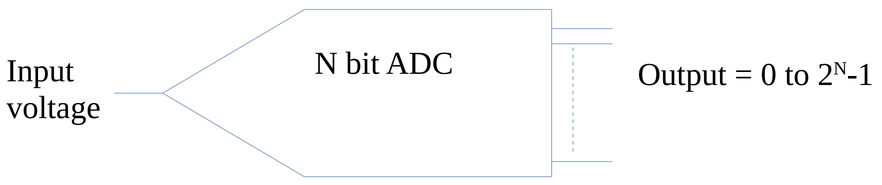
The SysTick timer, the ADC and PWM

In this lab you will explore various aspects of timing and the Analogue to Digital Converter in the STM32L031.

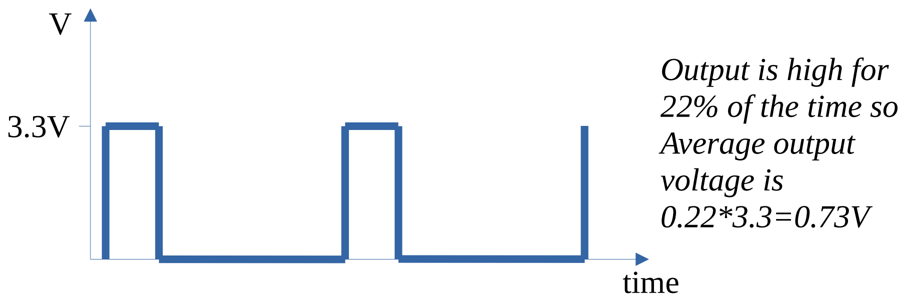
Background.
Analogue to digital converters (ADC's) allow computer systems measure the magnitude of voltage signals – not just whether they are high or low. ADC's output a value which is proportional to the input voltage. The range of values and ADC can output depends on how many data bits it has to work with. For example:

an 8 bit ADC can output values in the range 0 to 2^8-1 or 0 to 255
a 10 bit ADC can output values in the range 0 to 2^{10}-1 or 0 to 1023
a 12 bit ADC can output values in the range 0 to 2^{12}-1 or 0 to 4095

Each ADC has a certain input voltage range. For example, the 12 bit ADC in the STM32L031 has an input voltage range of 0 to 3.3V. If the input is 0V, the output number is 0. If the input is 3.3V, the output number will be 4095. If the input is 1.65V then the output number will be 2048.

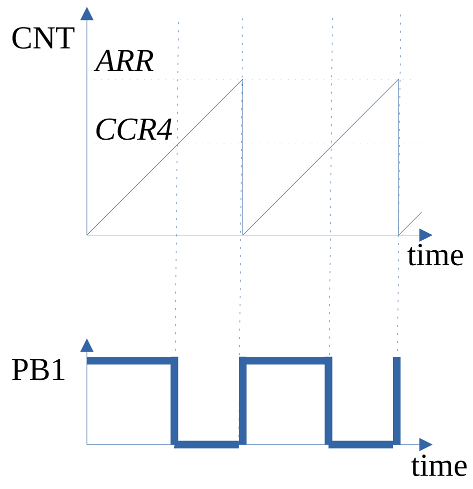


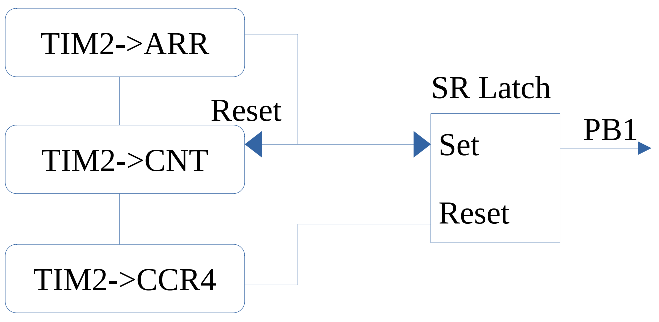
Digital to analogue converters (DAC's) perform the opposite task: they take a number and convert it to a proportional voltage. Typically DAC's are not found on budge microcontrollers. These devices usually use a different mechanism to produce "fake" analogue output: They switch an output pin high and low quickly and vary the percentage time during which the output is high. In the case of our microcontroller, if the output is high 100% of the time then the average output voltage will be 3.3V. If the output is high for 50% of the time then the average output voltage will be 1.65V. This technique is known as Pulse Width Modulation (PWM). Typically PWM is performed using a timer just like we the one we used to produce sound in a previous lab.


V
3.3V
time
*Output is high for 22% of the time so Average output voltage is 0.22*3.3=0.73V*

Part 1: PWM and ADC

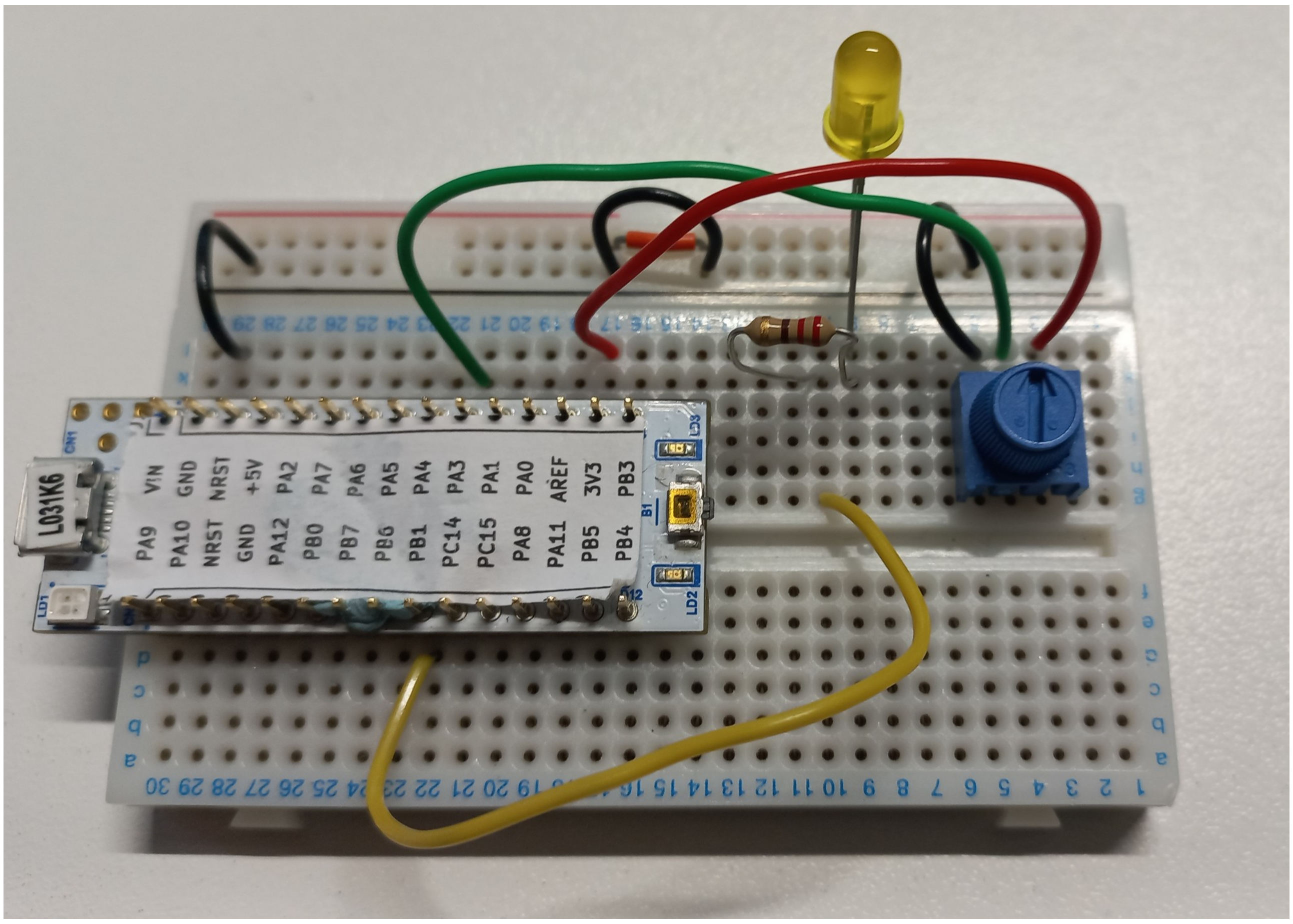
In this lab, Timer 2 (TIM2) is configured to increment the value in CNT until it reaches the value in ARR. The value in CNT is then reset. The value is CCR4 adjusts the length that PB1 is high in an output period.





Tasks
Wire the STM32L031 board as shown below. It includes a component you may not have seen before: a potentiometer. Think of this as a volume knob. As you turn the knob, the voltage on the centre pin of the potentiometer varies between 0 and 3.3V.
Download the **analog_io** starter code from Brightspace
Verify that the brightness of the Yellow LED varies as you turn the potentiometer
Modify the **doPWM** function so that it accepts an argument in the range 0 to 100 (percent).
Verify that the LED brightness still varies through the full range as you adjust the potentiometer.
The PWM system runs at 976 Hz. Change this to 20kHz by changing the value loaded into ARR in the **initPWM** function. You will also need to modify the code in doPWM so that a full turn of the potentiometer corresponds to the full range of brightness of the LED.

PB1 connects to the resistor in series with the yellow LED.
PA3 connects to the middle pin of the potentiometer.



Part 2: The Systick timer

Tasks

Download the **systick** starter code from Brightspace
Verify that the onboard green LED blinks on and off.
Modify the delay code such that it uses the global variable **milliseconds** to implement a calibrated delay function e.g. delay(1000) will result in a delay of 1 second.
Verify that this is working correctly by turning the onboard LED on for 10 seconds, and then off for 10 seconds. You can use the stopwatch feature on your phone to verify the accuracy of the delay function.