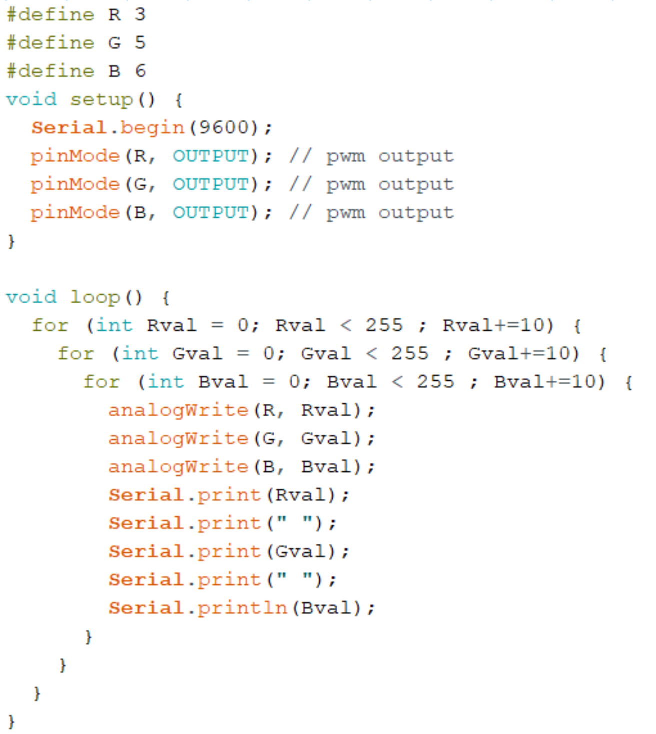
HW3 Simon Ng

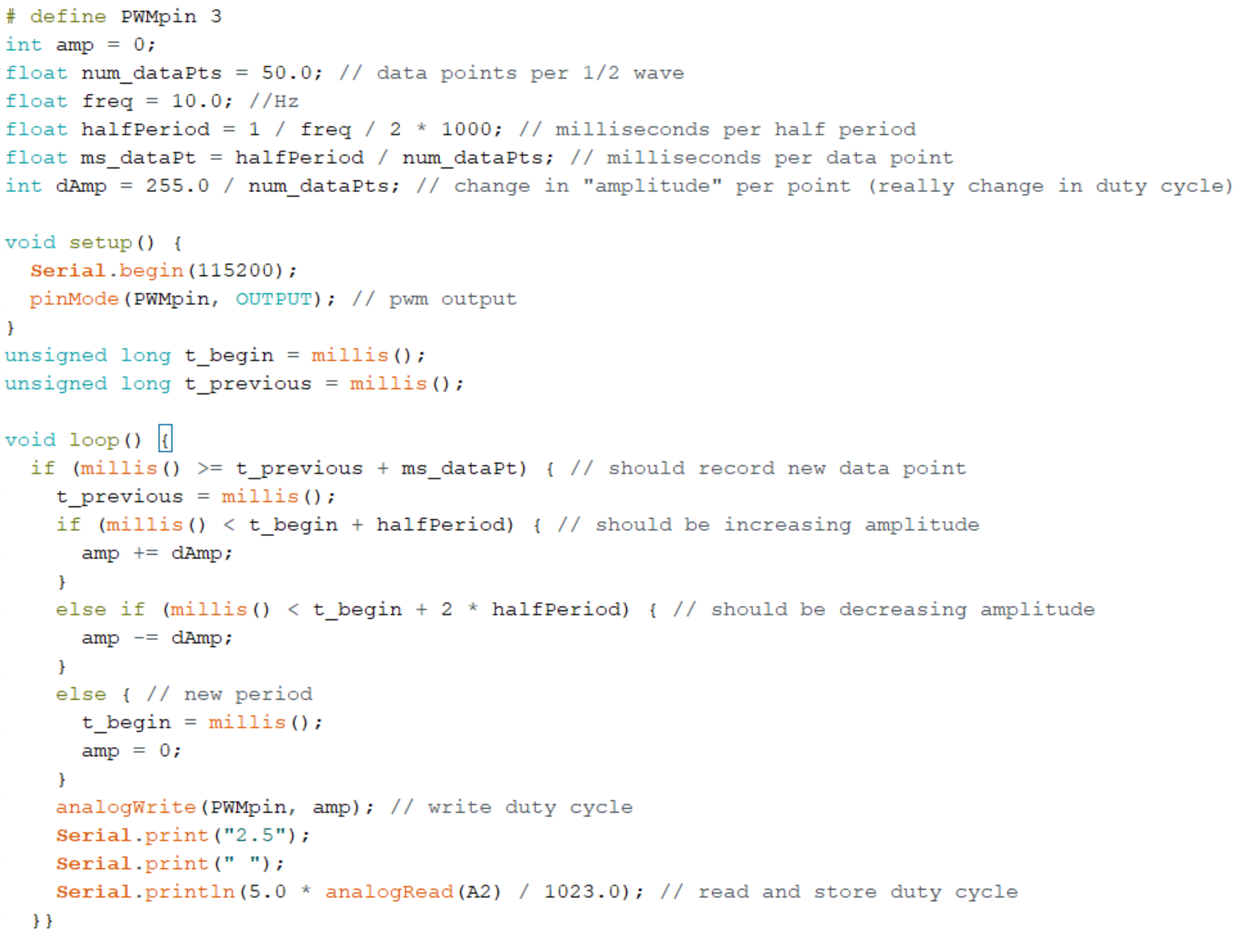
Friday, April 17, 2020 12:57

1. I used 1k resistors to give a final calculated current of 1.8-3mA for each internal LED. I set the PWM values to increment by 10 since incrementing by 1 and ranging over 2^{24} combinations took too long.



```
#define R 3
#define G 5
#define B 6
void setup() {
  Serial.begin(9600);
  pinMode(R, OUTPUT); // pwm output
  pinMode(G, OUTPUT); // pwm output
  pinMode(B, OUTPUT); // pwm output
}

void loop() {
  for (int Rval = 0; Rval < 255 ; Rval+=10) {
    for (int Gval = 0; Gval < 255 ; Gval+=10) {
      for (int Bval = 0; Bval < 255 ; Bval+=10) {
        analogWrite(R, Rval);
        analogWrite(G, Gval);
        analogWrite(B, Bval);
        Serial.print(Rval);
        Serial.print(" ");
        Serial.print(Gval);
        Serial.print(" ");
        Serial.println(Bval);
      }
    }
  }
}
```

2.
   a. It is very difficult to tell whether my program is doing what I want since the amplitudes all range from 0 to 5. I wasn't sure if my code was right, so I built a low pass filter with 3dB at 11.37Hz to troubleshoot. I was able to confirm my code was correct by changing the triangle wave frequency to 1Hz with 50 data points, which very clearly showed the triangle wave form. Then I changed back to 10Hz and removed the low pass filter to view the results without filtering. I could see some sort of pattern, but everything ranged from 0 to 5 and it was impossible to discern anything like a triangle wave.



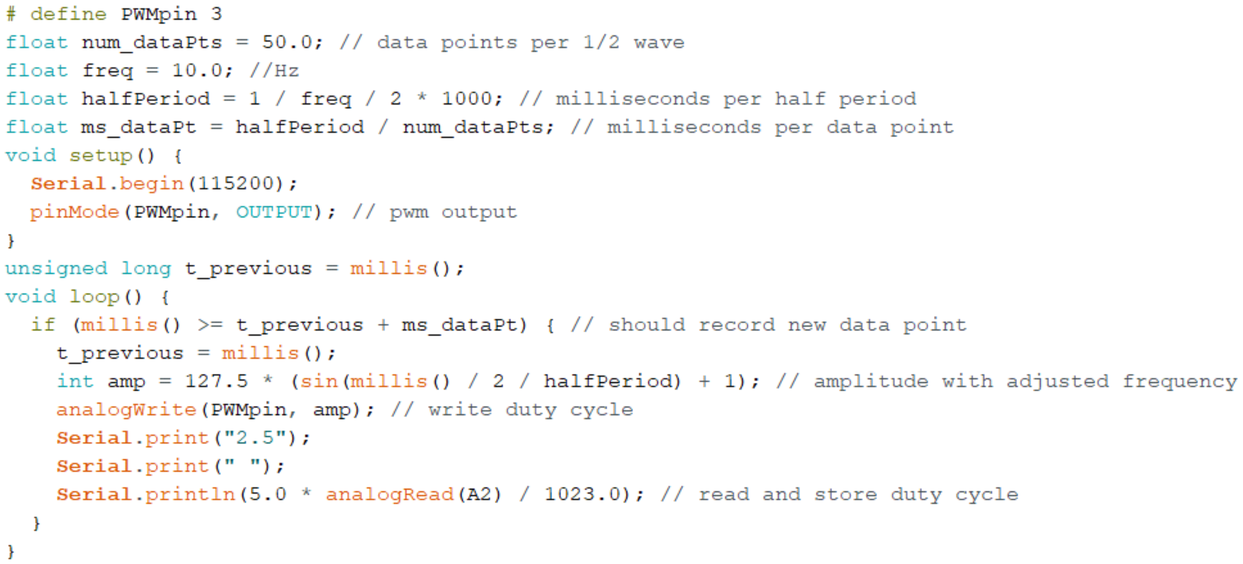
```
# define PWMpin 3
int amp = 0;
float num_dataPts = 50.0; // data points per 1/2 wave
float freq = 10.0; //Hz
float halfPeriod = 1 / freq / 2 * 1000; // milliseconds per half period
float ms_dataPt = halfPeriod / num_dataPts; // milliseconds per data point
int dAmp = 255.0 / num_dataPts; // change in "amplitude" per point (really change in duty cycle)

void setup() {
  Serial.begin(115200);
  pinMode(PWMpin, OUTPUT); // pwm output
}
unsigned long t_begin = millis();
unsigned long t_previous = millis();

void loop() {
  if (millis() >= t_previous + ms_dataPt) { // should record new data point
    t_previous = millis();
    if (millis() < t_begin + halfPeriod) { // should be increasing amplitude
      amp += dAmp;
    }
    else if (millis() < t_begin + 2 * halfPeriod) { // should be decreasing amplitude
      amp -= dAmp;
    }
    else { // new period
      t_begin = millis();
      amp = 0;
    }
    analogWrite(PWMpin, amp); // write duty cycle
    Serial.print("2.5");
    Serial.print(" ");
    Serial.println(5.0 * analogRead(A2) / 1023.0); // read and store duty cycle
  }}
```

b. To obtain a 1000Hz 3dB point, I used connected 560 ohm resistors in parallel to give an equivalent resistance of 140 ohms and a 105 ceramic capacitor for a value of 10^{-6} F. This gives a 3dB point of 1137Hz. My signal still looks very bad, but vaguely triangular with lots of noise.

c. To obtain a 10Hz 3dB point, I kept the four 560 ohm resistors and replaced the 105 ceramic capacitor with a 10 uF electrolytic capacitor. This produced a more clear triangular wave, though it still wasn't extremely clean. I tried changing the triangle wave frequency to 1Hz, and this produced a very clean triangle wave.

d. To obtain a 1Hz 3dB point, I replaced the four 560 ohm resistors with a 1k resistor in series with a 560 ohm resistor, giving a 3dB point of 1.02Hz. This produced a muddled, unclear pattern. However, when I changed the triangle wave frequency to 1Hz, it produced a nice triangle wave form similar to part C., indicating that the 1Hz 3dB low pass filter filtered out the 10Hz triangle wave I wanted to generate.

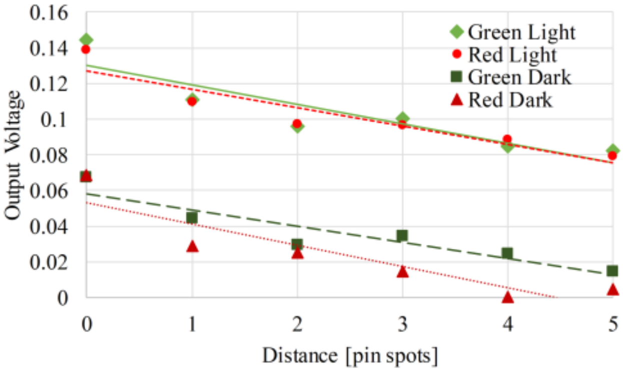
e. When I replaced my triangle wave with a sin wave, I ran into problems with the frequency of the sin wave which I fixed within the sin() function to adjust the frequency. My signal was still bad, until I realized by electrolytic capacitor had gotten loose and wasn't connected. Once I reconnected it, I observed the same pattern as with the triangle wave, where no filtering produced only 0 or 5 V, 1000Hz 3dB point gave a wave resembling a sine wave, and 10Hz gave a good looking sine wave. My results differed for the 1Hz 3dB point; for the sine wave, the result was still a clear sine wave, though I'm not sure why that would be.



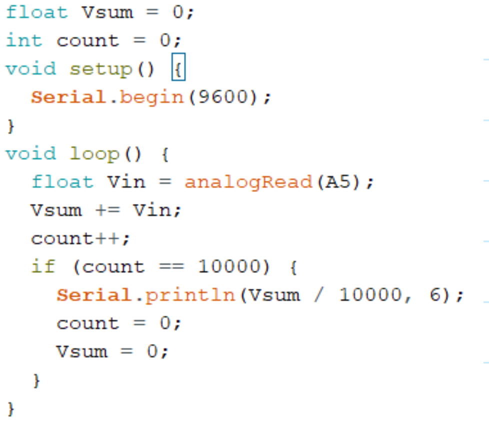
```
# define PWMpin 3
float num_dataPts = 50.0; // data points per 1/2 wave
float freq = 10.0; //Hz
float halfPeriod = 1 / freq / 2 * 1000; // milliseconds per half period
float ms_dataPt = halfPeriod / num_dataPts; // milliseconds per data point
void setup() {
  Serial.begin(115200);
  pinMode(PWMpin, OUTPUT); // pwm output
}
unsigned long t_previous = millis();
void loop() {
  if (millis() >= t_previous + ms_dataPt) { // should record new data point
    t_previous = millis();
    int amp = 127.5 * (sin(millis() / 2 / halfPeriod) + 1); // amplitude with adjusted frequency
    analogWrite(PWMpin, amp); // write duty cycle
    Serial.print("2.5");
    Serial.print(" ");
    Serial.println(5.0 * analogRead(A2) / 1023.0); // read and store duty cycle
  }
}
```

3.

a.



Phototransistor output voltage as a function of green or red LED facing directly into sensor at different distances and in a light room or a dark room. Trend lines are linear best fit and each data point is obtained from an average of 10,000 readings.



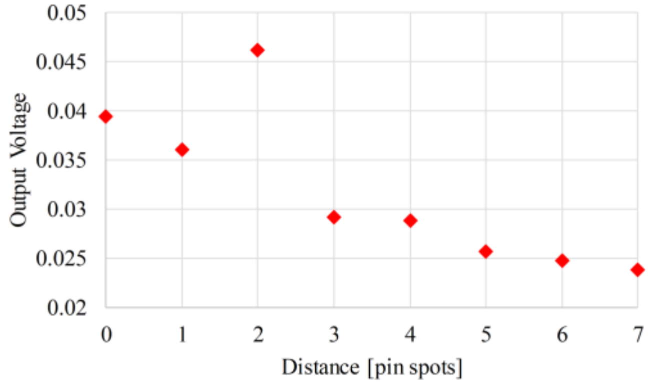
```
float Vsum = 0;
int count = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  float Vin = analogRead(A5);
  Vsum += Vin;
  count++;
  if (count == 10000) {
    Serial.println(Vsum / 10000, 6);
    count = 0;
    Vsum = 0;
  }
}
```

b. Based on the graphs, it does make a difference if the room is light or dark. In darkness, the output voltages are much lower.

c. I used two 560 ohm resistors in parallel, with a total resistance of 280 ohms, for the green LED, and one 10k resistor for the red LED to somewhat control for the luminosity. Both LEDs performed similarly, though the red LED produced slightly lower voltages. This could easily be a difference in the luminosity, as I only adjusted them visually, so I can't really make any reasonable conclusions about the phototransistor's discrimination between red or green LEDs.

d. There is an angular orientation dependence. As I turned the green LED upward, the output phototransistor voltage decreased. The maximum signal appears to be when the LED is aimed directly at the sensor, which makes sense based on what the spec sheets for the LED indicate about their luminescence as a function of viewing angle.

e. I can detect close by objects based on scattering off an object in an otherwise mostly dark room. The output value does increase as I move an object (the RGB LED from problem 1 [unplugged of course]) closer to the LED and sensor. However, because I have a partition between the emitting red LED and the phototransistor, at very close distances (i.e. 0 and 1 pin spots away, the reflecting surface is almost on the partition, so less light is able to reflect to the other side of the partition. It is possible that before this point, the reflected light follows a 1/r rule, though this could be mistaken.


Output Voltage
Distance [pin spots]