101 Starter Kit Series

ARD-01

BA SIC

Learn Arduino BASICS

Tutorials

OSEPP™

www.osepp.com

Contents:

UNO Specifications P1

Tutorial 1: Loading the First Sketch P2

Tutorial 2: Controlling Digital Outputs P5

Tutorial 3: Using Digital Input P8

Tutorial 4: An LED Game P12

Tutorial 5: Building Voltage Meter P15

Tutorial 6: Using Buzzer to Play a Melody P19

Tutorial 7: Counting Down with a 7 Segment LED P23

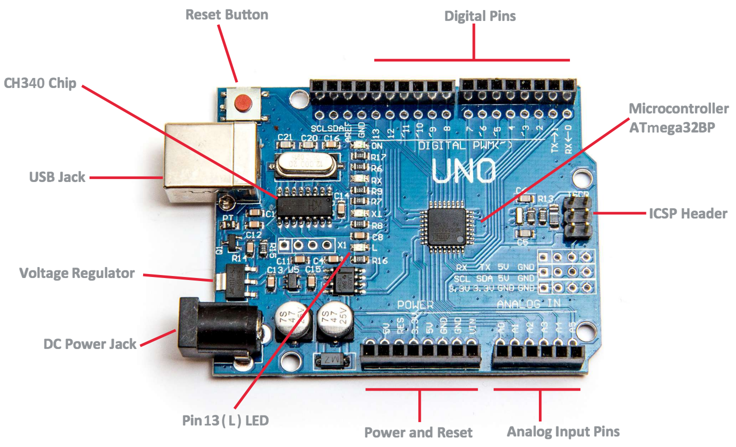
Tutorial 8: Powering the UNO Using Batteries P26

Stay Connected P28

After Sales Support P29

OSEPP™

UNO Specifications



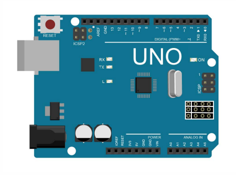
Microcontroller	**ATmega328P**
Clock Speed	16 MHz
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Operating Voltage	5V
Input Voltage	6-12 V
Digital I/O Pin Count	14 (including 6 for PWM output)
Analog Input Pin Count	6
Dimensions	2.95 x 2.13 x 0.61 inches (75.0 x 54.0 x 15.5 mm)
Power Source	USB or external DC power supply

Tutorial 1
Loading the First Sketch

In this tutorial we will show you how to load a sketch onto your UNO board. A sketch is the name that Arduino uses to refer to the code that runs on an Arduino board. We will start with a simple sketch that toggles the LED on your board.

I – Things you'll need:

You will need an UNO board, a USB cable, and a Windows PC. The UNO board will be powered by your PC through the USB cable, so we won't need a power adapter.



UNO

USB Cable

Windows PC

II - Download the Arduino Environment:

a) Download the latest Arduino environment from here:
http://arduino.cc/en/Main/Software.

b) Once the download is completed, unzip and install the Arduino IDE.

III – Connect the Board to Your PC

a) Attached the USB cable to the UNO board.

b) Attached the other end of the USB cable to your Windows PC.

c) Windows will begin its driver installation process.

IV – CH340 Serial–USB Converter Driver

The UNO board uses the CH340 USB to Serial Converter chip. Most operating systems have the CDC drivers pre–installed which means it automatically recognizes and installs the driver for the board. Therefore, you shouldn't need to install any extra software.

However, there are a wide range of operating systems out there, so if you run into driver problems you can download and install the driver.

a) Download the drivers and install:

https://www.wch-ic.com-downloads/CH341SER_ZIP.html

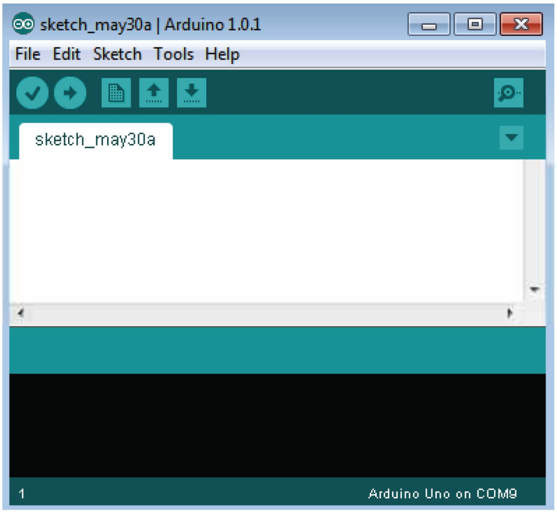
V – Download Tutorial Codes

a) Download tutorial codes from here:

https://osepp.com/wp-content/uploads/2013/07ARD101_Tutorial_Code.zip

VI – Start the Arduino Environment and Upload your Sketch

a) Go back to the arduino-x.y.z folder and double click the arduino.exe file. The below Windows will appear.



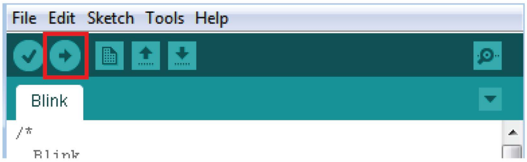
**Images are developed using Fritzing

b) Load the blinking LED example sketch by clicking "File→Examples→Basics→Blink", and another Windows with the Blink sketch will appear.

c) Select your board by clicking "Tools → Board → Arduino UNO".

d) Select the virtual serial communication port by clicking "Tools→Serial Port→COM X".

e) Upload the sketch to the UNO board by clicking the Upload button.

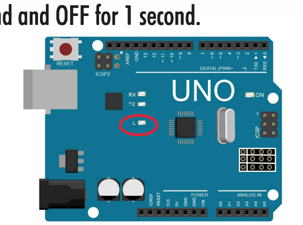


**Images are developed using Fritzing

f) When the upload is completed, you will see the “Done Uploading” message.

VII – Blinking LED Sketch

At this point, the LED labeled L on your board should be blinking ON and OFF. The LED should light ON for 1 second and OFF for 1 second.



You can experiment with the sketch and see how it alters the behavior of the LED. For instance, try changing the delay(1000) to delay (2000) or changing delay(1000) to delay(500).

Useful Note: *A typical Arduino sketch consists of two main procedures, a “setup” functions and a “loop” procedure.   The “setup” function gets executed one time only every time the board is power cycled or reset, and it is mainly used for defining the default or initial behavior. The “loop” function gets executed continuously.*

```
/* Tutorial 1
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
  This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

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

Things we have covered in this tutorial:

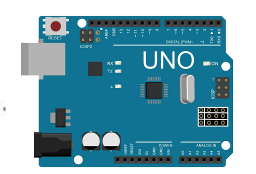
- How to setup the Arduino environment
- The CH340 USB to Serial Converter
- How to load an Arduino sketch

Tutorial 2
Controlling Digital Outputs

In this tutorial, we will show you how to use the UNO digital output signals to control an array of LEDs. We will show you how to wire up a simple LED circuit using a breadboard and how to correctly size the resistor for the LED circuit.

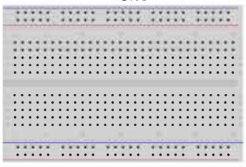
I – Things you'll need:

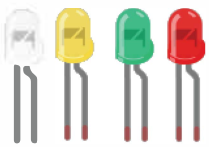
You will need an UNO board, a USB cable, a PC, a bread-board, LEDs, resistors, and jumper cables for assembling your LED circuit.

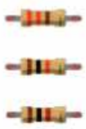

UNO

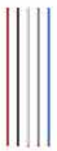
USB Cable

Windows PC


Breadboard


LEDs


Resistors


Jumper cables

II – Background Information:

The digital pins on the UNO are rated at 5V. This means when it drives a logic-1, it will measure 5V on the pin, and when it drives a logic-0, it will measure 0V. In input mode you can apply a signal as high as 5V to it without damaging it. The UNO has 14 digital pins that can be configured as input or output.

In this tutorial, we will configure 4 digital pins as outputs and use it to light up four LEDs one at a time. An LED stands for "light emitting diode", it emits light when an electrical current is flowing through it. When connecting the LED to the UNO board, we will need to put a resistor in series to limit the amount of electrical current flowing through it. If you put in a resistor with a very large resistance, the amount of current flowing through will be very small, and the LED will not be very bright. If you put in a resistor with a very small resistance (or not putting in a resistor), the amount of current flowing through the LED will be very high and you could blow out the LED.

To size the resistor, you will need to look at the LED data sheet and find out how much current (If) it can tolerate and what its forward voltage (Vf) is. Typically, a small LED will be able to handle up to 20mA of forward current and a voltage drop of 2V. Knowing this, we can use ohm's law to determine the size of the resistor we should use.

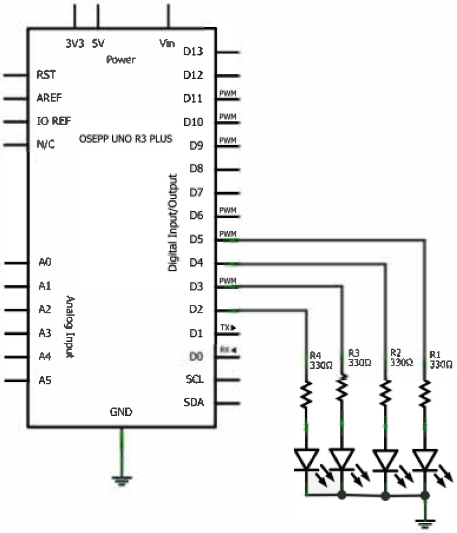
We know that the UNO will output 5V at logic 1, we can use this equation to find out the minimum resistor value:

$V = I \times R$
$R = V / I$
$R = (5V - Vf) / I = (5V - 2V) / 20\ mA = 150\ ohm$

150 ohm is the smallest resistor we can use without damaging the LED.

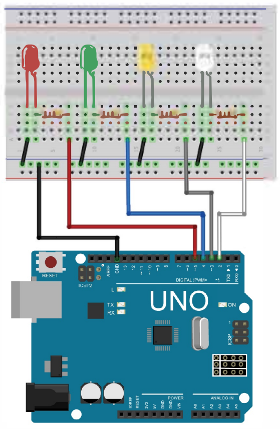
Question: If we use a 330ohm resistor, what will be the current?

<u>III – Schematic Diagram:</u>
On the right is the schematic diagram. We will connect a 330 ohm resistor in series to each LED. We will use digital pin 2, 3, 4, and 5.



IV – Wiring the Breadboard:

Let's wire up the breadboard. The lead of the positive terminal of the LED is longer than the negative terminal. Be sure to wire the positive terminal to the resistor and the negative terminal to ground.



V – Writing the Sketch:

Start a new project and copy and paste the below sketch into the Arduino environment. This sketch will setup digital pin 2, 3, 4, and 5 as output using the pinMode function and will toggle each pin one at a time using the digitalWrite function.

```
/*
Tutorial 2 - Digital Output
*/

int LED0  = 2;    // Use digital pin 2 to drive the white LED
int LED1  = 3;    // Use digital pin 3 to drive the yellow LED
int LED2  = 4;    // Use digital pin 4 to drive the green LED
int LED3  = 5;    // Use digital pin 5 to drive the red LED

void setup() {
  // initialize digital pin 2 to 5 as output:
  pinMode(LED0, OUTPUT);
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
}

void loop(){
  // Toggle each LED at a time with a 500ms delay
  digitalWrite(LED0, HIGH);
  delay(500);
  digitalWrite(LED0, LOW);
  delay(500);
  digitalWrite(LED1, HIGH);
  delay(500);
  digitalWrite(LED1, LOW);
  delay(500);
  digitalWrite(LED2, HIGH);
  delay(500);
  digitalWrite(LED2, LOW);
  delay(500);
  digitalWrite(LED3, HIGH);
  delay(500);
  digitalWrite(LED3, LOW);
  delay(500);
}
```

Once the sketch is loaded onto the UNO, you should see each LED light up one at a time. From white, yellow, green, red, and then back to white.

Things we have covered in this tutorial:

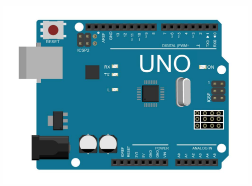
- The digital output pin
- How to size a resistor when driving an LED

Tutorial 3
Using Digital Input

In this tutorial we will show you how to read a digital signal using the UNO and report the signal state using the serial port monitor. We will show you how to connect a simple input circuit using a tact switch and a pull up resistor.

I – Things you'll need:

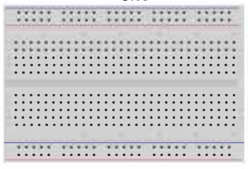
You will need an UNO board, a USB cable, a PC, a bread-board, a tact switch, a 10kohm resistors, and jumper cables for assembling your circuit.



UNO

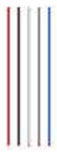
USB Cable

Windows PC


Breadboard

Tact Switch

1 x 10 kohm Resistors


Jumper cables

II – Background Information:

In the previous tutorial we mention that the digital pins of the UNO board are rated at 5V. This means you can apply a signal as high as 5V without damaging it.

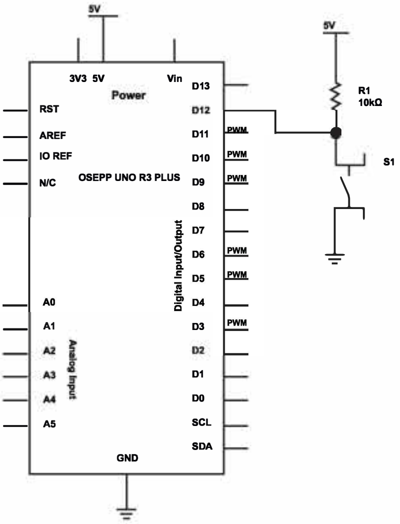
In this tutorial we will experiment with the digital input pin. To input a logic-1, we will use a 10k ohm pull up resistor connected to the 5V power supply.   To input a logic-0, we will use a tact switch to short the digital input pin to ground.

By default, the tact switch is an open circuit, so the electrical current would flow from the 5V power supply to the D12 pin. The amount of current flowing into the digital pin is very low because the input pin is a CMOS input. (This current is referred to as a Cgs leakage current and typically in the μA range.) Because there is very little current flow, the voltage drop across the 10kohm resistor will be close to 0V. If you measure the voltage at the input pin, you should see a 5V. When the tact switch is pressed, a short circuit will form and the digital input pin will be grounded.   If you measure the voltage on the input when the switch is pressed you should see 0V.

Useful Note: *The resistor in this circuit is to prevent the 5V from shorting to ground when the tact switch is pressed. With a 10kohm resistor you will only be sinking 0.5mA of current from the power supply. If you use a resistor with very low resistance, you could over load and damage the power supply.*

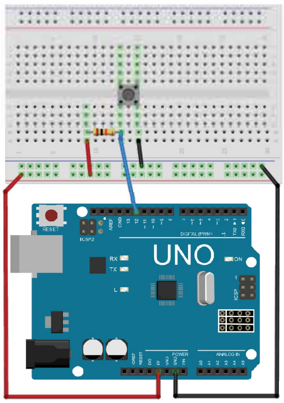
III – Schematic Diagram:

On the right is the schematic diagram. We will use digital pin 12.



IV – Wiring the Breadboard:

Let's wire up the breadboard. Be sure to pay attention to the orientation f the tact switch. If the orientation is incorrect, the input signal will always be grounded.



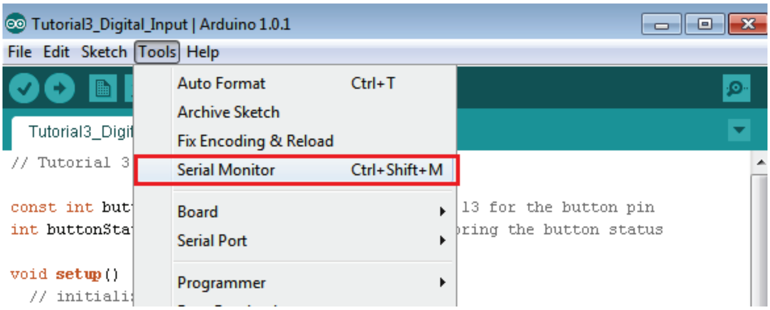
**Images aredeveloped using Fritzing

V – Writing the Sketch:

Like in the previous tutorial, start a new project and copy and paste the code below into the Arduino environment. This sketch will setup digital pin 12 as input using the pinMode function and will uses the digitalRead function to read the state of the pin. The sketch will poll the pin state once every second. In addition, the sketch enabled the serial port and will output the pin state on the serial port once every second.

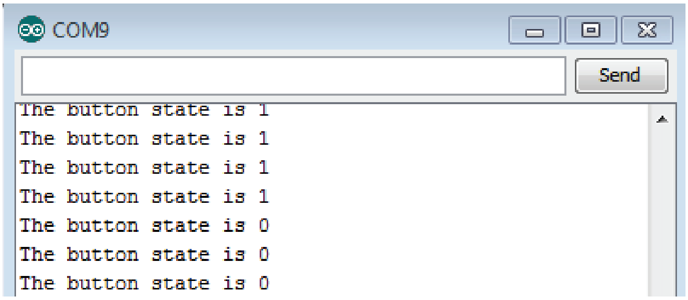
```
/*
Tutorial 3 - Digital Input
*/

const int buttonPin = 12;     // Use digital pin 12 for the button pin
int buttonState = 0;          // variable for storing the button status

void setup() {
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);

  // initialize the serial port;
  Serial.begin(9600);  // start serial for output
}

void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // Output button state
  Serial.print("The button state is ");
  Serial.println(buttonState);

  // Delay 1000ms
  delay(1000);
}
```

To read the output from the serial port: Click Tools –> Serial Monitor (or hit Ctrl + Shift + M).



```
// Tutorial 3

const int but                    13 for the button pin
int buttonSta                    oring the button status

void setup()
  // initiali
```

You should see the default button state is 1. When the button is pressed, the button state is 0.



```
The button state is 1
The button state is 1
The button state is 1
The button state is 1
The button state is 0
The button state is 0
The button state is 0
```

**Images are developed using Fritzing

Things we have covered in this tutorial:

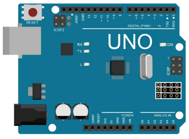
- Digital input pin
- Using serial monitor

Tutorial 4
An LED Game

In this tutorial we will show how to combine the use of digital input and output pins to build a simple LED game with the UNO board. In this game we will blink each LED in a loop from white, to yellow, to green, to red, and then back to white. The goal of this game is to stop the LED at the exact moment when the green LED is lit by pressing the tact switch. If you get it right the game will speed up and the difficulty will increase.

I – Things you'll need:

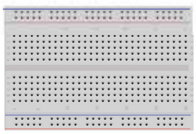
You will need an UNO board, a USB cable, a PC, a bread-board, LEDs, resistors, jumper cables, and a tact switch for assembling the circuit.



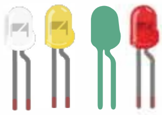
UNO

USB Cable

Windows PC



Breadboard



LEDs

Tact Switch

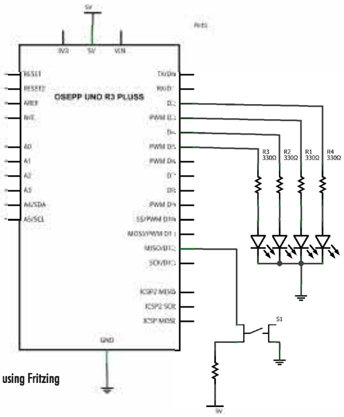
4x 330ohm + 1x 10kohm Resistors

Jumper cables

**Images are developed using Fritzing

II – Schematic Diagram:

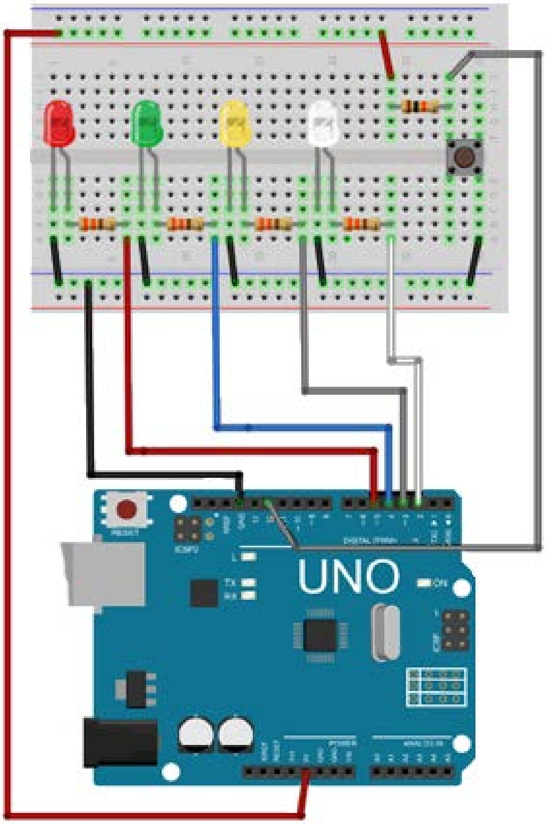
On the right is the schematic diagram. We will use digital pin 12 for input, digital pin 2, 3, 4, and 5 for output.



**Images are developed using Fritzing

III – Wiring the Breadboard

Let's wire up the breadboard.



**Images are developed using Fritzing

IV – Writing the Sketch:

Start a new project and copy and paste the next page sketch into the Arduino environment. After the sketch is loaded, you will see the LEDs blink one at a time. Each LED will light up for 200ms and turn off for 200ms at the start of the game. If you manage to hit the button at the exact moment when the green LED is lit, the game will speed up by 20ms. At the end, the LEDs will toggle so fast that all four LEDs will light up at the same time. To restart the game, hit the reset button on the UNO board.

```
/*
  Tutorial 4 - Digital Input and Output Game
  In this game, the LED will loop from white, yellow, green, red
  then back to white.  The goal is to press the push button at the exact
  moment when the green LED is ON. Each time you got it right, the LED
  will speed up and the difficulty will increase.
*/

int currentLED = 2;
int delayValue = 200;

void setup() {
  // initialize digital pin 12 as input;
  pinMode(12, INPUT);   // button input

  // initialize digital pin 2 to 5 as output:
  pinMode(2, OUTPUT);   // white LED
  pinMode(3, OUTPUT);   // yellow LED
  pinMode(4, OUTPUT);   // green LED
  pinMode(5, OUTPUT);   // red LED
}

int checkInput() {
  if (digitalRead(12) == 0) {
     return 1;
  } else {
    return 0;
  }
}

void loop(){
  // Check if the button is press at the right moment
  if (digitalRead(12) == 0) {
     if (currentLED == 4) {
       // Blink the correct (green) LED
       digitalWrite(4, HIGH);
       delay(200);
       digitalWrite(4, LOW);
       delay(200);
       digitalWrite(4, HIGH);
       delay(200);
       digitalWrite(4, LOW);
       delay(200);

       // Speed up the LEDs
       delayValue = delayValue - 20;

    } else {
       // Blink the wrong LED
       digitalWrite(currentLED, HIGH);
       delay(200);
       digitalWrite(currentLED, LOW);
       delay(200);
       digitalWrite(currentLED, HIGH);
       delay(200);
       digitalWrite(currentLED, LOW);
       delay(200);
    }
  }
  // Loop LED from white --> yellow --> green --> red
  digitalWrite(currentLED, HIGH);
  delay(delayValue);
  digitalWrite(currentLED, LOW);
  delay(delayValue);
  currentLED = currentLED + 1;
  if (currentLED > 5) {
     currentLED = 2;
  }
}
```

<u>Things we have covered in this tutorial:</u>

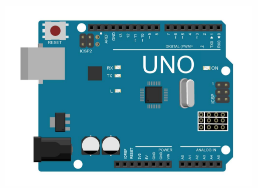
- Using digital input and output pin together.

Tutorial 5
Building Voltage Meter

In this tutorial we will show you how use the analog pins on the UNO as a voltage meter. We will show you how to build a voltage divider circuit using a fixed resistor and a potentiometer. We will output the voltage on the serial port monitor.

I – Things you'll need:

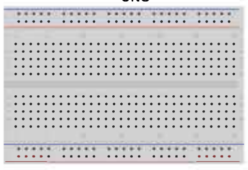
You will need an UNO board, a USB cable, a PC, a bread-board, a 1 kohm potentiometer, a 1kohm resistors, and jumper cables for assembling your circuit.



UNO

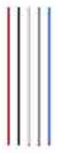
USB Cable

Windows PC


Breadboard

1kohm potentiometer

1x 1kohm Resistor


Jumper cables

**Images are developed using Fritzing

II – Background Information:

The analog pins on the UNO read the voltage applied and convert it into a digital format that can be processed by the microcontroller. We will feed an adjustable voltage into an analog pin using a simple voltage divider circuit. We will use a fixed 1k ohm resistor as the upper half of the divider and a 1k ohm potentiometer as the lower half. The resistance of the potentiometer will sweep from 0 ohm when adjust to one extreme, and to 1k ohm when adjusted to the other extreme. With this circuit, we will be able to feed in a voltage from 0V to 2.5V into the board. The voltage we will feed into the analog sensor can be calculated by:

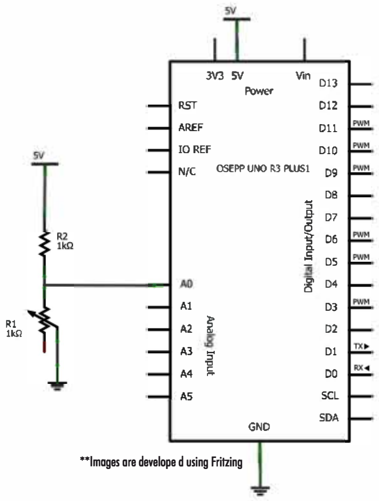
Vanalog_input = 5V x [R1 / (R1 + 1kohm)]

R1 is the resistance of the potentiometer. When R1 is adjusted to 1kohm, the voltage at the analog pin will be 2.5V. When R1 is adjusted to 0 ohm, the voltage at the analog pin will be 0V. The analog pins on the Uno have the same voltage restrictions as the digital pins. Do not apply a voltage larger than 5V.

Useful Note: *If you want to increase the voltage divider circuit voltage range from 0V to 5V, you can remove the 1k ohm fixed resistor and connect the potentiometer like this: Pin 1: 5V power supply; Pin 2 (Center): A0 pin; Pin 3: Ground.*

III – Schematic Diagram:

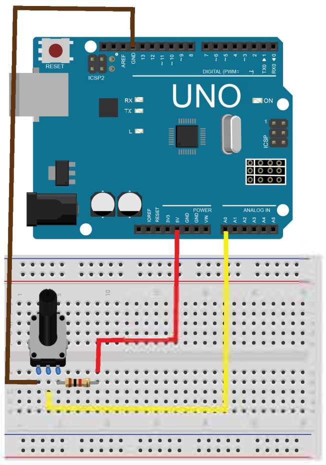
Here is the schematic diagram for the voltage divider circuit. We will use analog pin A0.



**Images are develope d using Fritzing

IV – Wiring the Breadboard:

Let's wire up the bread board.



**Images are developed using Fritzing

IV – Writing the Sketch:

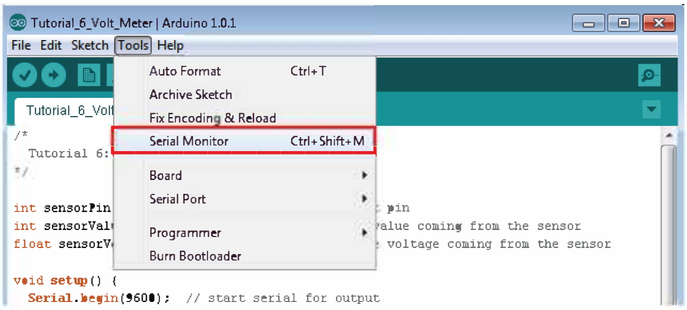
Start a new project and copy and paste the below sketch into the Arduino environment. This sketch uses the analogRead function to read the voltage level on the analog pin A0. The analogRead function will return 0 when the sensed voltage is 0V. It will return 1023 when the sensed voltage is 5V.

Useful Note The reason the function outputs between 0 and 1024 is because the analog to digital converter (ADC) inside the UNO is a 10-bit A-to-D converter. With a 10-bit converter, you will get a resolution of 5V / 1024 ~= 4.89mV.

This sketch will convert the value from the analogRead function and report the voltage value onto the serial port.

```
/*
  Tutorial 5: Volt Meter
*/

int sensorPin = A0;    // select the analog input pin
int sensorValue = 0;   // variable to store the value coming from the sensor
float sensorVoltage = 0; // variable to store the voltage coming from the sensor

void setup() {
  Serial.begin(9600);  // start serial for output
}

void loop() {
  // Read the value from the analog input pin
  // A value of 1023 = 5V, a value of 0 = 0V
  int sensorValue = analogRead(sensorPin);

  // Convert sensor value to voltage
  float sensorVoltage= sensorValue*(5.0/1023.0);

  // print sensor value
  Serial.print("The voltage is ");
  Serial.println(sensorVoltage);

  // delay by 1000 milliseconds:
  delay(1000);
}
```

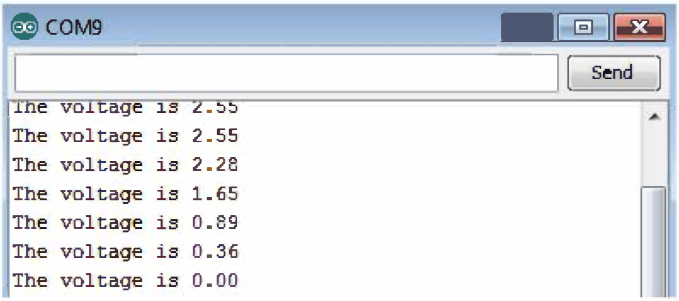
To launch the serial monitor, click Tools→Serial Monitor.



```
/*
  Tutorial 6:
*/

int sensorPin                                      pin
int sensorVal                                     value coming from the sensor
float sensorV                                     voltage coming from the sensor

void setup() {
  Serial.begin(9600);  // start serial for output
```

**Images aredeveloped using Fritzing

If you adjust the potentiometer you will see the voltage change on the serial port monitor.



```
The voltage is 2.55
The voltage is 2.55
The voltage is 2.28
The voltage is 1.65
The voltage is 0.89
The voltage is 0.36
The voltage is 0.00
```

**Images are developed using Fritzing

<u>Things we have covered in this tutorial:</u>

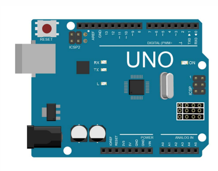
- Using the analog sensor on the UNO
- Using potentiometer

Tutorial 6
Using Buzzer to Play a Melody

In this tutorial we will show you how use the UNO to drive a buzzer. You will learn you use the buzzer to play a scale. We will also show you how to play a short melody with an example sketch.

I – Things you'll need:

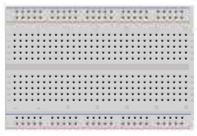
You will need an UNO board, a USB cable, a PC, a bread-board, a buzzer, resistors, jumper cablesm and an NPN transistor for assembling the buzzer circuit.



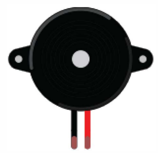
UNO

Mini USB to Standard USB

Windows PC


Breadboard

2N3904 NPN transistor


Buzzer

2x 1kohm Resistor

Jumper cables

**Images aredeveloped using Fritzing

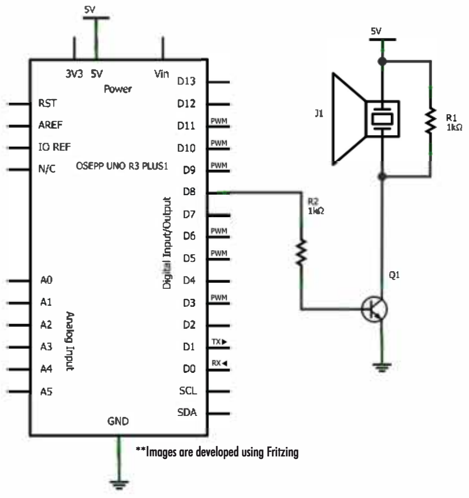
II – Background Information:

This buzzer has the following specifications and characteristics:

SPECIFICATIONS AND CHARACTERISTICS

Sound pressure	70dBA/ 10cmmin.	[at 2kHz, 5V0-P rectangular wave, measuring temperature: 25±5°C, humidity: 60±10%]
Operating temperature range	-10 to +70°C	
Maximum input voltage	30V0-P max.	[without DC bias]

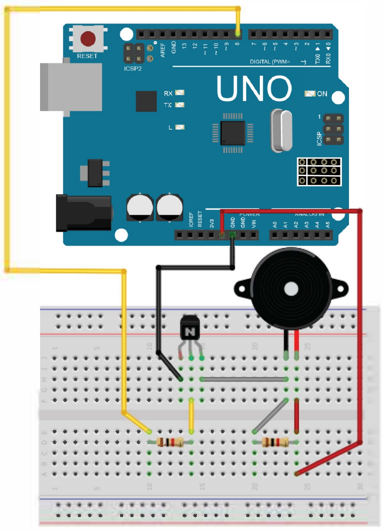
When applying a rectangular wave to the buzzer, a tone will get generated. The higher the frequency of the rectangular wave, the higher frequency the buzzer will produce.



**Images are developed using Fritzing

III – Schematic Diagram:

On the right is the schematic diagram. We will use digital pin 8 to drive the buzzer.



IV – Wiring the Breadboard:

Let's wire up the bread board. Be sure to pay attention to the positive and negative terminal of the buzzer. Also pay attention to the connection of the NPN transistor. The pin on the left is the emitter, the center pin is the base, and the pin on the right is the collector.

**Images are developed using Fritzing

V – Writing the Sketch:

Start a new project and copy and paste the code bellow into the Arduino environment. These two sketches use the tone function to generate the rectangular wave to the buzzer. The first sketch will play the C scale. The second sketch will pay a melody. You can also load this sketch from the example tab in the Arduino environment.

```
/*
  Tutorial 6a: Simple Scale Sweep
*/

int buzzerPin = 8;    // Using digital pin 8

#define NOTE_C6  1047
#define NOTE_D6  1175
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_G6  1568
#define NOTE_A6  1760
#define NOTE_B6  1976
#define NOTE_C7  2093

void setup() {
  // nothing to setup
}

void loop() {

  //tone(pin, frequency, duration)
  tone(buzzerPin, NOTE_C6, 500);
  delay(500);
  tone(buzzerPin, NOTE_D6, 500);
  delay(500);
  tone(buzzerPin, NOTE_E6, 500);
  delay(500);
  tone(buzzerPin, NOTE_F6, 500);
  delay(500);
  tone(buzzerPin, NOTE_G6, 500);
  delay(500);
  tone(buzzerPin, NOTE_A6, 500);
  delay(500);
  tone(buzzerPin, NOTE_B6, 500);
  delay(500);
  tone(buzzerPin, NOTE_C7, 500);
  delay(500);
}
```

```
/*
Tutorial 6b: Playing an Melody
*/

 #include "pitches.h"

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3,
NOTE_C4};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4,4,4,4,4 };

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000/noteDurations[thisNote];
    tone(8, melody[thisNote],noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(8);
  }
}

void loop() {
  // no need to repeat the melody.
}
```

<u>Things we have covered in this tutorial:</u>

- Making sounds with a buzzer.

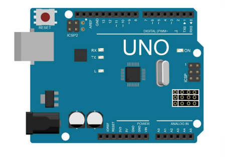
Tutorial 7

Counting Down with a 7 Segment LED

In this tutorial we will show you how to use the UNO to drive a 7 segment LED display. You will learn how to wire up the LED display, and how to make a simple count down from 9 to 0.

I – Things you'll need:

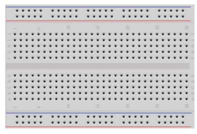
You will need an UNO, a USB cable, a PC, a bread-board, 7 segment LED, resistors, and jumper cables for assembling the circuit.

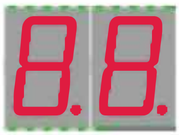


UNO

USB Cable

Windows PC


Breadboard


7 Segment LED Display

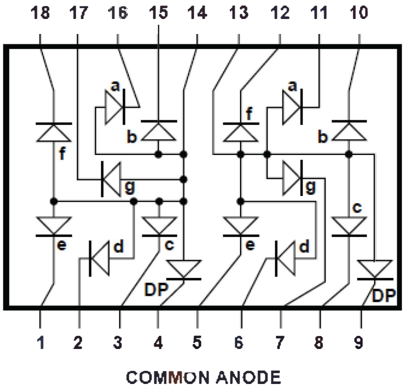
7x 330ohm Resistor

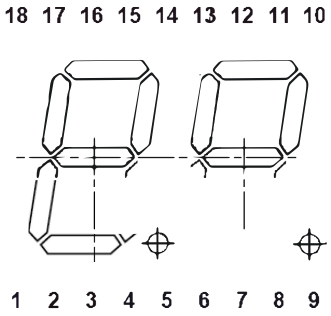
Jumper cables

**Images are developed using Fritzing

II – Background Information:

We will be using a two digit 7-segment LED display for this tutorial. We're only going to use the right digit. This LED display is a common anode device. To light up an LED segment we will need to apply 5V to the common anode, pin 13, and then ground any of the seven segments through a 330ohm resistor. We will use the UNO digital output pins to drive these LED segments. We will need 7 pins to drive the 7 segments of one digit.

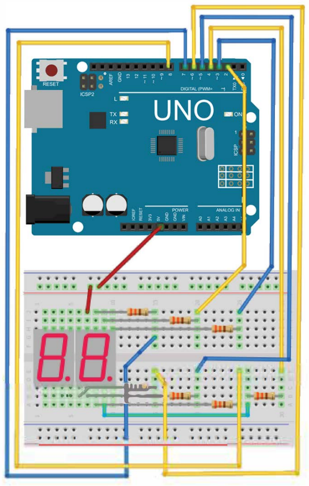




III – Wiring the Breadboard:

Let's wire up the bread board. We will map the digital pin as follows:

- Digital pin 2: segment a
- Digital pin 3: segment b
- Digital pin 4: segment c
- Digital pin 5: segment d
- Digital pin 6: segment e
- Digital pin 7: segment f
- Digital pin 8: segment g



IV – Writing the Sketch:

Start a new project and copy and paste the code below into the Arduino environment. This sketch initializes digital pin 2 to 8 as output. We define a lookup table that we use to find the LED display segment pins that make up each number from 0 to 9. Once the sketch is loaded on the board, you will see the number decrement from 0 to 9 continuously.

```
//  Tutorial 7: 7 Segment LED
//
// Define the LED digit patters, from 0 - 9
// Note that these patterns are for common anode displays
// 0 = LED on, 1 = LED off:

//                                  Digital pin: 2,3,4,5,6,7,8
//                                                  a,b,c,d,e,f,g
byte seven_seg_digits[10][7] = { { 0,0,0,0,0,0,1 },  // = 0
                                                { 1,0,0,1,1,1,1 },  // = 1
                                                { 0,0,1,0,0,1,0 },  // = 2
                                                { 0,0,0,0,1,1,0 },  // = 3
                                                { 1,0,0,1,1,0,0 },  // = 4
                                                { 0,1,0,0,1,0,0 },  // = 5
                                                { 0,1,0,0,0,0,0 },  // = 6
                                                { 0,0,0,1,1,1,1 },  // = 7
                                                { 0,0,0,0,0,0,0 },  // = 8
                                                { 0,0,0,1,1,0,0 }   // = 9
                                                };

void setup() {
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}

void sevenSegWrite(byte digit) {
  byte pin = 2;
  for (byte segCount = 0; segCount < 7; ++segCount) {
    digitalWrite(pin, seven_seg_digits[digit][segCount]);
    ++pin;
  }
}

void loop() {
  for (byte count = 10; count > 0; --count) {
   delay(1000);
   sevenSegWrite(count - 1);
  }
  delay(3000);
}
```

Things we have covered in this tutorial:

- Using the 7-segment LED display.

Tutorial 8

Powering the UNO Using Batteries

In this tutorial we will highlight the things that you will need to pay attention to when trying to power the UNO board using AA batteries.

A AA battery is the most common type of battery used in portable electronic products. The voltage and capacity depends on what chemicals they are made from. These chemicals include Zinc-carbon, Alkaline, Li-FeS2, NiCd, NiMH, and most recently Li-ion. Off the shelf, the most common battery that you will find for an un-rechargeable type is the alkaline battery, and the rechargeable type is the NiMH battery. Typically, their capacity is in the 2300mAh – 2700mAh range.

Useful Note: *The unit mAh means mA-hour. Assuming you are using a single 2700mA battery, if your application consumes 2700mA, your battery will last for about an hour.*

Before designing your circuit, you should think ahead on the type of battery you will be using, how many batteries you will be connecting and how they will be connected, as these will affect how the input stage of the power circuit is designed.

Each AA alkaline type battery will output around 1.5V in normal condition, however, the NiMH type battery will output around 1.2V in normal conditions.

If you connect four batteries in series, then you will get 6V when using alkaline batteries and you will get 4.8V when using NiMH batteries.

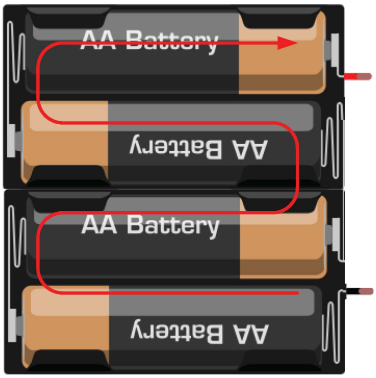
Useful Note: *The UNO uses a MC33269D-5.0V 5V linear regulator to regulate its input voltage. This voltage regulator has a 1V drop, when you supply a 6V into the input, it can still output 5V and function correctly.*

Useful Note: *Can I connect four NiMH batteries and tap it into the 5V power rail directly? A NiMH battery will output 1.2V normally, but just after charged, it could be outputting 1.4V, so you could be supplying 5.6V into the 5V rail. Verify the voltage with a volt-meter. Avoid connecting more than 5V directly to the 5V rail.*

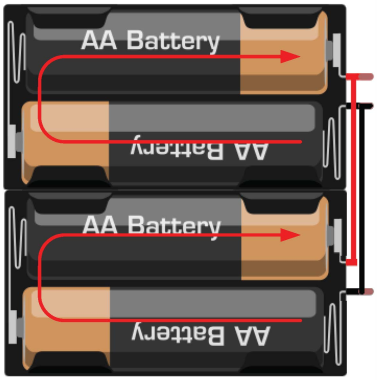
If you want to power your UNO using either type of batteries, you can use a boost converter. The AA battery holder connects the four AA batteries in series, so your voltage will range from 4.8V to 6V while fully charged. You can put in a boost converter that converts the voltage up to 7V, then supply the 7V to the VIN terminal of the UNO. This method will work, but it is very inefficient.

A better method is to modify the battery holder, so that it will connect the four batteries with two in series, and two in parallel. This will give you 2.4V to 3V. Using a 5.0V boost convert, you can convert this voltage to 5V and supply it directly to the 5V terminal of the UNO. By bypassing the onboard 5.0V linear regulator, this solution will have less power loss and your battery will last longer.

Four batteries all in series



Four batteries with two in series



**Images are developed using Fritzing

Another alternative is to use a 9V battery. You will need a 9V connector (included in kit). You simply plug it into the DC Power Jack. You should see the green light to indicate that it is powered.

STAY CONNECTED

Stay connected with us to get the latest product information, free giveaways, and contests. It is also a great way to let us know what's on your mind. We love to hear from our customers about what we are doing right but most importantly, how we can improve our products and services!

http://www.facebook.com/OSEPP.ArduinoCompatible

https://twitter.com/diy_w_osepp

Subscribe by going to our website

AFTER SALES SUPPORT

Our team of engineers at OSEPP can't guarantee that every product we sell will be perfect; we make mistakes along the way. For that reason, we put a lot of effort into making sure that all complaints, comments, and concerns be dealt with promptly and to the customer's satisfaction!

We strive to provide the "Best after sales support" in the industry.

For anyone who has taken the time to contact us (for any reason), we thank you!

Here's how you can contact us:

Technical Support	support@osepp.com
General Inquiries	info@osepp.com
Sales Related	sales@osepp.com