PH435 Lab 2

Sankalp Gambhir
180260032

September 13, 2020

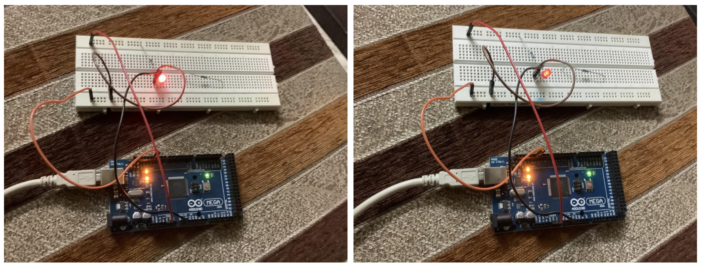
Note: I was playing around on my own Arduino Mega, forgot and took screenshots on it instead of the Uno. The code should still compile fine for the Uno, I did not use anything extreme. I hope it isn't an issue, I'd disconnected everything by the time I realised.

A. Background and buildup: `analogWrite()` function

A.1) **Pins 3, 5, 6, 9, 10, and 11.** Marked by ~ on the board itself, confirmed with the Arduino documentation.
The `analogWrite` system can be checked using an LED connected to it, followed by a large resistor, to ensure staying within current limits.

A.2) **0-255.** The value is limited by being an 8-bit unsigned interger type. A simple change in data type cannot mitigate this, since the counter on the Arduino itself is 8-bit only. There is supposedly another 16-bit counter per the ATmega328P datasheet, so not sure if that could be used to expand this further. However, in any case, the minute changes achievable by this method for the duty cycle might be bottlenecked by other hardware even *if* they were of any practical purpose.

A.3) Connected circuit is as in Figure 1.

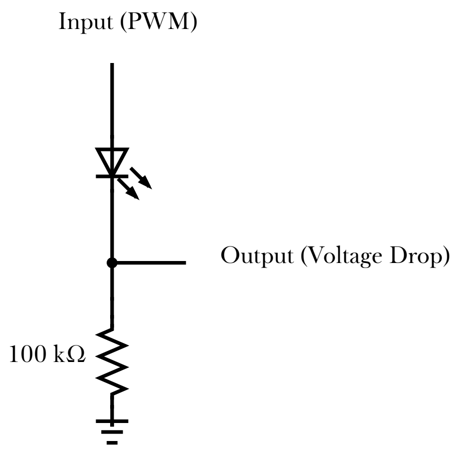


(a) Input `255/HIGH` (b) Input `10`

Figure 1: Connected PWM Circuit.

1. The PWM setup works by adjusting the duty cycle of a digital output to obtain an apparent average analog voltage. This exploits the fact that due to persistence of vision, the light output of the LED connected to this voltage will also be averaged out, to appear dimmer over a period significantly longer than the PWM frequency. A slow-mo camera can easily catch this.

2. The voltage is measured with the circuit in Figure 2a with the code in Figure 2b. A photo of the physical system has been attached already in Figure 1.
   The output can be inaccurate due to current flowing, so this is mitigated by simply using enormous resistors in series. The output received consistently relayed 2.01V, close enough for a red LED (expected 1.8-2.1V).



(a) Diagram of the circuit used.

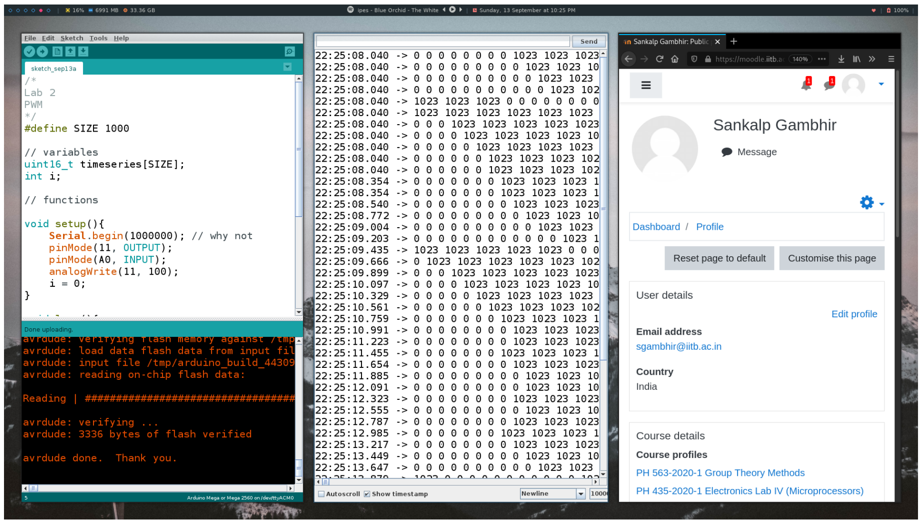
```
void setup(){
    Serial.begin(115200);   // why not
    pinMode(11, OUTPUT);    // PWM pin
    pinMode(A0, INPUT);     // Voltage measuring pin
    analogWrite(11, 130);
}

void loop(){
    // convert to voltage and subtract from Vcc to get the drop
    Serial.println(5 - 5 * ((float)analogRead(A0) / (float) 1023));
}
```

(b) Code sent to the Arduino

Figure 2: Measuring LED threshold voltage

A.4) We can store the output of the analog pin via an analog read (code in Appendix) and return it via serial after a set number of iterations. A possible other solution is to return the current output via serial every time it is read, with a baud rate of say, 1M, not sure if that would give more resolution, but it would be less reliable due to cables and USB interfaces being involved, not to mention, syncing serials of Arduino and the PC.

A rough calculation of the duty cycle can be done by counting the numbers of zeroes and 1023's obtained and taking their ratio. This was correct to within $3 - 4\%$ for values I tested, ranging between 50 and 200. The difference may be attributed to the frequency we have, getting around 20 readings for each cycle of the PWM wave.



```
/*
Lab 2
PWM
*/
#define SIZE 1000

// variables
uint16_t timeseries[SIZE];
int i;

// functions

void setup(){
    Serial.begin(1000000); // why not
    pinMode(11, OUTPUT);
    pinMode(A0, INPUT);
    analogWrite(11, 100);
    i = 0;
}
```

```
Done uploading.
avrdude: verifying flash memory against /tmp
avrdude: load data flash data from input fil
avrdude: input file /tmp/arduino_build_44309
avrdude: reading on-chip flash data:

Reading | ####################################

avrdude: verifying ...
avrdude: 3336 bytes of flash verified

avrdude done.  Thank you.
```

```
22:25:08.040 -> 0 0 0 0 0 0 0 0 1023 1023 1023
22:25:08.040 -> 0 0 0 0 0 0 0 0 0 1023 1023 10
22:25:08.040 -> 0 0 0 0 0 0 0 0 0 0 1023 1023
22:25:08.040 -> 0 0 0 0 0 0 0 0 0 0 0 1023 102
22:25:08.040 -> 1023 1023 1023 0 0 0 0 0 0 0 0
22:25:08.040 -> 1023 1023 1023 1023 1023 1023
22:25:08.040 -> 0 0 0 1023 1023 1023 1023 1023
22:25:08.040 -> 0 0 0 0 1023 1023 1023 1023 10
22:25:08.040 -> 0 0 0 0 0 1023 1023 1023 1023
22:25:08.040 -> 0 0 0 0 0 0 1023 1023 1023 102
22:25:08.040 -> 0 0 0 0 0 0 1023 1023 1023 102
22:25:08.354 -> 0 0 0 0 0 0 0 1023 1023 1023 1
22:25:08.354 -> 0 0 0 0 0 0 0 1023 1023 1023 1
22:25:08.540 -> 0 0 0 0 0 0 0 0 1023 1023 1023
22:25:08.772 -> 0 0 0 0 0 0 0 0 0 1023 1023 10
22:25:09.004 -> 0 0 0 0 0 0 0 0 0 0 1023 1023
22:25:09.203 -> 0 0 0 0 0 0 0 0 0 0 0 0 1023 1
22:25:09.435 -> 1023 1023 1023 1023 1023 0 0 0
22:25:09.666 -> 0 1023 1023 1023 1023 1023 102
22:25:09.899 -> 0 0 0 1023 1023 1023 1023 1023
22:25:10.097 -> 0 0 0 0 1023 1023 1023 1023 10
22:25:10.329 -> 0 0 0 0 0 1023 1023 1023 1023
22:25:10.561 -> 0 0 0 0 0 0 1023 1023 1023 102
22:25:10.759 -> 0 0 0 0 0 0 0 1023 1023 1023 1
22:25:10.991 -> 0 0 0 0 0 0 0 0 1023 1023 1023
22:25:11.223 -> 0 0 0 0 0 0 0 0 1023 1023 1023
22:25:11.455 -> 0 0 0 0 0 0 0 1023 1023 1023 1
22:25:11.654 -> 0 0 0 0 0 0 0 0 1023 1023 1023
22:25:11.885 -> 0 0 0 0 0 0 0 0 0 1023 1023 10
22:25:12.091 -> 0 0 0 0 0 0 0 0 0 1023 1023 10
22:25:12.323 -> 0 0 0 0 0 0 0 0 1023 1023 1023
22:25:12.555 -> 0 0 0 0 0 0 0 0 0 1023 1023 10
22:25:12.787 -> 0 0 0 0 0 0 0 0 1023 1023 1023
22:25:12.985 -> 0 0 0 0 0 0 0 1023 1023 1023 1
22:25:13.217 -> 0 0 0 0 0 0 0 0 1023 1023 1023
22:25:13.449 -> 0 0 0 0 0 0 0 0 0 1023 1023 10
22:25:13.647 -> 0 0 0 0 0 0 0 0 0 0 1023 1023
```

Sankalp Gambhir

Dashboard / Profile

User details

Email address
sgambhir@iitb.ac.in

Country
India

Course details

Course profiles
PH 563-2020-1 Group Theory Methods
PH 435-2020-1 Electronics Lab IV (Microprocessors)

A.5) The question was quite confusing about its intention, so I am assuming we had to double the value of a DC analog signal after the PWM filtering. We can simply use a voltage pump to double the analog voltage. See Figure 3. The PWM wave itself can be easily used as a clock here.

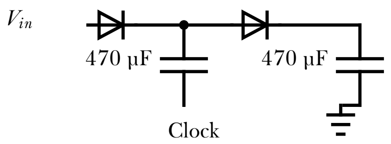


Figure 3: Voltage doubler.

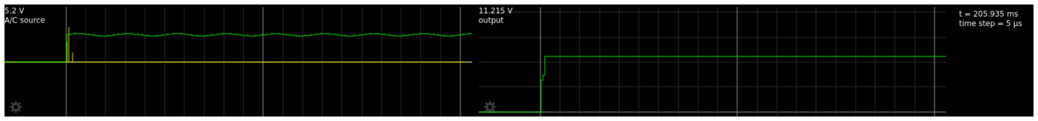


Figure 4: Voltage doubler simulation.

Appendix

Arduino

The code uploaded to the Arduino for testing. The size was defined as a compiler directive for easily changing it everywhere without any issues.

```
/*
Lab 2
PWM
*/
#define SIZE 1000

// variables
uint16_t timeseries[SIZE];
int i;

// functions

void setup(){
    Serial.begin(1000000); // why not
    pinMode(11, OUTPUT);
    pinMode(A0, INPUT);
    analogWrite(11, 100);
    i = 0;
}

void loop(){
  if(i == SIZE){
    for(int j=0; j<i;j++){
        Serial.print(timeseries[j]);
        Serial.print(' ');
      }
    i = 0;
    Serial.print('\n');
    }
  else{
    timeseries[i] = analogRead(A0);
    i++;
  }
}
```