SCMU2018/2019

ARDUINO DEVELOPMENT PLATFORM (LAB2)

Voltage divider

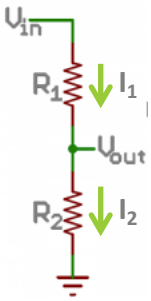
A voltage divider is a simple circuit which turns a large voltage into a smaller one, are commonly used to create reference voltages.

Using just two series resistors and an input voltage, we can create an output voltage that is a fraction of the input.

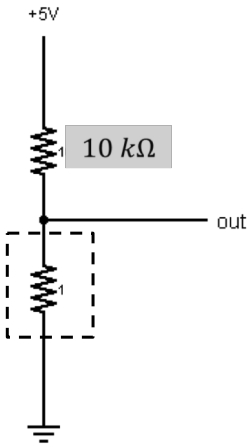
Voltage division is the result of distributing the input voltage among the components of the divider.

Voltage divider

- $V_{out} = R_2 * I_2$
- $I_2 = I_1$ and $R_{total} = R_1 + R_2$
- $I = V_{in} / (R_1+R_2)$
- $V_{out} = R_2 * (V_{in} / (R_1 + R_2))$



2 Pin Analog Sensors = var. resistor



Take two sensors: Use the Serial Monitor and find the range of input values you get for each sensor.

Analog Sensors

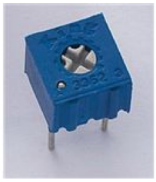
Voltage Divider Calculation

$$V_{R1} = V_{CC} \cdot \left(\frac{R_1}{\mathrm{R}_{Total}} \right)$$

$$V_{R2} = V_{CC} \cdot \left(\frac{R_2}{\mathrm{R}_{Total}} \right)$$

$$R_{Total} = R_1 + R_2$$

Potentiometer



A potentiometer is a variable resistance

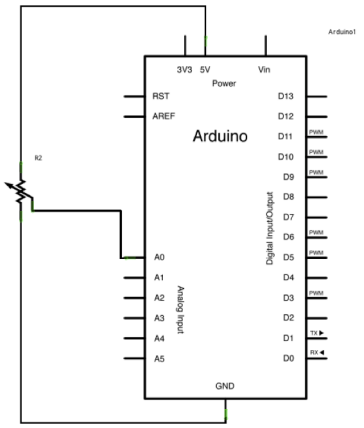
When powered with 5V, the middle pin outputs a voltage between 0V and 5V, depending on the position of the knob on the potentiometer

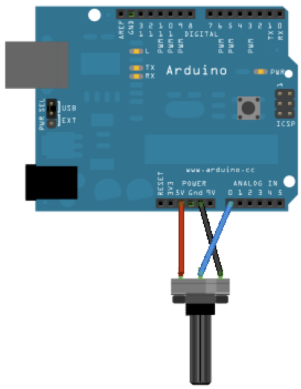
A potentiometer is a perfect demonstration of a variable voltage divider circuit

The voltage is divided proportionate to the resistance between the middle pin and the ground pin

Potentiometers have a lot of uses, essentially as a basic control (e.g. continuous speed control or position control)

Potentiometer – experiment 1





Potentiometer – experiment 1

By turning the shaft of the potentiometer, you change the amount of resistance on either side of the wiper which is connected to the center pin of the potentiometer.

This changes the voltage at the center pin.

When the resistance between the center and the side connected to 5 volts is close to zero (and the resistance on the other side is close to 10 K), the voltage at the center pin nears 5 volts.

When the resistances are reversed, the voltage at the center pin nears 0 volts, or ground.

This voltage is the analog voltage that you're reading as an input.

Potentiometer – experiment 1

Goal: Read Analog Voltage

Reads an analog input on pin 0, converts it to voltage, and prints the result to the serial monitor.

Steps:

- Initialize the serial communication, at 9600 bits/s
- In the main loop read the analog value from A0 (int value between 0 – 1023)
- Change the values from 0-1023 to a range that corresponds to the voltage the pin is reading, (float value = sensorValue * (5.0V/1023.0)
- Print the information to the serial monitor window

Potentiometer – experiment 1

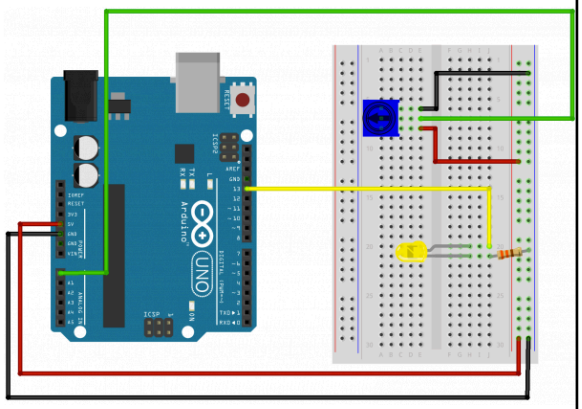
```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.print(sensorValue);
  Serial.print(" - ");
  Serial.println(voltage);
  delay(500);
}
```

Potentiometer – experiment 2

A potentiometer to control the blink rate of an LED.

Material:

- 1 - LED
- 1 - 330 Resistor
- 1 - Potentiometer 10K



Potentiometer – experiment 2

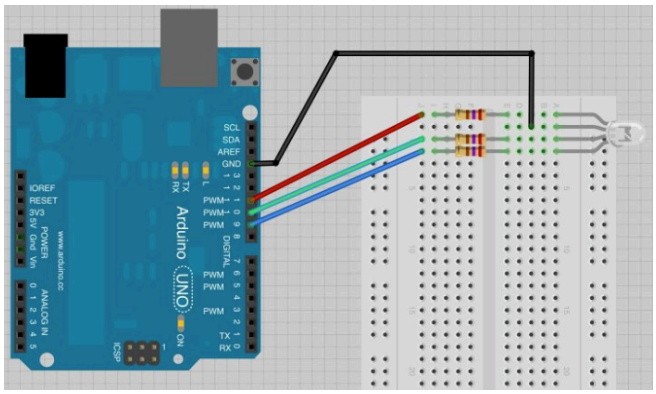
```
int sensorPin = 0;    // The potentiometer is connected to analog pin 0
int ledPin = 13;      // The LED is connected to digital pin 13

void setup() // this function runs once when the sketch starts up
{
  pinMode(ledPin, OUTPUT);
  // No need to configure sensorPin as an input.
  // The reason is that this is an "analog in" pin.
  // Since they're always used as inputs, there is no need to
  // specifically configure them.
}

void loop() // this function runs repeatedly after setup() finishes
{
  int sensorValue;
  sensorValue = analogRead(sensorPin);
  // It returns an integer number that ranges from 0 (0 Volts) to 1023 (5 Volts).

  digitalWrite(ledPin, HIGH);     // Turn the LED on
  delay(sensorValue);             // Pause for sensorValue milliseconds
  digitalWrite(ledPin, LOW);      // Turn the LED off
  delay(sensorValue);             // Pause for sensorValue milliseconds
}
```

RGB LED – experiment

Use analogWrite() to control the color of a RGB LED.

Material:

- 1 - RGB LED
- 3 - 330 Resistors



RGB LED - experiment

```
int redPin = 11;
int greenPin = 10;
int bluePin = 9;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}

void loop() {
  setColor(255, 0, 0);  // red
  delay(1000);
  setColor(0, 255, 0);  // green
  delay(1000);
  setColor(0, 0, 255);  // blue
  delay(1000);
  setColor(255, 255, 0);  // yellow
  delay(1000);
  setColor(80, 0, 80);  // purple
  delay(1000);
  setColor(0, 255, 255);  // aqua
  delay(1000);
}

void setColor(int red, int green, int blue) {
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

Photoresistor – experiment

A photoresistor changes resistance based on how much light the sensor receives.

Arduino can not directly interpret resistance (rather, it reads voltage), we need to use a voltage divider.

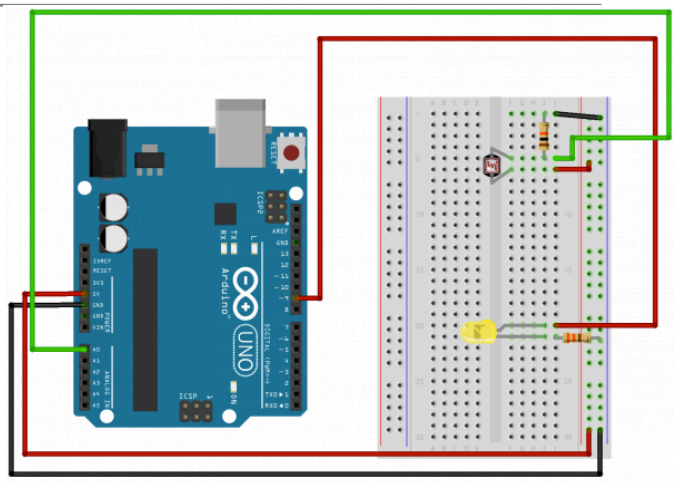
This voltage divider will output a high voltage when there is a lot of light and a low voltage when little or no light is present.

Experiment: photoresistor (light sensor) to control the brightness of a LED.

Photoresistor – experiment

Material:

- 1 - LED
- 1 - 330 Resistor
- 1 - Photoresistor
- 1 - 10K Resistor



Photoresistor – experiment

map function

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

Description

Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc.

Note that, it does not constrain values to within the range, because out-of-range values are sometimes intended and useful. The constrain() function may be used either before or after this function, if limits to the ranges are desired.

Photoresistor – experiment

map function

Parameters

value: the number to map

fromLow: the lower bound of the value's current range

fromHigh: the upper bound of the value's current range

toLow: the lower bound of the value's target range

toHigh: the upper bound of the value's target range

Returns

The mapped value.

Photoresistor – experiment

map function

Mathematic function:

```
long map(long x, long in_min, long in_max,
                                long out_min, long out_max)
{
        return (x - in_min) * (out_max - out_min) /
                                (in_max - in_min) + out_min;
}
```

Photoresistor – experiment

constrain function

constrain(x, a, b)

Description: Constrains a number to be within a range.

Parameters

x: the number to constrain, all data types

a: the lower end of the range, all data types

b: the upper end of the range, all data types

Returns

x: if x is between a and b

a: if x is less than a

b: if x is greater than b

Photoresistor – experiment

constrain function

Example

limits range of sensor values a value between 10 and 150

sensVal = constrain(sensVal, 10, 150);

Photoresistor – experiment

```
const int sensorPin = 0;
const int ledPin = 9;
int lightLevel, high = 0, low = 1023;

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

void loop()
{
  lightLevel = analogRead(sensorPin);
  manualTune();  // manually change the range from light to dark
  analogWrite(ledPin, lightLevel);
}

void manualTune()
{
  lightLevel = map(lightLevel, 0, 1023, 0, 255);
  lightLevel = constrain(lightLevel, 0, 255);
}
```

NTC Thermistor– experiment

Thermistor is a special type of resistance that changes values depending on the ambient temperature.

Thermistor PTC (Positive Temperature Coefficient): this type of thermistor has a positive temperature coefficient -> its resistance increments as temperature increases.

Thermistor NTC (Negative Temperature Coefficient): this type has a negative temperature coefficient -> its resistance decrements as temperature increases.

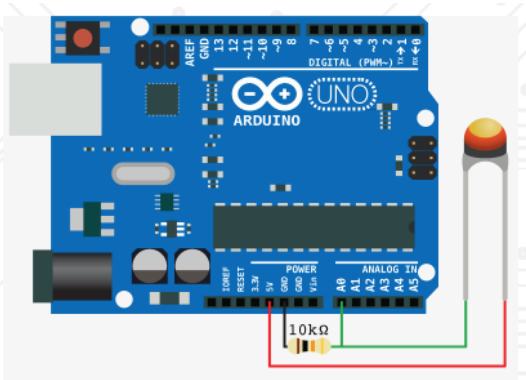
The nominal value of the thermistor is normally 25 degrees Celsius, in this case we will use a 10K thermistor. It can read temperatures between -40 and +125 degrees Celsius.

NTC Thermistor– experiment

Calculate the temperature based on the value of the resistance of a thermistor

Material:

- 1 - NTC 10K
- 1 - 10K Resistor



NTC Thermistor– experiment

The basic thermistor circuit uses a voltage divider to read a voltage at a point between two known resistors.

Analog to Digital Converter (ADC) reading = (Voltage at Pin / 5V) * 1023

The Steinhart-Hart equation used to convert the linear ADC voltage readings to temperature.

$$\frac{1}{T} = A + B\ln(R) + C(\ln(R))^3$$

Where:
A =0.001129148
B = 0.000234125
C = 8.76741E-08
T = temp in Kelvins
R = resistance in Ohms

```
#include <math.h>
#define ThermistorPIN 0                 // Analog Pin 0 set to the measured Vcc.
float pad = 10000;//9850;               // balance/pad resistor value, set this to
                                        // the measured resistance of your pad resistor
float thermr = 10000;                   // thermistor nominal resistance

float Thermistor(int RawADC) {
  long Resistance;
  float Temp;  // Dual-Purpose variable to save space.

  Resistance=pad*((1024.0 / RawADC) - 1);
  Temp = log(Resistance); // Saving the Log(resistance) so not to calculate  it 4 times later
  Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 * Temp * Temp * Temp));
  Temp = Temp - 273.15;  // Convert Kelvin to Celsius
  return Temp;           // Return the Temperature
}

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

void loop() {
  float temp;
  temp=Thermistor(analogRead(ThermistorPIN));       // read ADC and  convert it to Celsius
  Serial.print("Celsius: ");
  Serial.print(temp,1);                             // display Celsius
  Serial.println("");
  delay(5000);                                      // Delay a bit...
}
```

PIR - Pyroelectric InfraRed Sensors

The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR.

The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor).

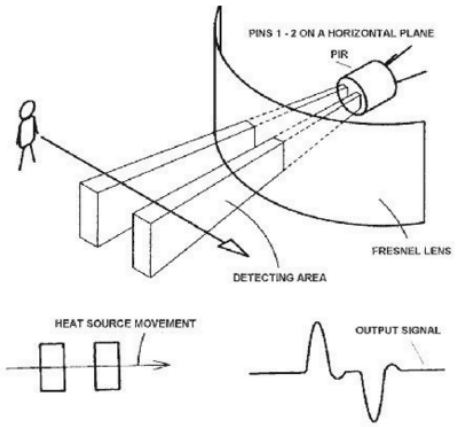
When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors.

PIR - Pyroelectric InfraRed Sensors

When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves.

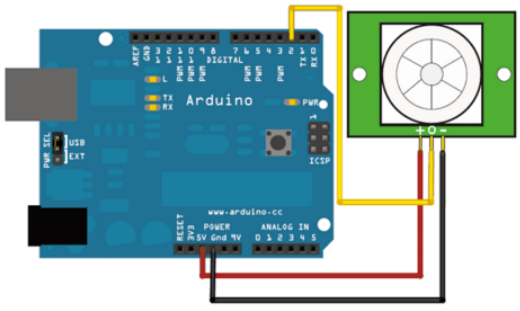
When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

PIR - Pyroelectric InfraRed Sensors



PIR - experiment

Power the PIR with 5V and connect ground to ground. Connect the output to a digital pin.



PIR - experiment (code1)

```
int ledPin = 13;     // choose the pin for the LED
int inputPin = 2;    // choose the input pin (PIR sensor)
int pirState = LOW; // we start, assuming no motion detected
int val = 0;         // variable for reading the pin status

void setup() {
  pinMode(ledPin, OUTPUT);       // declare LED as output
  pinMode(inputPin, INPUT);      // declare sensor as input
  Serial.begin(9600);
}
…
```

PIR - experiment (code1)

```
void loop(){
  val = digitalRead(inputPin);        // read input value
  if (val == HIGH) {                  // check if the input is HIGH
    digitalWrite(ledPin, HIGH);       // turn LED ON
    if (pirState == LOW) {
      // we have just turned on
      Serial.println("Motion detected!");
      // We only want to print on the output change, not state
      pirState = HIGH;
    }
  } else {
    digitalWrite(ledPin, LOW); // turn LED OFF
    if (pirState == HIGH){
      // we have just turned of
      Serial.println("Motion ended!");
      // We only want to print on the output change, not state
      pirState = LOW;
    }
  }
}
```

PIR - experiment (code2)

Switches a LED according to the state of the sensors output pin. Determines the beginning and end of continuous motion sequences.
(http://playground.arduino.cc/Code/PIRsense)

The sensor's output pin goes to HIGH if motion is present. However, even if motion is present it goes to LOW from time to time, which might give the impression no motion is present. This program deals with this issue by ignoring LOW-phases shorter than a given time, assuming continuous motion is present during these phases.

PIR - experiment (code2)

```
//VARS
//the time we give the sensor to calibrate (10-60 secs according
to the datasheet)
int calibrationTime = 30;

//the time when the sensor outputs a low impulse
long unsigned int lowIn;

//the amount of milliseconds the sensor has to be low
//before we assume all motion has stopped
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;

int pirPin = 3;    //the digital pin connected to the PIR sensor's
output
int ledPin = 13;
```

PIR - experiment (code2)

```
//SETUP
void setup(){
  Serial.begin(9600);
  pinMode(pirPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(pirPin, LOW);

  //give the sensor some time to calibrate
  Serial.print("calibrating sensor ");
  for(int i = 0; i < calibrationTime; i++){
      Serial.print(".");
      delay(1000);
  }
  Serial.println(" done");
  Serial.println("SENSOR ACTIVE");
  delay(50);
}
```

PIR - experiment (code2)

```
void loop(){
     if(digitalRead(pirPin) == HIGH){
       digitalWrite(ledPin, HIGH);   //the led visualizes the
                                     //sensors output pin state
       if(lockLow){
          //makes sure we wait for a transition to LOW
          //before any further output is made:
          lockLow = false;
          Serial.println("---");
          Serial.print("motion detected at ");
          Serial.print(millis()/1000);
          Serial.println(" sec");
          delay(50);
       }
       takeLowTime = true;
     }
...
```

PIR - experiment (code2)

```
...
    if(digitalRead(pirPin) == LOW){
      digitalWrite(ledPin, LOW); //the led visualizes the sensors output pin state
      if(takeLowTime){
       lowIn = millis();        //save the time of the transition from high to LOW
       takeLowTime = false;     //make sure this is only done
                                //at the start of a LOW phase
       }
      //if the sensor is LOW for more than the given pause,
      //we assume that no more motion is going to happen
      if(!lockLow && millis() - lowIn > pause){
          //makes sure this block of code is only executed again after
          //a new motion sequence has been detected
          lockLow = true;
          Serial.print("motion ended at ");      //output
          Serial.print((millis() - pause)/1000);
          Serial.println(" sec");
          delay(50);
          }
      }
}
```

References

1. “Intro to Arduino: Zero to Prototyping in a Flash!”, from Sparkfun Electronics.
2. Arduino documentation: https://www.arduino.cc/en/Guide/HomePage
3. Sparkfun experiments:https://learn.sparkfun.com/tutorials/