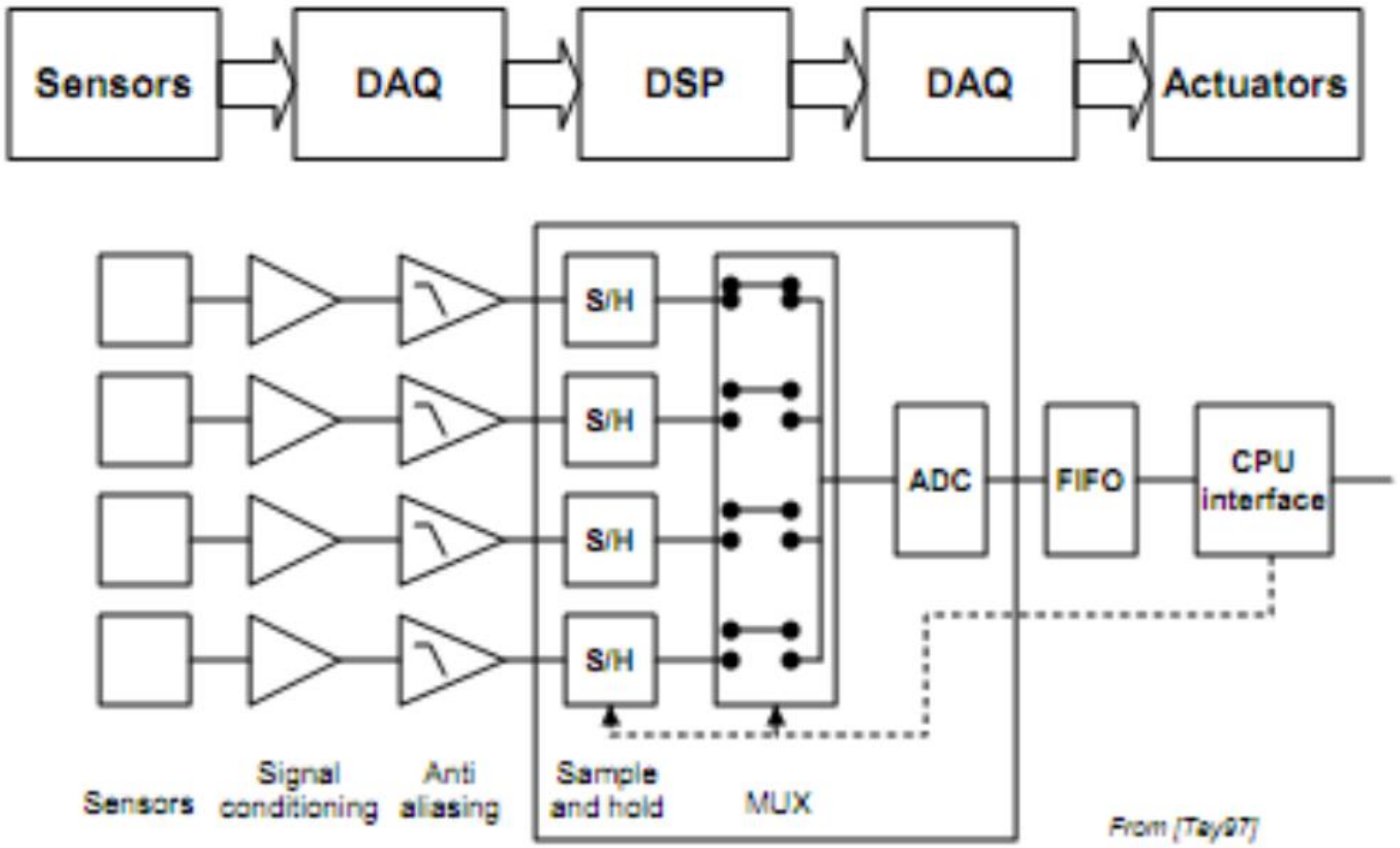
Data Acquisition II

- Sample and hold
- Multiplexing
- Analog to digital conversion
- Digital to analog conversion

Architecture of data acquisition systems



From [Tay97]

ANALOG $\Leftrightarrow$ DIGITAL CONVERSION

Physical world is analog (mostly)

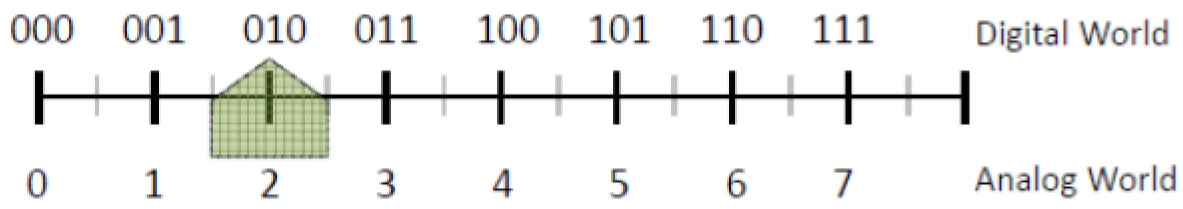
Analog is infinitely variable, while digital is discrete in both time and value

Analog/digital conversion can be part of the instrument or part of the actuator, e.g.

- Incremental encoder – analog position gets encoded into digital pulse
- Stepper motor – digital step pulse gets transformed to analog position

ANALOG ⇔ DIGITAL CONVERSION

To be processed by computers, all information must be converted to a binary representation possessing a finite number of distinct values that combine logical low (0) and logical high (1):



Digital coding is arbitrary, but natural binary order is most often used

ANALOG ⇔ DIGITAL CONVERSION

For notational compactness and ease of reading, binary (base 2) representations are often expressed in octal (base 8) or hexadecimal (base 16):

$$10100110_2 = 0246_8 = 166_{10} = A6_{16}$$

Arduino code:

Binary:	`B10100110`	(leading '`B`', only 8-bit values)
Octal:	`0246`	(leading '`0`')
Decimal:	`166`	(no formatting needed)
Hexadecimal:	`0xA6`	(leading "`0x`", chars `0-9`, `A-F`, `a-f`)

DIGITAL REPRESENTATION

When using natural binary order, the digital code consists of an *N*-bit binary word:

$$b_{N-1}\ b_{N-2}\ b_{N-3}\ \ldots\ b_1\ b_0$$

where b_{N-1} is the most-significant bit (MSB) and b_0 is the least-significant bit (LSB)

8-bit digital word: ***byte*** (or octet).

4-bit digital word: ***nibble*** (or quartet).

DIGITAL REPRESENTATION

The fractional value of the digital word is:

$$2^{-1}\, b_{N-1} + 2^{-2}\, b_{N-2} + 2^{-3}\, b_{N-3} + \dots + 2^{-N}\, b_0$$

with a range of $0 \longrightarrow (1 - 2^{-N})$ and a precision of 2^{-N}

Digital fractions sometimes notated with leading point notation:

$$.1011_2 = .6875_{10}$$

Example:

- 4-bits
- Range $= 0 \longrightarrow (1 - 2^{-4}) = 0 \longrightarrow \frac{15}{16}$ (nearly 1)
- Precision (absolute and relative) $= 2^{-4} = \frac{1}{16}$
- Code of 1011 $= \frac{1}{2} + \frac{1}{8} + \frac{1}{16} = \frac{11}{16} = .6875$

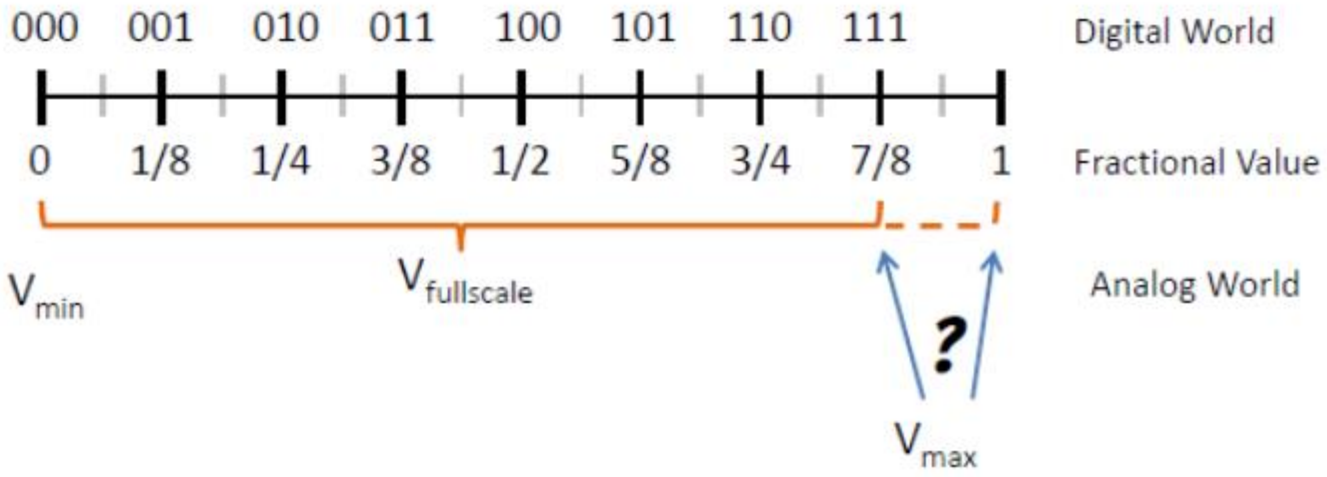
DIGITAL REPRESENTATION

- Digital codes have no inherent units, so scaling must be defined for code to have real-world meaning.

- How to assign relationship between binary code and analog values? It depends on the application…

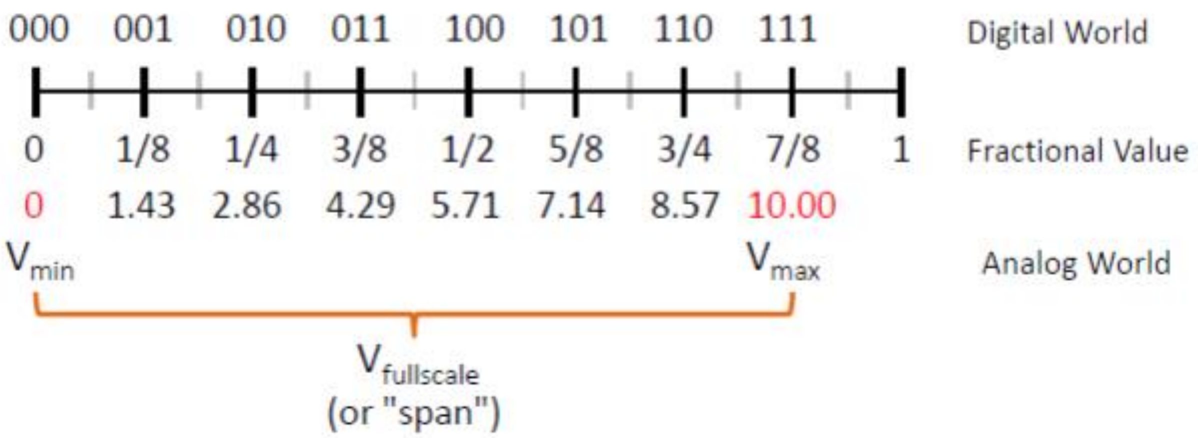
DIGITAL REPRESENTATION

For a unipolar signal, it seems obvious to associate V_{min} with binary zero. Do we associate V_{max} with binary 2^N (fractional value of 1), or with binary 2N-1 (and its fraction value of $1-2^N$)?



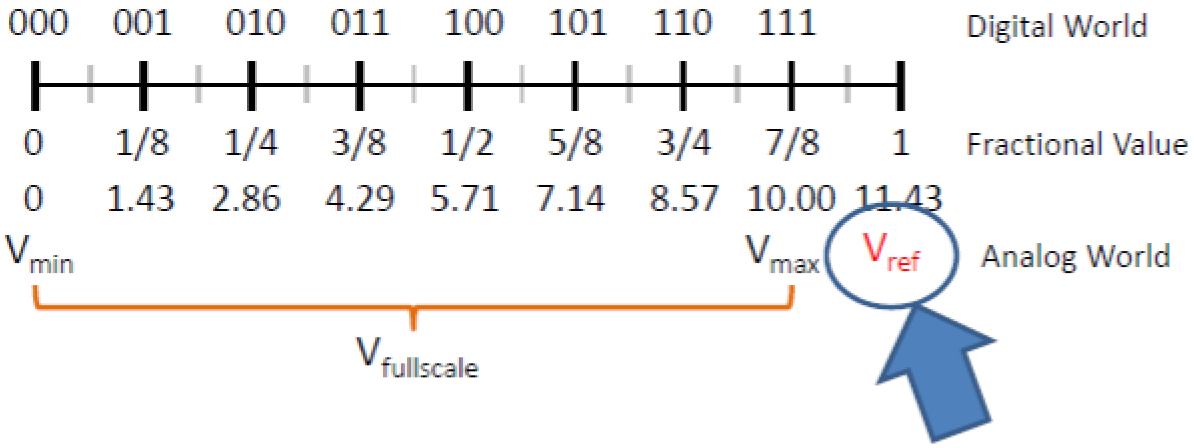
DIGITAL REPRESENTATION

To maximize the conversion range for N-bit coding of a *unipolar* value, we often associate V_{min} with binary zero and V_{max} with binary 2^N-1. For example, for a 0-10V signal, we may assign values such that:



DIGITAL REPRESENTATION

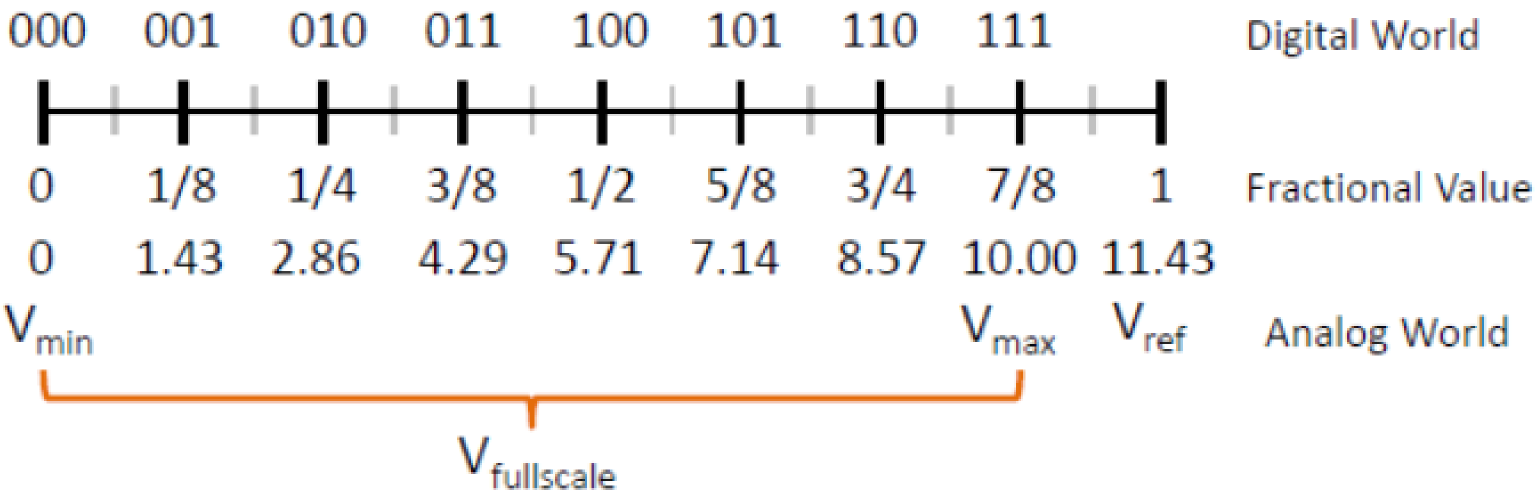
However, it is common to associate a reference value of V_{ref} with the digital code used to denote a fraction of 1



DIGITAL REPRESENTATION

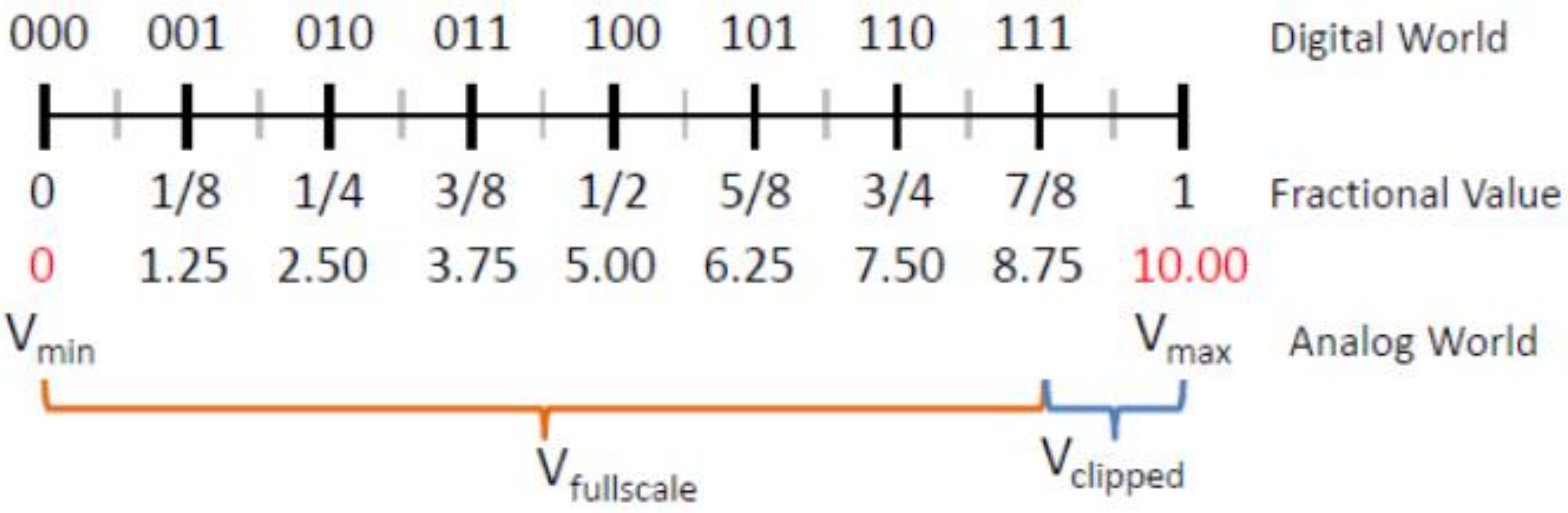
In such an arrangement, with V_{min} = 0,

$$V_{\text{fs}} = V_{\text{max}} = V_{\text{ref}}\left(\frac{2^N - 1}{2^N}\right)$$



DIGITAL REPRESENTATION

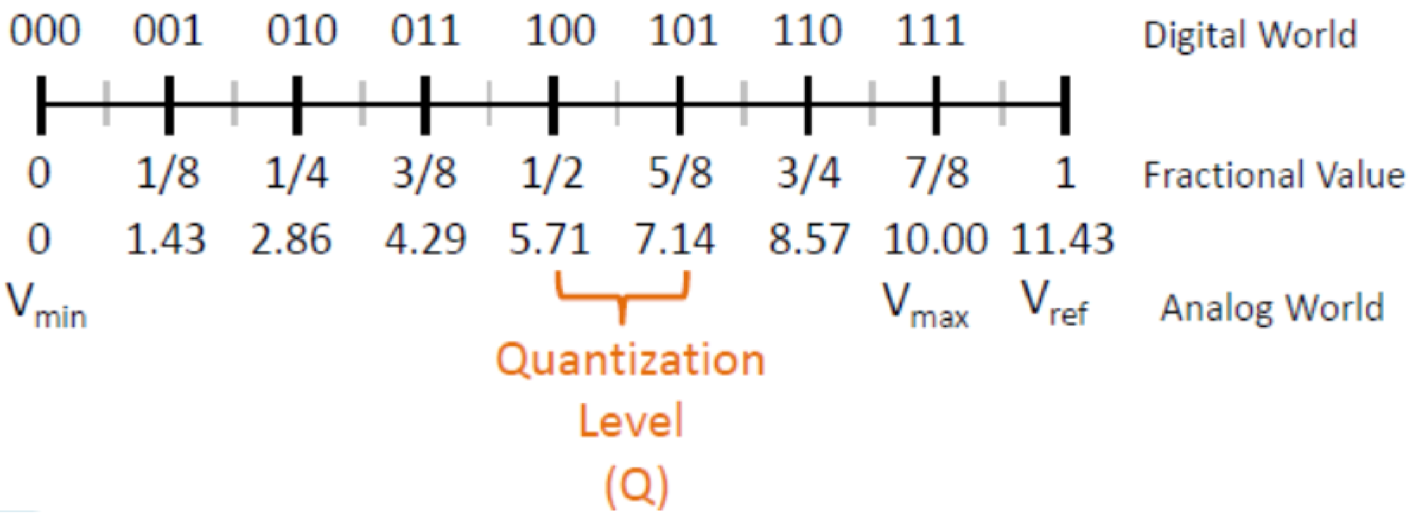
If V_{min} is less than binary zero, or V_{max} is more than binary 2^N-1, then *amplitude clipping* may result.



DIGITAL REPRESENTATION

Quantization Level: change in analog output associated with each discrete step

- We assume this step size to be fixed, although in sophisticated applications, it may vary across the signal range, or adapt to signal characteristics.



DIGITAL REPRESENTATION

Quantization Level (Q)

- Signal change associated with LSB
- For an N-bit converter:
  - $Q = \frac{V_{\mathbf{fullscale}}}{2^N - 1} = \frac{V_{\mathbf{ref}}}{2^N}$
- Quantization ***error*** ranges from $-\frac{Q}{2}$ to $+\frac{Q}{2}$

Example:

- 4-bits, 12 volts full-scale, binary code of 1001
- $Q = 12V / (2^N - 1) = .8V$
- $|Q_{\text{error}}| \leq .4V$
- $V_{analog} = Q \times (8 \cdot 1 + 4 \cdot 0 + 2 \cdot 0 + 1) = .8V \times 9 = 7.2V$

DIGITAL REPRESENTATION

Quantization

- An irreversible process
- A source of information loss
- Increasing the number of bits lowers information loss, but usually raises the cost and processing time

Example:

- 10 volts full-scale, with 1% precision required
- $Q = 1\% \cdot 10 = 0.1V$
- $N \geq \log_2 [V_{FS}/Q + 1] = 6.7$ bits $\Rightarrow$ at least 7 bits

INTEGER CODES

Unipolar Voltages

- Can be coded to unsigned integers
- Example – 0-5 volts coded to 3 bit unsigned integer

Voltage	Digital Value	Decimal Equivalent
0	000	0
0.71	001	1
1.43	010	2
2.14	011	3
2.86	100	4
3.57	101	5
4.29	110	6
5	111	7

INTEGER CODES

Bipolar Coding

- Two's Complement

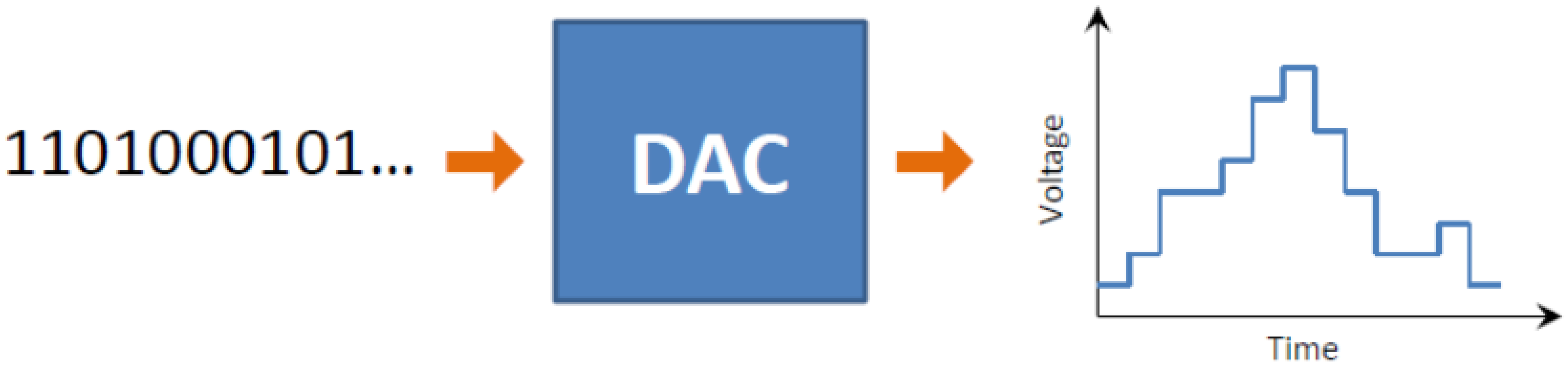
Voltage(1)	Voltage(2)	Digital Value	Decimal Equivalent
+3.75	+5	011	3
+2.50	+3.33	010	2
+1.25	+1.67	001	1
0	0	000	0
-1.25	-1.67	111	-1
-2.50	-3.33	110	-2
-3.75	-5	101	-3
-5	--	100	-4

- MSB is treated with a weighting of -2^{N-1}

$$-3 = -2^{(3-1)} + 1 = -2^2 + 1 = -4 + 1 \rightarrow 101$$

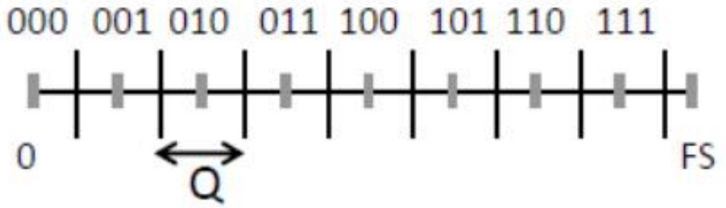
DIGITAL-TO-ANALOG CONVERTER (DAC)

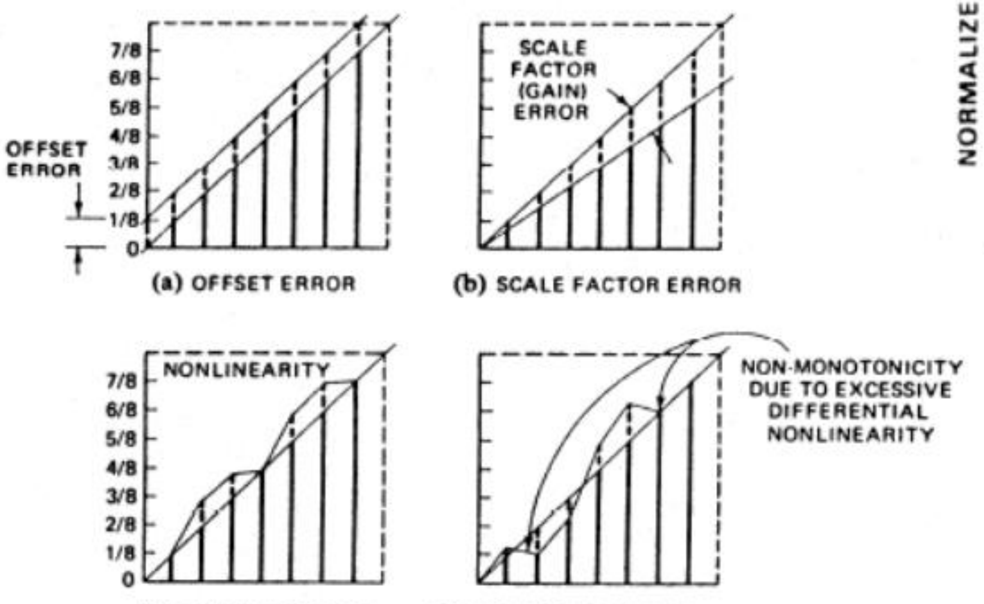
Converts digital values to analog outputs of either voltage or current



DIGITAL-TO-ANALOG (D/A) CONVERSION

Ideal DA Conversion:



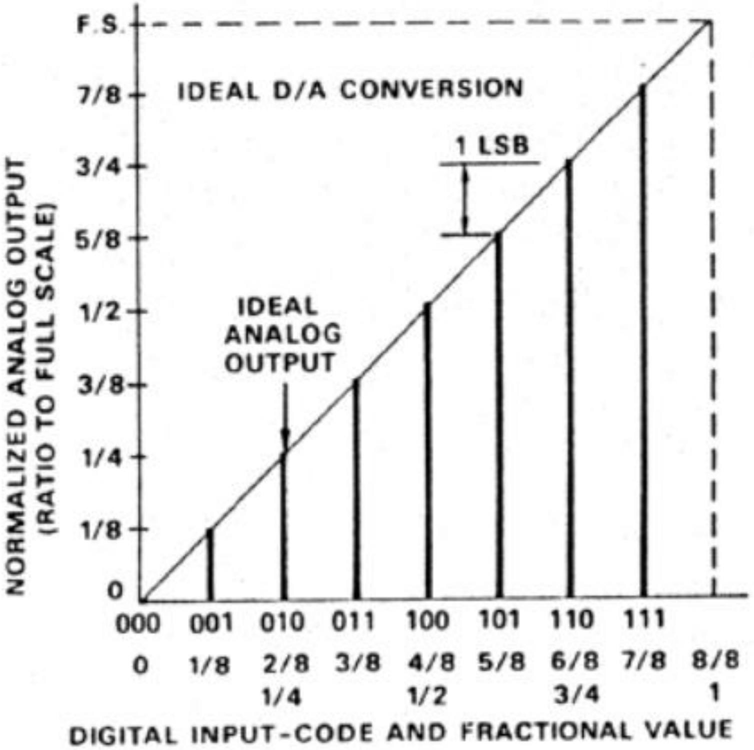


(a) OFFSET ERROR

(b) SCALE FACTOR ERROR

(c) LINEARITY ERROR

(d) NON-MONOTONICITY (DUE TO EXCESSIVE DIFFERENTIAL NONLINEARITY)



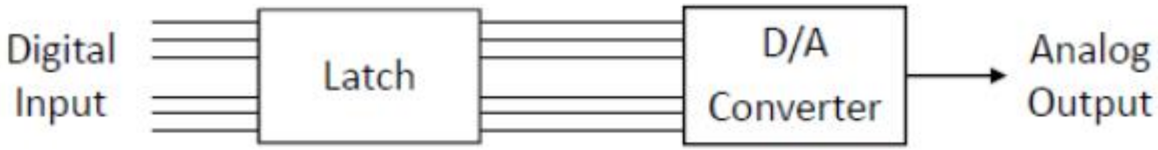
DIGITAL-TO-ANALOG (D/A) CONVERSION

Digital value is stored in a register (latch), then converted. Duration of conversion is called *settling time*.

Output of the DAC remains the same until the next value is sent to the register (latch) – a zero-order hold.

Basic concept:

$$V_{OUT} = \underbrace{(b_{N-1} \cdot 2^{N-1} + b_{N-2} \cdot 2^{N-2} + \cdots + b_1 \cdot 2^1 + b_0 \cdot 2^0)}_{\text{Integer equivalent of binary code}} \cdot Q$$



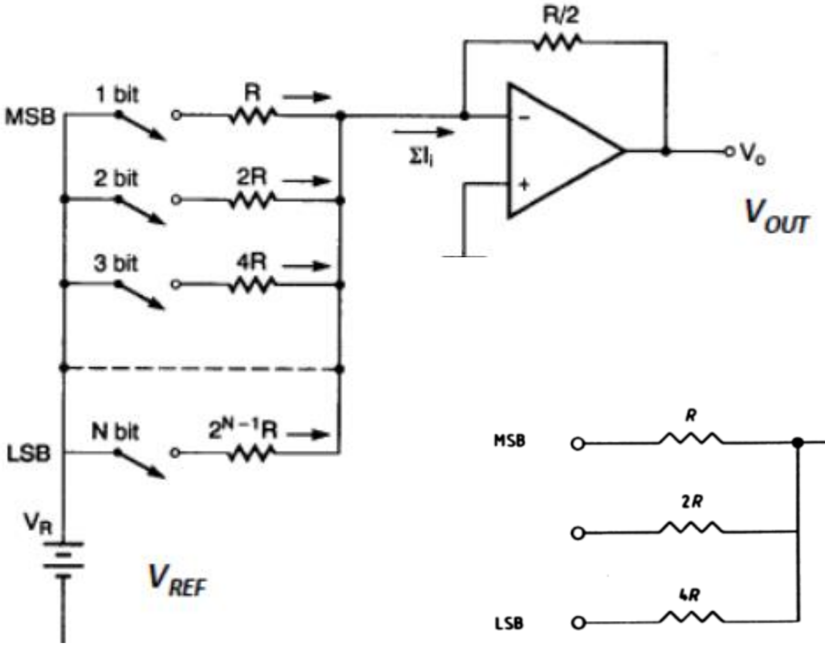
DIGITAL-TO-ANALOG (D/A) CONVERSION

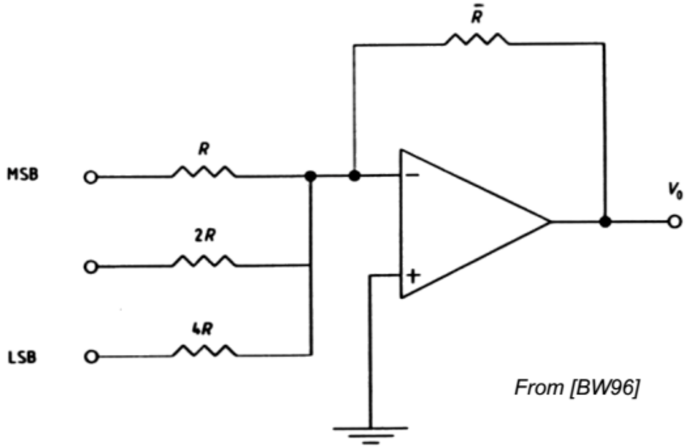
Weighted (Scaled) Resistor DAC

- Fast, but not practical for high bit count due to expense of high precision resistors across a wide magnitude range
- Virtual ground at inverting op-amp input
- Requires
  - Accurate reference voltage.
  - Higher precision resistor.

$$V_{OUT} = -\Sigma I_i \frac{R}{2} = -V_{\text{REF}}\left(\frac{b_{N-1}}{2} + \frac{b_{N-2}}{4} + \cdots + \frac{b_0}{2^N}\right)$$

Can $|V_{\text{OUT}}| = |V_{\text{REF}}|$?



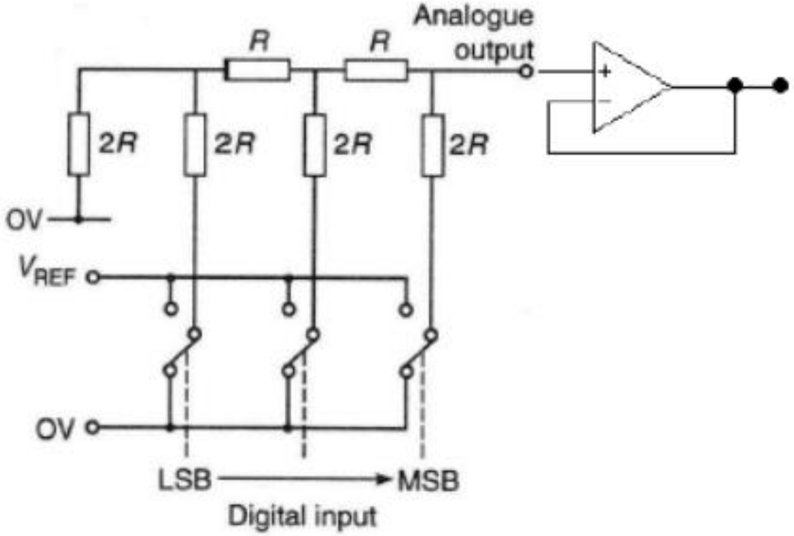


From [BW96]

DIGITAL-TO-ANALOG (D/A) CONVERSION

R/2R Ladder DAC

- Uses just two resistance values (2R and R, closely matched)
- Input switches define a specific resistor divider network



If only MSB (100) is asserted:

$$V_{OUT} = \frac{4}{8} \cdot V_{REF} = \frac{1}{2} \cdot V_{REF}$$

If all bits are asserted (111):

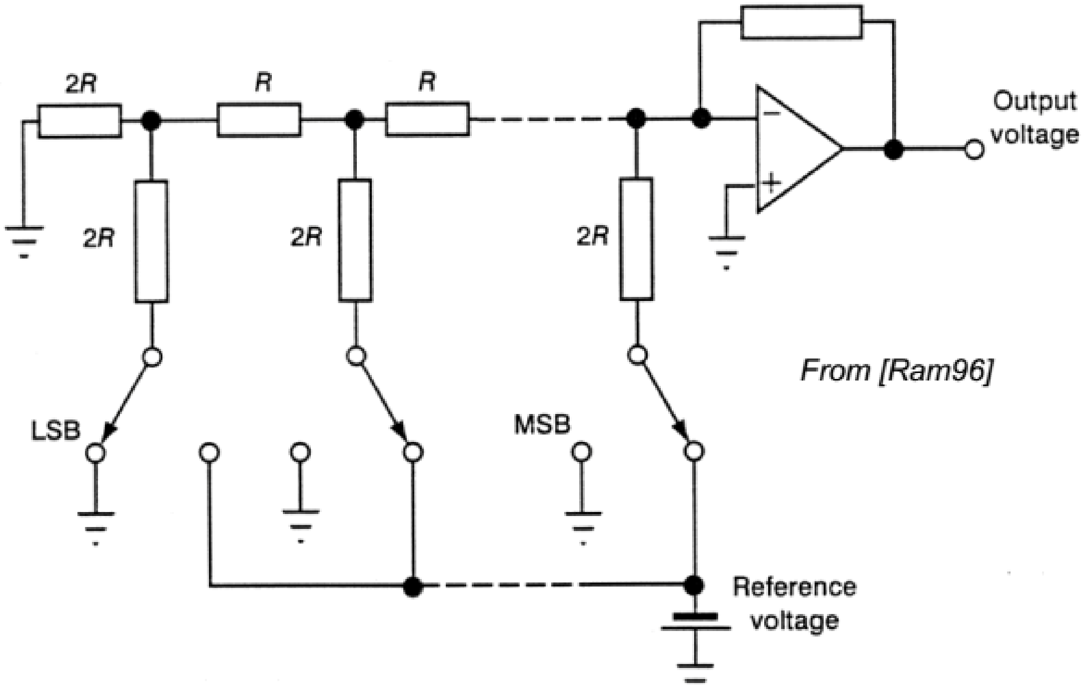
$$V_{OUT} = \frac{7}{8} \cdot V_{REF}$$

$$V_{OUT} = (b_{N-1} \cdot 2^{N-1} + b_{N-2} \cdot 2^{N-2} + \cdots + b_1 \cdot 2^1 + b_0 \cdot 2^0) \cdot \frac{V_{REF}}{2^N}$$

- Bipolar output can be achieved by substituting the ground with a negative voltage source.

R-2R ladder

- **Basic elements**
  - 2N resistors
  - An op-amp
  - N switches

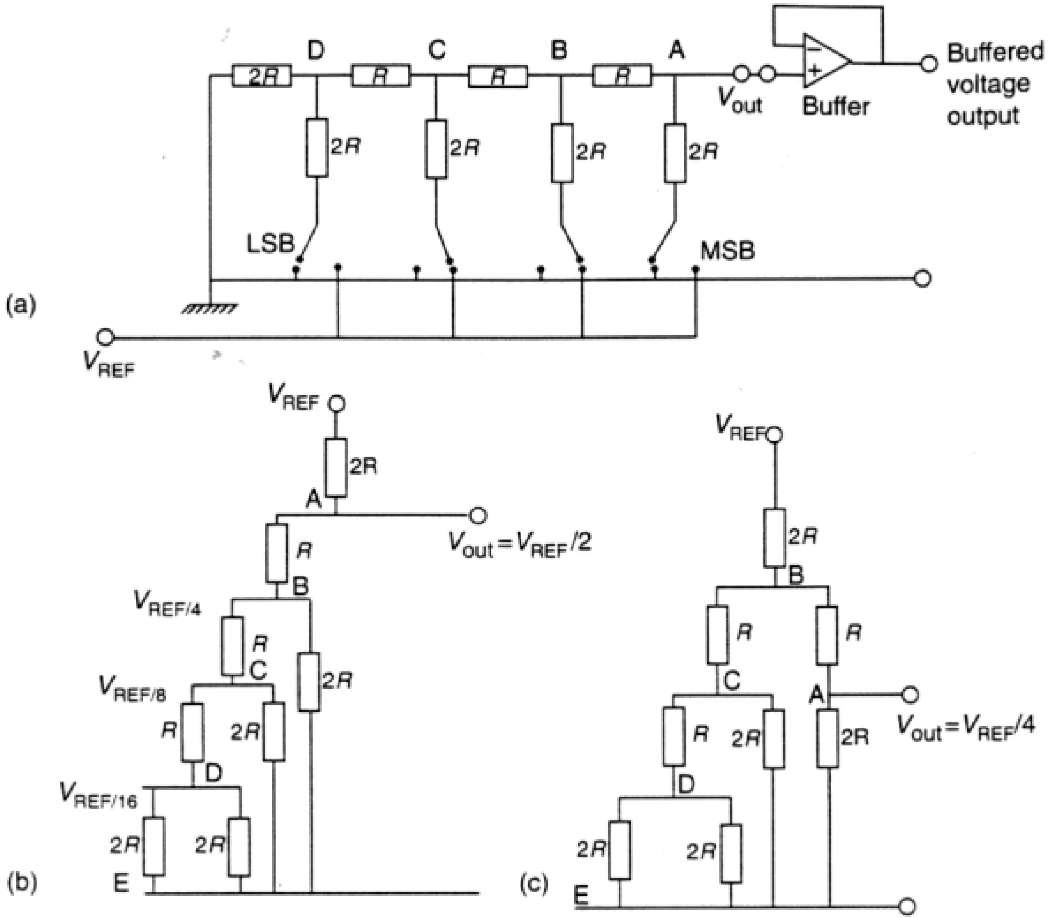


From [Ram96]

- **Operation**
  - Switches
    - When bit I_k=1, the corresponding switch is connected to V_{REF}
    - When bit I_k=0, the corresponding switch is connected to GND
  - Assume all the legs but one are grounded
    - The one connected to V_{REF} will generate a current that flows towards the inverting input of the op-amp
    - This current is halved by the resistor network at each node
    - Therefore, the current contribution of each input is weighted by its position in the binary number

R-2R ladder

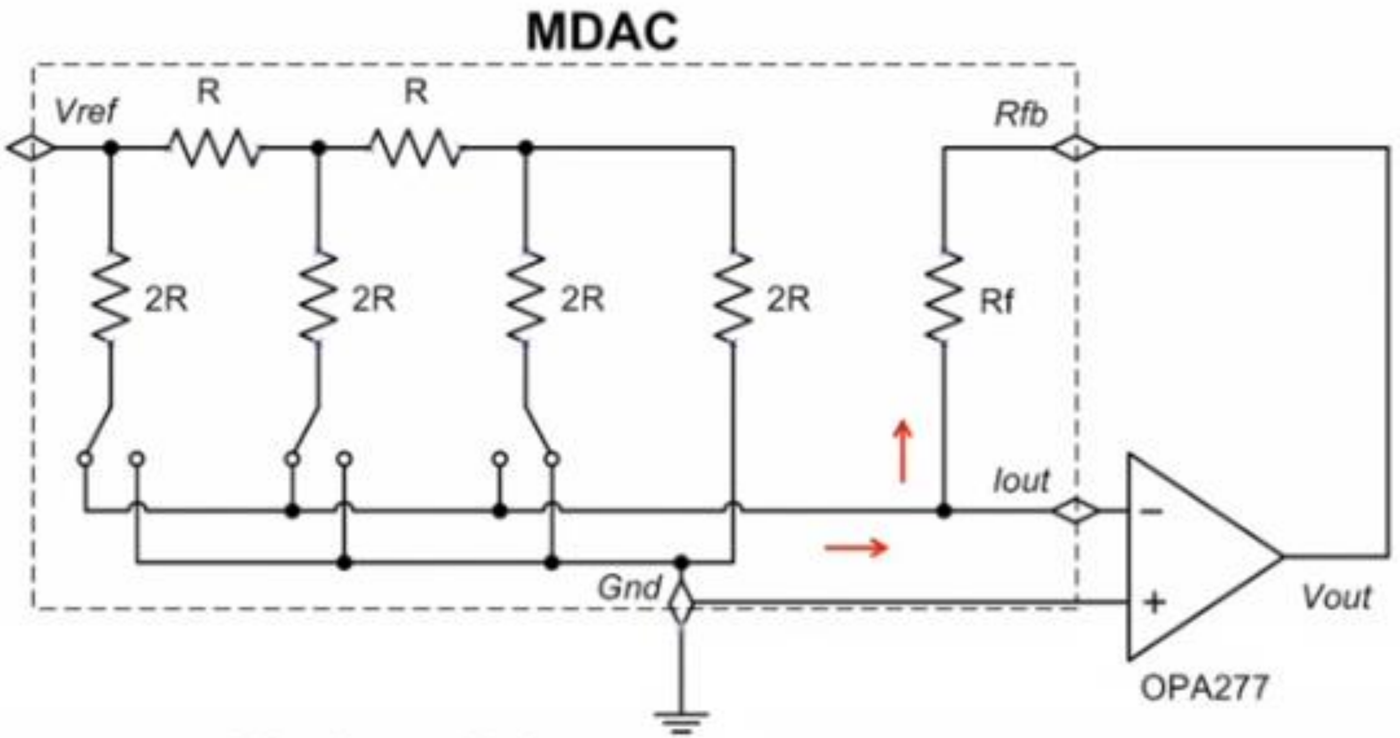
- **The R-2R operation is better understood by redrawing the resistor network**
  - In (b) only the MSB is ON
  - In (c) only the next bit to the MSB is ON



From [Tay97]

Multiplying DAC

MDAC current to voltage conversion



$$Vout = -\left(\frac{Vref}{R}\right) \times \left(\frac{Code}{2^N}\right) \times Rf$$

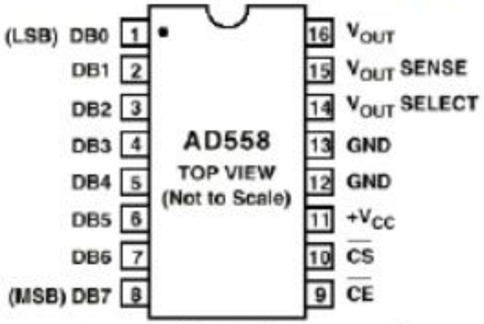
DIGITAL-TO-ANALOG (D/A) CONVERSION

Multiplying DAC

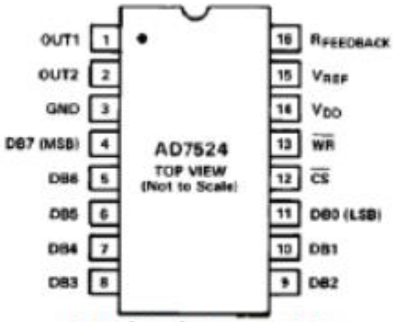
- Conventional DAC has internal reference voltage V_{REF} that is derived from the fixed power supply.
- Multiplying DAC has an externally supplied reference voltage.

$$V_{OUT} = (b_{N-1} \cdot 2^{N-1} + b_{N-2} \cdot 2^{N-2} + \cdots + b_1 \cdot 2^1 + b_0 \cdot 2^0) \cdot \frac{V_{REF}}{2^N}$$

- Advantages:
  - Use a constant frequency sinusoidal reference signal to achieve amplitude modulation, i.e. let $V_{REF} = V_R \sin(\omega t)$.
  - External V_{REF} can be precisely controlled to compensate for drift.



Internally Referenced



Multiplying DAC (Externally Referenced)

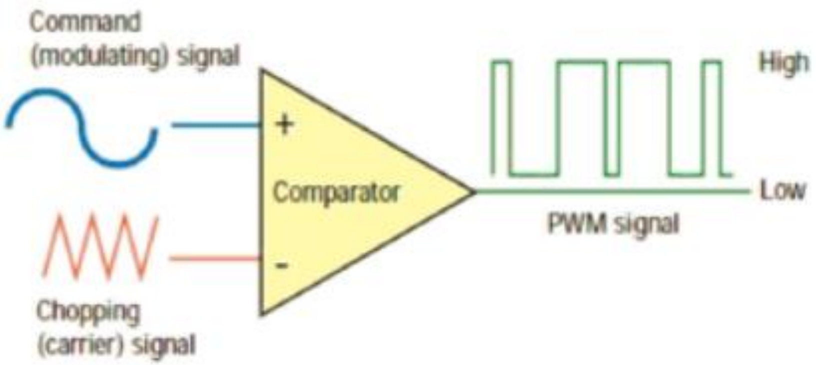
DIGITAL-TO-ANALOG (D/A) CONVERSION

Resistor methods rely on voltage dividers

- Many precision resistors necessary
- Wasted energy dissipated as heat

Pulse Width Modulation (PWM)

- Rectangular pulse wave
- Duty cycle controls average voltage
- Very high frequency
- Need a low-pass filter to remove the sharp transitions at edges of the pulses!
- About 90% efficiency



DIGITAL-TO-ANALOG (D/A) CONVERSION

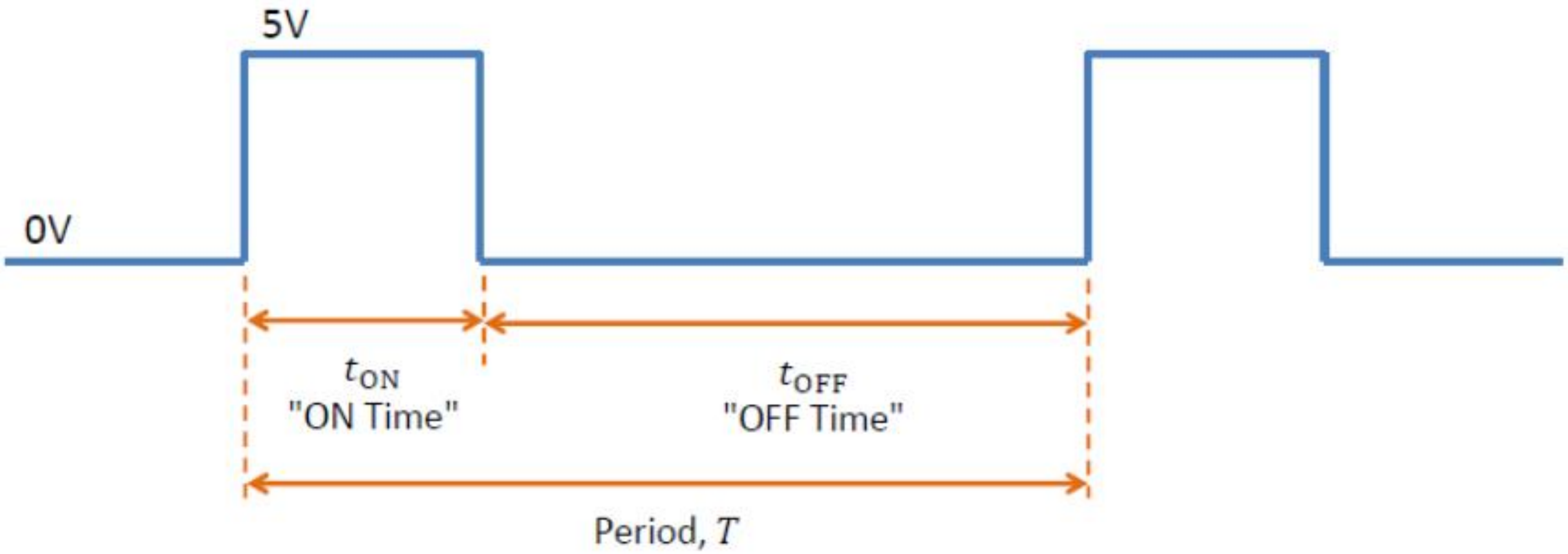
Pulse Width Modulation (PWM)

- Poor man's DAC
- Low pass filtering the PWM signal can produce an analog signal whose magnitude is proportional to the pulse width of the PWM signal
- For motor/motion control, the motor/motion system will act as the low pass filter
- Unipolar output
- Best suited when an analog output is needed but does not require a high resolution DAC

PULSE WIDTH MODULATION (PWM)

$$\text{Duty Cycle} = \frac{t_{ON}}{T} \Rightarrow 0 - 100\%$$

$$\text{Frequency (rad/sec)} = \frac{1}{T}$$



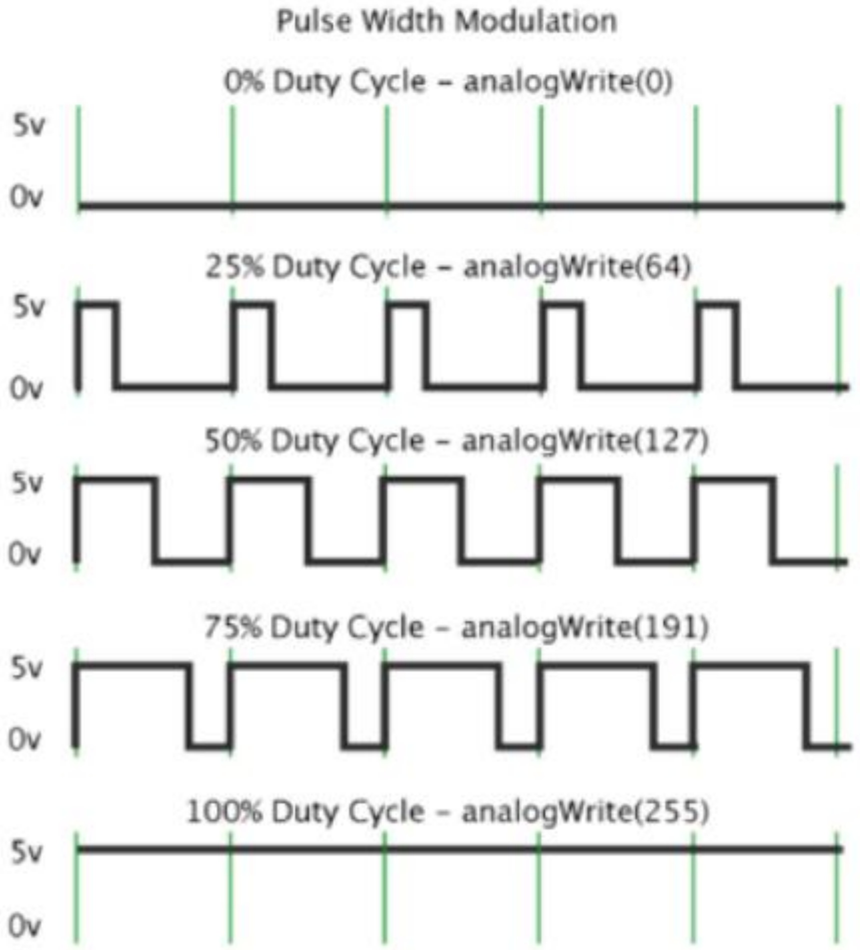
PULSE WIDTH MODULATION (PWM)

8-bit resolution:

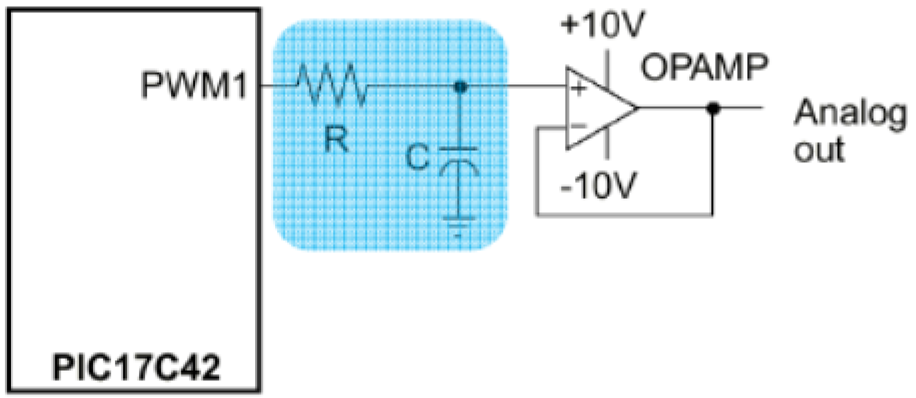
- 100%/255 => 0.39% per step
- 5V/255 => 19.6 mV per step

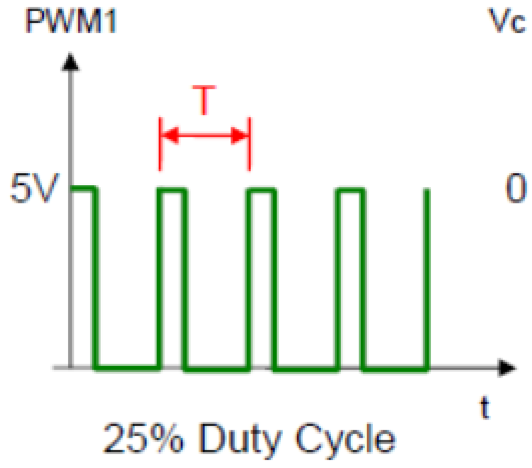
Arduino default PWM frequency:

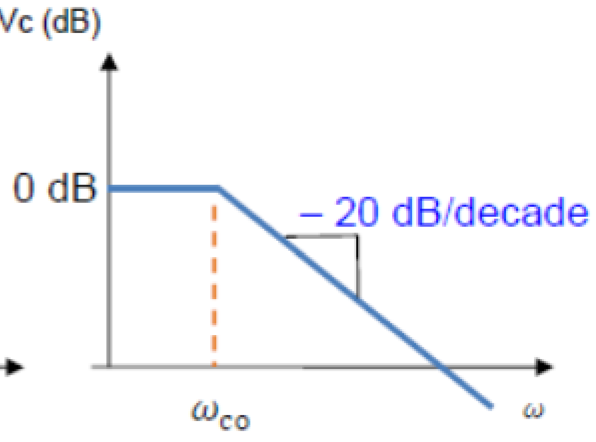
- Pins 5/6: ~976 Hz
- Pins 3/9/10/11: ~488 Hz
- Frequency can be increased to as much as 62.5 kHz by altering timer control registers
    - Pins 5/6: TCCR0B
    - Pins 9/10: TCCR1B
    - Pins 3/11: TCCR2B



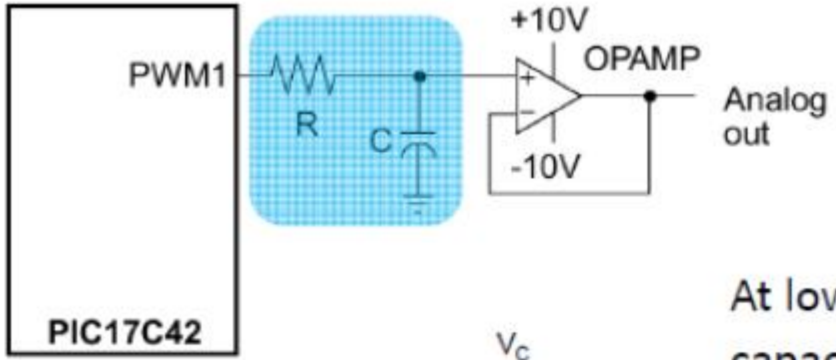
DIGITAL-TO-ANALOG (D/A) CONVERSION

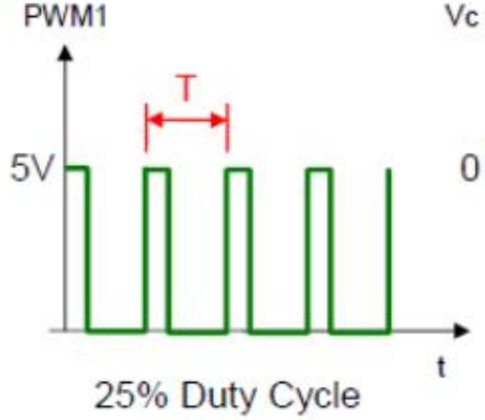


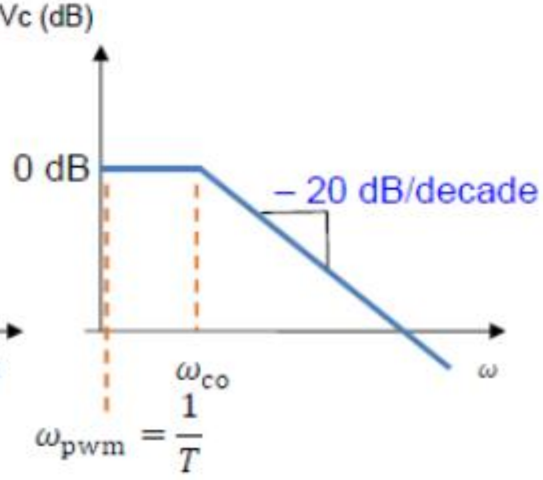




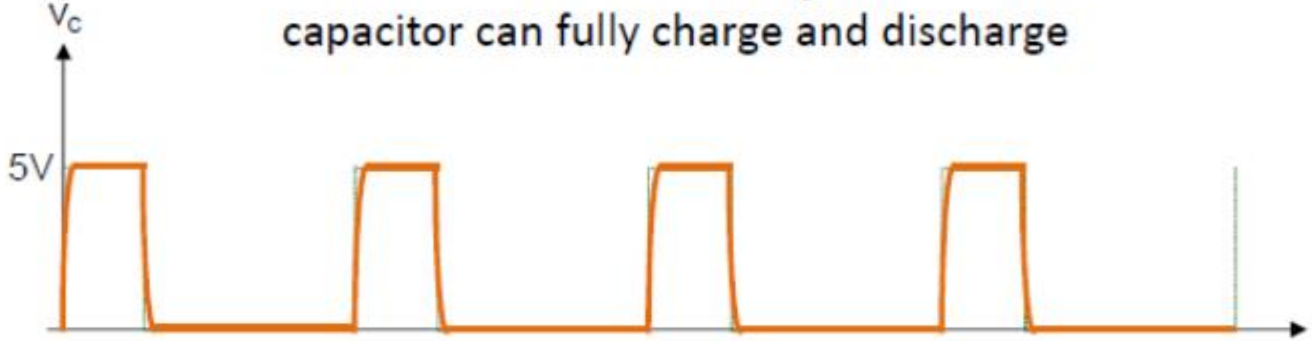
DIGITAL-TO-ANALOG (D/A) CONVERSION



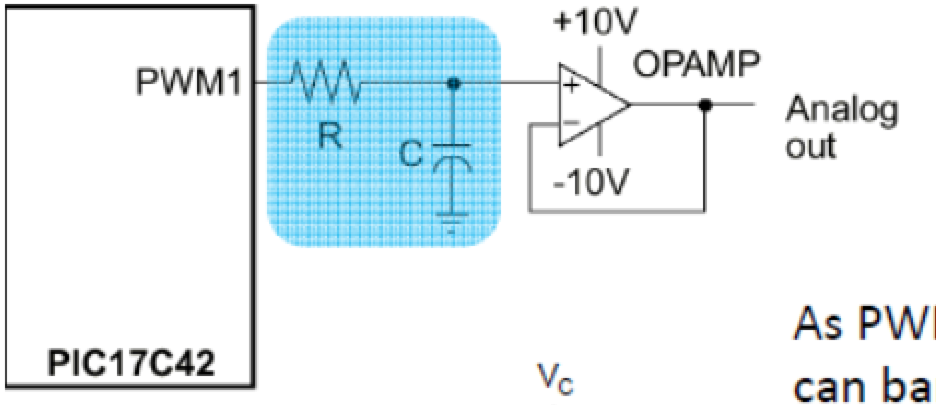


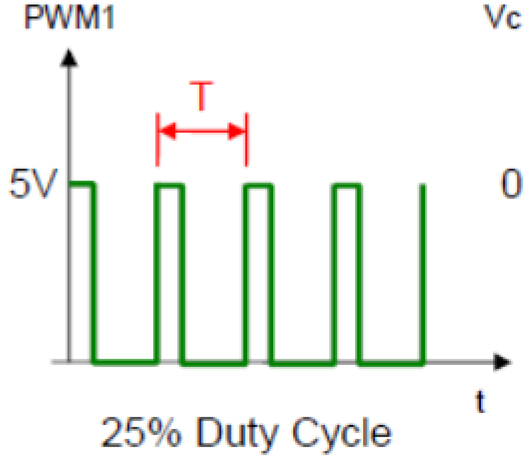


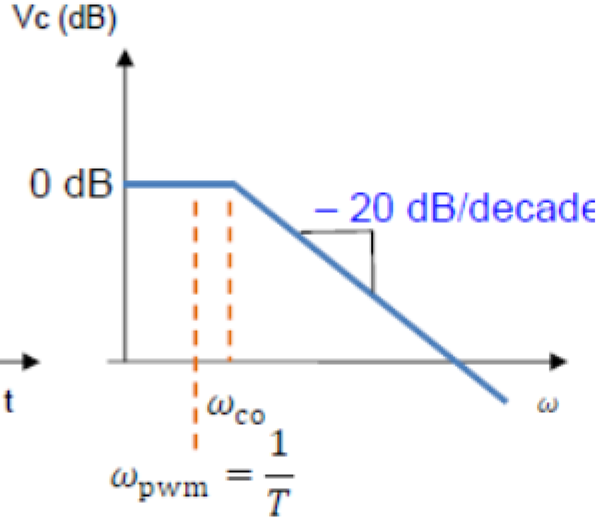
At low PWM frequency ($\omega_{pwm} \ll \omega_{co}$), capacitor can fully charge and discharge



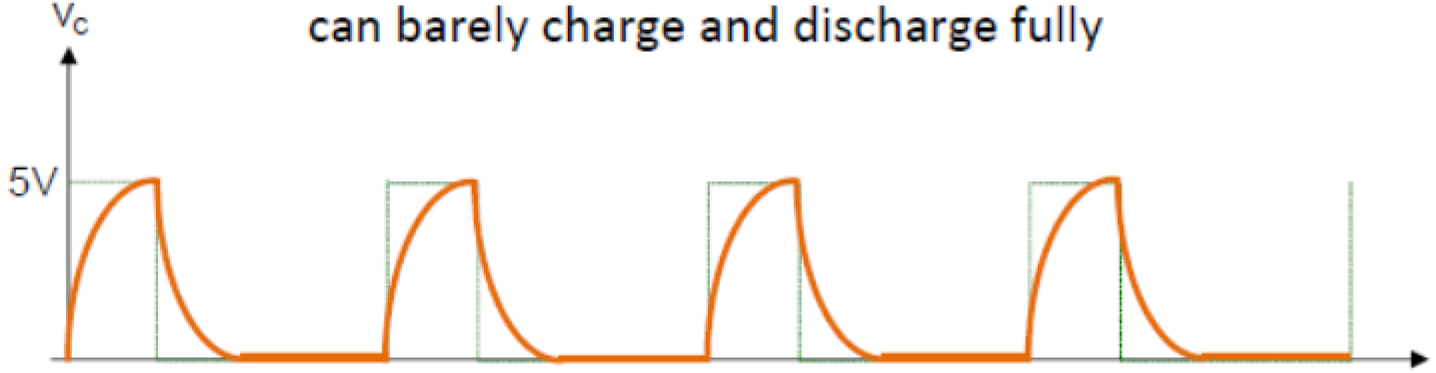
DIGITAL-TO-ANALOG (D/A) CONVERSION



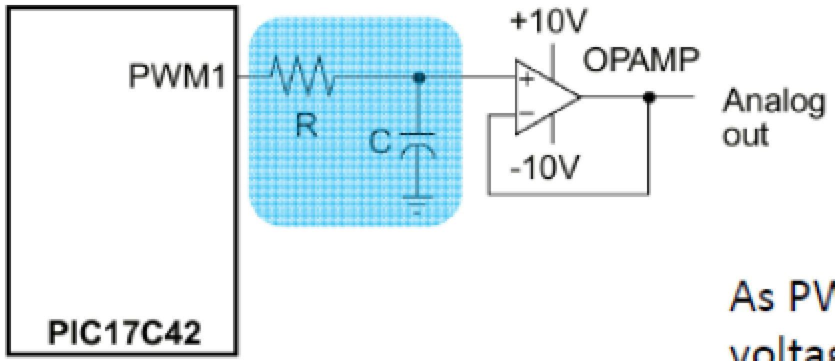


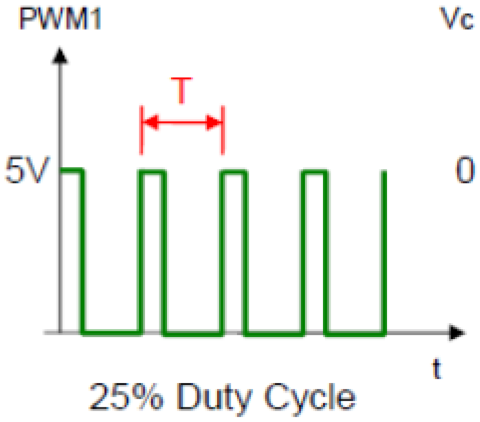


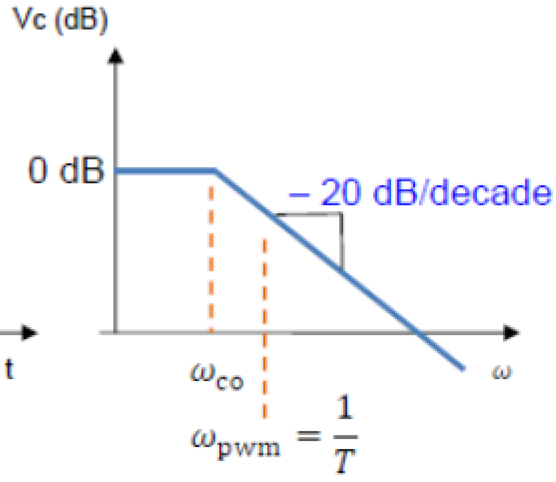
As PWM frequency increases, capacitor can barely charge and discharge fully



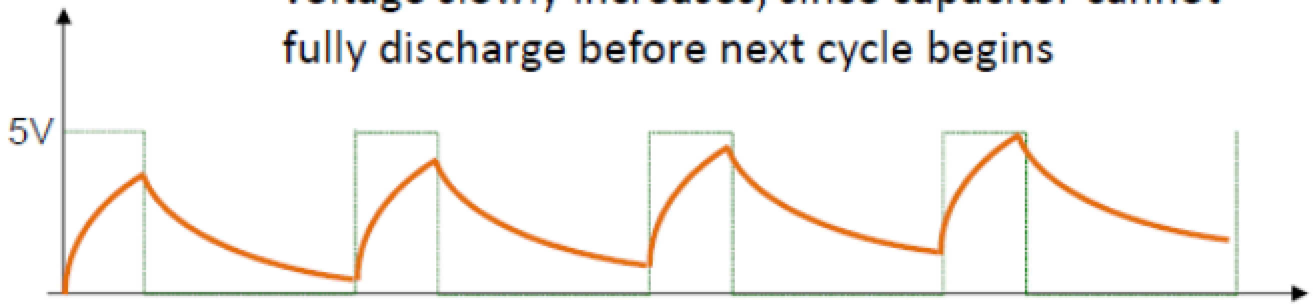
DIGITAL-TO-ANALOG (D/A) CONVERSION



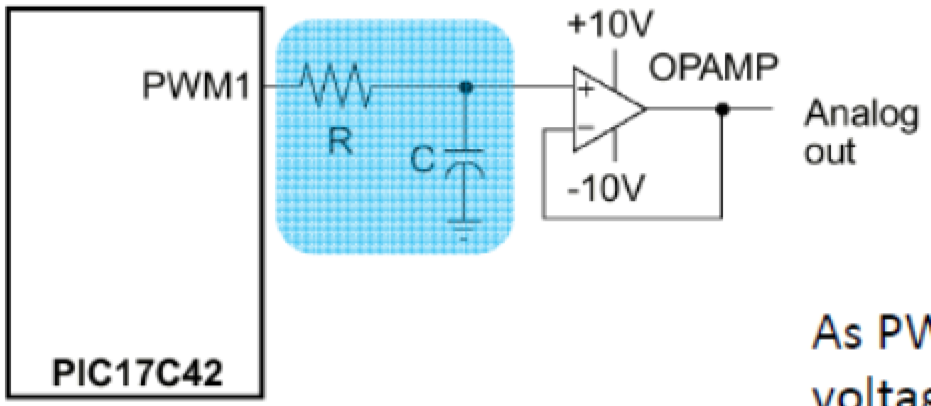


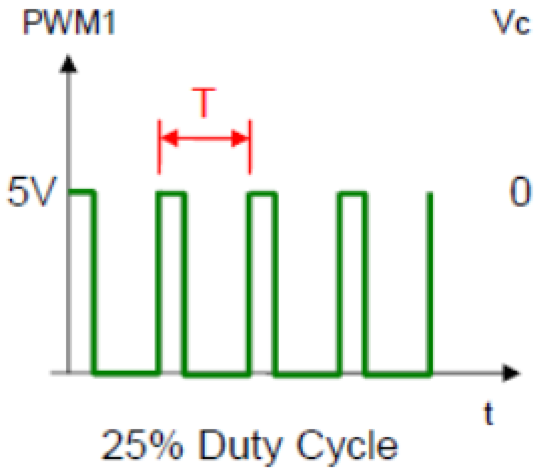


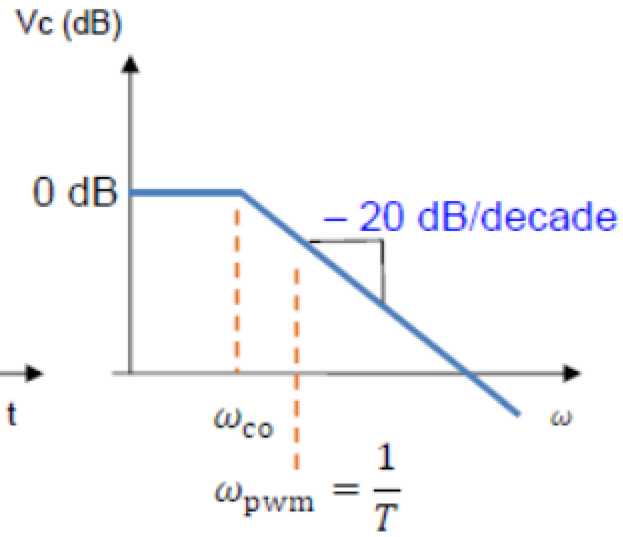
As PWM frequency increases further, capacitor voltage slowly increases, since capacitor cannot fully discharge before next cycle begins



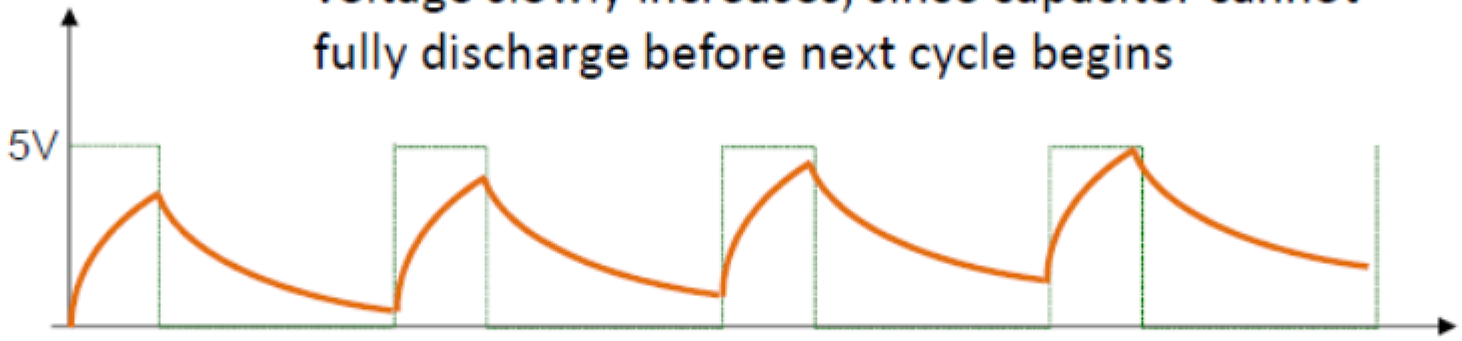
DIGITAL-TO-ANALOG (D/A) CONVERSION



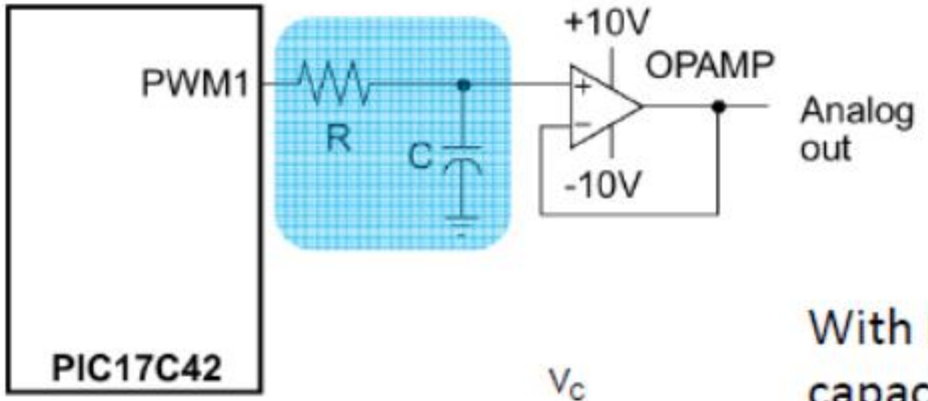


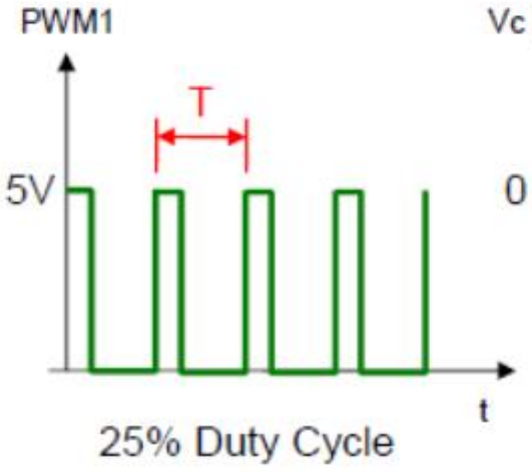


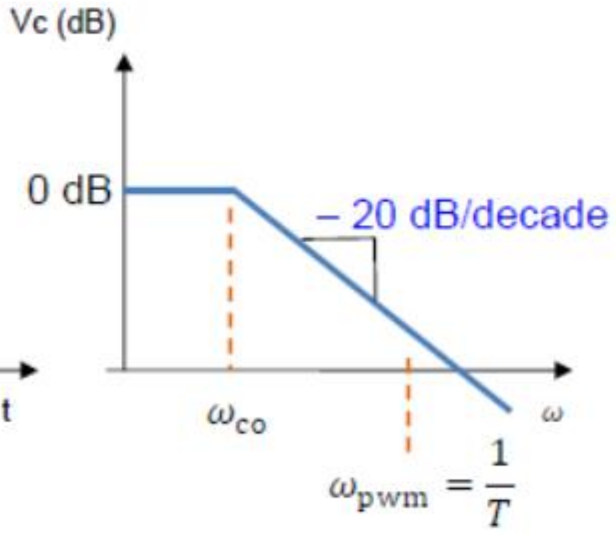
As PWM frequency increases further, capacitor voltage slowly increases, since capacitor cannot fully discharge before next cycle begins



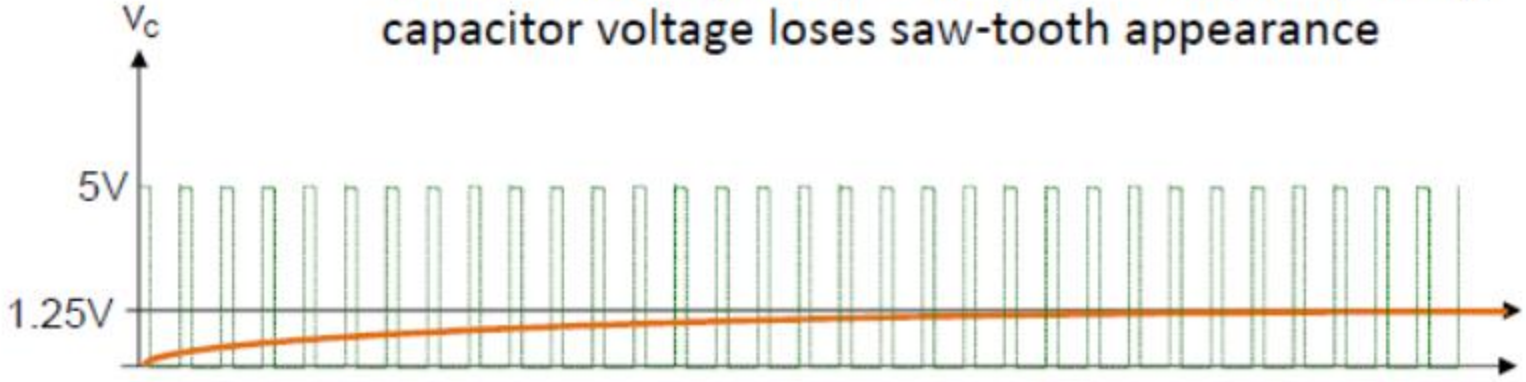
DIGITAL-TO-ANALOG (D/A) CONVERSION





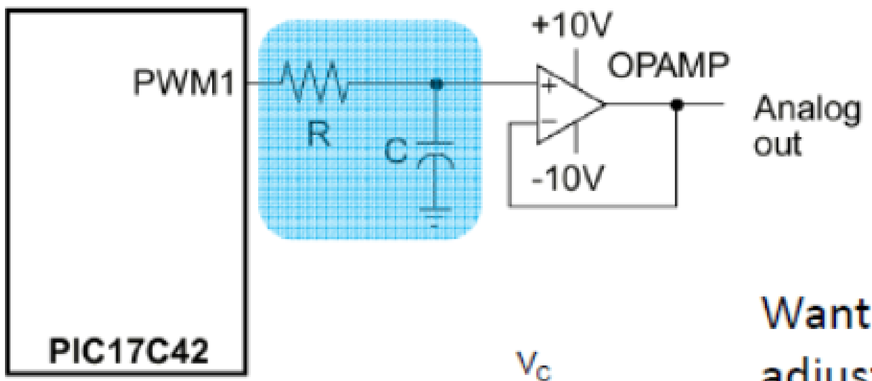


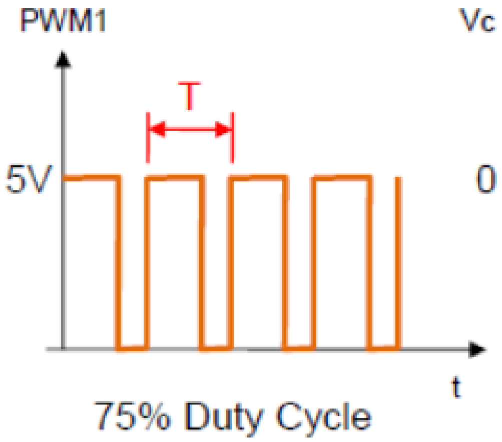
With PWM frequency well above cutoff frequency, capacitor voltage loses saw-tooth appearance

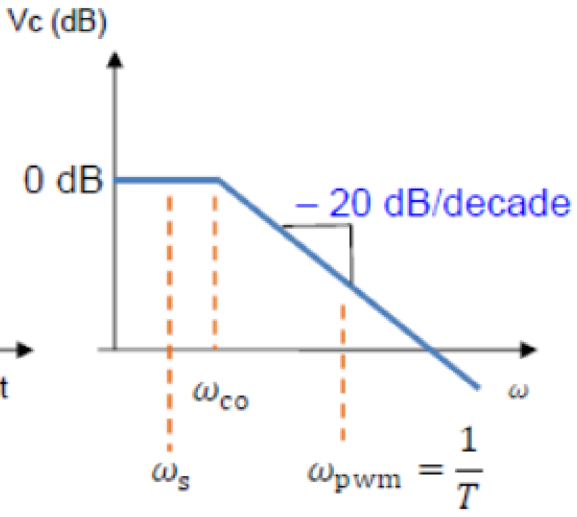


$$V_{C,\,\text{steady state}} = V_{\text{pwm}} \times \text{Duty Cycle \%}$$

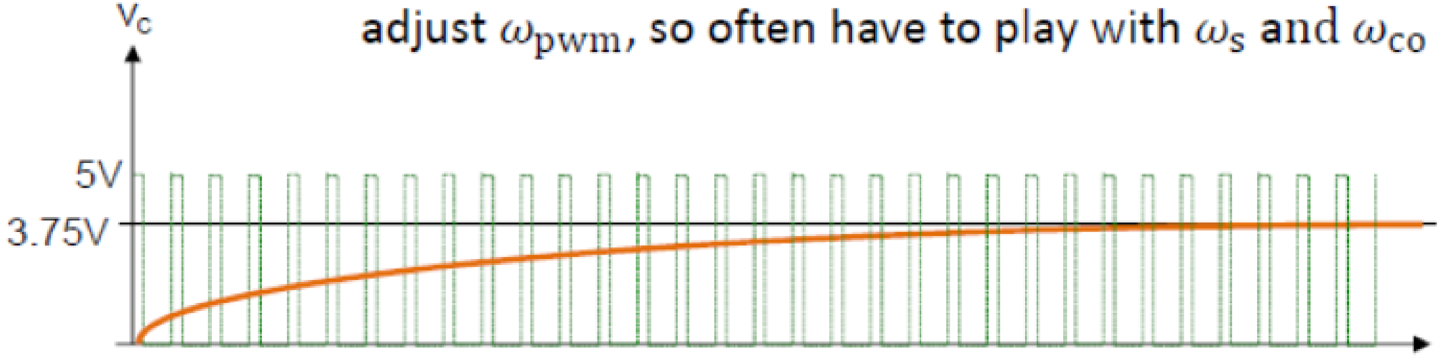
DIGITAL-TO-ANALOG (D/A) CONVERSION







Want signal frequency ω_s less than ω_{co}. Can't always adjust ω_{pwm}, so often have to play with ω_s and ω_{co}



$V_{C,\,\text{steady state}} = V_{\text{pwm}} \times$ Duty Cycle %

RC Low-pass Filter Design for PWM

Calculated peak-to-peak ripple voltage and settling time at a given PWM frequency and cut-off frequency or values of R and C.

http://sim.okawa-denshi.jp/en/PWMtool.php

How to adjust Arduino PWM frequencies

Pins 5 and 6: controlled by Timer 0 in fast PWM mode
(cycle length = 256)

Setting	Divisor	Frequency
0x01	1	62500
0x02	8	7812.5
0x03	64	976.5625 <--DEFAULT
0x04	256	244.140625
0x05	1024	61.03515625

TCCR0B = (TCCR0B & 0b11111000) | <setting>;