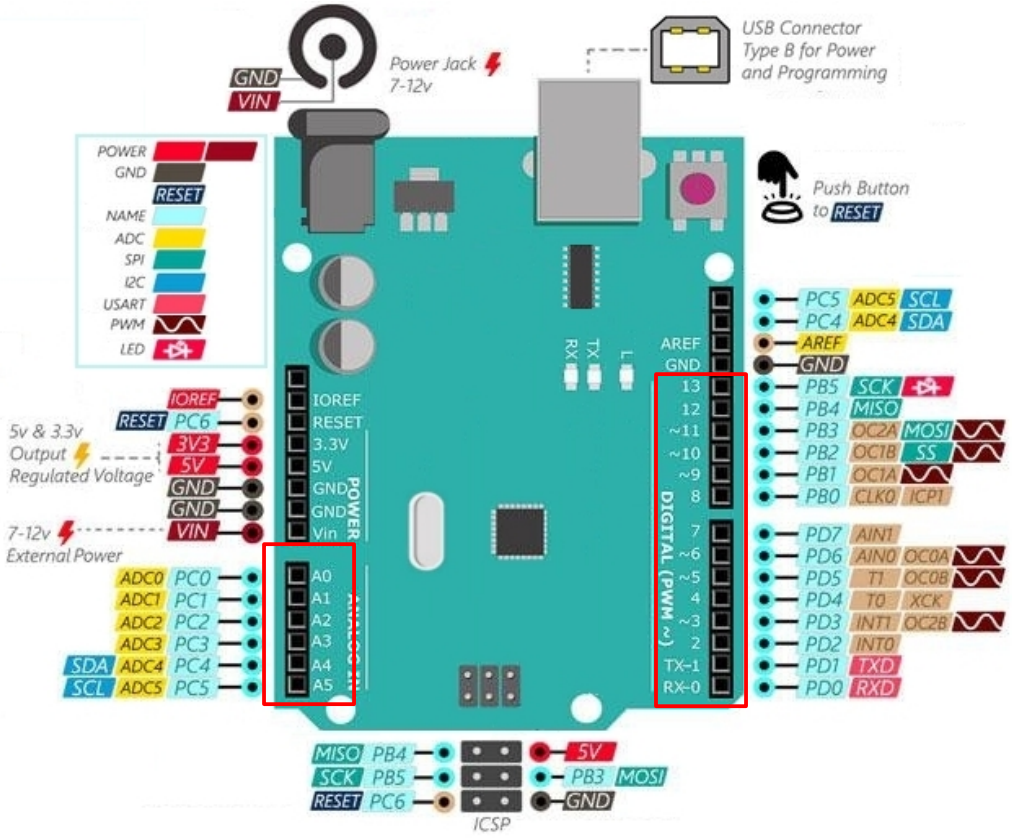
ATMEGA328/Arduino Uno – I/O Pins – OUTPUT PWM
https://github.com/teaksoon/lmaewapm



We can only supply 0V or 5V at full Current to the OUTPUT I/O Pins from our Program Codes

What if we want need supply something between the 0V and 5V or between no-Current and full Current to our OUTPUT I/O Pin?

Answer is: We cannot supply anything other than 0V or 5V or full Current directly to our OUTPUT I/O Pin with our Program Codes BUT we can “fake it” by using PWM – Pulse Wave Modulation

Example: We want to supply a “fake” HALF full Voltage or HALF full Current to our OUTPUT I/O Pin with our Program Codes

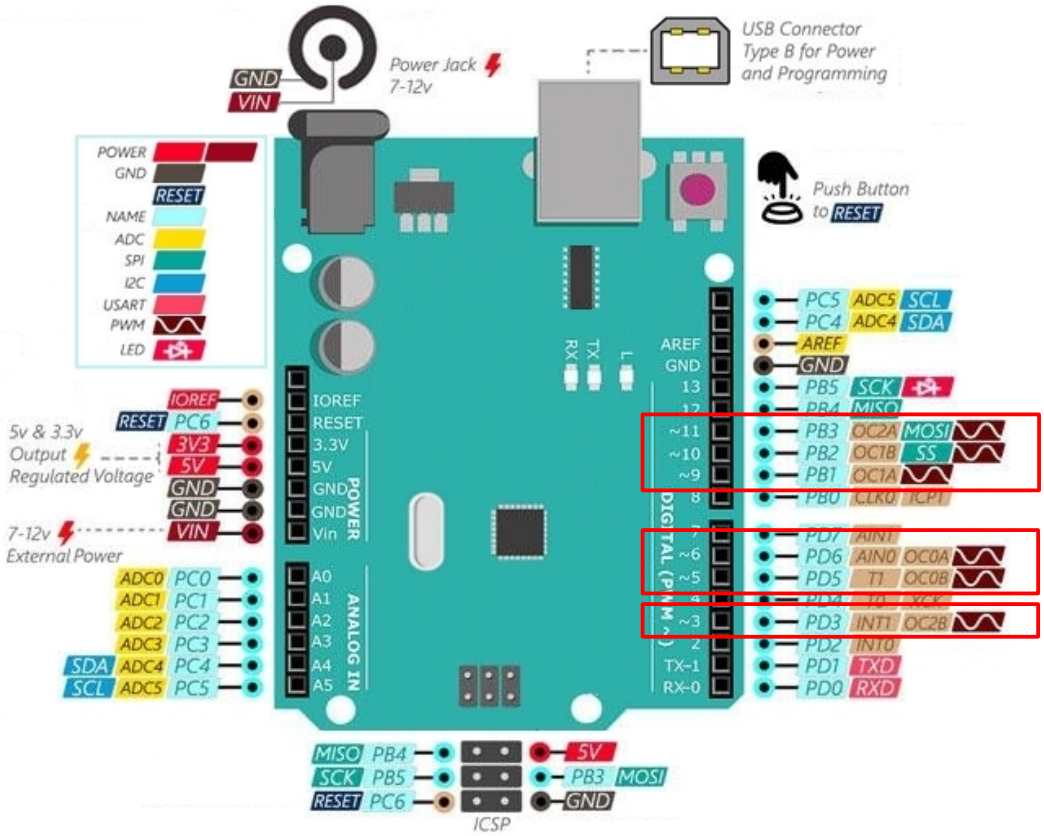
We make the OUTPUT I/O Pin to have 0V, then followed by a 5V and then repeat them forever at high-speed with a precise timing. The VOLTAGE on that OUTPUT I/O Pin will be become “0V,5V,0V,5V,... We will no longer have the “feeling” of a full 0V or full 5V or a full Current on that Pin. The “feeling” will be something in the middle, because we are getting an alternating 0V(no-current) and 5V(full-current) at high-speed on that OUTPUT I/O Pin

Alternating between 0V and 5V is called “PWM at 50% duty cycle”

Instead of repeating the “0V,5V” Voltage combo, we can repeat a different Voltage combo, for example: with “0V,0V,5V” Voltage combo, we will get a different PWM effect

Repeating “0V,0V,5V” combo is called “PWM at 25% duty cycle”
or
Repeating “5V,5V,0V” combo is called “PWM at 75% duty cycle”
...
We can use any OUTPUT I/O Pin to do the PWM. Making different “PWM at n% duty cycle” manually can be quite troublesome. Fortunately, the **ATMEGA328 micro-controller have special OUTPUT I/O Pins that can do the PWM for us** and in the “C-Language with Arduino Library”, there is a function called **analogWrite()**. This function make it very easy for us to make the PWM OUTPUT I/O Pin to run different PWM % duty cycle

ATMEGA328/Arduino Uno – I/O Pins – OUTPUT PWM
https://github.com/teaksoon/lmaewapm



NOT all OUTPUT I/O Pin has the built-in PWM abilites that can be used with the Arduino **analogWrite() function**. There are **6 PWM Pins on the Arduino Uno board**, they are labelled with "~" Symbol (**~3,~5,~6,~9,~10,~11**) or ATMEGA328 Pin PD3,PD5,PD6,PB1,PB2,PB3

After any of those Pins is **set as OUTPUT I/O Pin**, We can use the **analogWrite()** function from the Arduino Library to do PWM on the selected PWM OUTPUT I/O Pin for us

To run the PWM at precise timing, it needs clock timers, the ATMEGA328 has 3 internal timers (Timer 0,Timer 1,Timer 3). Below are the Timers used by each of the PWM Pins (When PWM is running at those pin, avoid using the affected timers for other purpose).

```
PWM Pin  ~3,PD3 uses Timer 2
PWM Pin  ~5,PD5 uses Timer 0
PWM Pin  ~6,PD6 uses Timer 0
PWM Pin  ~9,PB1 uses Timer 1
PWM Pin ~10,PB2 uses Timer 1
PWM Pin ~11,PB3 uses Timer 2
Note: delay(),millis(),micros() uses Timer 0; tone() uses Timer 2;
Servo Class uses Timer 1
```

analogWrite(pinNumber, value) function takes 2 parameter for PWM generation.

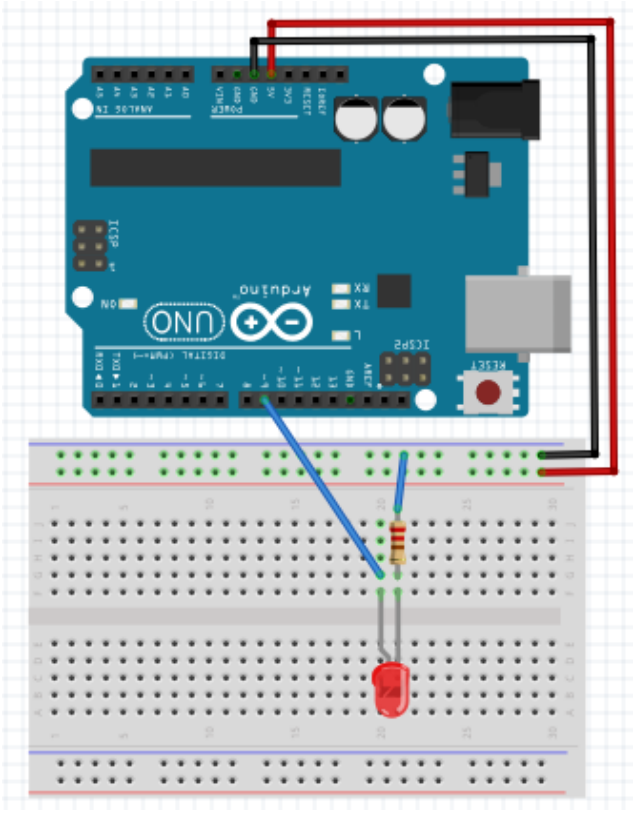
pinNumber – can by any of the 6 PWM Pins, 3,5,6,9,10,11

value – is a number range from 0 to 255

```
255 represents PWM at 100% duty cycle 5V
  0 represents PWM at   0% dity cycle 0V
191 represents PWM at  75% duty cycle between 2.5V and 5V
127 represents PWM at  50% duty cycle between 0V and 5V
 64 represents PWM at  25% duty cycle between 0V and 2.5V
```

We can choose any number between 0 and 255, the %duty cycle will be adjusted accordingly for us internally

ATMEGA328/Arduino Uno – I/O Pins – OUTPUT PWM
https://github.com/teaksoon/lmaewapm



1x Computer with Arduino IDE Software
1x USB 2.0 Type A/B Data Cable
1x Arduino Uno Board
Jumper Wires

1x Solderless Breadboard
1x 5mm bulb LED
1x 220 Ohm resistor

LED +ve to Arduino Pin 9
LED -ve to Resistor to Arduino GND

Upload the following Program and watch the LED.

You may also open the Serial Monitor to see the PWM value that is being fed into the analogWrite() function, at the same time see the effects on the LED

```
#define PWM_START    0    // 0 to 255
#define PWM_END     30    // 0 to 255

int pwmDir   = 1;
int pwmValue = 0;

void setup() {
  pinMode(9, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  if (pwmValue < PWM_START || pwmValue > PWM_END) {
    pwmValue = pwmValue-pwmDir-pwmDir;
    pwmDir   = -pwmDir;
    delay(100);
  }
  Serial.print("\nPWM Value = ");Serial.print(pwmValue);
  analogWrite(9,pwmValue);
  pwmValue += pwmDir;
  delay(100);
}
```

The LED gradually increase in brightness and then gradually decrease in brightness. They are just different rate of a very fast “flickers”.

That is the effect of the PWM, alternating between 0V and 5V at different “Duty Cycle”.

We can also use this to control many other device, when we need to “vary” the Voltage and Current to control things like sound, speed and etc...