WEAPONS ROBOTICS AND CONTROL ENGINEERING
UNITED STATES NAVAL ACADEMY

Lab 4

Controlling the LED in HSL Color Space

In [ ]:

```
from colorsys import hls_to_rgb
import analogio
import pwmio
import time
from board import *

# make a list of my declared knobs and lights
knob_pins = [A0, A1, A2]
light_pins = [GP13, GP14, GP15]

# assign knob and led pins
knobs = [analogio.AnalogIn(k) for k in knob_pins]
lights = [pwmio.PWMOut(l, frequency=5000, duty_cycle=0) for l in light_pins]

MAX = 2**16
GAMMA = 3.0

while True:
    # use get knob values and convert to a value from 0-1
    h,l,s = knobs[0].value/MAX, knobs[1].value/MAX, knobs[2].value/MAX
    # convert hls values to rgb values 0-255
    rgb = hls_to_rgb(h,l,s)

    # set each led pin brightness to gamma corrected rgb value
    for i in range(len(lights)):
        lights[i].duty_cycle = int(((rgb[i]/255)**GAMMA)*MAX)

    time.sleep(0.05)
```

Intuitive Color Control

In [ ]:

```python
from colorsys import hls_to_rgb
import analogio
import pwmio
import time
from board import *

# make a list of my declared knobs and lights
knob_pins = [A0, A1]
light_pins = [GP13, GP14, GP15]

# assign knob and led pins
knobs = [analogio.AnalogIn(k) for k in knob_pins]
lights = [pwmio.PWMOut(l, frequency=5000, duty_cycle=0) for l in light_pins]

MAX = 2**16
GAMMA = 3.0

while True:
    # use get knob values and convert to a value from 0-1
    # scale lightness to 0-50%
    # leave saturation at 100%
    h,l,s = knobs[0].value/MAX, knobs[1].value/(MAX*2), 1
    # convert hls values to rgb values 0-255
    rgb = hls_to_rgb(h,l,s)

    # set each led pin brightness to gamma corrected rgb value
    for i in range(len(lights)):
        lights[i].duty_cycle = int(((rgb[i]/255)**GAMMA)*MAX)

    time.sleep(0.05)
```

Return of Blinky

In [ ]:

```
from colorsys import hls_to_rgb
import analogio
import pwmio
import time
from board import *

# make a list of my declared knobs and lights
knob_pins = [A0, A1, A2]
light_pins = [GP13, GP14, GP15]

# assign knob and led pins
knobs = [analogio.AnalogIn(k) for k in knob_pins]
lights = [pwmio.PWMOut(l, frequency=5000, duty_cycle=0) for l in light_pins]

MAX = 2**16
GAMMA = 3.0

while True:
    # use get knob values and convert to a value from 0-1
    # scale lightness to 0-50%
    # leave saturation at 100%
    h,l,s = knobs[0].value/MAX, knobs[1].value/(MAX*2), 1
    # convert hls values to rgb values 0-255
    rgb = hls_to_rgb(h,l,s)

    # get third knob value and scale from 0-1
    rest = knobs[2].value/MAX

    # set each led pin brightness to gamma corrected rgb value
    for i in range(len(lights)):
        lights[i].duty_cycle = int(((rgb[i]/255)**GAMMA)*MAX)

    # use time.sleep to adjust rate of blink
    time.sleep(rest)

    # blink (turn led off) if knob value is lower than 25%
    if rest > 0.25:
        for i in range(len(lights)):
            lights[i].duty_cycle = 0

        # use time.sleep to adjust rate of blink
        time.sleep(rest)
```

Bonus: Colored Heartbeat

In [ ]:

```
from colorsys import hls_to_rgb
import analogio
import pwmio
import time
from math import sin, pi
from board import *

# make a list of my declared knobs and lights
knob_pins = [A0, A1, A2]
light_pins = [GP13, GP14, GP15]

# assign knob and led pins
knobs = [analogio.AnalogIn(k) for k in knob_pins]
lights = [pwmio.PWMOut(l, frequency=5000, duty_cycle=0) for l in light_pins]

MAX = 2**16
GAMMA = 3.0

# fade using corrected sin wave
def fade(x):
    return (sin((2*pi)*(x*0.01))+1)/2

# create list of brightness values along sin wave
# ensure brightness values don't exceed 65535
# ternary operator and list comprehension!
brightness = [fade(x) if fade(x) < 65535 else 65535 for x in range(1,101)]

while True:
    # cycle throguh brightness values
    for bright in brightness:
        # use get knob values and convert to a value from 0-1
        # scale lightness to 0-50%
        # leave saturation at 100%
        h,l,s = knobs[0].value/MAX, (knobs[1].value/MAX) *(bright/2), 1
        rgb = hls_to_rgb(h,l,s)

        # get third knob value and scale from 0-1
        rest = knobs[2].value/MAX

        # set each led pin brightness to gamma corrected rgb value
        for i in range(len(lights)):
            lights[i].duty_cycle = int(((rgb[i]/255)**GAMMA)*MAX)

        # use time.sleep to adjust rate of heartbeat
        time.sleep(0.01 * rest)
```