Chapter 4: Analog Output

EE2405

嵌入式系統與實驗

Embedded System Lab

Data conversion

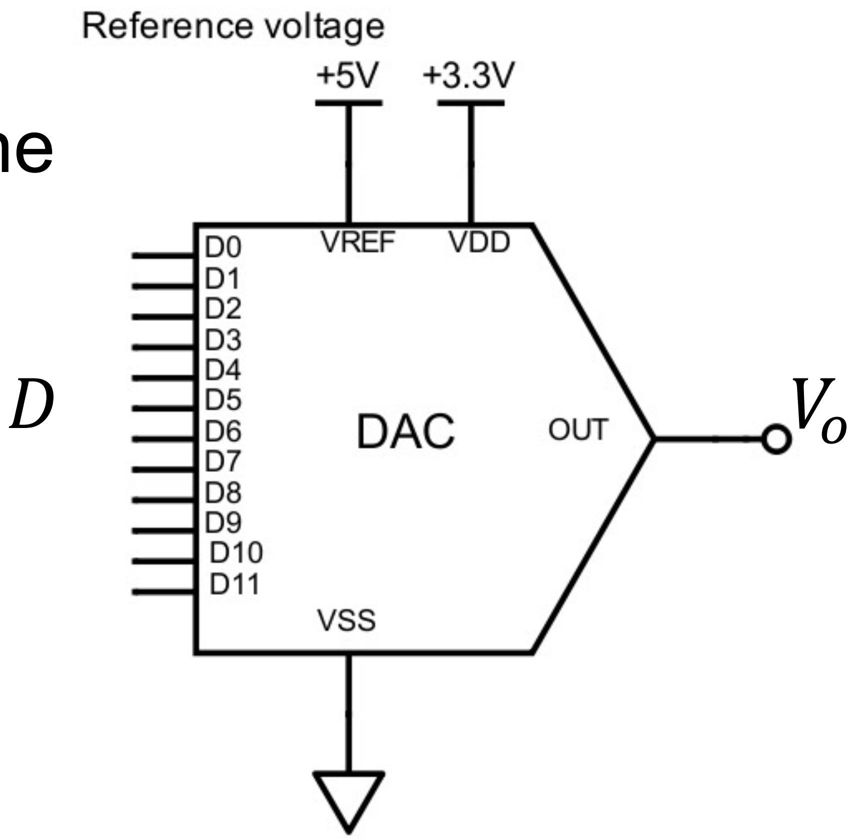
- Microcontrollers compute and store in digital domain, but we need to handle analog signals.
- For example, from a microphone or temperature sensor, we have to convert them into digital form.
  - Analog to digital conversion (ADC)
- After processing the data, they may then need to convert digital data back to analog form, for example to drive a loudspeaker or dc motor.
  - Digital to analog conversion (DAC)

Digital to Analog converter

- A digital-to-analog converter (DAC) converts a binary input number into an analog output.
  - *Vref* is used to define output range
  - Output voltage:

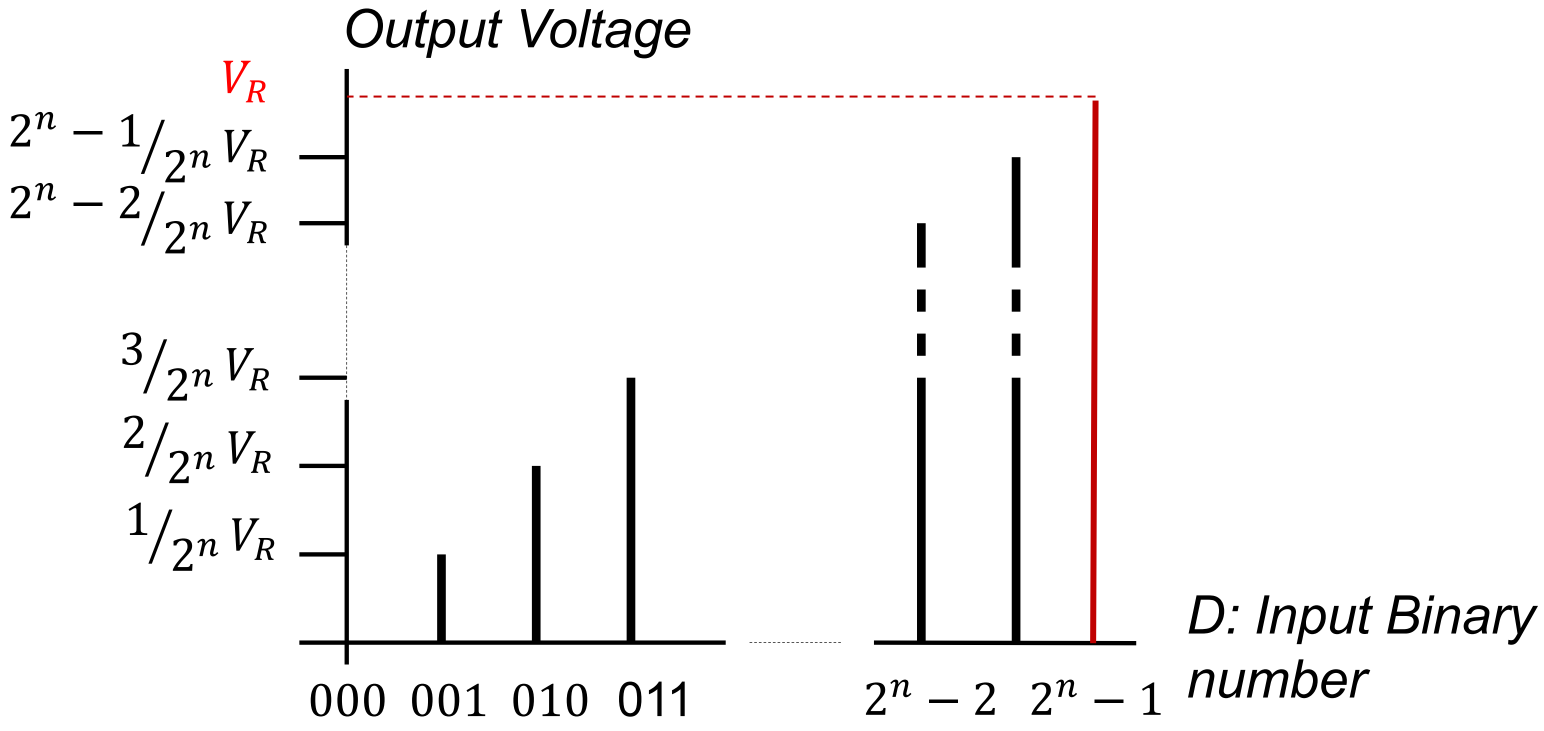
$$V_o = \frac{D}{2^n} V_{REF}$$

  - *D* is the binary number of D11-D0

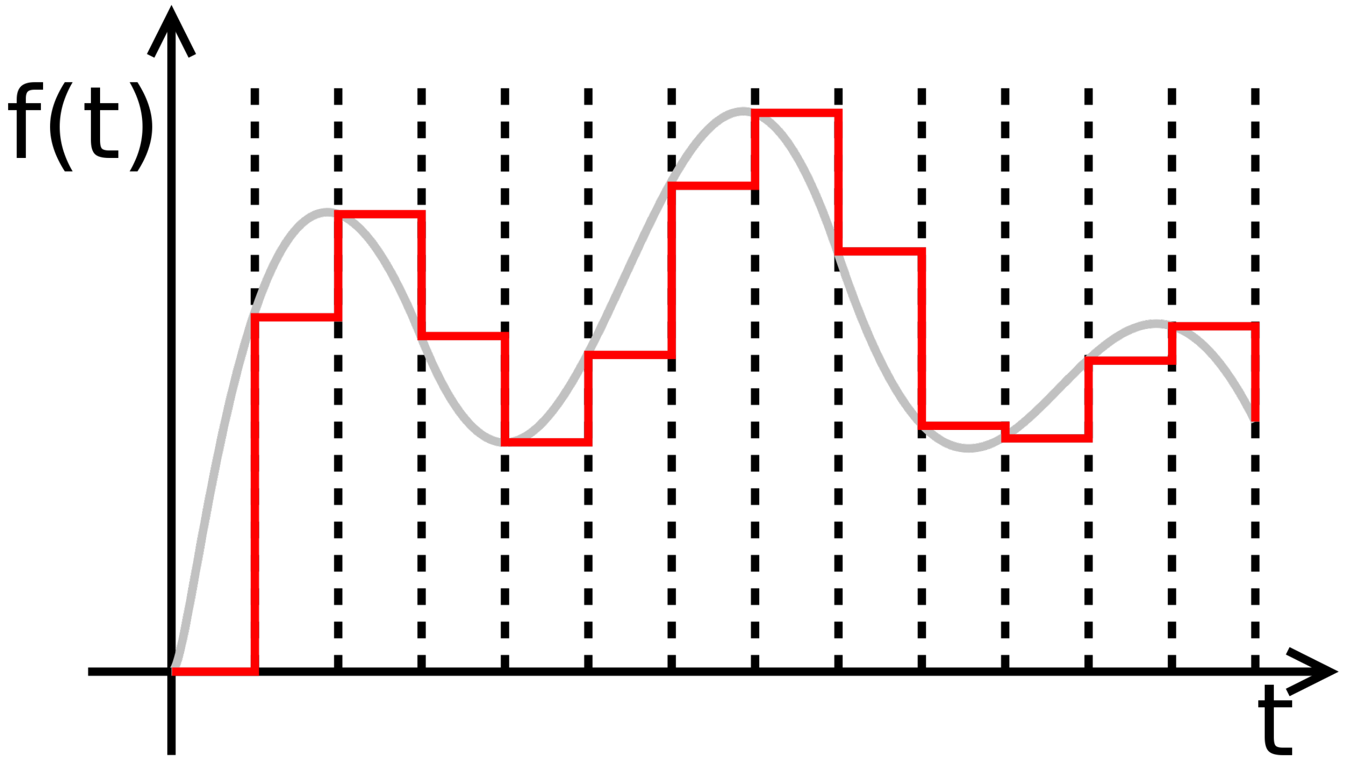


DAC conversion curve

- ST discovery IoT node has two 12-bit DACs; there will therefore be 2^{12} steps in its output characteristic, i.e. 4096. Normally it uses its own power supply voltage, i.e. 3.3 V, as voltage reference. The step size, or resolution, will therefore be 3.3/4096, i.e. 0.806 mV.



- Piecewise constant output of a conventional DAC lacking a reconstruction filter. In a practical DAC, a filter or the finite bandwidth of the device smooths out the step response into a continuous curve.



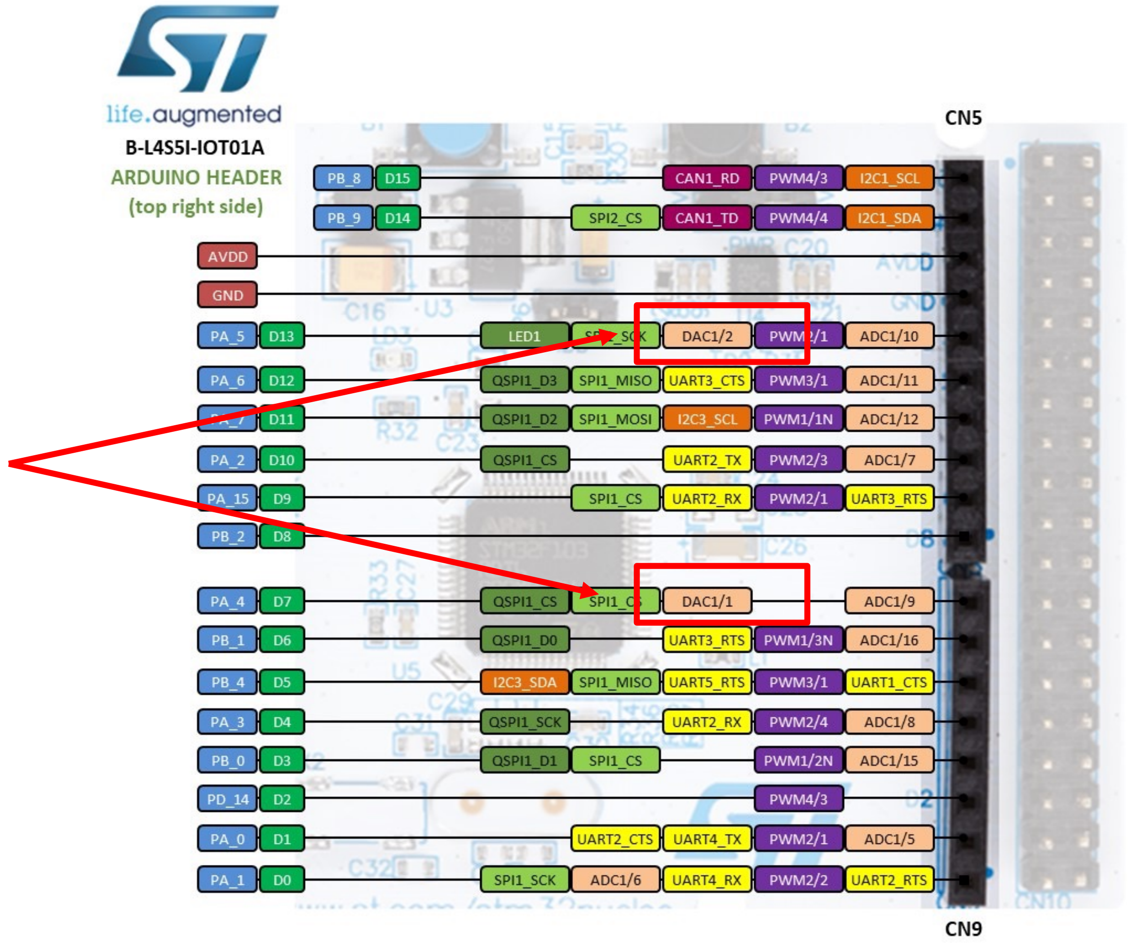
AnalogOut

- AnalogOut API

Function	Usage
AnalogOut	Create a AnanlogOut connected to the specified pin
write	Set the output voltage, specified as a percentage (float)
write_u16	Set the output voltage, specified as an unsigned short in the range [0x0, 0xFFFF]
read	Return the current output voltage, measured as a percentage (float)
operator =	A operator shorthand for write()

DAC Pins

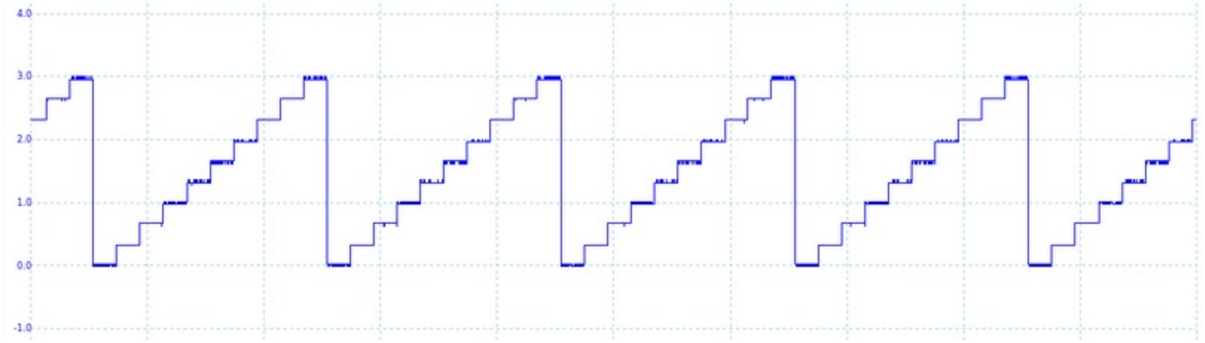
A two-channel 12-bit 1Msps DAC.



life.augmented
B-L4S5I-IOT01A
ARDUINO HEADER
(top right side)
CN5
PB_8 D15 CAN1_RD PWM4/3 I2C1_SCL
PB_9 D14 SPI2_CS CAN1_TD PWM4/4 I2C1_SDA
AVDD
GND
PA_5 D13 LED1 SPI1_SCK DAC1/2 PWM2/1 ADC1/10
PA_6 D12 QSPI1_D3 SPI1_MISO UART3_CTS PWM3/1 ADC1/11
PA_7 D11 QSPI1_D2 SPI1_MOSI I2C3_SCL PWM1/1N ADC1/12
PA_2 D10 QSPI1_CS UART2_TX PWM2/3 ADC1/7
PA_15 D9 SPI1_CS UART2_RX PWM2/1 UART3_RTS
PB_2 D8
PA_4 D7 QSPI1_CS SPI1_CS DAC1/1 ADC1/9
PB_1 D6 QSPI1_D0 UART3_RTS PWM1/3N ADC1/16
PB_4 D5 I2C3_SDA SPI1_MISO UART5_RTS PWM3/1 UART1_CTS
PA_3 D4 QSPI1_SCK UART2_RX PWM2/4 ADC1/8
PB_0 D3 QSPI1_D1 SPI1_CS PWM1/2N ADC1/15
PD_14 D2 PWM4/3
PA_0 D1 UART2_CTS UART4_TX PWM2/1 ADC1/5
PA_1 D0 SPI1_SCK ADC1/6 UART4_RX PWM2/2 UART2_RTS
CN9

Generating a Sawtooth Waveform

```
#include "mbed.h"
AnalogOut  aout(PA_4);
DigitalOut dout(LED1);

int main(void)
{
    while (1) {
        // change the voltage on the digital output pin by 0.1 * VCC
        //  and print what the measured voltage should be (assuming VCC = 3.3v)
        for (float i = 0.0f; i < 1.0f; i += 0.1f) {
            aout = i;
            printf("aout = %1.2f volts\n", aout.read() * 3.3f);
            // turn on the led if the voltage is greater than 0.5f * VCC
            dout = (aout > 0.5f) ? 1 : 0;
            ThisThread::sleep_for(1000);
        }
    }
}
```



Generating a Sine Waveform

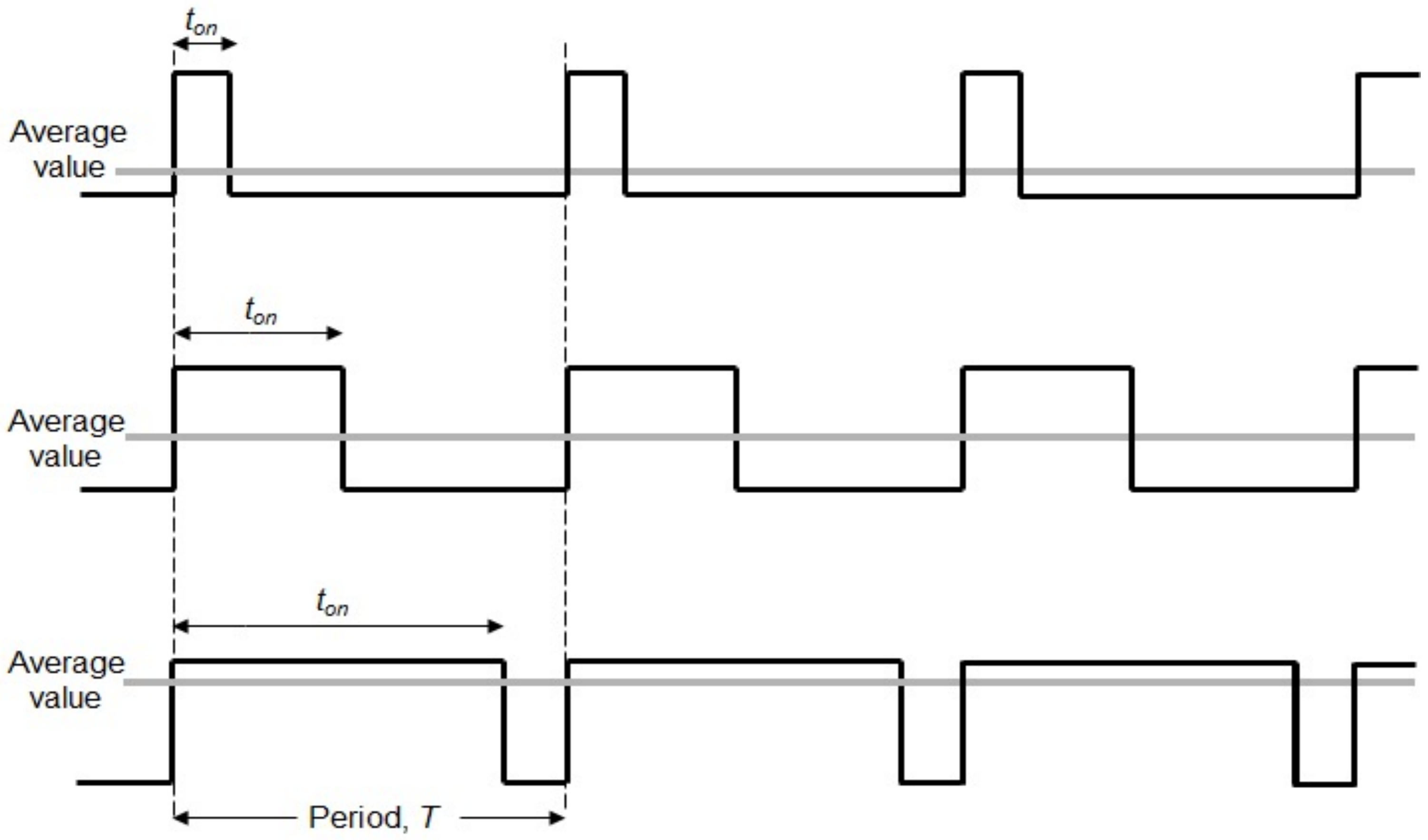
```
#include "mbed.h"

const double pi = 3.141592653589793238462;
const double amplitude = 0.5f;
const double offset = 65535 / 2;

AnalogOut aout(PA_4);

int main()
{
    double rads = 0.0;
    uint16_t sample = 0;

    while (1) {
        // sinewave output
        for (int i = 0; i < 360; i++) {
            rads = (pi * i) / 180.0f;
            sample = (uint16_t)(amplitude * (offset * (cos(rads + pi))) + offset);
            aout.write_u16(sample);
        }
    }
}
```

- Here we apply the **cos( )** function, part of the C standard library.
- We add offset DC to generate only positive output voltages.

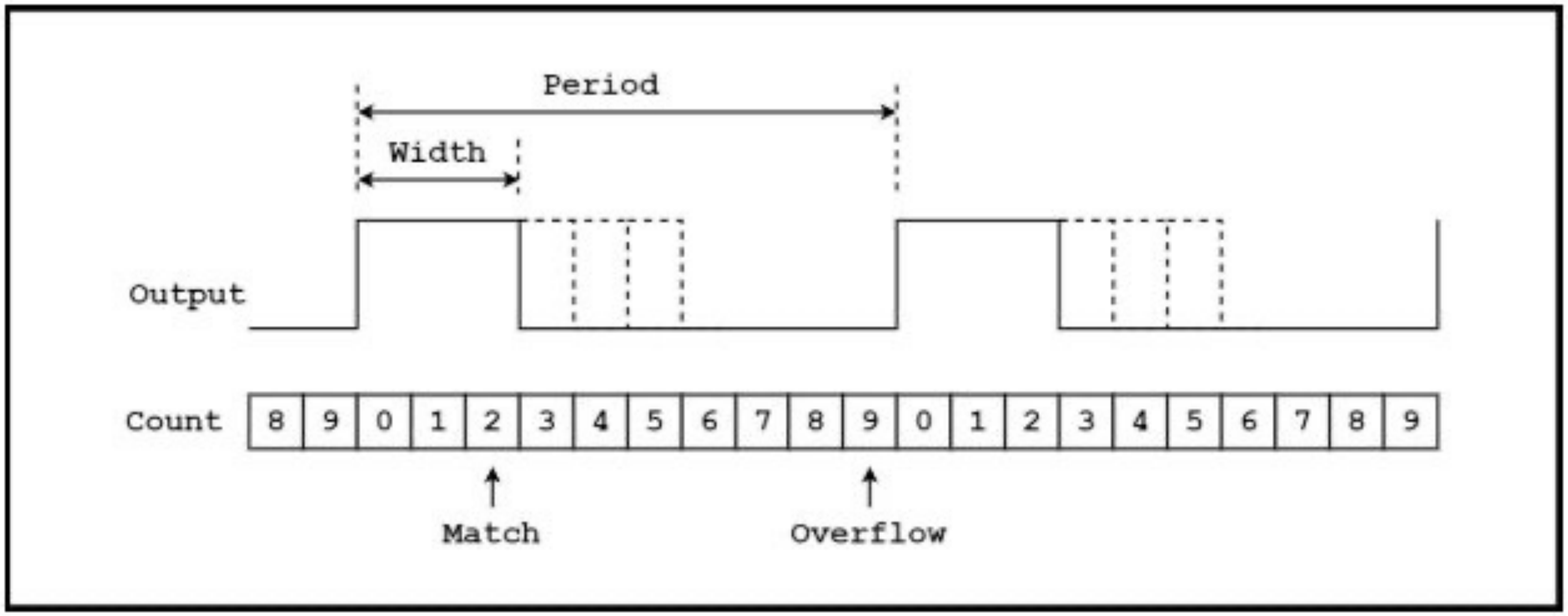
Pulse Width Modulation (PWM)

- Pulse width modulation (PWM) is a fixed-frequency rectangular signal with varying duty cycle.
  - $duty\ cycle = \frac{signal\ on\ time}{signal\ period} \times 100\%$
- A circuit usually take an average of a PWM as a control signal.
  - For example, LED or motor.

PWM Examples



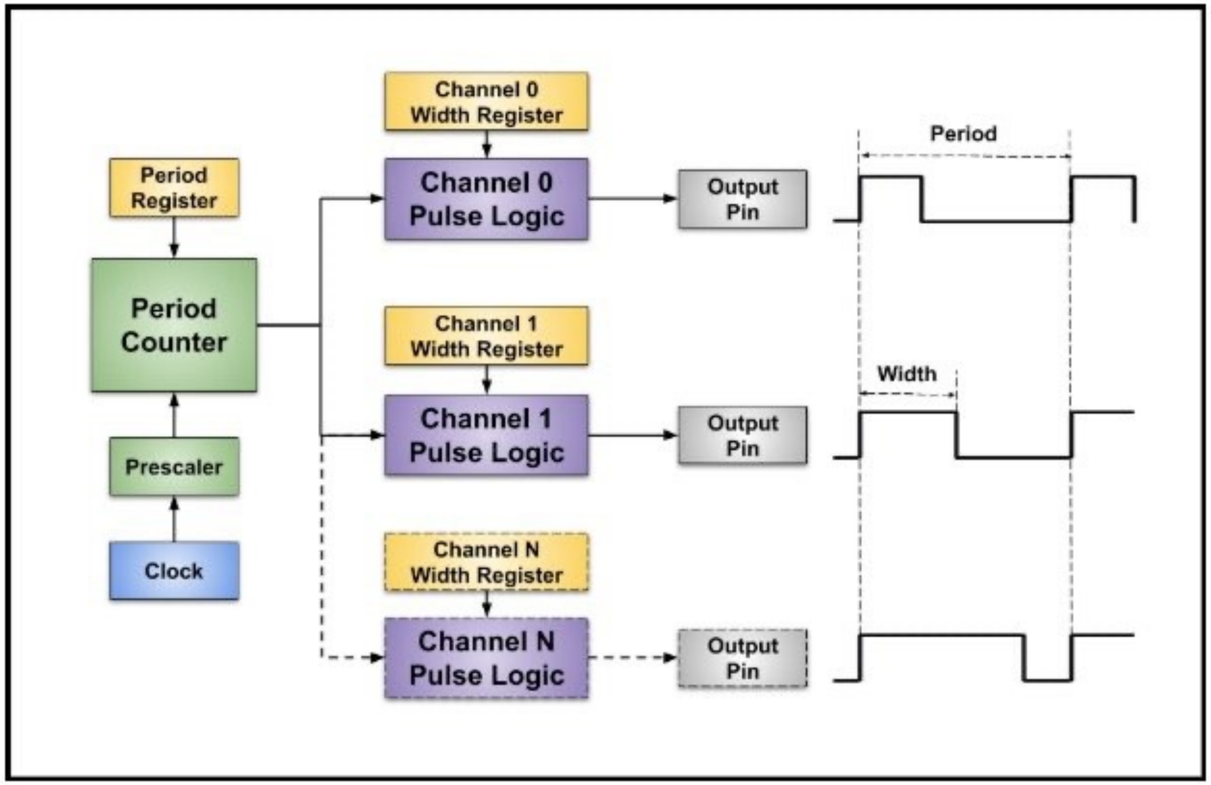
PWM Timers



https://www.allaboutcircuits.com/technical-articles/introduction-to-microcontroller-timers-pwm-timers/

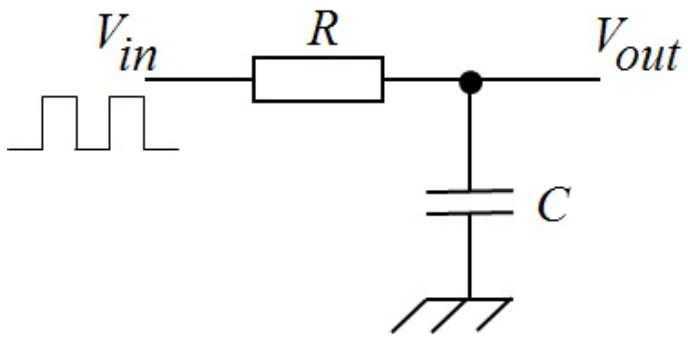
PWM Channels

- We can generate different duty cycle waveforms with the same period (from the same Period Counter).
- In the example, we have one timer to generate 3 channels.

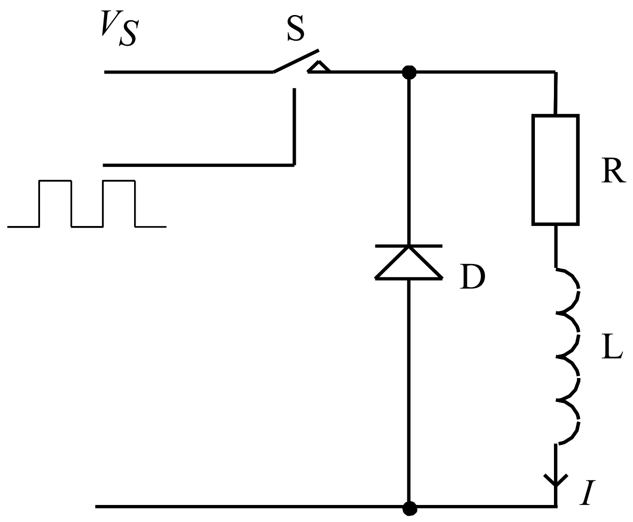


Simple Averaging Circuits

- The average PWM value can be extracted from the PWM stream in a number of ways.
- We can use a low-pass filter, like the resistor-capacitor combination below left. In this case, and as long as PWM frequency and values of R and C are well- chosen, V_{out} becomes an analog output, with a bit of ripple, and the combination of PWM and filter acts just like a DAC.
- Alternatively, if we switch the current flowing in an inductive load, below right, then the inductance has an averaging effect on the current flowing through it. This is important, as the windings of any motor are inductive, so we can use this technique for motor control.



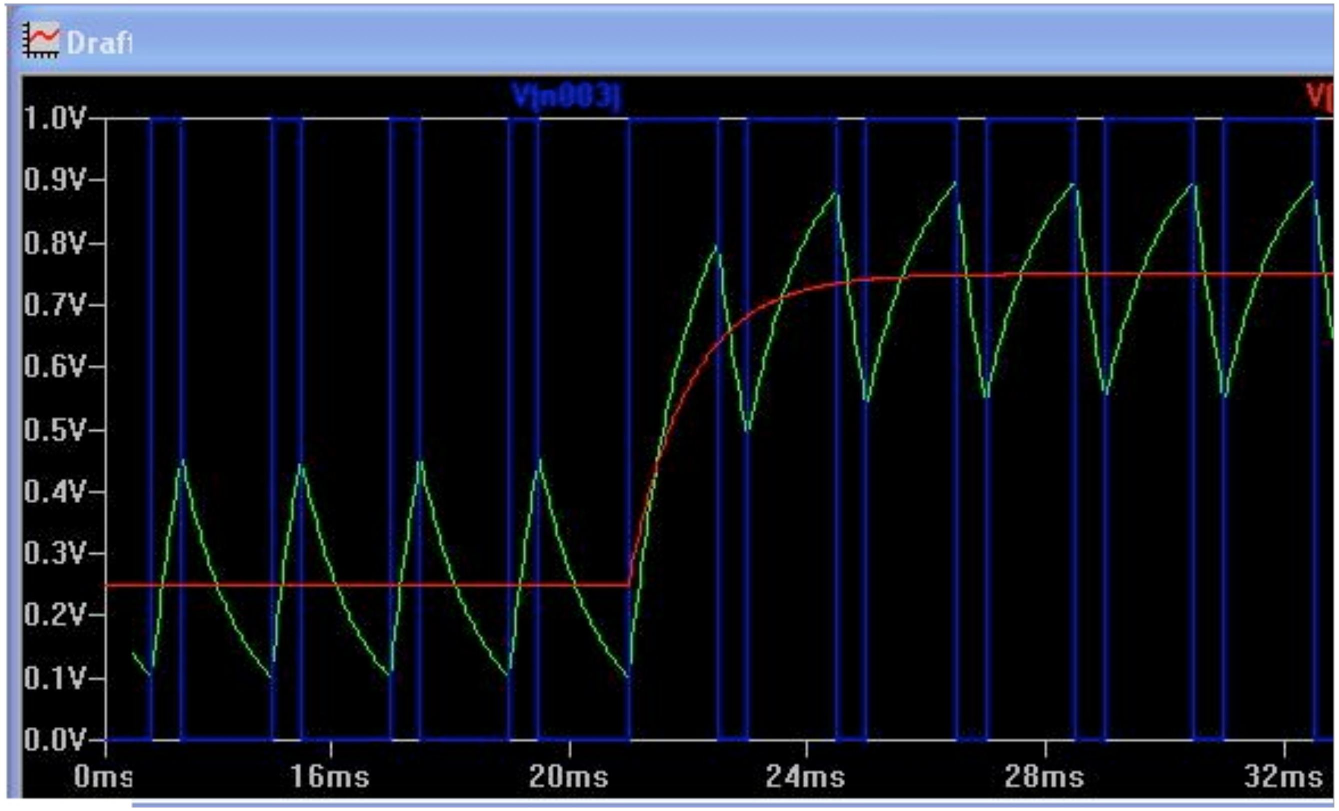
Resistor-capacitor low-pass filter



An inductive load

PWM Filtering Example

- One pole Low-pass with PWM Step Change from D=0.25 to D = 0.75 compared to a second filter with an input step from 0.25V to 0.75V



http://ltwiki.org/images/8/82/PWM_Filters.pdf

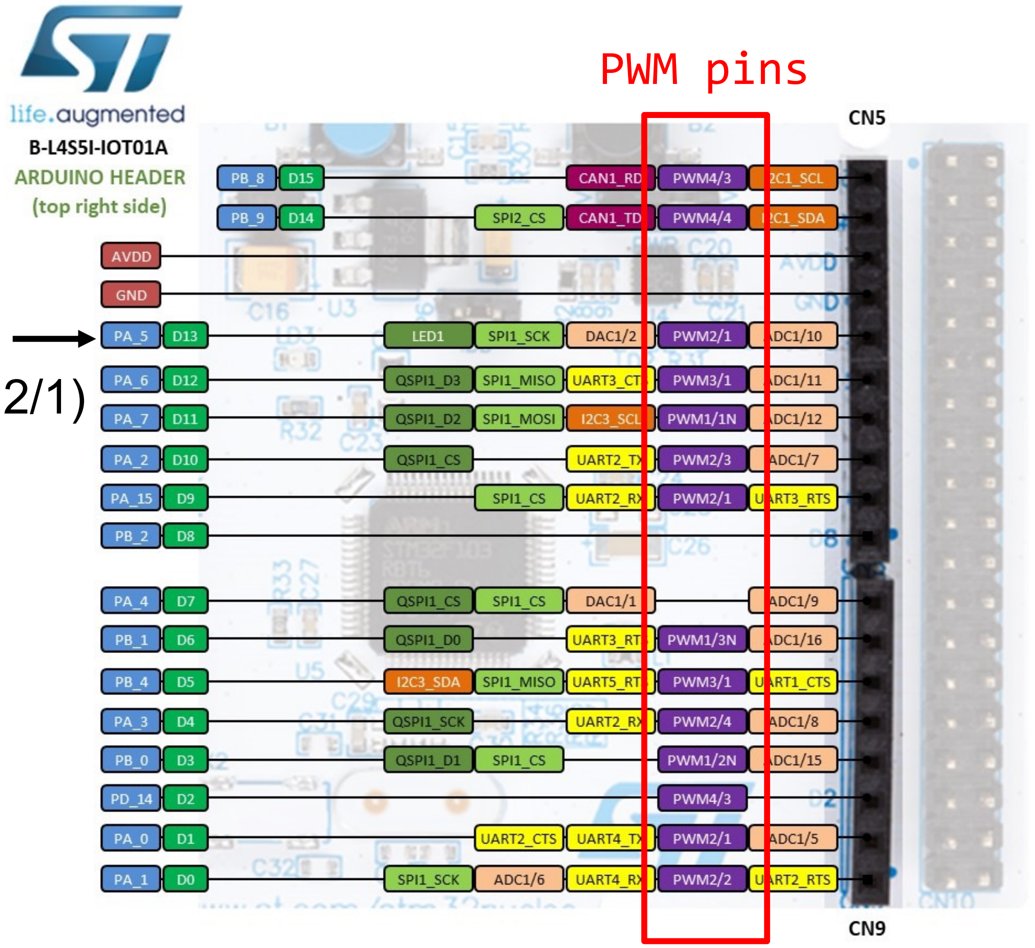
PwmOut

- PwmOut API

Functions	Usage
PwmOut	Create a PwmOut object connected to the specified pin
write	Set the output duty-cycle, specified as a normalised float (0.0 – 1.0)
read	Return the current output duty-cycle setting, measured as a normalised float (0.0 – 1.0)
period	Set the PWM period, specified in seconds (float), keeping the duty cycle the same.
period_ms	Set the PWM period, specified in milliseconds (int), keeping the duty cycle the same.
period_us	Set the PWM period, specified in microseconds (int), keeping the duty cycle the same.
pulsewidth	Set the PWM pulse width, specified in seconds (float), keeping the period the same.
pulsewidth_ms	Set the PWM pulse width, specified in milliseconds (int), keeping the period the same.
pulsewidth_us	Set the PWM pulse width, specified in microseconds (int), keeping the period the same.
operator =	An operator shorthand for write().

PWM Pins

LED1 is also driven by a PWM pin (PWM2/1)


life.augmented
B-L4S5I-IOT01A
ARDUINO HEADER
(top right side)
PWM pins
CN5
PB_8 D15 CAN1_RD PWM4/3 I2C1_SCL
PB_9 D14 SPI2_CS CAN1_TD PWM4/4 I2C1_SDA
AVDD
GND
PA_5 D13 LED1 SPI1_SCK DAC1/2 PWM2/1 ADC1/10
PA_6 D12 QSPI1_D3 SPI1_MISO UART3_CTS PWM3/1 ADC1/11
PA_7 D11 QSPI1_D2 SPI1_MOSI I2C3_SCL PWM1/1N ADC1/12
PA_2 D10 QSPI1_CS UART2_TX PWM2/3 ADC1/7
PA_15 D9 SPI1_CS UART2_RX PWM2/1 UART3_RTS
PB_2 D8
PA_4 D7 QSPI1_CS SPI1_CS DAC1/1 ADC1/9
PB_1 D6 QSPI1_D0 UART3_RTS PWM1/3N ADC1/16
PB_4 D5 I2C3_SDA SPI1_MISO UART5_RTS PWM3/1 UART1_CTS
PA_3 D4 QSPI1_SCK UART2_RX PWM2/4 ADC1/8
PB_0 D3 QSPI1_D1 SPI1_CS PWM1/2N ADC1/15
PD_14 D2 PWM4/3
PA_0 D1 UART2_CTS UART4_TX PWM2/1 ADC1/5
PA_1 D0 SPI1_SCK ADC1/6 UART4_RX PWM2/2 UART2_RTS
CN9

Generating a PWM Waveform

```
#include "mbed.h"

PwmOut led(LED1);

int main()
{
    // specify period first
    led.period(4.0f);        // 4 second period
    led.write(0.50f);        // 50% duty cycle
    //led = 0.5f;            // same as led.write()
    //led.pulsewidth(2);     // set duty cycle time in sec
    while (1);
}
```

This program generates a 0.25 Hz pulse (4s period) with 50% duty cycle, i.e. a perfect square wave, appearing on pin LED1. The LED1 will cycle from dim to bright.

Chapter Review

- A digital-to-analog converter (DAC) converts an input binary number to an output analog voltage, which is proportional to that input number.
- DACs are widely used to create continuously varying voltages, for example to generate analog waveforms.
- Pulse Width Modulation (PWM) provides a way of controlling certain analog quantities, by varying the pulse width of a fixed frequency rectangular waveform.
- PWM is widely used for controlling flow of electrical power, for example led brightness or motor control.