Engineering Principles Name_____
Electrical Engineering
Color Wheel 4.6.1 Doc: Images Date_____ Team _____

## Problem statement

Develop a persistence-of-vision color wheel utilizing 8 RGB diodes and an Arduino UNO R3 board.

## Documentation: Pictures

1. Create a new graphics file using Paint.Net. Set the image size to 8 pixels high by 64 pixels wide. Draw a picture to be displayed on Arduino. A black background is recommended.
2. Save this picture and upload it to your group's file space on Canvas. File names cannot contain and spaces or special characters.

## Documentation: Image Translation

3. Open the picture using GIMP. Gimp is an image translation program. It can read most graphics file formats and can output them in different formats. We will be exporting to "C" code which can be read by Arduino.
    a. Select File->Export As
    b. change "All export images" to "C source ode (*.c)"
    c. Replace the file extension (.png) with (.c)
    d. Select Export
    e. **Uncheck all the boxes** and select Export again.
4. Open the file using notepad to see what it looks like. See below for a description of the code. If your file looks different than the example, review the instructions carefully and regenerate the file.
5. Upload this file to your group's file space in Canvas.
6. Discuss with your team the image file you created and the structure of the C file you exported.

## Documentation: Image Refinement

7. Work with your team to create additional images as necessary.

Name_____

Date_____ Team _____

## Image file as a C Structure

In C, Structures (struct) are compound variables containing multiple related pieces of information. The graphics images you created using GIMP defines a structure which creates a variable called `gimp_image`. It contains everything you need to know to display the image. Inside the `struct` are the `width` (64), `height` (8) and `bytes_per_pixel` (3) along with an array of characters called `pixel_data` representing the individual pixel values.

```
static const struct {
  unsigned int   width;
  unsigned int   height;
  unsigned int   bytes_per_pixel; /* 2:RGB16, 3:RGB, 4:RGBA */
  unsigned char  pixel_data[64 * 8 * 3 + 1];
} gimp_image = {
  64, 8, 3,

"\377^\304\0\0\0\0\0\377^\304\0\0\0\0\0\377^\304\0\0\0\0\0\377^\
304"
```

Most of the pixel data is represented in octal format. Each byte or character begins with a backslash followed by one to three digits ranging from 0 to 7. The largest number is /377 which is the binary number 11 111 111. In decimal this is 255. In hex it would be written 0xFF. You may also see text characters in the data list – the pixel value is the ASCII value of the character (A = 65, a = 97).

Pixels are stored in the array by color, row, and then column (R1C1 Red, R1C1 Green, R1C1 Blue, R1C2 Red, R1C2 Green, R1C2 Blue, ...). The pixel array index can be calculated by adding color (0, 1 or 2) to (row * width + column ) * bytes_per_pixel.

To access elements of a struct, use the struct variable name followed by a dot (.) followed by the element name within the struct.

```
gimp_image.width = 64;  // sets the image width to 64
pixelValue = gimpImage.pixel_data[35];  // retrieves the 36th pixel
```