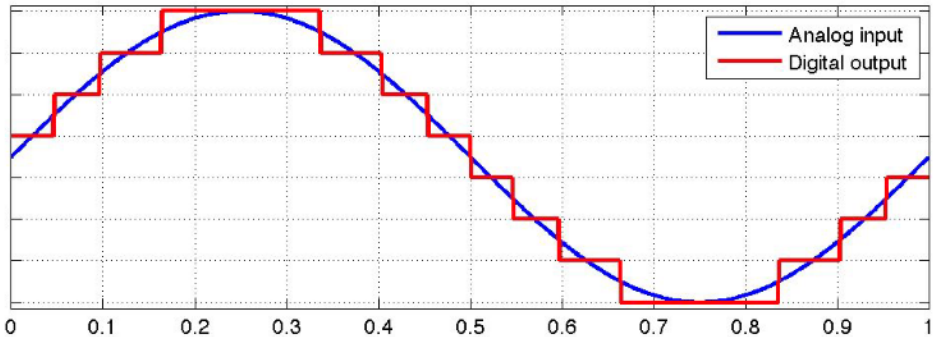
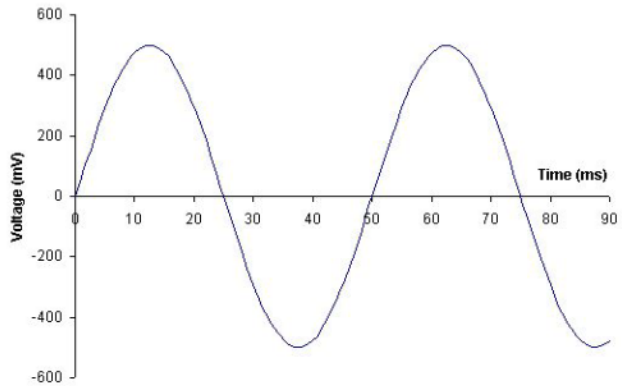
Signals, Time, Timer and Timing

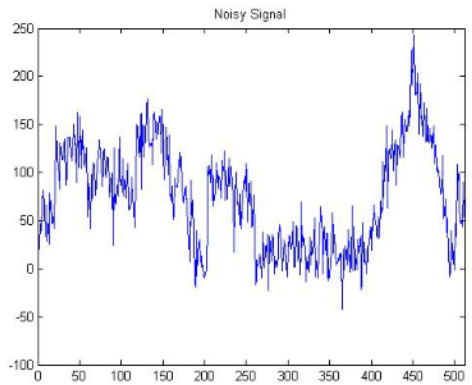
Programming and Development of Embedded Systems

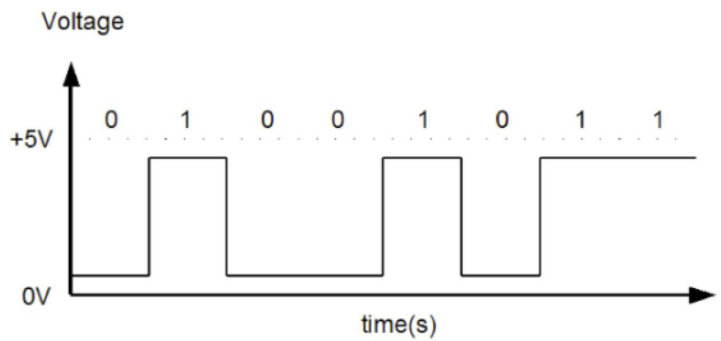
Electrical Signals

- A Signal is defined as **physical quantities** which contains **information**
- Signals are passed between devices in order to send and receive information
- Signals mainly can be classified as
  - Analog signals
  - Digital signals



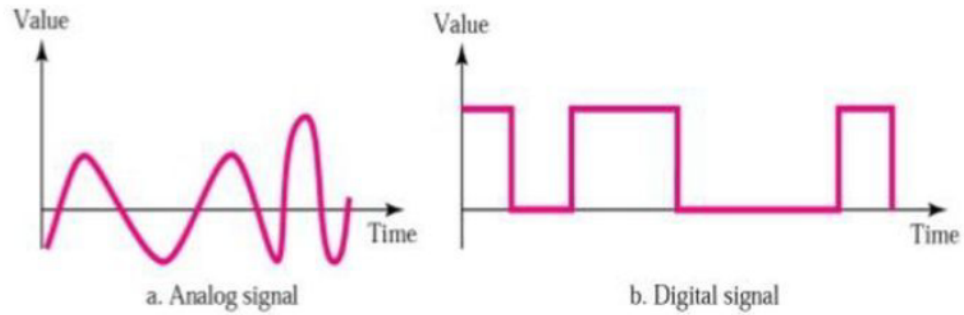




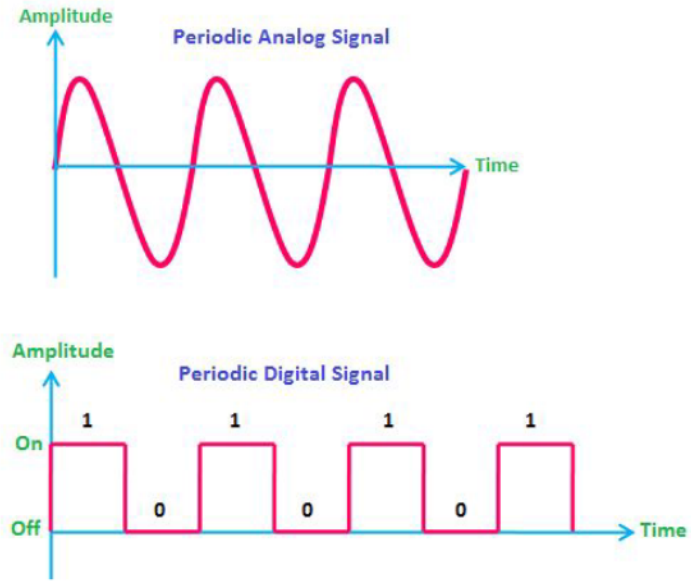


Periodic and Non-periodic Signals

- A Periodic signal is a signal which **repeats** itself intervally
  - Examples: Sine, cosine, square, rectangular and triangular signals, and etc.
- A Non-periodic signal is a signal which **does not repeat** itself after a specific interval
  - Examples: Sound signals from radio and all types of noise signals, and etc.



a. Analog signal b. Digital signal

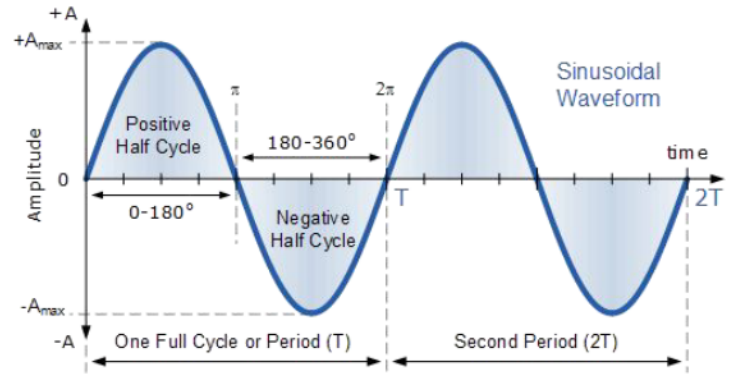


Periodic Signals

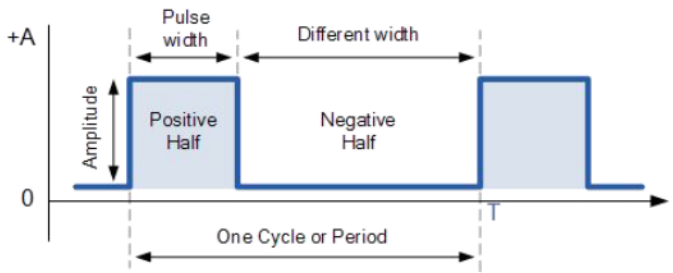
- Some Periodic Signal Waveforms

$$\text{Frequency} = \frac{1}{\text{Periodic time}} \quad \text{or} \quad f = \frac{1}{T} \text{ Hz}$$

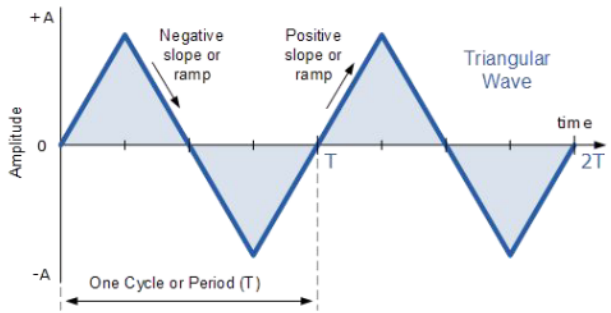
$$\text{Periodic time} = \frac{1}{\text{Frequency}} \quad \text{or} \quad T = \frac{1}{f} \text{ sec}$$



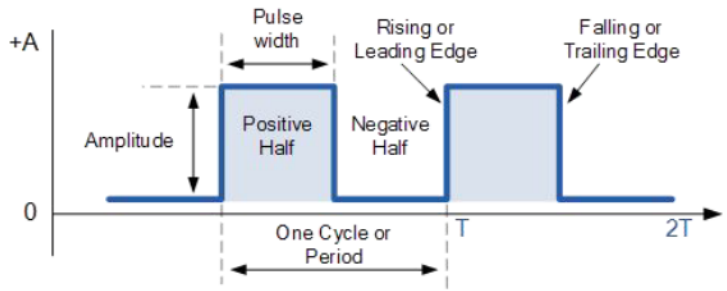
A Sine Wave Waveform



A Rectangular Wave Waveform



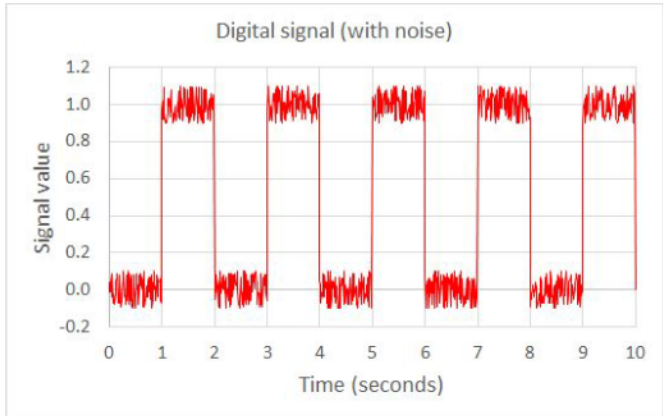
A Triangular Waveform

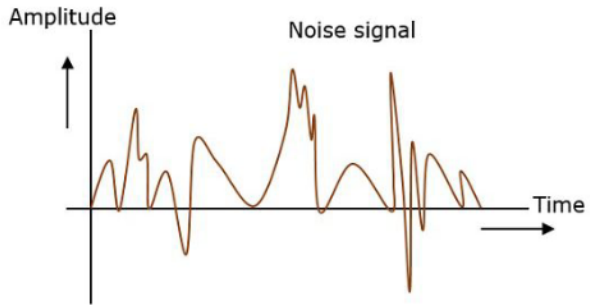


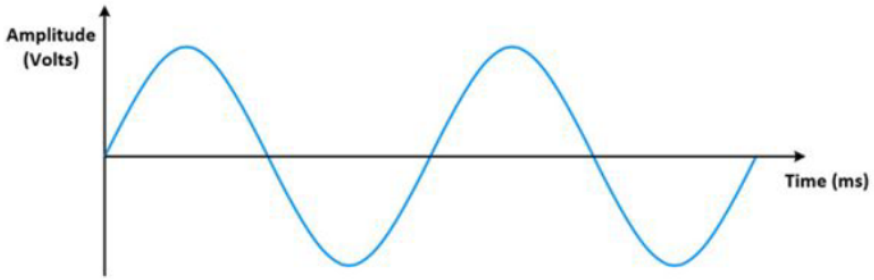
A Square Wave Waveform

Analog Signals

- Analog Signal
  - It can be defined as a signal having continuous and infinite number of values
  - Most of the things observed in the nature are analog
    - Temperature, Pressure, Sound, Voltage, Current and etc.
  - The circuits that process analog signals are called as analog circuits or systems. e.g. motor speed controller
  - Disadvantage of Analog Systems
    - Less accuracy and versatility, more noise effects, more distortion and more effect of weather

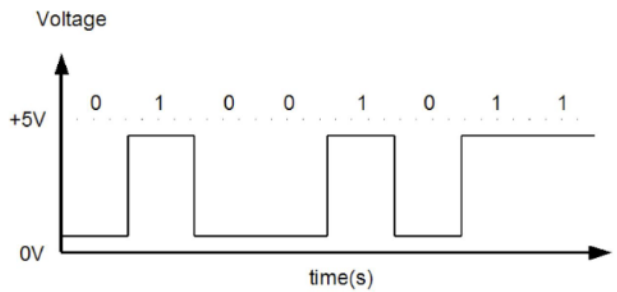


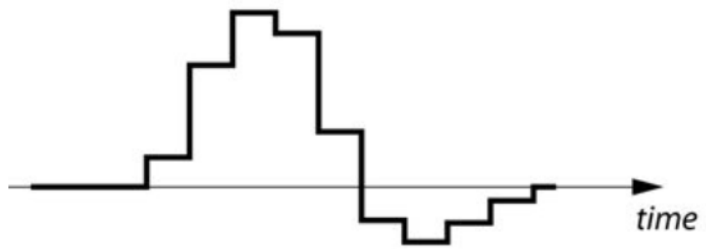




Digital Signals

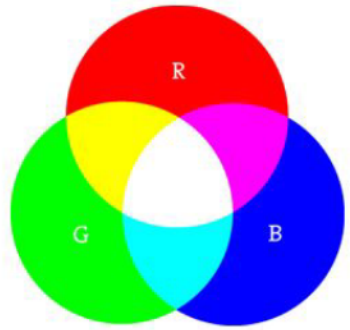
- Digital Signal
  - Is defined as a signal which has a finite number of discrete values or levels
  - Like binary signals which have two discrete values or levels
  - The circuits that process digital signals are called digital circuits or systems
    - Like Microprocessors, Counters, Registers and etc.
  - Advantage of Digital Systems
    - More accuracy
    - More versatility
    - Less distortion
    - Easy communicate
    - Possible storage of information
  - Usually is generated by A to D converters



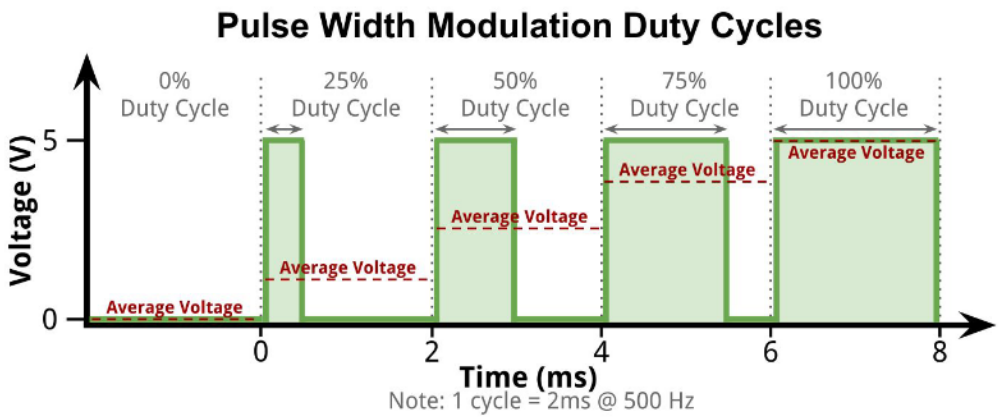


PWM Signals

- Pulse Width Modulation (PWM)
  - Is a fancy term for describing a rectangular digital signal
  - Is a technique for getting analog results with digital means
  - Is used in a variety of applications like control of
    - The brightness of an LED, the angle of a servo motor, generating clock signal and etc.



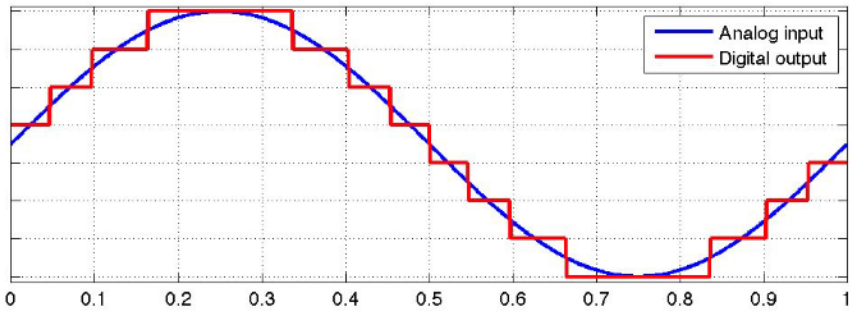
- Examples
  - Control the brightness of an LED by adjusting the duty cycle
  - With an RGB (red green blue) LED, control how much of each of the three colors you want in the mix of color by dimming them

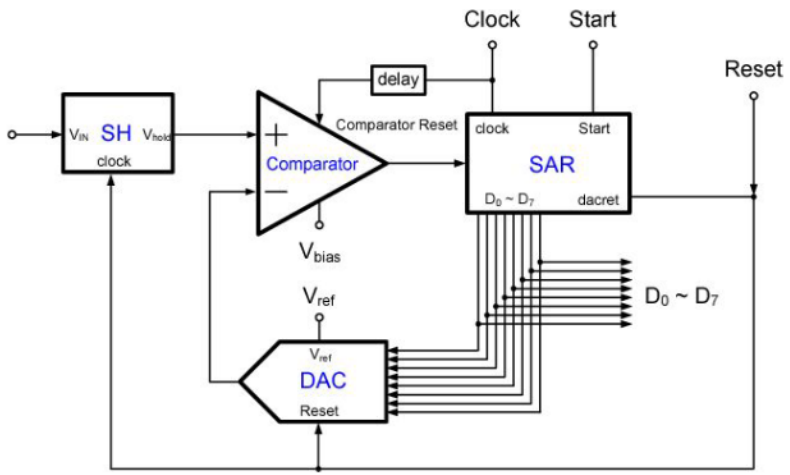


Duty Cycle describes the proportion of 'on' time to the regular interval or 'period' of time

Analog to Digital Converter

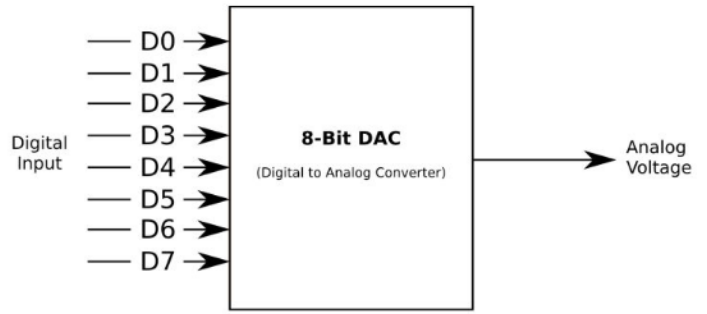
- ADC Resolution: Number of the output bits
- Sample Rate: Number of samples per second (Hz)
- Reference Voltage: System voltage (Vref)
- Voltage Resolution
  - The change in voltage required to guarantee a change in the output code level
  - It is the least significant bit (LSB) voltage
  - Vres = Vref / 2 **ADC Resolution**
- Digital Value = Analog Voltage / Vres
- Nyquist–Shannon sampling law
  - Sampling rate > 2 × the bandwidth of the analog signal
  - Bandwidth = fmax - fmin (Hz)

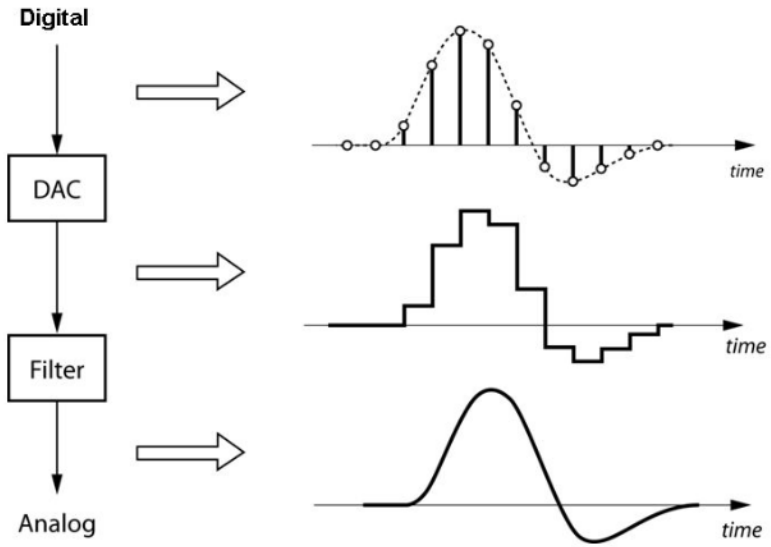




Digital to Analog Converter

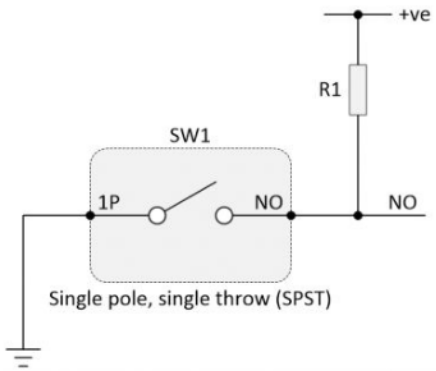
- DAC Resolution
  - Number of the input bits
- Reference Voltage
  - System voltage (V_{ref})
- Voltage Resolution
  - The least significant bit (LSB) voltage
  - $V_{res} = V_{ref} / 2^{\text{DAC Resolution}}$
- Analog Voltage
  - V_{out} = Digital Value × V_{res}

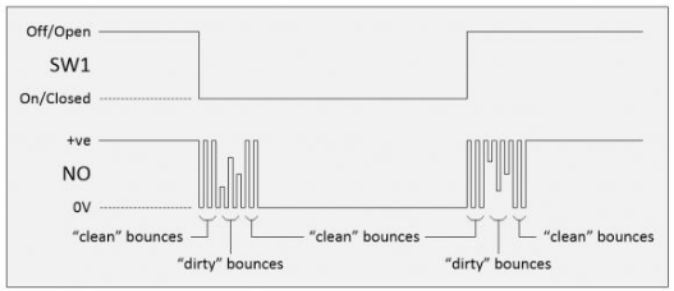


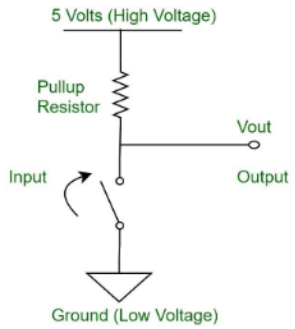


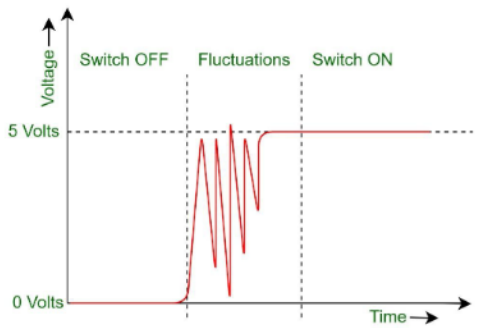
Switches and Contact Bouncing

- Metal contacts are used in mechanical switches and electromechanical relays
- Signal bouncing happens when the state of the switch is toggled
- Mechanical bounce of the contacts which makes and breaks the circuit causes bouncing
- Contact bouncing happens even if there is no mechanical bounce due to:
  - An inductive effect caused by the switch (the switch acts as an inductor)
  - The transmission line effects in the signal trace
- Contact bouncing causes multiple states changes per user's action



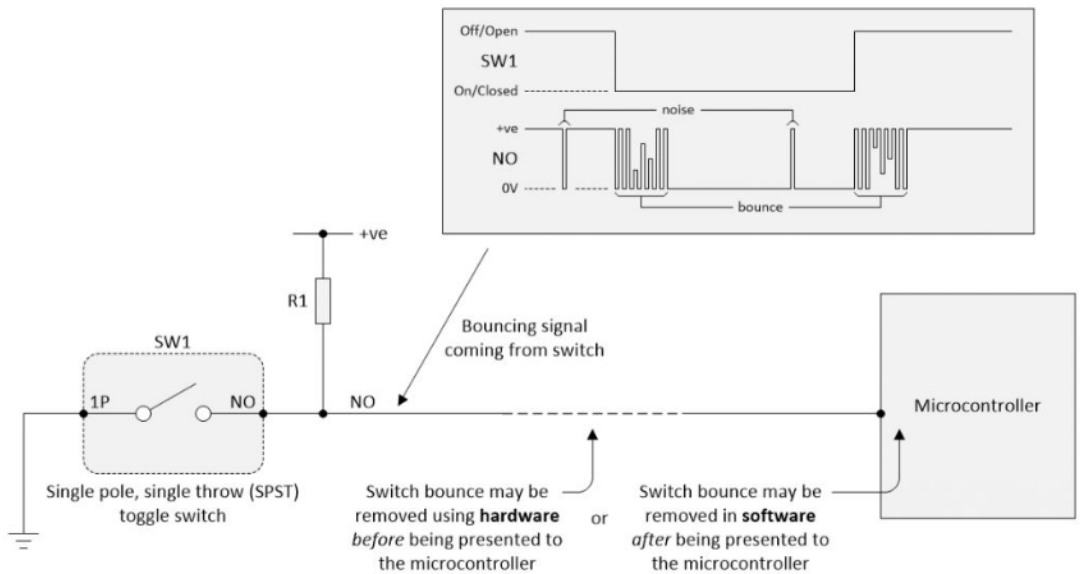


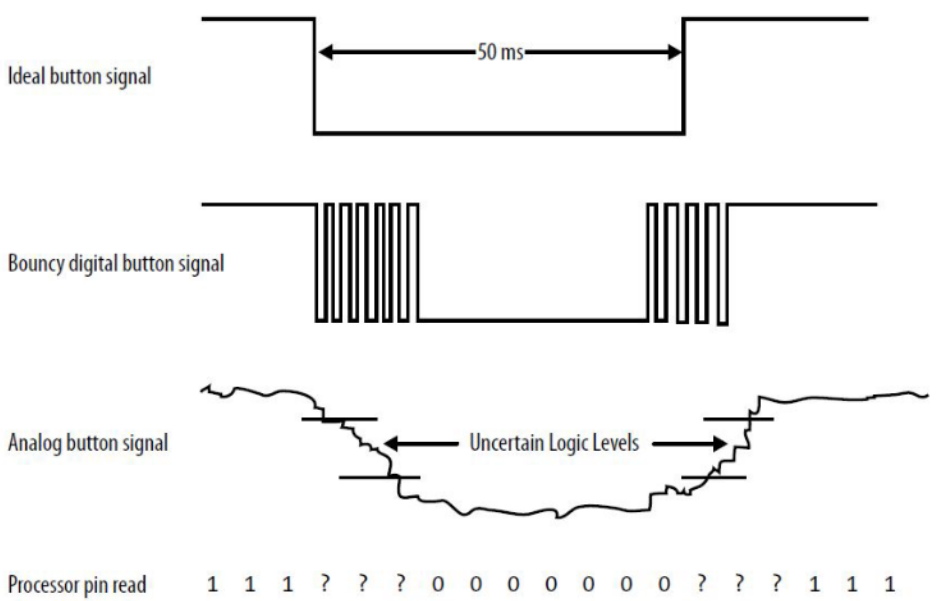




Switches and Contact Bouncing

- To the microcontroller the button appears to be pressed many times for extremely short durations, when you think you've just pressed it once.

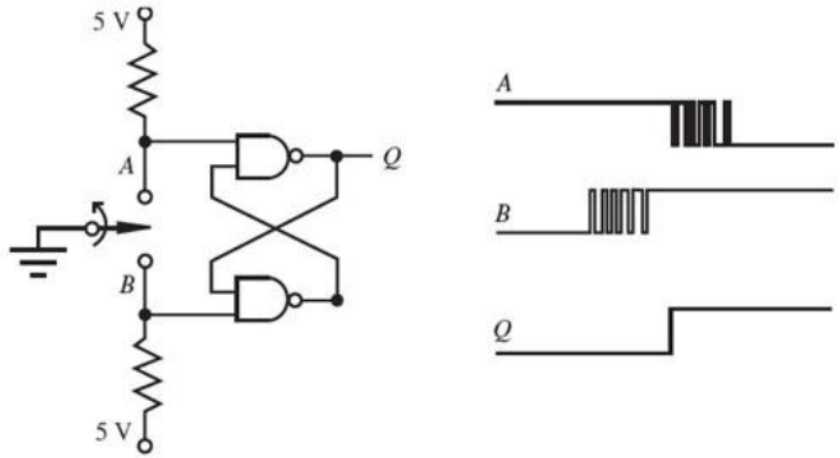




An analog view of what could happen when a button is pressed and only slowly goes into effect. There are parts of the analog signal where the signal is neither high nor low, but somewhere in between. This leads to indeterminate values of the digital signal.

Switches and Contact Debouncing

- Debouncing is all about making sure that when a switch push or release event happened.
- Debouncing can be done in hardware or software
- Software debouncing
  - Read the datasheet of the switch for the period of uncertainty
  - To avoid the garbage near the rising and falling edges, you'll need to look for a long period of consistent signal
    - How long? it depends on the switch and how fast you want to respond to the user.
  - To debounce the switch, take multiple samples of the pin at a periodic interval several times faster than you'd like to respond
  - When there have been several consistent samples, alert the rest of the system that the switch has been changed
  - For example if a button is hitting 10 times per second and it is down half the time (50 ms), we can sample every 5ms and look for five continued consistent samples



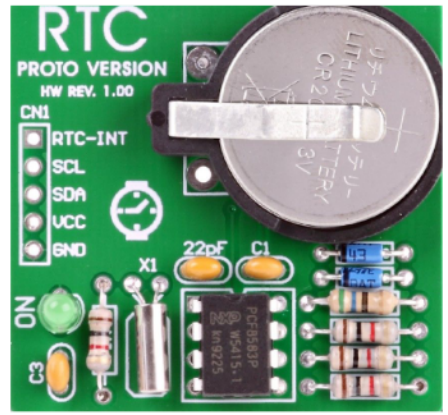
The Mechanical switch takes a time on the order of milliseconds to settle to its final contact.
The NAND gate latch take a time in the order of nanoseconds to supply the output voltage.

Electrical Signals (Teensyduino Functions)

- **pinMode** is used to set a pin as **INPUT / INPUT_PULLUP / INPUT_PULLDOWN** or **OUTPUT**.
- **digitalWrite** is used to write **HIGH(1)** or **LOW(0)** to a digital pin.
- **digitalRead** is used to read the value of a digital pin.
- **analogRead** is used to read the value of an analog pin.
- **analogWrite** is used to write a value to an analog or a pwm pin.
- **analogWriteDAC0** is used to write a value to the **DAC0** pin.
- **analogWriteDAC1** is used to write a value to the **DAC1** pin.
- **analogReadResolution** is used to change the resolution of an input analog pin
- **analogWriteResolution** is used to change the resolution of an output analog pin
- **analogReference** is used to set the type of the voltage reference
- **analogWriteFrequency** is used to change the frequency of the clock of the output analog pins.

Time, Timers and Timing

- Real-Time Clock (RTC)
  - A device to keep track of date and time.
  - Most RTCs use a dedicated 32.768 kHz crystal timer
    - The same frequency used in **quartz** clocks and watches
    - Oscillators with exactly 2^{15} cycles per second
  - RTCs often have an alternate source of power
    - They can also be powered by a battery.
    - E.g. in Teensy a 3V coin cell battery can be connected to VBAT.
  - RTCs can be external or internal. In Teensy and ESP32 there are internal RTCs.
  - In Teensy you can use the Time Library to work with the RTC
  - In ESP32 you can use the POSIX standard time library (time.h) to work with the RTC



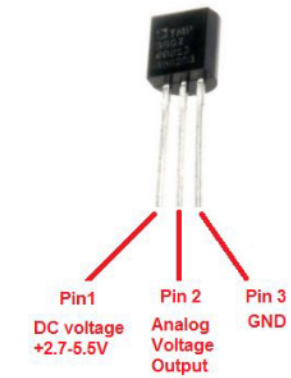
Time, Timers and Timing

- System Time
  - Special registers to keep track of system time
    - In milliseconds and microseconds
    - Counters (32 bits) are used. Counters will overflow.
      - E.g. milliseconds counter overflows after approx. 50 days
  - In Arduino you can use:
    - **millis**() to get milliseconds since the system began running the current program
    - **micros**() to get microseconds since the system began running the current program
    - These function can be used for timing, measuring time and making delays.
      - Look at Metro Library and Timing Functions
- Programmable Interval Timers (PIT): Hardware timers which are used to trigger events in regular repeating intervals. Look at IntervalTimer

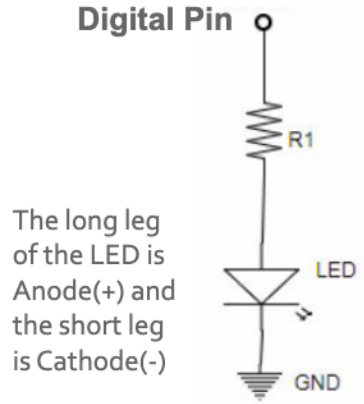
Exercises

- 1. Connect your TMP36 temperature sensor to an **analog** pin on Teensy
  - Read the ambient temperature every second and print it to the terminal.
  - For timing, use the delay() function.
  - Be careful to correct the pins correctly. Look at the datasheet.

Pin 1 to 3.3v
Pin 2 to A14
Pin3 to GND



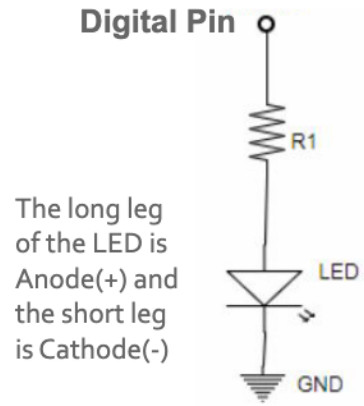
- 2. Connect an LED series with a 120Ω resistor to a digital pin on Teensy
  - Make the LED blinking every second using a **digital** signal
  - For timing, measure time using millis()



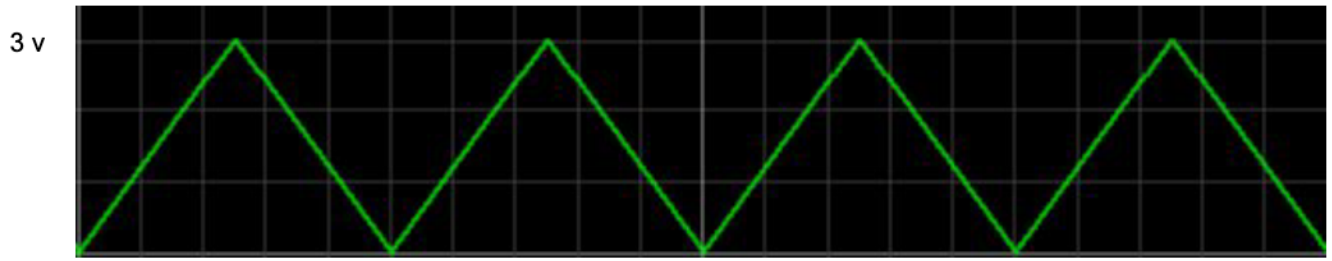
- 3. Connect an LED series with a 120Ω resistor to a pwm pin on Teensy
  - Make the LED fading in/out using **PWM** signals.
  - For timing use the Metro library.

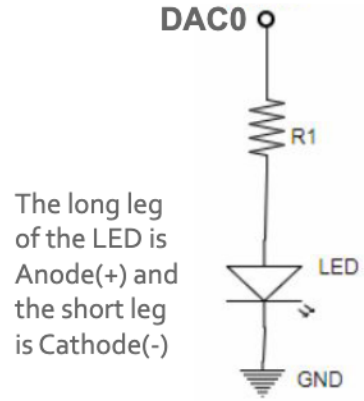
Exercises

- 4. Connect an LED series with a 120Ω resistor to a pwm pin on Teensy
  - Make the LED fading in/out with step 5 using **PWM** signals.
  - For timing use an interval timer with interval 50 ms.



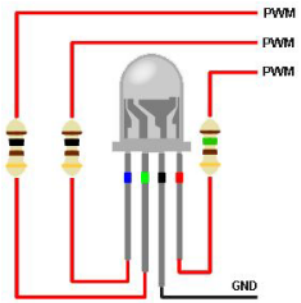
- 5. Connect an LED series with a 120Ω resistor to the **DAC0** on Teensy
  - Make the LED fading in/out using a periodic triangle signal
    - Every 50 ms increase/decrease the voltage with step 0.15v
  - For timing, use the delay() function





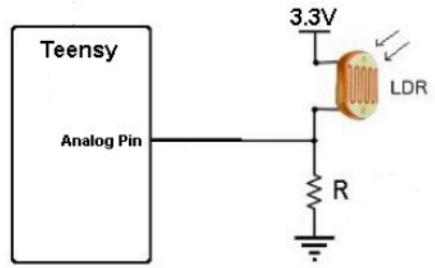
Exercises

- 6. Connect an RGB LED to series with three 120 resistors to 3 analog pins on Teeny
  - Make the LED shining with random colors using **PWM** signals.
  - For timing, use Metro. Every 500ms change the color.
  - For the pinout of the LED look at the datasheet.



- 7. Connect a pushbutton to a digital input pin on Teensy
  - Enable the internal pull-up resistor and read the pin and print it to the terminal.
  - Try the button. Debounce the button using Bounce.h with interval 10ms

- 8. Connect your LDR to an analog pin on Teensy (R = 4.7k)
  - Turn the built-in LED on Teensy on if the ambient light intensity is lesser than 10 lux(1.0 ftc)
  - Look at the datasheet of the LDR in Photoresistor CdS 4 - 7 kohm



- 9. Using an interval timer make an example code which
  - Shows the necessity of using the type qualifier volatile for shared variables

Electrical Signals

- Some useful links
  - Electrical Waveforms
  - Analog vs. Digital
  - Pulse Width Modulation
  - Analog to Digital Conversion
  - Digital to Analog and Analog to Digital Conversion
  - Analog to Digital Converter library for the Teensy
  - A Guide to Debouncing
  - Debounce Your Noisy Buttons
  - Contact Bounce and Debouncing
  - Arduino Tutorial - Button
  - TMP36 Temperature Sensor Arduino Tutorial
  - Turn LED On and Off Through LDR
  - Using An LDR Sensor With Arduino
  - Tutorial 2: RGB LED
  - Arduino - Fading an LED
  - Pulse Width Modulation
  - Bounce Library
  - How to Measure Light Intensity