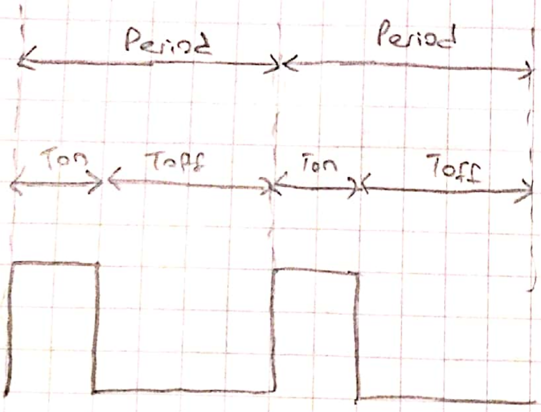
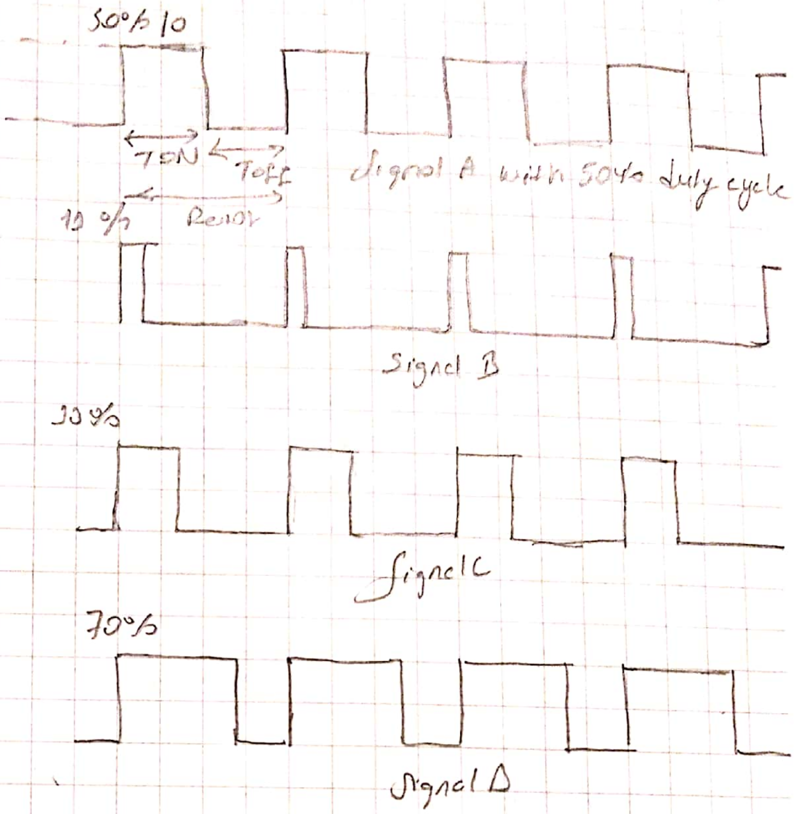
PWM

Pulse Width Modulation (PWM) Sinyal (Darbe) Genişlik Modülasyonu

Nerelerde kullanılır?

- Motor sürücüler
- Voltaj düzenleyiciler
- telekominikasyon kodlama ve çözme





Period = 1 / Frequency

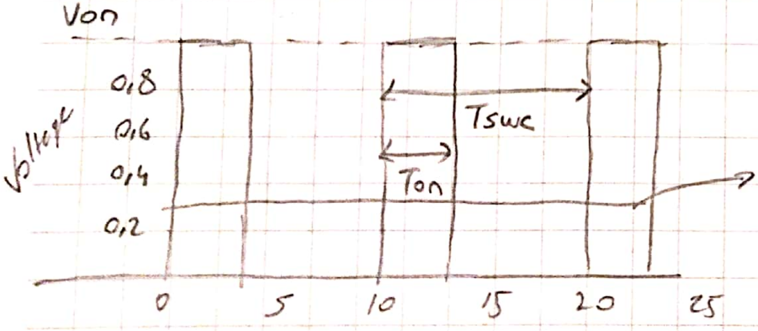
Period = Ton + Toff

$$\underbrace{\text{Duty cycle}}_{\Downarrow} = \frac{Ton}{(Ton+Toff)}*100$$

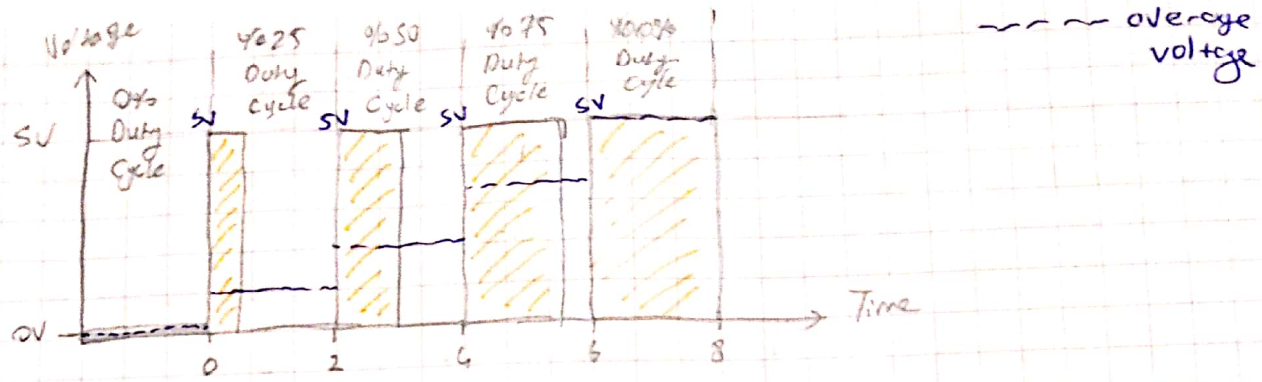
<On Percentage>

Doluluk
Boşluk Oranı

SORU: Aşağıdaki şekilde ortalama gerilim değerlerini söyleyiniz?



$$\langle Vout \rangle = \frac{Ton}{Tswc} , Von$$



Sinyalin Etkin (Effective) ve Ortalama (mean, average) değeri nedir?

Alternatif bir akımın RMS değeri sabit bir dirence yükünden geçen ve aynı miktarda ısı enerjisi üreten DC akımın değerine eşittir. RMS Karesel Ortalama Değer (Root Mean Square) anlamına gelir ve etkin değer, efektif değer olarak da isimlendirilir.

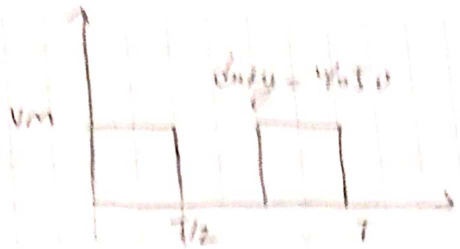
RMS ⇒ Karesel Ortalama Değer $= \sqrt{\dfrac{S_1^2+S_2^2+\cdots+S_n^2}{n}} = \sqrt{\dfrac{\sum_{n=1} S_n^2}{n}}$

$$F_{rms} = \sqrt{\frac{1}{T}\int_0^F f(t)^2\,dt}$$

° Bir işaretin Ortalama Değeri (Mean Value) ise o işaretin zaman ekseni ile arasında kalan alanı ifade etmektedir ve aşağıdaki formül kullanılarak hesaplanır.

$$F_{MeanValue} = \frac{1}{T}\int_0^F f(t)\,dt$$

Kare Dalganın Etkin Değeri



$$V_{Rms} = \sqrt{\frac{1}{T}\int_{t_0}^{t_0+T} v(t)^2 dt}$$

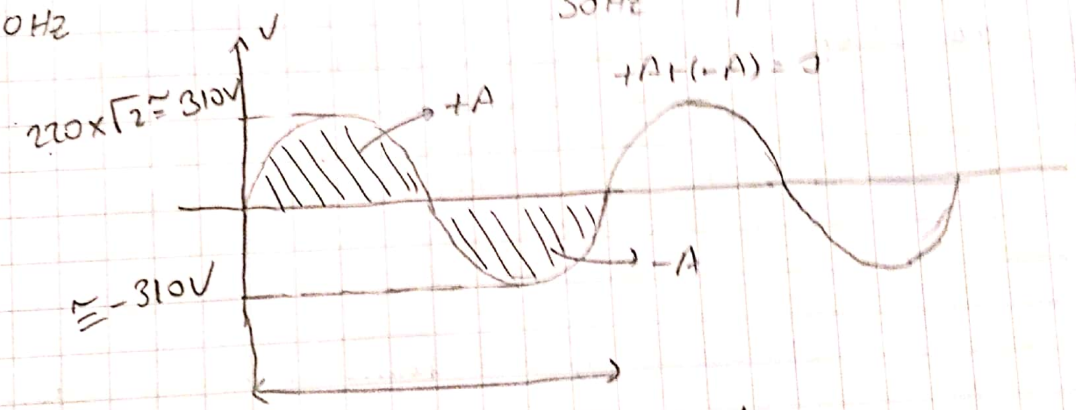
$$V_{ems} = \sqrt{\frac{1}{T}\int_{0}^{T/2} Vm^2 dt} = \sqrt{\frac{1}{T} Vm^2 t \Big|_0^{T/2}} = \sqrt{\frac{1}{T} Vm^2 \left(\frac{T}{2} - 0\right)}$$

$$V_{ems} = \sqrt{\frac{1}{T} Vm^2 \frac{T}{2}} = \frac{Vm}{\sqrt{2}}$$

110 VAC
60 Hz

Şebeke Gerilimi

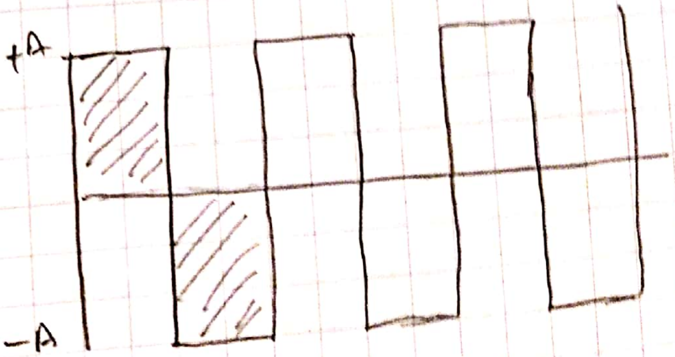
220 VAC
50 Hz
} → RMS olan değer



$$T = 0{,}02\,sn = 20\,ms = \frac{1}{50\,Hz}$$

SORU: Simetrik Bipolar Kare Dalganın Ortalama Değeri nedir?

$$V_{DC} = \frac{1}{T}\int_0^T V(t)dt$$



$$V_{DC} = 0$$

PWM Frekansı Hesaplama

- Period = (Timer_Tick_Frekansı / PWM Frekans) - 1
- PWM Frekans = Timer_Tick_Frekansı / (Period + 1)
- Timer_Tick_Frekansı = (Timer_Clock / Prescaler +1)
- Timer_Clock ⇒ Timer'in bağlandığı hattın clock'u
- Prescaler ⇒ bu değeri biz veriyoruz.

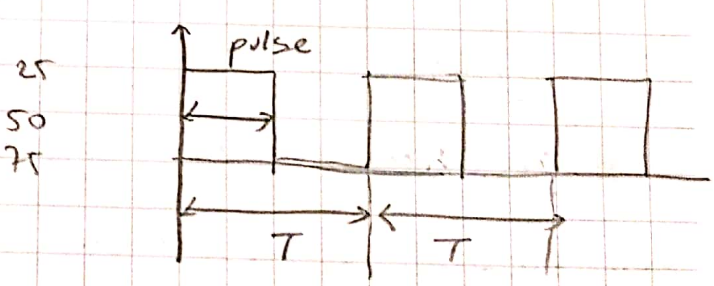
sayaç çözünürlüğü

- Timer_Tick_Frekansı ⇒ Timer sayacının her bir artışında geçen sürenin frekansı

Pulse - Duty Cycle Arası Bağlantı

- Pulse = [(periyot+1) x %DC] - 1 = [(9999+1) * 100 / 100] - 1

Pulse = 9999 //



ÖRNEK 1 : Frekansı 1 Hz olan PWM Sinyal

Timer_Tick_Frekansı = (Timer_Clock / Prescaler +1)

imer_Tick_Frekansı = (84000000 / 8399+1)

imer_Tick_Frekansı = 10000

WM Frekans = Timer_Tick_Frekansı / (Period+1)

M_Frekans = 10000 / (9999+1) = 1Hz

- Frekans 1 Hz olduğu için led hep aynı şiddette yanıyormuş gibi göze görünür. Hatta frekans değeri gözle görülür. Çünkü göz 24 Hz altındaki değerleri algılar.
- Duty Cycle %100 dursa Pulse ne olur? 9999 → Vled = 3.3V
- Duty Cycle %75 dursa Pulse ne dur? 7499 → Vled = 2,475V
- Duty Cycle %50 dursa Pulse ne dur? 4999 → Vled = 1,65V
- Duty Cycle %25 dursa Pulse ne olur? 2499 → Vled = 0,875V

→ 0,875VDc
→ 0,25sn led yanar
0,75sn led yanar

```
//1Hz PWM isareti

_HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 2499); //%25 doluluk oranı
_HAL_Delay(2000);
_HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 4999); //%50 doluluk oranı
_HAL_Delay(2000);
_HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 7499); //%75 doluluk oranı
_HAL_Dely(2000);
_HAL_TIM_SET_COMPARE(&htim2, TIM_CHANNEL_1, 9999); //%100 doluluk oranı
_HAL_Delay(2000);
```

→ 1,65 VDc

ÖRNEK2: Frekansı 10 KHz olan PWM sinyal

- Timer_Tick_Frekansı = (Timer_Clock / Prescaler+1)
- Time_Tick_Frekansı = (84000 000 / 83+1)
- Timer_Tick_Frekansı = 1000000
- PWM Frekans = Timer_Tick_Frekansı / (Period+1)
- PWM Frekans = 1000 000 / (99+1) = 10KHz

Ayrı ayrı gelistirelim, Göz farkedemez. Vortalama değeri düşümünü gösterir.

- Duty-cycle %100 dursa Pulse ne olur? 99 → Vled = 3,3V → 1,24
- Duty-cycle %75 dursa Pulse ne olur? 74 → Vled = 2,475V → 1,49
- Duty-cycle %50 dursa Pulse ne olur? 49 → Vled = 1,65V → 1,74
- Duty-cycle %25 dursa Pulse ne olur? 24 → Vled = 0,875V → 1,99

PIC 16F628

/ /

ÖRNEK3 Pwm frekansı 50Hz ayarlama

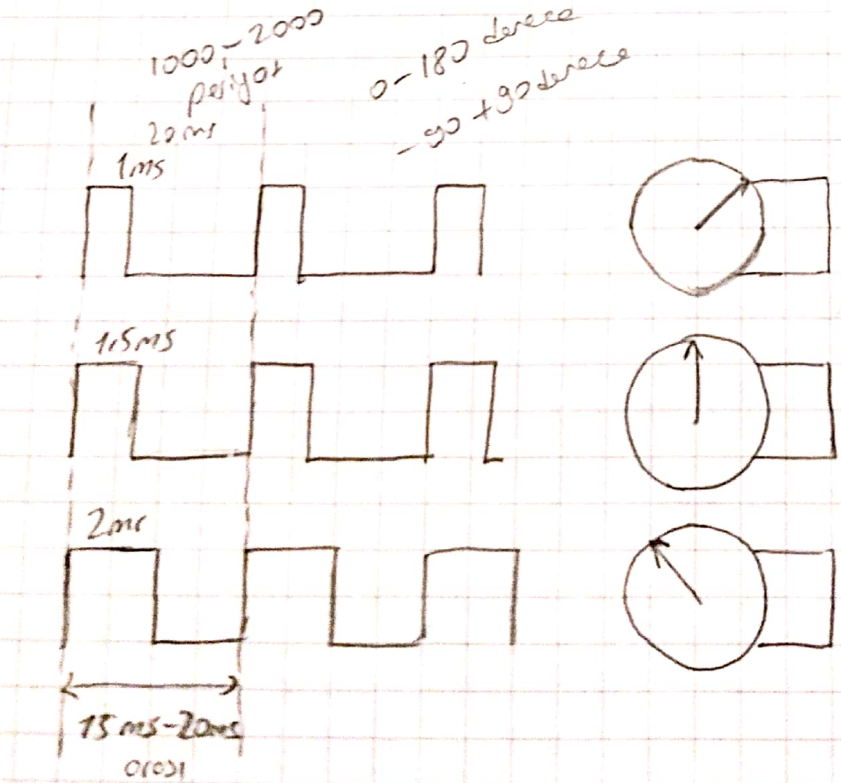
- Period = (Timer-Tick-Frekansı / PWM Frekans) - 1
- PWM Frekans = Timer_Tick_Frekansı / (Period +1)
- Timer-Tick-Frekansı = (Timer-Clock / Prescaler +1)
- Periyot = 20ms counter periyot = 20000-1 = 19999
- Çözünürlük 1µs → 20000 x 1µs = 20ms
- Çözünürlük 1µs → Sayaç frekansı 1/1µs = 1000000 Hz

Çiyo! PWM Frekans = Timer-Tick_Frekansı /(Period +1)

50 = Timer_Tick_Frekansı / (1999+1)

Timer-Tick-Frekansı = 1000000 Hz = 84000000 / (Prescaler +1) ↗ 83

< 0, 90 derecelerde counter değeri ne olmalıdır. >



- 20ms de 1ms ise
- 2000 de x? 1000

- x = (1ms * 2000) / 20ms
- X = 1000

1,5 msn için 1500

2msn için 2000

a) CCS C ve PIC ile PWM kullanarak DC motorun 2 buton yardımı ile hızını azaltıp, arttıracağız.

○ PIC16F877A ○ BC547 Transistör ○ 1 adet motor ○ 2 adet buton ○ 220Ω direnç

```
#include <16F877A.h>  // Kullanacağımız pic'in kütüphanesini ekliyoruz

# FUSES NOWDT         // Watch dog timer kapatıyoruz.
# FUSES NOBROWNOUT    // Brownout reset kapatıyoruz.
#FUSES NOLVP          // Düşük voltaj programlama kapalı

#use delay (crystal = 400000)  // Osilatör frekansı ayarlanıyor.

# use fast-io (c)
unsigned long int K;  // data = adc değeri, K pwm değeri olacak

// Buton pinlerini tanımlıyoruz.
# define HIZLANDIR PIN_D0
# define YAVAŞLAT PIN_D1

void main()
{

// PIC ayarlarını yapıyoruz
   setup_psp (PSP_DISABLED);
   setup_spi (SPI_SS_DISABLED);
   setup_timer_0 (RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1 (T1_DISABLED);  // T1 timer kapatıyoruz.
   setup_timer_2 (T2_DIV_BY_16, 255, 1);  // PWM için timer2 kullanılıyor, max 255
   set_tris_c (0X00); // Tüm C portu sıfırlanıyor.
   setup_ccp1 (CCP_PWM);  // CCP1 ayarlanıyor PIC C2 portunu kullanacağız.
   set_pwm1_duty (K);  // PWM'in ayarlıyoruz.
   delay_us (20);

   while(1)
   {

   while (input (HIZLANDIR)) {  // While ile HIZLANDIR butonuna basılıp basılmadığını kontrol.
   K = K+20;  // K değişkeninin değerini 20 arttırıyoruz.
   set_pwm1_duty (K);  // K değişkeninin değerini pwm'e aktarıyoruz.
   delay_ms (100);  // 100ms bekletiyoruz.
   }
```

```
    while (input(YAVASLAT)){  // while ile YAVASLAT butonuna basılıp basılmadığı kontrolü
    k=k-20;    // K değişkeninin değeri 20 azaltılır.
    set_pwm1_duty(k);    // K değişkeni PWM'e aktarılır.
    delay_ms(100);    //100ms bekle
    }

    }
  }
```

```
#include <16f877a.h>
#device adc=3
#use delay (clock=4000000)
#fuses XT, NOWDT, NOPUT, NOLVP, NOCPD, NOPROTECT, NODEBUG, NOBROWNOUT, NOWRT

#use fast_io(a)
#use fast_io(b)
#use fast_io(c)

#define use_portb_lcd TRUE
#include <lcd.c>
unsigned int deger;  →  float voltaj;
void main()
{
  setup_psp(PSP_DISABLED);
  setup_timer_1(T1_DISABLED);
  setup_CCP2(CCP_OFF);

  setup_timer_2(T2_DIV_BY_4,255,1);
  setup_CCP1(CCP_PWM);

  setup_adc(adc_clock_div_32);
  setup_adc_ports(ALL_ANALOG);  //AN4

  set_tris_a(0X20);  // 0b00100000
  set_tris_b(0X00);
  set_tris_c(0X00);

  set_adc_channel(4);
  delay_us(20);
  output_c(0X00);
lcd-init();
printf(lcd-putc, "\f DCMOTOR \n HIZ KONTROLU");
delay-ms(1000);

set-pwm1-duty(0);

While(True) {
deger = read-adc();
voltaj = deger / 51,0;

output_high(pin-c3);
output-low(pin-c4);

// if (deger > 255)
// deger = 255;

set_pwm1_duty(deger);

printf(lcd-putc, "\f Voltaj = %f V", voltaj);
lcd_gotoxy(1,2);
printf(lcd-putc, " Deger = %u", deger);
delay_ms(1000);

  }
}
```