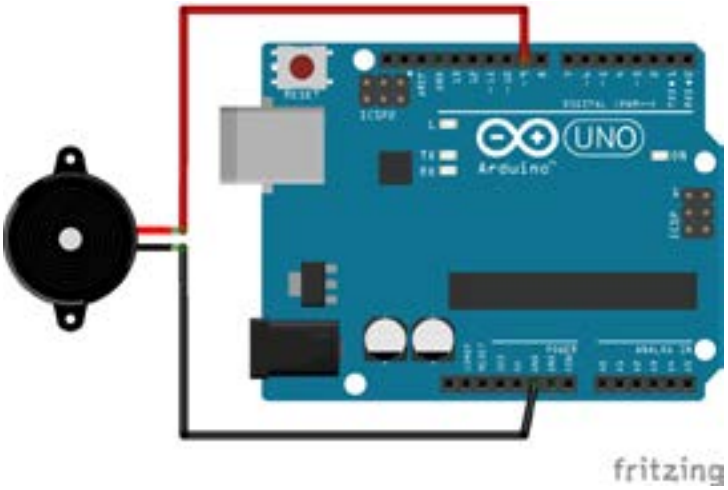
- Arduino forum
- Arduino: piezo buzzer



CIRCUIT DIAGRAM

This circuit diagram shows that you connect the positive lead (read) of the piezo buzzer to digital pin 9 of the arduino and the negative lead (black) to the ground (GND) pin of the arduino.

For the tutorial please visit this web page.



FRITZING DIAGRAM

This fritzing diagram shows how straight forward this example is to set up. Connect the positive (red) lead of the buzzer directly into the digital pin with PWM.

For the tutorial please visit this web page.

CODE EXAMPLE

As the code is too long to fit on this page, please refer to it directly from the tutorial here. The included comments on the code are very helpful to establish what is going on. It explains that you have to establish the frequency of each individual note and then how to arrange it into a distinguishable tune.

Click here for another well known tune you can also try. It establishes notes and melody in a different way from the tutorial above.

RELATED CHAPTERS

1.8 Piezo electric sensor

3. REFERENCES

Wikipedia, 2012. Available from <http://en.wikipedia.org/wiki/Potentiometer>. [03/12/2012].

Arduinology, 2012. Available from <http://arduinology.wordpress.com/tag/voltage-divider>. [03/06/2012].

CQbiz, 2012. Available from <http://www.qrbiz.com/buy_Photo-Resistor>. [03/12/2012].

Shenzen Winsun Technology. Available from <http://winsun168.en.made-in-china.com>. [03/12/2012].

Transformative design. Available from: <http://transformativedesign.pbworks.com/w/page/22424425/Arduino%20Tutorial%201%3A%20Kit%20Review>. [03/01/2013]

Sedoniatech. Available from: <http://www.sedoniatech.com.au/ping)))-ultrasonic-range-sensor.htm>. [03/01/2013]

Sparkfun. Available from: <https://www.sparkfun.com/products/252>. [03/01/2013]

Mindkits. Available from: <http://www.mindkits.co.nz/store/sensors/piezo-element>. [03/01/2013]

Arduino.cc. Availabe from: http://arduino.cc/en/Tutorial/Ping?from=Tutorial.UltrasoundSensor>. [03/01/2013]

Arduino.cc. Available from <http://arduino.cc/en/Tutorial/ADXL3xx>. [03/01/2013]

Garagelab. Available from: <http://garagelab.com/profiles/blogs/tutorial-piezo-element-as-a-sensor-with-arduino>. [03/01/2013]

Amos, S. W., S. R. Amos, and G. W. A. Dummer. Newnes Dictionary of Electronics. OXFORD: NN, 1999. Print.

Howard, David. The Penguin Dictionary of Electronics. 4th ed. London: Penguin, 2005. Print.

Platt, Charles. Encyclopedia of Electronic Components. Farnham: O'Reilly, 2012. Print.