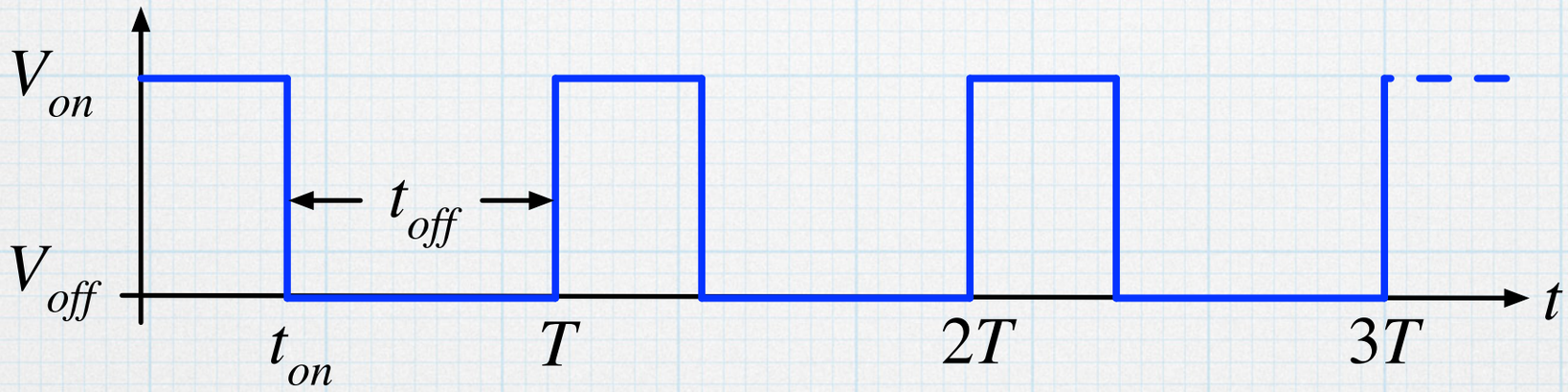
# Pulse-Width Modulation (PWM)

Pulse-width modulation is used in many switching situations, particularly where switching is used to control analog voltages and currents.

A PWM signal is pulse wave, in which the high time can controlled (modulated) by an external signal.

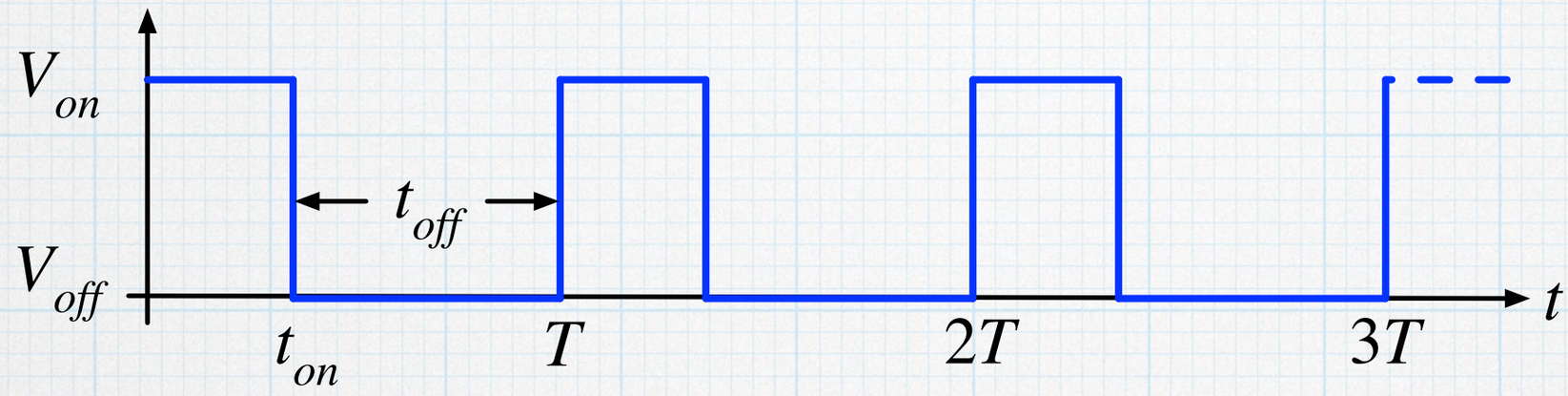
Since PWM is often used to control a switch, we use "on" and "off" to characterize the two states, rather than "high" and "low".



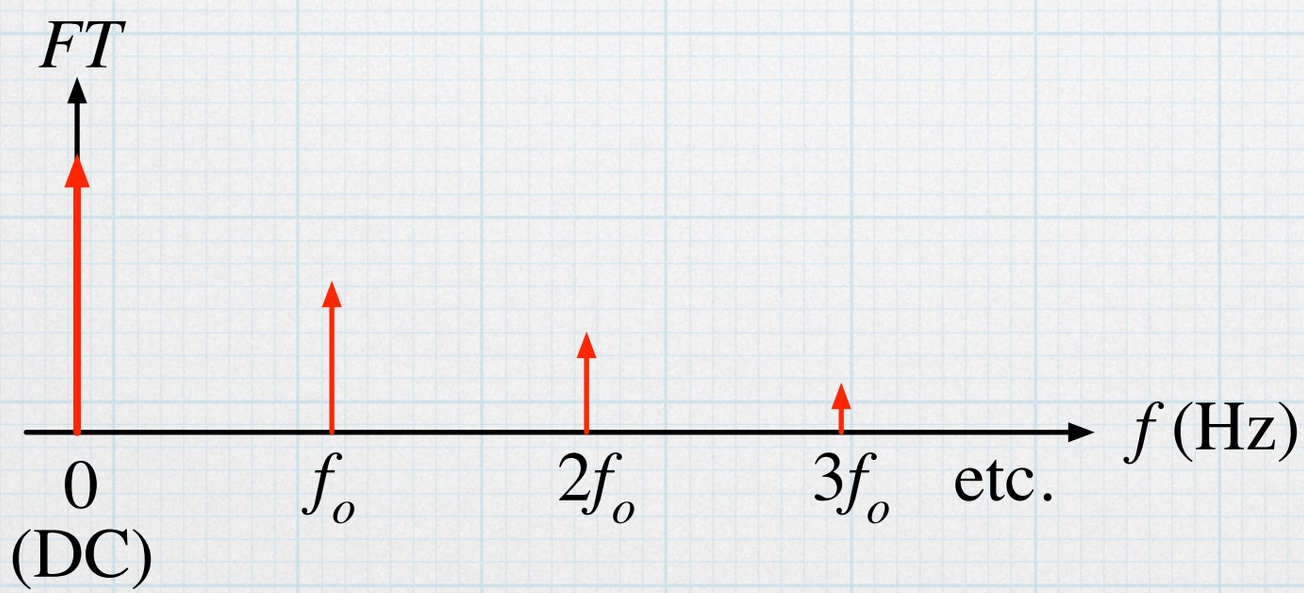

The frequency (and period) of the signal is fixed. The frequencies can range from a few 10s of Hz to several MHz. The corresponding periods are a few milliseconds to a few tenths of a microsecond. Of course, the speed will the type of circuit needed to generate the pulse train.

Obviously, $t_{on} + t_{off} = T$ — as $t_{on}$ changes, $t_{off}$ must change in the opposite fashion. Theoretically, the "high" time can be varied from 0 to $T$, but sometimes there may limitations that prevent $t_{on}$ from ranging all the way to the extreme values.

PWM signals are usually described in terms of a *duty cycle,* defined as $D = t_{on}/T$. The value of the duty cycle ranges between 0 and 1. Then, $t_{on} = D \cdot T$ and $t_{off} = (1 - D)T$.




Thinking in EE 224 terms, the Fourier Transform of the signal will have a DC value and harmonics based on the fundamental frequency of the signal.




As the duty cycle changes, the relative heights of the DC and harmonic "spikes" will change, but the frequencies are fixed.

The DC value is the average value of the waveform is easily calculated:

$$v_{DC} = \frac{1}{T}\left[\int_0^{t_{on}} V_{on}dt + \int_{t_{on}}^{T} V_{off}dt\right]$$

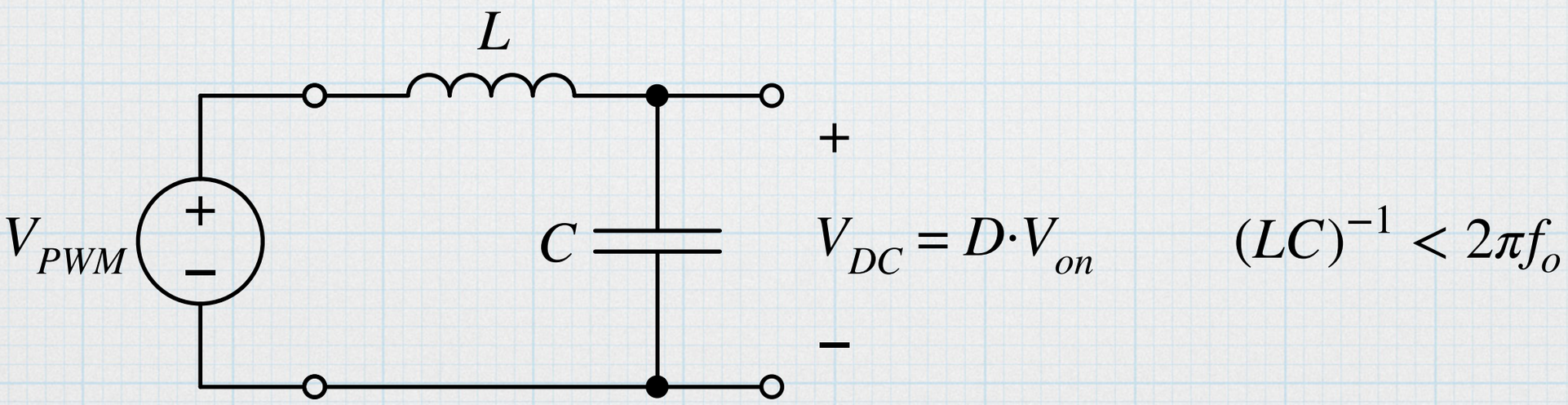
$$= V_{on}\left(\frac{t_{on}}{T}\right) + V_{off}\left(\frac{T - t_{on}}{T}\right)$$

$$= V_{on} \cdot D + V_{off} \cdot (1 - D)$$

If $V_{off} = 0$, then $V_{DC} = D \cdot V_{on}$.

This is a very nice result — a variable DC voltage controlled by the duty cycle of the PWM signal.

We can extract the DC easily by using a low-pass filter with $f_c << f_o$ to attenuate the harmonic components.



Generating PWM signals is a standard feature of most microcontrollers. For example, Arduino has several pins ready for pwm. Use the analogWrite function to set the pwm. The default frequency is 490 Hz, but that can be changed, if needed. The pwm value can range between 0 and 255. ( 0 for $D = 0$ and 255 for $D = 1$. ) The program below ramps the level from min to max and back repeatedly.

pwm_demo | Arduino 1.8.16

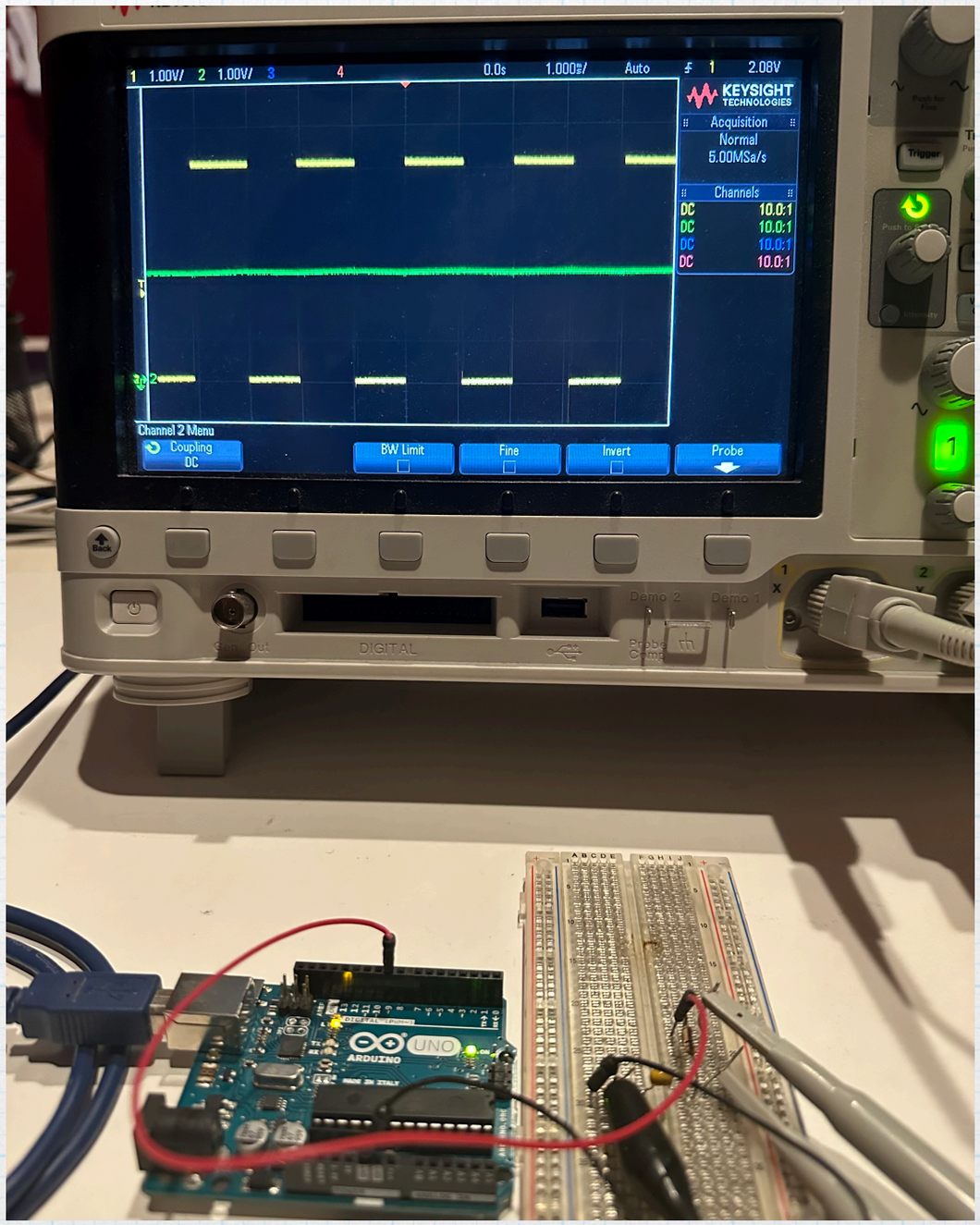
pwm_demo §

```
int pwm_pin = 9;       // the PWM pin
int pwm_level = 0;     // currrent pwm level
int fadeAmount = 5;    // how much to change by

void setup() {
  pinMode(pwm_pin, OUTPUT); // declare pin 9 to be an output:
}

void loop() {
  analogWrite(pwm_pin, pwm_level);  //set the level

  pwm_level = pwm_level + fadeAmount;  //increment for the next iteration

  // reverse the direction oat the end
  if (pwm_level <= 0 || pwm_level >= 255)
    fadeAmount = -fadeAmount;

  delay(60);   // wait for 60 milliseconds so we humans can see it
}
```

Done uploading.

```
Sketch uses 1144 bytes (3%) of program storage space. Maximum is 32256 bytes.
Global variables use 13 bytes (0%) of dynamic memory, leaving 2035 bytes for local vari
```

19 Arduino Uno on /dev/cu.usbmodem2201




1 1.00V/ 2 1.00V/ 3 4
0.0s
1.000s/
Auto
1 2.08V
KEYSIGHT TECHNOLOGIES
Acquisition
Normal
5.00MSa/s
Channels
DC 10.0:1
DC 10.0:1
DC 10.0:1
DC 10.0:1
Channel 2 Menu
Coupling
DC
BW Limit
Fine
Invert
Probe

There are simple circuits that can be used to generate PWM signals. A standard method is to generate a periodic ramp or triangle wave and use a comparator to produce the PWM square wave.

The upper two op amps form an oscillator circuit the sets the frequency and provides square- and triangle-wave outputs. (See EE 230 notes for a detailed explanation.)

The lower op amp is a simple comparator. Changing the voltage at the reference input changes the PWM duty cycle.