# **Vaja 7** – PWM z STM32F4 Discovery

1. **Cilj naloge**: S pomočjo programskega okolja **STM32CubeIDE** in HAL knjižnicami sprogramirajte mikroprocesor tako, da boste generirali PWM signal in ga tudi krmilili na izbranem izhodu STM32F4 Discovery. Za preizkus potrebujete osciloskop.
2. **Postopek inicializacije periferije**.
3. Zaženite **STM32CubeIDE** in ustvarite nov STM32 projekt (pod zavihkom *information Center*). V zavihku *Board selector* s pomočjo filtrov *Vendor*, *Type* in MCU/MPU *Series* izberite ustrezno razvojno ploščo (v našem primeru STM32F401C-Discovery), kliknite *Next,* projekt poimenujte **vaja7\_PWM** in kliknite *Finish* (na možnosti opcije za prenastavitev periferije izberite *Yes*, izbrana naj bo tudi opcija perspektive za STM32CubeMX).
4. V levem *Pinout* oknu razširite nabor možnosti za *Timers* ter za časovnik *TIM1. Clock Source* nastavite kot ***Internal Clock***. Prvi kanal aktivirajte kot ***PWM Generation CH1****.* Kateri pin ste omogočili? \_\_PE9\_\_\_\_\_\_\_\_. Kaj se izpiše poleg pina? TIM\_CH1.
5. V *Clock Configuration* spremenimo takt časovnika *APB1 Timer Clocks (MHz)* na **16 MHz**.
6. V Oknu *Configuration* kliknemo za *TIM1* Vrednost *Prescaler* v zavihku *Counter Settinngs* določite tako, da bo časovnik delal s frekvenco **1 MHz.** Koliko je vrednost *Perscaler* (namig; delitelj) ? \_\_\_\_16\_\_\_\_\_\_\_\_\_\_\_\_\_.
7. Parameter *Counter Period* nastavimo na **100** in s tem še dodatno znižamo takt časovnika. Koliko znaša sedaj? \_\_\_\_\_\_\_\_10\_\_kHz.
8. V *PWM Generation Chanel* nastavite *Pulse (16 bits value)* na **50**. Kaj pomeni ta parameter? Namig – največja vrednost je lahko 100 odstotkov (znak za odstotek v polja ne pišemo). Ta parameter pomeni širina signala.
9. Na izbrani izhodni PWM pin priključite sondo osciloskopa (ne pozabite sondo ozemljiti na GND). Vključite osciloskop in ustrezno nastavite merilno območje za x in y os.
10. Sedaj generirajte kodo tako, da enostavno kliknete ikono *Save* in po potrebi še enkrat potrdimo generiranje kode.
11. **Programiranje v** **IDE:**
12. V skrajno levem oknu *Project Explorer* poiščemo **main.c** datoteko pod *Core* 🡪 *Src* 🡪 *main.c* (dvokliknite na datoteko, odpre se tekstovni urejevalnik za main.c).
13. Poiščite prenastavljeni parameter *Pulse* (ki je 50) v vaši kodi in prepišite ukaz, ki ga je generiral CubeMX:  
    \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_sConfigOC.Pulse = 50;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
14. V *User code begin 2* inicializiramo časovnik za PWM z ukazom:

HAL\_TIM\_PWM\_Start(&TIM\_HANDLE\_TYPE, TIM\_CHANNEL\_1);

Ime tega paramtera boste našli v vrstici pod *Private variables ----*. Del zgornje kode v rdečem torej zamenjajte z **imenom** te ročice (TIM\_HandleTypeDef **XXXX**;). Drugi parameter je izbana številka kanala za generiranje PWM signala.

1. V *User code begin 4* ne napišemo ničesar!!!
2. **Naložitev kode (*Run*) in opazovanje delovanja:**
3. Kodo preverite s tipko ***Build*** (ikona za kladivce - *Debug*). Ko je preverjanje končano lahko preverimo, če smo med pisnjem kode naredili kakšno napako sintakse, sicer se pod kodo v oknu *Console* izpiše 0 errors.
4. Priklopite STM32F4Discovery na vaš računalnik preko USB kabla.
5. S tipko ***Run***(zelena ikona za play – *Run as STM…*)prenesete program na STM32F0Discovery.
6. V oknu *Edit Configuration* kliknemo OK. Nekaj sekund bo na ploščici STM izmenično utripala zelena in rdeča LED, ko je program naložen, sveti LED rdeče.
7. Z osciloskopom preverite izhodni signal (uporabite sondo za priklop na ustrezni pin, ne pozabite na GND) in delovanje **posnemite** s telefonom (MP4 datoteka).
8. **Dodatne nastavitve kode v programu IDE:**
9. V kodi spremenite vrednost širine pulza na 25 %. Zapišite popravljeni ukaz v kodi:

\_\_\_\_\_\_\_sConfigOC.Pulse = 12,5;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ .   
Ponovno naložite program in s telefonom **posnemite** signal na zaslonu osciloskopa.

1. V *User code begin 1* deklariramo spremenljivko :

uint16\_t dutyCycle = 10;

V *User code begin 3* prepišemo naslednje ukaze:

htim1.Instance->CCR1 = dutyCycle;

dutyCycle+=10;

if(dutyCycle>90) dutyCycle=10;

HAL\_Delay(1000);

Ponovno naložite program in s telefonom **posnamite** signal na zaslonu osciloskopa.

Zapišite kaj počnejo ukazi v 1.,2. in 3. vrstici (v user code begin 3):

1. Nastavi spremenljivko duty cycle.
2. Spremeljivki prišteje 10.
3. Če ima spremenljivka duty cycle vrednost ki je večja od 90, nastavi spremenljivko na 10.
4. **Vaš projekt** (datoteko *main.c* [15%], slikovne izrezeke *Pinout* mikroporcesorjaiz CubeMX [5%] okno od *TIM1 configuration-parameter settings* [10%] *,* trije kratki MP4 videposnetki prikaza signalov na osciloskopu [15%], fotografija nastavitev osciloskopa z generiranim PWM signalom na zaslonu [10%]) **naložite v Github** kot nov *Repository* z imenom **Vaja7-PWM-STM32F4** [5%]**.** V *readme* datoteki zapišite vse **odgovore** [30%] na vprašanja ter **komentar** [10%] na delovanje. **Oceno** pridobite iz Github dokumentacije!
5. Dodatna video pomoč z razlago: [Video Tutorial 14](https://www.youtube.com/watch?v=vDvbO_BAYvc).