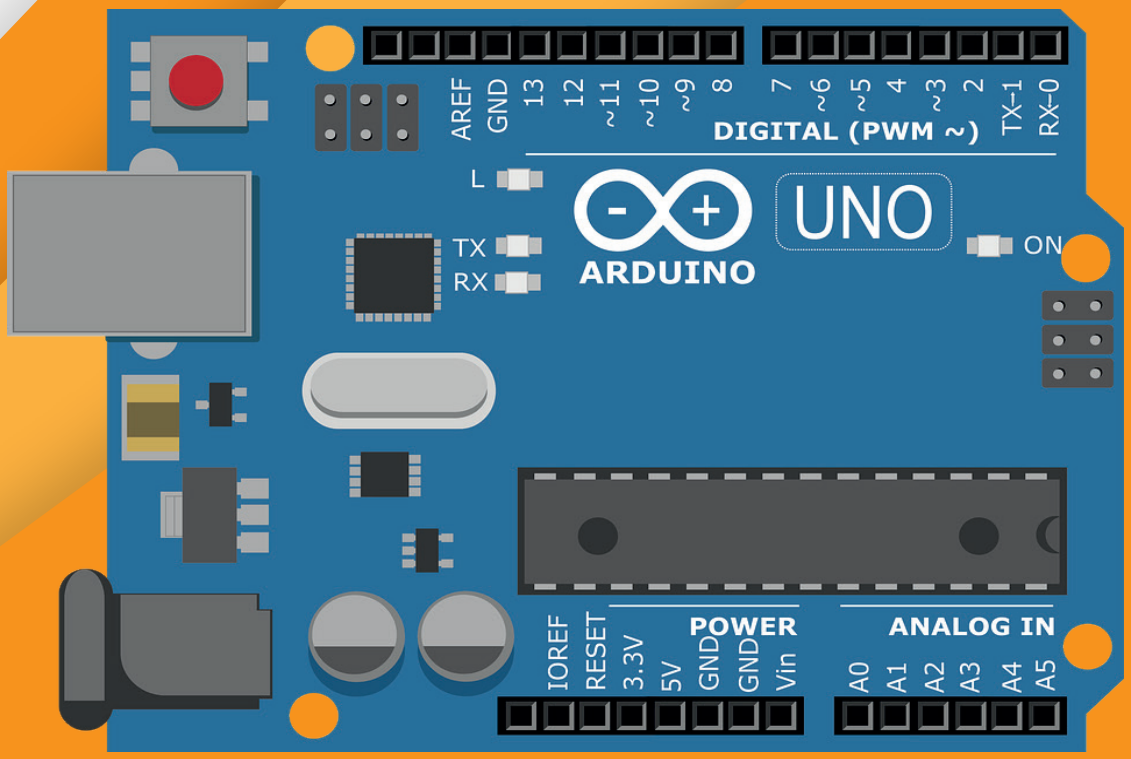
Rewise
Learning. Made Fun

BITBOX

CODE BOOKLET



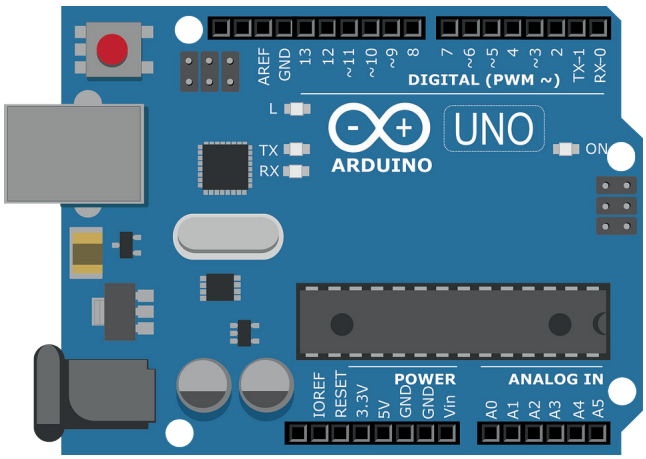
LearnThruTech

<CODE>

QUICK START

Equipment Required;

- Smartphone / Tablet / PC
- Bluetooth
- Arduino IDE Software
- Arduino UNO Board & USB Cable
- Electronics Kit Supplied



CODING SOFTWARE

You can download the Arduino IDE software from the Arduino website, use a search engine on your online browser;

www.arduino.cc/en/main/software

Make sure you download the right software compatible with your computer operating system; Windows, Mac or Linux. Follow the download procedure to install.

Alternatively you can use the online web editor instead of downloading the software by registering an account online.

KEY TERMS

{ } - At the beginning and end of each section, these curly brackets tell the Arduino board what relates to each section. They are the marker to begin or end your section.
Sections:
void setup ( ) - Where you put your set up and initial functions for the program. Note: The code you enter here will run once.
void loop ( ) - Putting your main code in here will make it run repeatedly in a loop.
void someFunctionName( ) – custom created function that can be called from **loop ( )** or **setup ( )**

INBUILT COMMANDS

Remember - Arduino is case sensitive and has numerous of inbuilt commands. If typed on correctly functions will turn orange, while variables types will turn blue.

Example of a function:
analogWrite() – send an analog (0-255) value to a pin.
digitalWrite() – send a digital value (HIGH,LOW) to a pin.

Example of a variable:
int – allows to define a natural number variable (like 1 or 400).
Other commands:
#include – allows to add a library (someone else's code) into your code.

NOTE

THE WRITING IN THIS DOCUMENT HIGHLIGHTED SHOULD NOT BE WRITEN INTO THE ARDUINO SOFTWARE.
Press enter after each section once, then type the following into the Arduino IDE software.

BITBOX CODE

Write the following into the Arduino software window above **void setup()** to declare the arduino pin variables and volume for the speakers, and brightness and colours for LED rings;

```
#include <Adafruit_NeoPixel.h>
#include <math.h>

#ifdef __AVR__
  #include <avr/power.h>
#endif
```

Below we define some of the variables that are used in the boombox. You do not need type anything that is after // and in grey colour. The comments here and in other parts of the code are here to help you to understand what is going on in a code.

```
#define N_PIXELS 24  // Number of pixels in strip
#define MIC_PIN   A0 // Microphone is attached to this analog pin
#define LED_PIN    6 // NeoPixel LED strip is connected to this pin

int count = 0;
int brightness = 255;
float volume = 0;
float last = 0;
float maxVolume = 0;
float minVolume = 1024;
float avgVolume = 0;
int currentPixel = 0;
bool musicOn = true;

Adafruit_NeoPixel strip = Adafruit_NeoPixel(N_PIXELS, LED_PIN, NEO_GRB
+ NEO_KHZ800);

int paletteThreshold = 1500;
int colourCount = 1;
```

void setup()

In the void setup() you will need to outline what the pins will do. All the RC pins below will be output pins as we are sending a signal/voltage to the pins. You will need to type the following within the curly brackets;

```
analogReference(EXTERNAL);
 strip.setBrightness(50);
 Serial.begin(9600);

 // initializing neopixel strip
 strip.begin();
 strip.show();

  for(int i = 0 ; i < N_PIXELS; i++){
       strip.setPixelColor(i, strip.Color(255, i*2, 30));
       delay(20);
       strip.show();
     }
 strip.clear();

 volume = analogRead(MIC_PIN);
```

```
void loop () {
```

Type the following into the Arduino software in-between the curly brackets, put the setup code below here, to run repeatedly;

```
  last = volume;
  volume = analogRead(MIC_PIN);
```

The int (where it is 1 now) in musicOn can be changed lower to make the microphone more sensitive or higher to make the microphone less sensitive.

```
  musicOn = abs(last - volume) > 1;
  Serial.println(volume);

  if (musicOn){
    brightness = 255;
      avgVolume = (avgVolume + volume)/2.0;

      if (volume > maxVolume) maxVolume = volume;
      if (volume < minVolume) minVolume = volume;
```

Choose one of the following LED lightning functions by uncommenting the one you want (removing //) and commenting out the other 2 (adding //). Remember, that you will need to write the function that implements the chosen one. Code for all them can be found later in the booklet.

```
      PulseColours();
      //PulseFull();
      //QuaterPulse();

   } else if(currentPixel != 0){
    strip.setPixelColor(currentPixel, strip.Color(0,0,0));
    currentPixel = currentPixel - 1;
    }
```

```
  else {
    brightness -= 5;
    strip.setBrightness(brightness);
  }
  strip.show();
  count++; // keeps count to enable reset minVolume and MaxVolume
  if (count > 500){
    minVolume = avgVolume;

    maxVolume = avgVolume;
    count = 0;
}
delay(10);
```

If you decided that you want to play music, but don't want the light you can comment out all lightening schemes.

```
  //PulseColours();
  //PulseFull();
  //QuaterPulse();
```

Pulse Colour Function

The code below sets the LED rings pixels to pulse colour scheme. Remember, this is one of the functions that control the lighting. So type out the one you picked earlier in loop(). If you have time can type them all out and change between them by commenting them in inside of the loop() function.

```
void PulseColours(){
    if (musicOn){
        int pulseSize = map(volume, minVolume, maxVolume, 0,  N_PIXELS);
        currentPixel = pulseSize;
        for (int i = 0; i < N_PIXELS; i++){
            if (i < pulseSize){
```

The if() statement below allows to change colour scheme by commenting out the unwanted one. Pick one of the functions below to choose between the rainbow scheme(first function) or specified colour one (second).

```
            strip.setPixelColor(i,Wheel(map(i, 0, N_PIXELS-1 ,30, 150)));
```

If you are using the line below – try changing Ocean to another colour scheme. You will find the full list in the other pages.

```
            //strip.setPixelColor(i,Ocean(map(i,0,strip.numPix-
els()-1,0,paletteThreshold)));                    } else{
                strip.setPixelColor(i, strip.Color(0,0,0));
            }
        }
    strip.show();
    }
}
```

The code turns all the LEDs on and adjusts their brightness based on the music. Remember, this is one of the functions that control the lighting. So type out the one you picked earlier in loop(). If you have time can type them all out and change between them by commenting them in inside of the loop() function.

```
void PulseFull(){

 if (musicOn){

  int pulseSize = map(volume, minVolume, maxVolume, 0,N_PIXELS);
  if (pulseSize > N_PIXELS/2){
   colourCount += (paletteThreshold/70)*pulseSize;
   }
  colourCount += 1;

  if (colourCount >= paletteThreshold){
   colourCount = 0;
   }
```

The part where it is written Sulfur can be changed to any of the colour scheme that are typed out below.

```
  uint32_t col = Sulfur(colourCount);

  for (int i = 0; i < N_PIXELS; i++){
     strip.setPixelColor(i,col);
  }

  strip.setBrightness(255.0 * pow((volume / maxVolume),2.5));
  strip.show();
  }
 }
```

The code turns all the LEDs on and adjusts their brightness based on the music. Remember, this is one of the functions that control the lighting. So type out the one you picked earlier in loop(). If you have time can type them all out and change between them by commenting them in inside of the loop() function.

```
void QuaterPulse(){
  if (musicOn){
    int division = 4;
    int equalParts = N_PIXELS/division;
    int pulseSize = map(volume, minVolume, maxVolume, 0, equalParts);
    int pixel = 0;

    for (int i = 0; i < equalParts; i++){
      for (int j = 0; j < equalParts; j++){
        pixel = i* equalParts + j;
        if (j < pulseSize){
```

The part where it is written Ocean can be changed to any of the colour scheme that are typed out below.

```
          strip.setPixelColor(pixel,Ocean(map(pixel,0,strip.numPixels()-1,0,paletteThreshold)));
//Choose your color scheme on this line!
        } else {
          strip.setPixelColor(pixel, strip.Color(0,0,0));
        }
      }
    }
    //strip.setBrightness(255.0 * pow(volume / maxVolume, 2));
    strip.show();
  }
}
```

The code turns all the LEDs on and adjusts their brightness based on the music. Remember, this is one of the functions that control the lighting. So type out the one you picked earlier in loop(). If you have time can type them all out and change between them by commenting them in inside of the loop() function.

```
void Circle(){

 if (musicOn){
  int pulseSize = map(volume, minVolume, maxVolume, 0,  N_PIXELS);
  currentPixel = pulseSize;
  for (int i = 0; i < N_PIXELS; i++){
     if (i < pulseSize){
```

The part where it is written Ocean can be changed to any of the colour scheme that are typed out below.

```
     strip.setPixelColor(i,Ocean(map(i,0,strip.numPixels()-1,0,paletteThreshold)));
    //use this line to change to another colour scheme
    //don't forget to comment out the one above.

     } else{
     strip.setPixelColor(i, strip.Color(0,0,0));
     }
  }
  //strip.setBrightness(255.0 * pow(volume / maxVolume, 2));
  strip.show();
  }
}
```

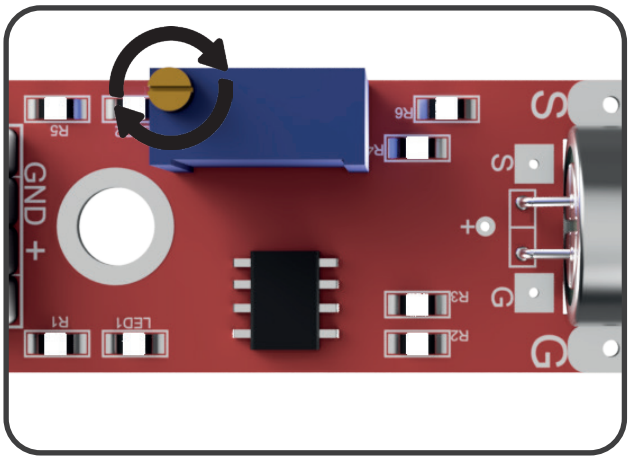
This function called Wheel() allows to LEDs to spin in rainbow colours.

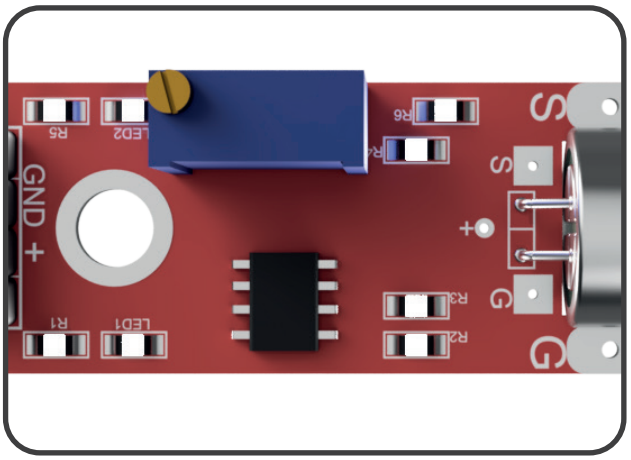
```
uint32_t Wheel(byte WheelPos) {

  if(WheelPos < 85) {
    return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  }
  else if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  else {
    WheelPos -= 170;
    return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

These are all of the colour schemes you can use to light up your boombox. Remember to type out the one you selected in your lightning function. You can also try to create your own!

```
uint32_t Rainbow(unsigned int i) {
  paletteThreshold = 1529;
  if (i > 1529) return Rainbow(i % 1530);
  if (i > 1274) return strip.Color(255, 0, 255 - (i % 255));    //violet -> red
  if (i > 1019) return strip.Color((i % 255), 0, 255);          //blue -> violet
  if (i > 764) return strip.Color(0, 255 - (i % 255), 255);     //aqua -> blue
  if (i > 509) return strip.Color(0, 255, (i % 255));           //green -> aqua
  if (i > 255) return strip.Color(255 - (i % 255), 255, 0);    //yellow -> green
  return strip.Color(255, i, 0);                               //red -> yellow
}
```

```
uint32_t Sunset(unsigned int i) {
  paletteThreshold = 1019;
  if (i > 1019) return Sunset(i % 1020);
  if (i > 764) return strip.Color((i % 255), 0, 255 - (i % 255));          //blue -> red
  if (i > 509) return strip.Color(255 - (i % 255), 0, 255);                //purple -> blue
  if (i > 255) return strip.Color(255, 128 - (i % 255) / 2, (i % 255));   //orange -> purple
  return strip.Color(255, i / 2, 0);                                       //red -> orange
}
uint32_t Ocean(unsigned int i) {
  paletteThreshold = 764;
  if (i > 764) return Ocean(i % 765);
  if (i > 509) return strip.Color(0, i % 255, 255 - (i % 255));      //blue -> green
  if i > 255) return strip.Color(0, 255 - (i % 255), 255);           //aqua -> blue
  return strip.Color(0, 255, i);                                     //green -> aqua
}
uint32_t PinaColada(unsigned int i) {
  paletteThreshold = 764;
  if (i > 764) return PinaColada(i % 765);
  if (i > 509) return strip.Color(255 - (i % 255) / 2, (i % 255) / 2, (i % 255) / 2);
//red -> half white
  if i > 255) return strip.Color(255, 255 - (i % 255), 0);           //yellow -> red
  return strip.Color(128 + (i / 2), 128 + (i / 2), 128 - i / 2);     //half white -> yellow
}
uint32_t Sulfur(unsigned int i) {
  paletteThreshold = 764;
  if (i > 764) return Sulfur(i % 765);
  if (i > 509) return strip.Color(i % 255, 255, 255 - (i % 255));  //aqua -> yellow
  if (i > 255) return strip.Color(0, 255, i % 255);                //green -> aqua
  return strip.Color(255 - i, 255, 0);                             //yellow -> green
}
uint32_t NoGreen(unsigned int i) {
  paletteThreshold = 1274;
  if (i > 1274) return NoGreen(i % 1275);
  if (i > 1019) return strip.Color(255, 0, 255 - (i % 255));         //violet -> red
  if (i > 764) return strip.Color((i % 255), 0, 255);                //blue -> violet
  if (i > 509) return strip.Color(0, 255 - (i % 255), 255);          //aqua -> blue
  if (i > 255) return strip.Color(255 - (i % 255), 255, i % 255);   //yellow -> aqua
  return strip.Color(255, i, 0);                                     //red -> yellow
```

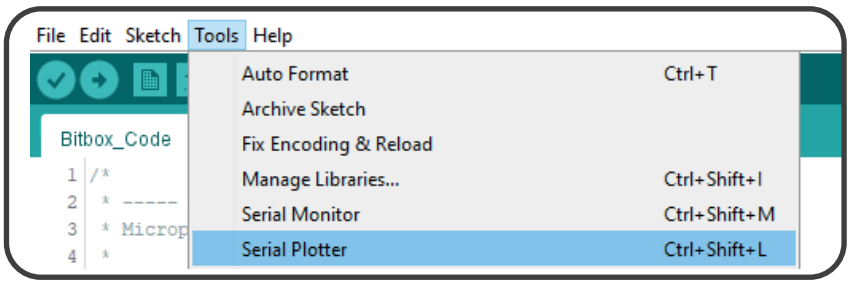




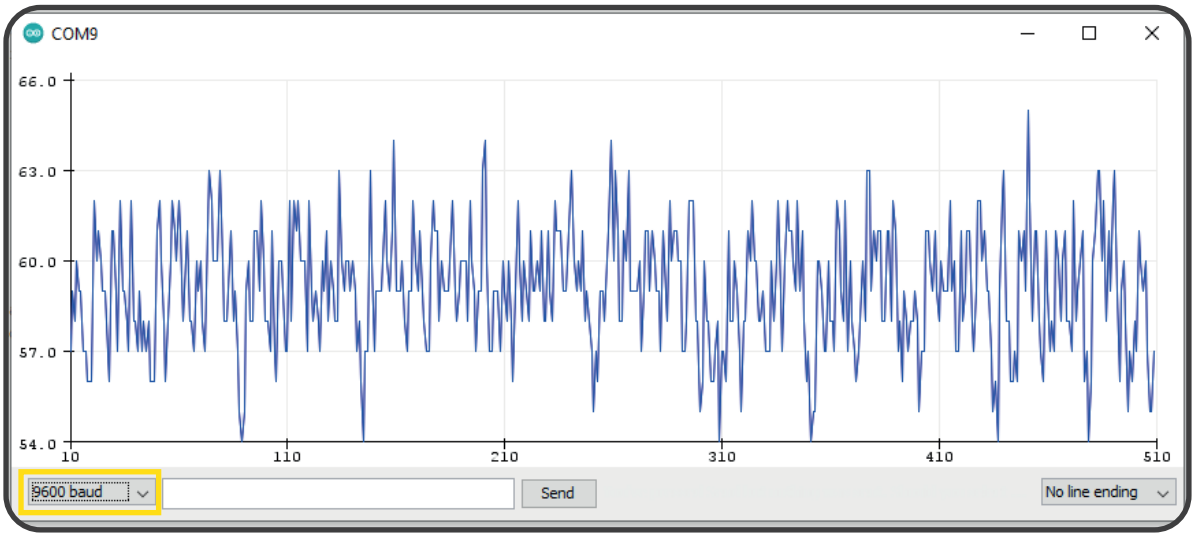
To ensure that the microphone is adjusted correctly you have to rotate the potentiometer (Copper screw on the top of blue box on the microphone).

The potentiometer must be set to value between 55-60. To do that, please upload the code to the Arduino, connect all cables and open Serial Plotter.

Tools -> Serial Plotter



When the window is displayed, please choose 9600 baud (Bottom Left Corner).



By rotation potentiometer you will notice that the numbers on left side will either increase or decrease.

This product is not a toy.

tuborial.com

Widely Recycled