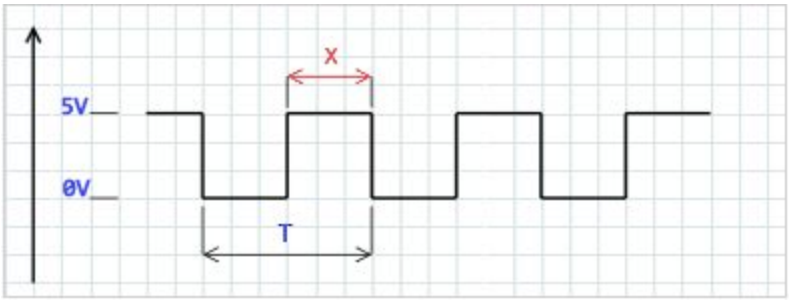
Wykład 4.

PWM - metoda modulacji sygnału prostokątnego poprzez regulację szerokości impulsu.

Załóżmy, że do mikrokontrolera podłączyliśmy diodę świecącą i zaczęliśmy migać nią w pętli. Dioda jest włączona przez sekundę, a przez kolejną pozostaje wyłączona i tak w koło:

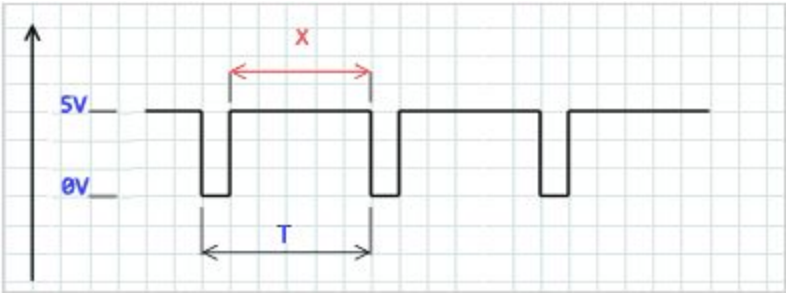


Wartość zaznaczona jako x, to czas, w którym dioda świeci. Natomiast T, to okres z jakim migamy LEDem. Z kolei jego odwrotność, czyli 1/T oznacza częstotliwość. Stosunek czasu, gdy dioda świeci, do czasu gdy jest wyłączona wynosi 1:1. Innymi słowy, pozostaje ona włączona jedynie przez 50% - wypełnienie sygnału.

Podsumowując informacje o naszym sygnale:

1. Amplituda (maksymalna wartość): 5V
2. Okres sygnału: 2 sekundy
3. Częstotliwość sygnał: 1/2 = 0,5 Hz
4. Wypełnienie sygnału: 50%

Tym razem dioda włączona jest przez około 5/6 czasu. Czyli wypełnienie wynosi około 83%:



Za każdym razem zmienia się parametr wypełnienie. Częstotliwość przebiegów pozostaje taka sama przy zachowaniu okresu.

Każdy timer w mikrokontrolerach AVR może pracować w **trybie licznika**, czyli może zliczać impulsy. Oczywiście zlicza od zera do swojej maksymalnej wartości, dla timerów 8-bitowych jest to wartość 255. Ta prosta funkcjonalność daje nam możliwość precyzyjnego odmierzania czasu. Możemy bowiem zaprojektować funkcję oczekującą dokładnie określony okres, po czym program przechodzi do wykonania dalszych zaplanowanych działań. Timer może wygenerować przerwanie wewnętrzne po przepełnieniu jego licznika. Dodatkowo mamy możliwość dostępu do stanu tego licznika w czasie rzeczywistym, możemy odczytywać jego wartość, a nawet ją zmieniać! Dostęp do liczników sprzętowych każdego timera jest możliwy poprzez specjalny **rejestr TCNTx** (**x** oznacza numer timera, czyli TCNT0 oznacza rejestr licznika timera0). Warto dodać, że rejestr ten w przypadku timerów 8-bitowych ma pojemność 1 bajta, 16-bitowych 2 bajty.

Tryb CTC (Clear Timer on Compare Mode) - w celu włączenia trybu CTC w naszym timerze, musimy skorzystać z **rejestru TCCRx** timera ustawiając bity o nazwie **WGMxx,** dodatkowo trzeba też skorzystać z innego przerwania (innego rejestru) obsługującego przepełnienie licznika. Rejestr ten nazywa się **OCRx**. Działa dwa razy wolniej od fast pwm. Po doliczeniu

się skrajnej wartości wartość fali prostokątnej zmienia się na przeciwną. Daje zawsze wypełnienie 50% i sygnał o stałej częstotliwości. Przydatne do grania dźwięków - ton zależy tylko od częstotliwości, a nie wypełnienia.

ATmega Timer/Counter 8bit Timer/Counter0

W trybie CTC rejestr OCR0A służy do manipulowania rozdzielczością licznika. W trybie CTC licznik jest zerowany, gdy wartość licznika (TCNT0) jest zgodna z OCR0A. OCR0A określa najwyższą wartość licznika, a więc także jego rozdzielczość. Ten tryb umożliwia lepszą kontrolę częstotliwości wyjściowej porównania. Wartość licznika (TCNT0) rośnie, dopóki nie pojawi się dopasowanie porównania między TCNT0 i OCR0A, a następnie licznik (TCNT0) jest kasowany.

ATmega Timer/Counter 16bit Timer/Counter1

Timer / Licznik (TCNT1), rejestry porównania wyników (OCR1A / B) i rejestr przechwytywania danych wejściowych (ICR1) są wszystkie 16-bitowe. Rejestry sterowania licznikiem (TCCR1A / B) są rejestrami 8-bitowymi i nie mają ograniczeń dostępu do procesora.

ATmega Timer/Counter 8bit Timer/Counter2

Timer / licznik (TCNT2) i rejestr porównania wyników (OCR2A i OCR2B) są rejestrami 8-bitowymi.

Tryb fast PWM jest prawdopodobnie najczęściej używanym trybem PWM, a także najprostszym. Licznik zapętla się od zera do maksymalnej wartości (255 lub 65 535 lub innej wartości zapisanej w rejestrze OCRxA, w zależności od konkretnego timera i trybu) w sposób ciągły z prędkością określoną przez prescaler. Po drodze licznik jest porównywany z zawartością rejestrów Output Compare (OCRnx). Gdy dochodzi do dopasowania, piny wyjściowe PWM można ustawić lub wyczyścić, a przerwania mogą zostać wyzwolone. Tryb fast PWM jest zasadniczo sprzętową wersją kodu demonstracyjnego pwm.c, który ciągle się zapętlał od 0 do 255 i porównywał wartość licznika ze zmienną cyklu pracy. Ale

korzystanie ze sprzętowego trybu PWM może być znacznie szybsze niż robienie tego samego w kodzie i nie wykorzystuje cykli procesora. Obliczanie częstotliwości PWM dla fast PWM jest łatwe. Weź częstotliwość zegara procesora, podziel ją przez wartość preskalera, a następnie podziel przez liczbę kroków na cykl. Na przykład: prescaler 8, 8-bitowy timera z 256 krokami i 1 MHz zegara procesora. Częstotliwość fast PWM 1 000 000 Hz / 8/256 = 488 Hz.