HERIOT - WATT UNIVERSITY

School of Engineering & Physical Sciences

Mechanical Engineering Laboratory

Arduino Experiments

Pulse Width Modulation: RGB LED Control

Introduction:

One of the tasks of Arduino is to control devices such as motors and light sources. Often this just involves switching these devices on an off, but in some instances they are used to control the speed of a motor or the intensity of a light source.

If we take controlling the speed of a motor as an example, then a commonly used method of controlling the motor speed involves keeping the motor continuously on, but varying the power that drives it. Arduino does not have a true analogue voltage output, however by adding a digital-to-analogue converter we can then then use the analogue signal to vary the power.

An alternative mode of controlling the speed of the motor is Pulse Width Modulation. Essentially it works by only involving “on” and “off” states. PWM varies the amount of time that the blinking pin or a motor spends HIGH vs. the time it spends LOW. If it spends most of its time LOW, the LED will look dim. Because the pin is blinking much faster than your eye can detect, the Arduino creates the illusion of a “true” analogue output.

The ratio of “on” time to the total time is called the duty cycle, and variation of the duty cycle can be used to control the motor speed.

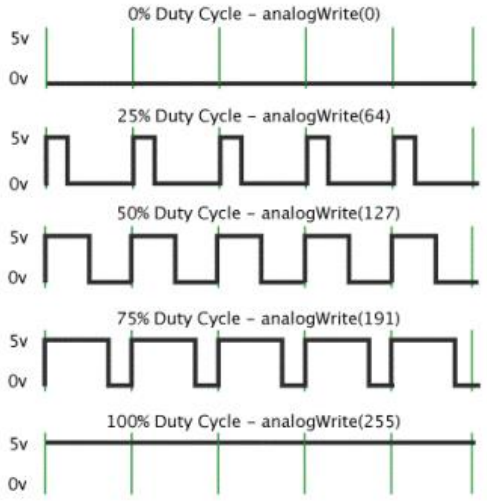

Figure 1: Pulse Width Modulation

RGB:

The RGB colour system constructs visible colours from combinations of Red, Green and Blue colours. The red, green and blue use 8 bits each, which have integer values from 0 to 255. This makes 256*256*256=16777216 possible colours.

In the RGB spectrum the main colours are:

- Red (255,0,0)
- Green (0,255,0)
- Blue (0,0,255)

Changing the three values would give you different colours. Some examples of colours are:

- Yellow (255,255,0)
- Purple (128,0,128)
- Navy (0,0,128)

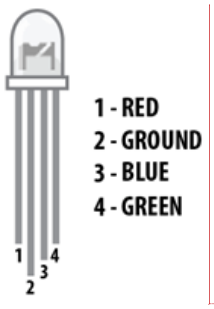

Figure 2: RGB LED

Commented [SJH1]: This will need to be checked. Some off-the-shelf RGB LEDs have the pins in a different order (e.g. R,B,G). Check with whatever we decide to order and update this accordingly.

Required Components:

- Arduino UNO
- Potentiometer
- LED RGB
- 3* 330Ω resistor for LED
- Breadboard
- Jumper cables

Procedure:

In this experiment, PWM will be used to control the light output of a multicolour LED.

PWM can be achieved with Arduino using the “analogWrite” command. The analogWrite command is called with 2 parameters, like this:

analogWrite(pin, value);

- pin – denotes which number pin on the Arduino the PWM is applied (note: only pins marked with a “~” on the Arduino connectors support PWM)
- value – is the duty cycle of the PWM signal to be applied, ranging from 0 (for 0%) to 255 (for 100%)

To display information on your monitor first add Serial.begin(9600); in setup(); then you can use Serial.print(); or Serial.println(); to display on Serial Monitor which is located on the top right in Arduino IDE.

Figure 3: Serial Monitor:

**As before uncommenting the method in loop(); will allow you to use it, just don't forget to comment it out after you done*

Use the code named rgb_led for these exercises

Task 1:

Write a program to light the LED in red, blue, and green. Try lighting the LED with other colours.

Task 2:

Write a program to slowly fade the red LED in and out, you may find the program "fading" under File/Examples/03.Analog useful.

Task 3:

Now modify the program to fade each LED sequentially (i.e. fade red in, red out, green in, green out)

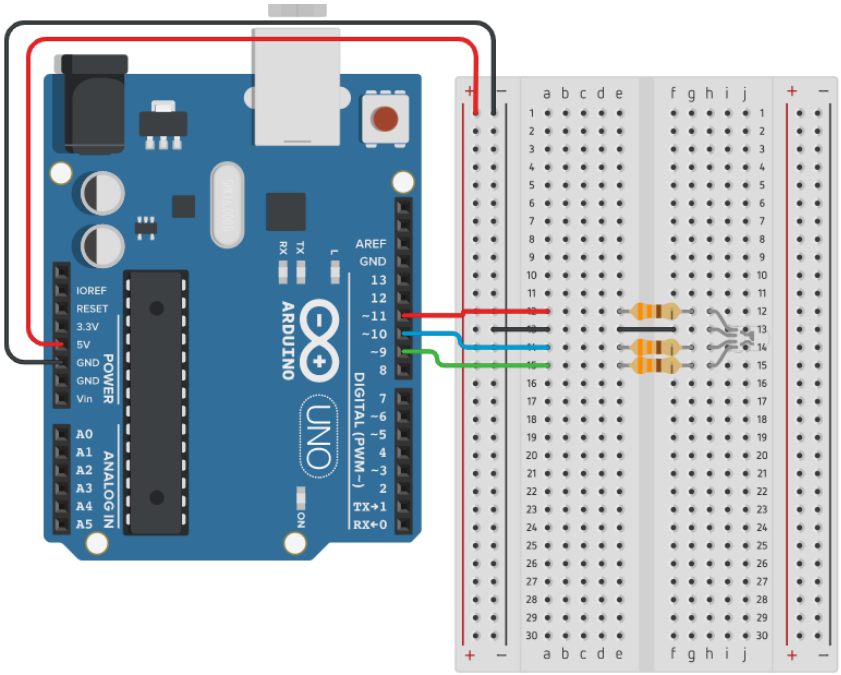

Figure 4: Circuit for Tasks 1,2,3

Task 4:

Once the program is working correctly for the previous step, make sure the potentiometer middle pin is connected to A0 on the Arduino and modify the program to change the speed of the fading depending on the potentiometer position. This can be done by reading the voltage at A0 and changing the value added to “fadeValue” each time (you may use the “ReadAnalogVoltage” example program to remind yourself on how to do this).

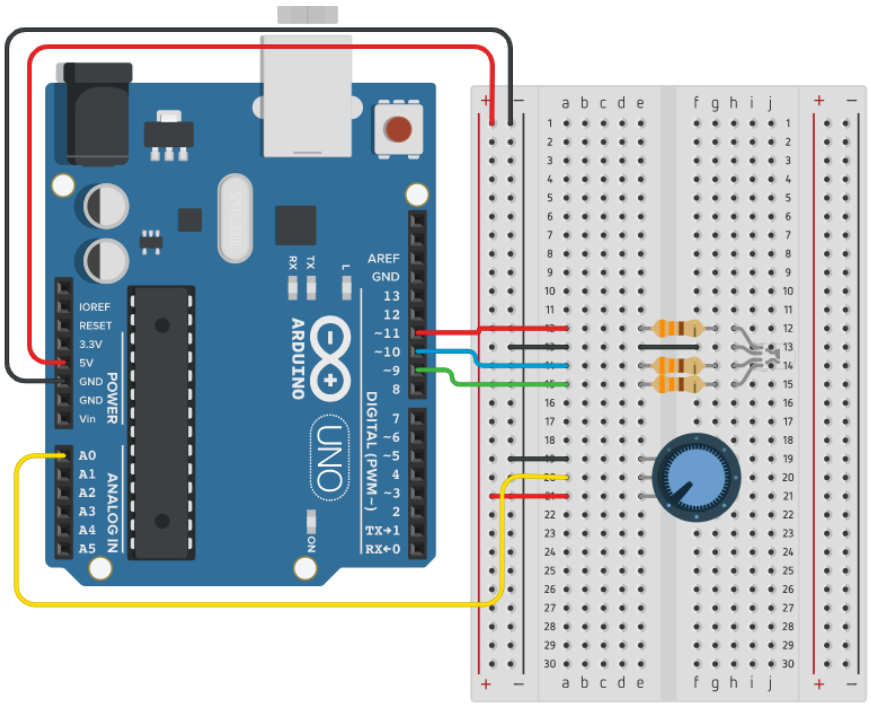

Figure 5: Circuit for Task 4