Embedded Software

CS 145/145L

Caio Batista de Melo

Mid-point Evaluations

- **Thanks for the feedback!**
- Some suggestions/comments we'll try doing this quarter:
  - Relate zybook/lecture concepts to real-world examples
  - Better connection between lectures
  - Grading turnaround
  - Provide some resources that explain how the software is working behind the scenes
- Some suggestions/comments we'll take into consideration for <u>future</u> quarters:
  - 8 AM class…
  - Recordings (won't do it this quarter, sorry!)
    - I'm happy to go over your questions about lecture on office hours or Edstem!
  - Environment setup
  - Sample demo for all projects

Agenda

- Project 3 Musical Notes
- Pulse Width Modulation (PWM)
- Analog to Digital

P3 Musical Notes

Project 3 Musical Notes

- When discussing frequencies, we only looked at collisions
- Also need to consider the recreated frequency
- Example:
  - 440Hz with 1ms timer
  - TH/TL = 1ms wait
  - Period = TH + TL = 2ms
  - Actual frequency = 1 / 2ms -> 500Hz
  - 13% error!

Project 3 Note Recreation Error

Note	Offset	Frequency (Hz)	Period (s)	TH / TL (s)	Wait (1ms resolution)	Actual Frequency	ABS Error
A	0	440.00	0.00227	0.001135	1	500	13.63
A#	1	466.16	0.00214	0.00107	1	500	7.25
B	2	493.88	0.00202	0.00101	1	500	1.23
C	3	523.25	0.00191	0.000955	1	500	4.44
C#	4	554.37	0.0018	0.0009	1	500	9.8
D	5	587.33	0.0017	0.00085	1	500	14.86
D#	6	622.25	0.0016	0.0008	1	500	19.64
E	7	659.26	0.00151	0.000755	1	500	24.15
F	8	698.46	0.00143	0.000715	1	500	28.41
F#	9	739.99	0.00135	0.000675	1	500	32.43
G	10	783.99	0.00127	0.000635	1	500	36.22
G#	11	830.61	0.0012	0.0006	1	500	39.8

Project 3 Note Recreation Error

Note	Offset	Frequency (Hz)	Period (s)	TH / TL (s)	Wait (0.1ms resolution)	Actual Frequency	ABS Error
A	0	220.00	0.0045	0.00225	23	217.3913	1.18
A#	1	233.08	0.0042	0.0021	21	238.0952	2.15
B	2	246.94	0.004	0.002	20	250	1.23
C	3	261.63	0.0038	0.0019	19	263.1578	0.58
C#	4	277.18	0.0036	0.0018	18	277.7777	0.21
D	5	293.66	0.0034	0.0017	17	294.1176	0.15
D#	6	311.13	0.0032	0.0016	16	312.5	0.44
E	7	329.63	0.003	0.0015	15	333.3333	1.12
F	8	349.23	0.0028	0.0014	14	357.1428	2.26
F#	9	369.99	0.0027	0.00135	14	357.1428	3.47
G	10	392.00	0.0025	0.00125	13	384.6153	1.88
G#	11	415.30	0.0024	0.0012	12	416.6666	0.32

Project 3 Note Recreation Error

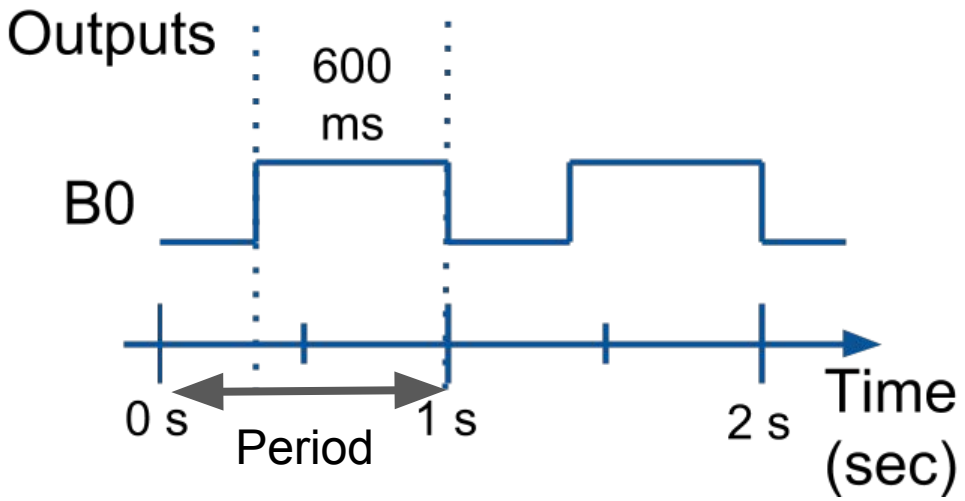
Note	Offset	Frequency (Hz)	Period (s)	TH / TL (s)	Wait (0.01ms resolution)	Actual Frequency	ABS Error
A	0	220.00	0.00454	0.00227	227	220.26431	0.12
A#	1	233.08	0.00429	0.002145	215	232.55813	0.22
B	2	246.94	0.00404	0.00202	202	247.52475	0.23
C	3	261.63	0.00382	0.00191	191	261.7801	0.05
C#	4	277.18	0.0036	0.0018	180	277.77777	0.21
D	5	293.66	0.0034	0.0017	170	294.11764	0.15
D#	6	311.13	0.00321	0.001605	161	310.559	0.18
E	7	329.63	0.00303	0.001515	152	328.94736	0.2
F	8	349.23	0.00286	0.00143	143	349.65034	0.12
F#	9	369.99	0.0027	0.00135	135	370.37037	0.1
G	10	392.00	0.00255	0.001275	128	390.625	0.34
G#	11	415.30	0.0024	0.0012	120	416.66666	0.32

Project 3 Note Recreation Error

Note	Offset	Frequency (Hz)	Period (s)	TH / TL (s)	Wait (0.01ms resolution)	Actual Frequency	ABS Error
A	0	440.00	0.00227	0.001135	114	438.59649	0.31
A#	1	466.16	0.00214	0.00107	107	467.28971	0.24
B	2	493.88	0.00202	0.00101	101	495.0495	0.23
C	3	523.25	0.00191	0.000955	96	520.83333	0.46
C#	4	554.37	0.0018	0.0009	90	555.55555	0.21
D	5	587.33	0.0017	0.00085	85	588.23529	0.15
D#	6	622.25	0.0016	0.0008	80	625	0.44
E	7	659.26	0.00151	0.000755	76	657.89473	0.2
F	8	698.46	0.00143	0.000715	72	694.44444	0.57
F#	9	739.99	0.00135	0.000675	68	735.29411	0.63
G	10	783.99	0.00127	0.000635	64	781.25	0.34
G#	11	830.61	0.0012	0.0006	60	833.33333	0.32

Pulse Width Modulation

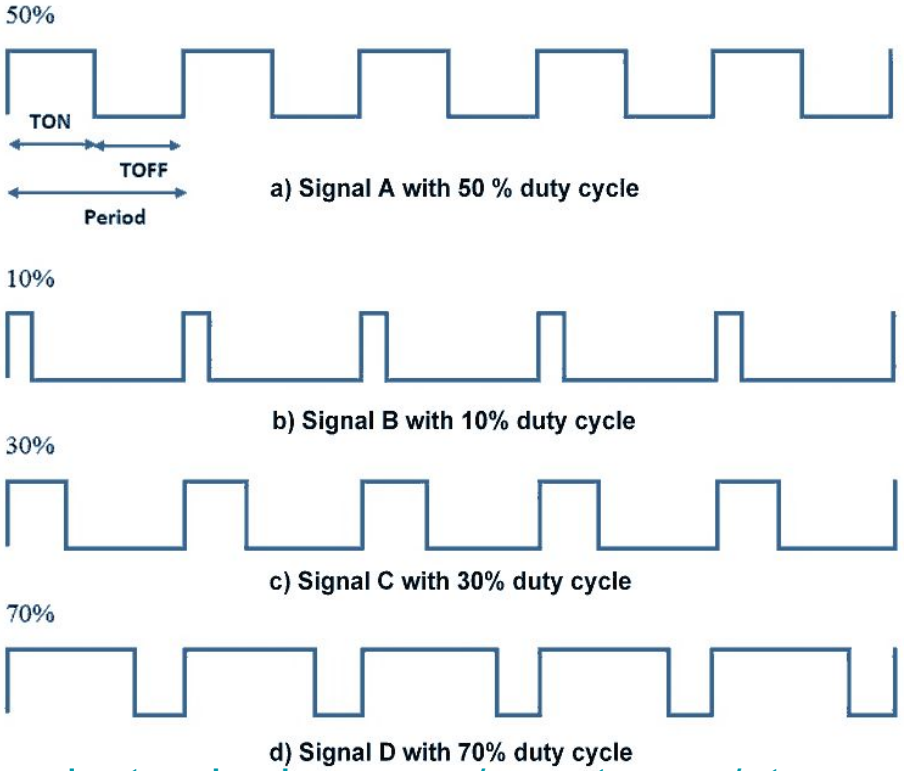
Pulse Width Modulation (PWM)



Duty Cycle: Percentage of period where output is 1

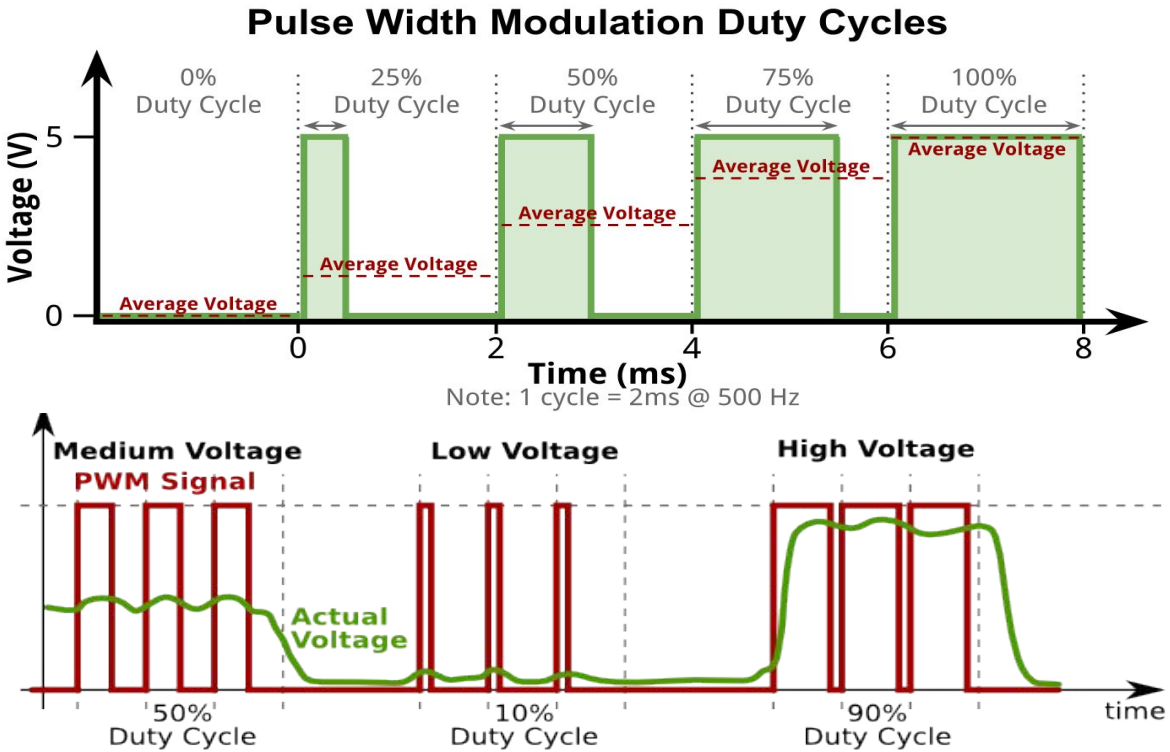
600ms / 1000ms -> 60%

PWMs with Different Duty Cycles



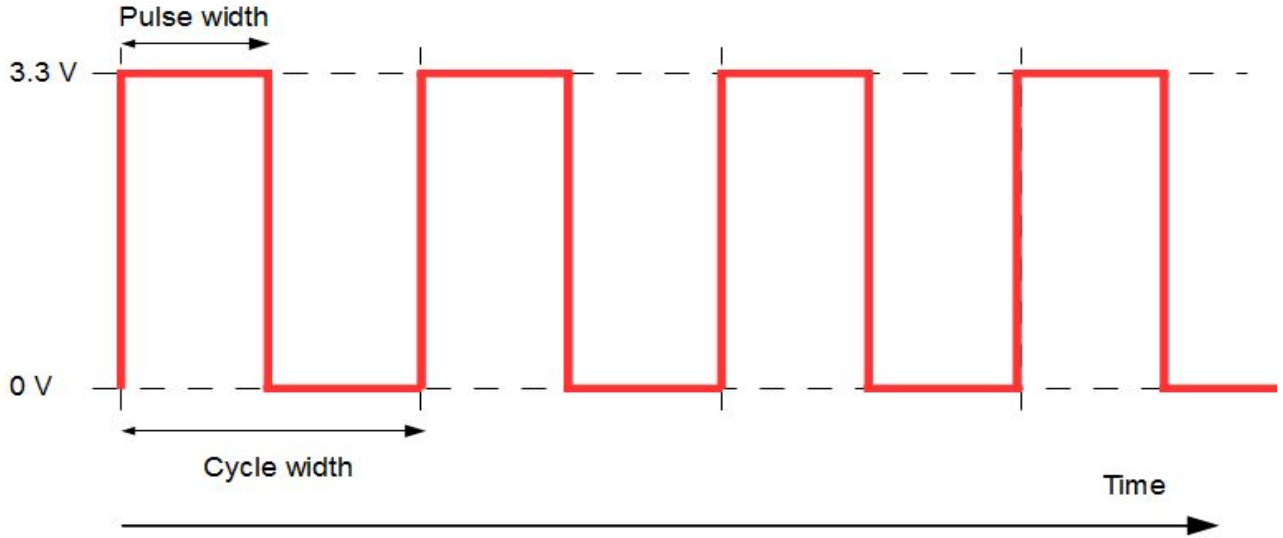
https://www.electronicwings.com/avr-atmega/atmega1632-pwm

Outcomes of Different Duty Cycles



https://www.youtube.com/watch?v=GQLED3gmONg

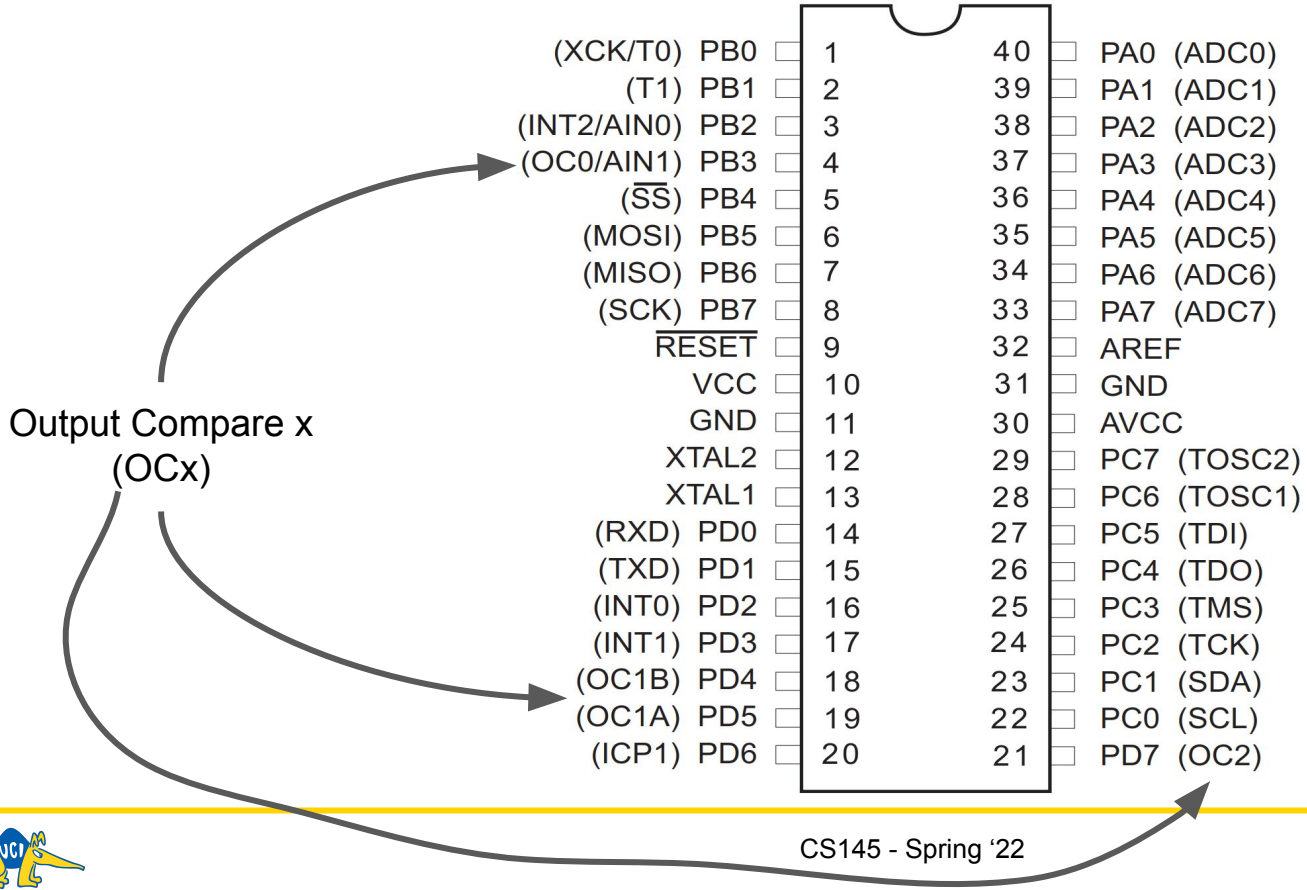
PWM with 50% Duty Cycle



http://www.lediouris.net/RaspberryPI/PWM/img/50.PNG

What we did for P1 and doing for P3's notes!

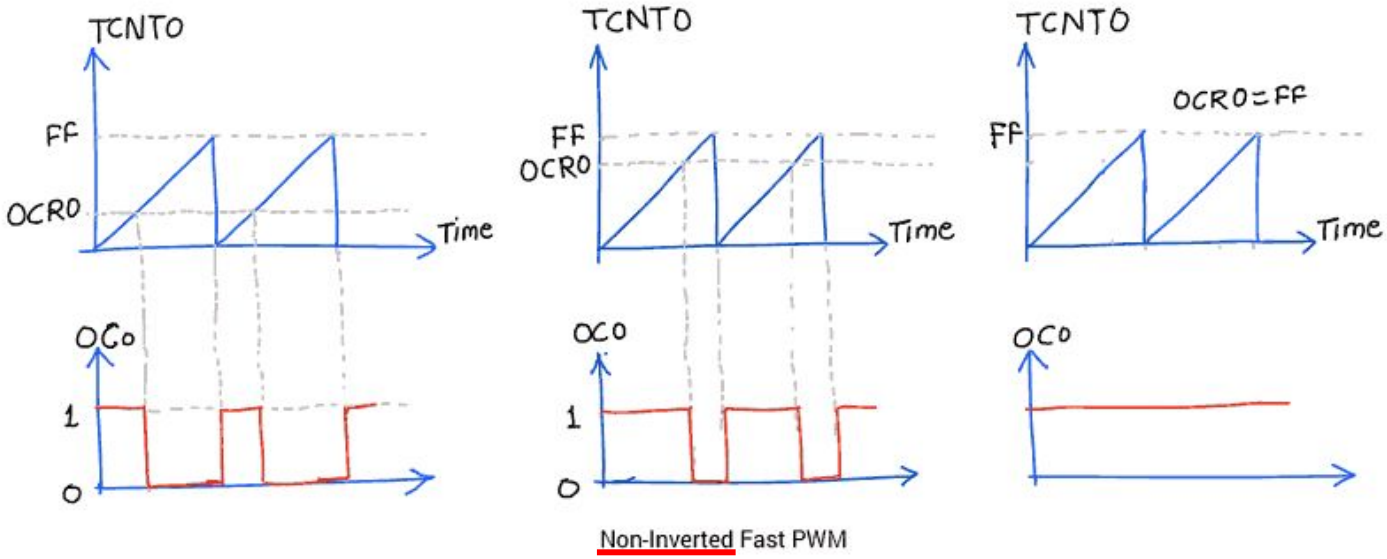
ATmega32 PWM Pins



ATmega32 PWM Modes

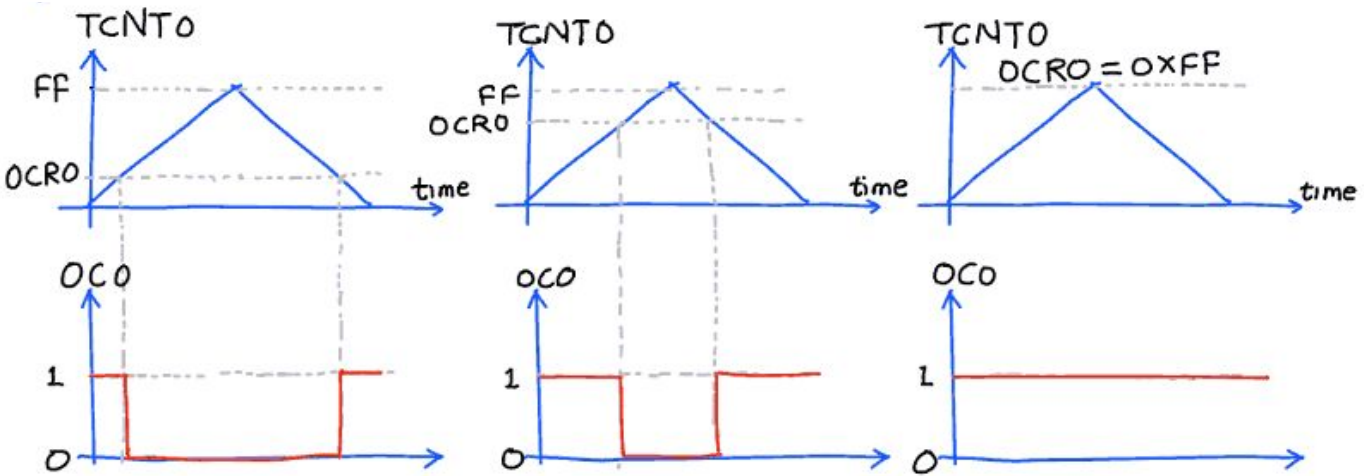
- Fast
  - Higher frequency
- Phase Correct
  - Result is centered in period
- Check page 73 of the manual

ATmega32 PWM Modes - Fast



https://www.electronicwings.com/avr-atmega/atmega1632-pwm

ATmega32 PWM Modes - Phase Correct



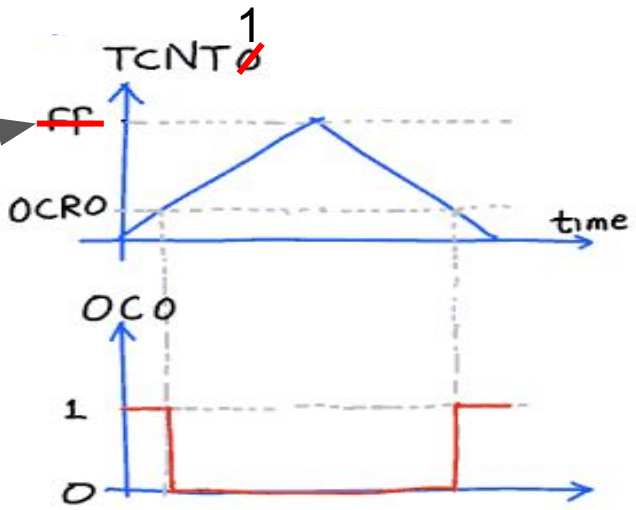
Non-Inverted Phase correct PWM

https://www.electronicwings.com/avr-atmega/atmega1632-pwm

Where can we use a PWM?

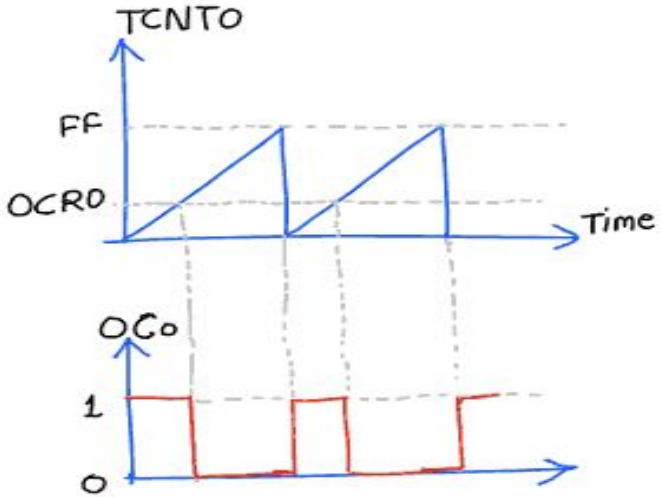
Project 1 LED blink using a PWM

- 50% duty cycle
- Pre-scaler might not allow exactly 1s
  - ➢ How to control period?
- Phase and Frequency Correct PWM Mode
  - ➢ Need to use Timer1 for this mode
  - ➢ Check pages 103 and 109 on the manual
- Same idea for P3 musical notes



Control Frequency of OC0

- OC1A/B might be blocked by the LCD…
- OC0 (PB3) might already be for your speaker!
- So how can we use timer0?
- Let's check page 80 of the manual
- Instead of starting at 0, shift it
  - By setting a seed!
- How can we set this seed?
  - Use an ISR

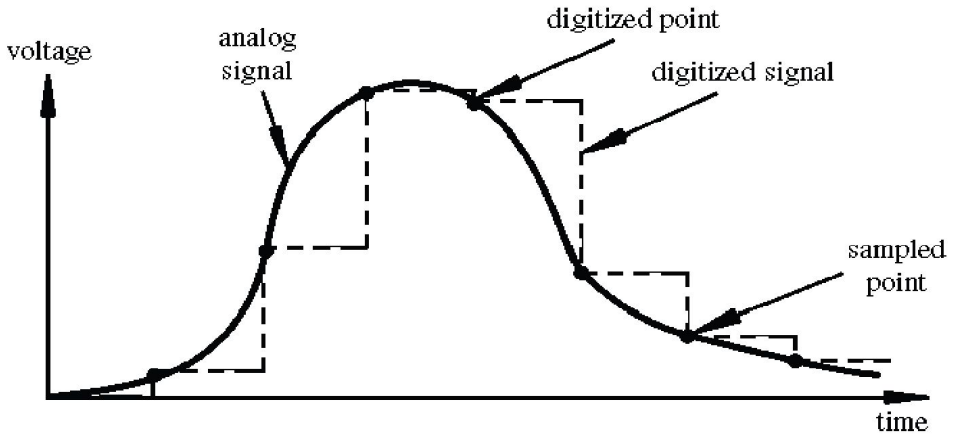


- Any (dis)advantages of using this instead of a wait function?

Analog-Digital Conversion

Analog-Digital Conversion (ADC)

- Digital has **two** values: **on** and **off**
- Analog has many (**infinite**) values
- Computers don't really do analog, they ***quantize***

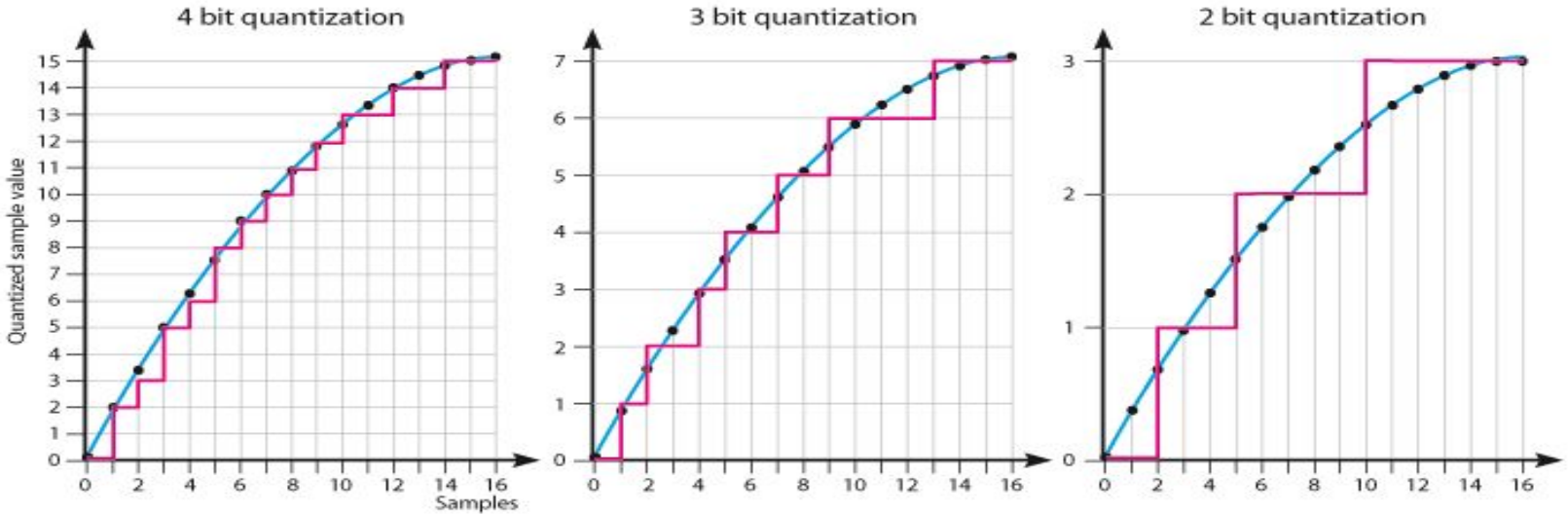


ADC Parameters

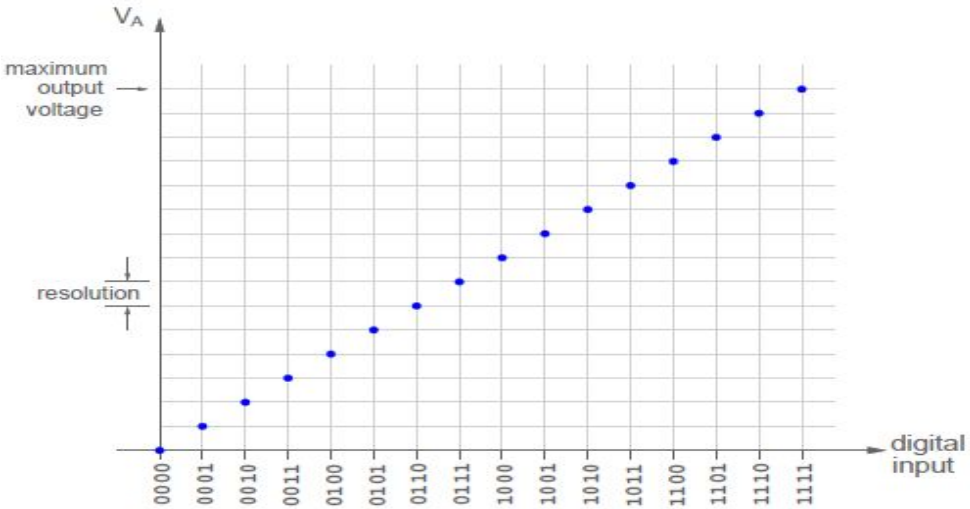
- Range
  - What are the minimum/maximum possible values
- Sampling Rate
  - How often we get a new data point
- Precision
  - How many bits we can use to represent the values

ADC 0~5V Example

Reads the voltage applied to an analog input pin and returns a number between **0 and 2^N-1** that represents the voltages between 0 and 5 V.



4-bit Quantization



b3	b2	b1	b0	Expected Decimal	Vout (V)
0	0	0	0	0	0
0	0	0	1	1	0.315
0	0	1	0	2	0.630
0	0	1	1	3	0.937
0	1	0	0	4	1.268
0	1	0	1	5	1.577
0	1	1	0	6	1.892
0	1	1	1	7	2.2
1	0	0	0	8	2.51
1	0	0	1	9	2.83
1	0	1	0	10	3.14
1	0	1	1	11	3.45
1	1	0	0	12	3.78
1	1	0	1	13	4.09
1	1	1	0	14	4.4
1	1	1	1	15	4.72

Intuitive Conversion

b3	b2	b1	b0	Expected Decimal	Vout (V)
0	0	0	0	0	0
0	0	0	1	1	0.315
0	0	1	0	2	0.630
0	0	1	1	3	0.937
0	1	0	0	4	1.268
0	1	0	1	5	1.577
0	1	1	0	6	1.892
0	1	1	1	7	2.2
1	0	0	0	8	2.51
1	0	0	1	9	2.83
1	0	1	0	10	3.14
1	0	1	1	11	3.45
1	1	0	0	12	3.78
1	1	0	1	13	4.09
1	1	1	0	14	4.4
1	1	1	1	15	4.72

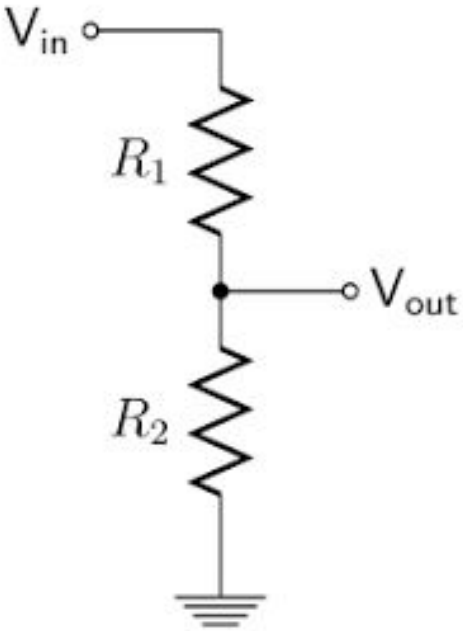
The pattern is repeated

- The MSB is contributing to half of the voltage.

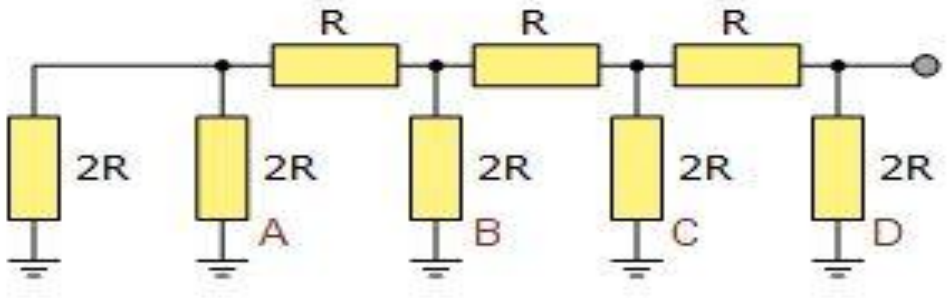
$V_{out} = b3 * 2^3 + b4 * 2^2 + b1 * 2^1 + b0 * 2^0$

b3 is contributing to 50% of the voltage
b2 to 25%
b1 to 12.5%
b0 to 6.25%

Intuitive Hardware



Voltage divider
(single bit)

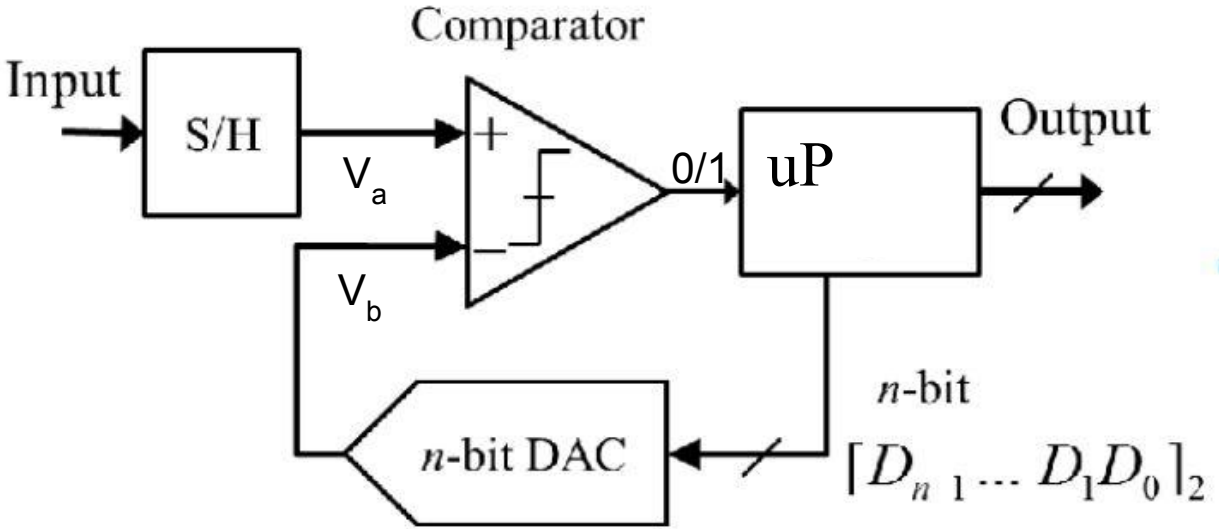


High quality signal without noise but expensive because resistances should have precise tolerances

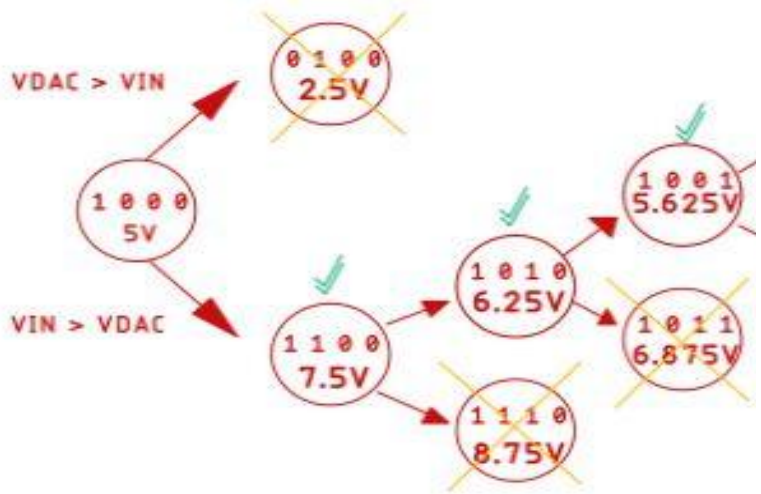
* e.g., audio systems

ADC Layout

Output	
0	Va < Vb
1	Va > Vb



ADC Result



Successive Approximation (Binary search)

- Let's say, the sampled input is 5.8V
- The reference of the ADC is 10V. When conversion starts, the register sets the most significant bit to 1 and all other bits to zero.
- This means the value becomes 1000, which means half of the 10V reference voltage.
- Now this voltage will be compared to the input voltage and based on the comparator output, the output of the register will be changed.

ADC on ATmega32

- Our microcontroller has a built-in ADC!
- We'll use it for P4
- And we'll talk about that next class!

See you next time :)

Q & A