VJEŽBA 2: STM32F407 ULAZI I IZLAZI OPĆE NAMJENE. A/D PRETVORNIK.

I. Cilj vježbe

Upoznati se s načinom upotrebe i programiranjem ulaza i izlaza opće namjene (engl. *General Purpose Input and Output* - GPIO) na mikroupravljaču STM32F407VG. Korištenje GPIO pina kao digitalnog ulaza i izlaza te kao ulaza za A/D pretvornik. Upoznavanje s CMSIS i Standard Peripheral Library.

II. Opis vježbe

Na laboratorijskim vježbama koristi se STM32F4 Discovery razvojna ploča kao hardverska komponenta dok se Atollic TrueStudio koristi kao pripadno integrirano razvojno okruženje. U vježbi se demonstriraju načini rada i programiranja ulaza i izlaza opće namjene (GPIO) mikroupravljača STM32F407VG. Za demonstraciju rada GPIO kao digitalnog izlaza odnosno ulaza koriste se LED odnosno tipkalo koji su ugrađeni na STM32F4 Discovery razvojnu ploču. Analogna funkcija pina i korištenje A/D pretvornika za mjerenje analognih veličina demonstriraju se spajanjem potenciometra na odgovarajući GPIO pin.

II.1. Ulazi i izlazi opće namjene (GPIO) mikroupravljača STM32F407

Ulazi i izlazi opće namjene (GPIO) je sučelje koje postoji gotovo na svim mikroupravljačima i predstavlja sučelje koje nema predefiniranu zadaću već se način rada i karakteristike podešavaju iz programskog koda tj. korisničke aplikacije. Svaki GPIO je generički pin mikroupravljača na koji je moguće spojiti određenu periferiju poput tipkala, LED, senzora i sl.
GPIO sučelje je na mikroupravljaču STM32F407VG () prisutno u obliku setova pinova opće namjene koji se nazivaju portovi i označavaju se kao GPIO_A, GPIO_B, GPIO_C, GPIO_D, GPIO_E. Svakom portu pripada ukupno 16 pinova.

Svaki GPIO pin može se konfigurirati softverski upisivanjem odgovarajućih vrijednosti u registre koji pripadaju GPIO portu od interesa. Moguće je konfigurirati sljedeće:

1) Način rada: ulazni način rada, izlazni način rada, alternativna funkcija te analogna funkcija.
   Registar: GPIOx_MODER

2) Tip izlaza: push-pull ili open drain
   Registar: GPIOx_OTYPER

3) Brzina rada pina: spora, srednja, brza ili velika
   Registar: GPIOx_OSPEEDR

4) Definiranje *defaultne* naponske razine: interni pull-up ili interni pull-down
   Registar: GPIOx_PUPDR

pri čemu je "x": A, B, C, D, E u slučaju STM32F407VG mikroupravljača (npr. konfiguracijski registar porta D za način rada je GPIOD_MODER). Pojedinačni pin određenog porta obično se označava kraticom PXn pri čemu je "X" oznaka porta, a "n" broj pina u rasponu od 0 do 15 (npr. četvrti pin porta D označava se s PD3). Detaljan opis rada GPIO sučelja s opisom svih registara moguće je pronaći u RM0090 Reference manual.

II.1.1. Izravni rad s registrima GPIO sučelja

Upisivanje odgovarajućih vrijednosti u registre mikroupravljača moguće je obaviti izravnim zapisivanjem vrijednosti na memorijske lokacije pojedinih registara. Memorijski rasponi gdje se nalaze registri svakog GPIO porta dani su u RM0090 Reference manual na stranici 65. dok se tzv. *offseti* svakog pojedinog registra od početne adrese mogu pronaći na str. 286. Upisivanje 32-bitnih vrijednosti u konfiguracijske registre najčešće se izbjegava zbog čitljivosti koda pa se konfiguracija GPIO najčešće obavlja pomoću bit operacija, postavljanjem samo određenih bitova registra u 0 ili 1. Primjer podešavanja registra porta D pod nazivom GPIOD_MODER dana je u tablici 2.1 gdje se bit na mjestu 30 postavlja u 1. Na taj način se pin PD15 postavlja u izlazni način rada.

Rad GPIO periferije određen je signalom takta. Kako bi se takt za GPIO sučelje aktivirao, potrebno je konfigurirati odgovarajuće registre dijela mikroupravljača zaduženog za reset i distribuciju takta (engl. *Reset and clock control* - RCC). Kako bi se omogućio signal takta za GPIO potrebno je postaviti odgovarajuću vrijednost u registar RCC AHB1

peripheral clock register (RCC_AHB1ENR), str. 180. u RM0090 Reference manual. U tablici 2.1 dan je primjer aktiviranja signala takta za GPIO port D (pin PD15 koristi se kao izlazni pin i postavlja se u logičk "1").

Tablica 2.1 Primjer podešavanja registara `GPIOD_MODER, GPIOD_DR` i `RCC_AHB1ENR.`

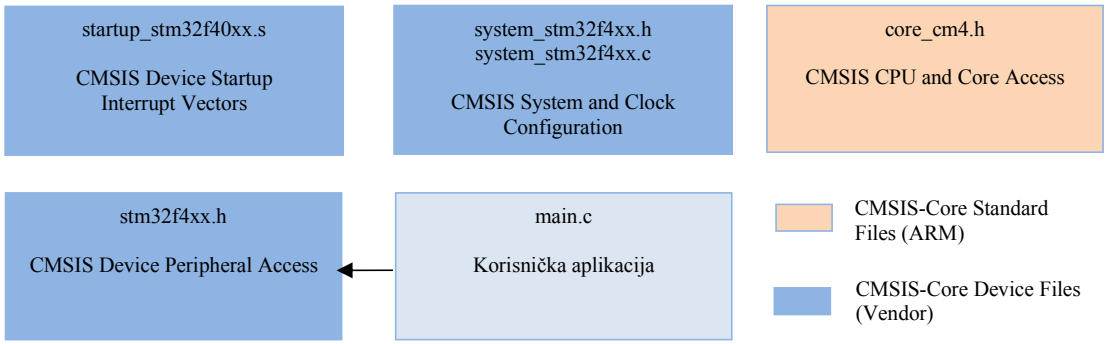
```
#define MASK(x) (1U << (x))

volatile unsigned int* GPIO_D = (unsigned int*)0x40020C00;
volatile unsigned int* GPIO_D_MODER_reg = (unsigned int*)0x40020C00;
volatile unsigned int* GPIO_D_ODR_reg = (unsigned int*)(0x40020C14);
volatile unsigned int* RCC_AHB1ENR_reg = (unsigned int*)(0x40023830);

*RCC_AHB1ENR_reg |= MASK(3);    //uključivanje takta za port D
*GPIO_D_MODER_reg |= MASK(30);  //PD15 se koristi kao izlazni pin
*GPIO_D_ODR_reg |= MASK(15);    //postavljanje PD15 u logičku "1"
```

II.1.2. Rad s registrima GPIO sučelja pomoću CMSIS

Iako je moguće pristupiti određenoj memorijskoj lokaciji deklariranjem i definiranjem pokazivača prema memorijskim adresama u dokumentaciji mikroupravljača, najčešće se koriste već postojeće deklaracije i definicije registara periferije. The ARM® Cortex® Microcontroller Software Interface Standard (CMSIS) je apstrakcijski sloj hardwarea za Cortex-M procesore koji je nezavisan od samog proizvođača, a sastoji se od nekoliko komponenata: CMSIS-CORE, CMSIS-RTOS, CMSIS-DSP, CMSIS-Driver, CMSIS-Pack, CMSIS-SVD, CMSIS-DAP i CMSIS-NN. Najvažnija je CMSIS-CORE komponenta koja sadrži datoteke vezane za podizanje sustava te datoteke koje omogućuju pristup jezgri procesora i periferiji. U slučaju STM32F407VG procesora CMSIS-CORE se sastoji od nekoliko datoteka kako je prikazano na slici 2.1.



Sl. 2.1 CMSIS-CORE datoteke.

Prilikom rada s GPIO sučeljem mikroupravljača iz korisničke aplikacije moguće je koristiti deklaracije i definicije koje se nalaze u zaglavlju naziva `stm32f4xx.h`. Naime, u ovoj datoteci nalaze se specifične informacije za STM32F407VG mikroupravljač, kao što su brojevi prekidnih rutina, podatkovne strukture, bit definicije registara i sl., a koje znatno olakšavaju programiranje i korištenje periferije poput GPIO. Dio `stm32f4xx.h` datoteke dan je u tablici 2.2 a) gdje je prikazana struktura podataka koja se može koristiti za rad s GPIO sučeljem. U tablici 2.2 b) dan je primjer podešavanja GPIO pina koji odgovara podešavanju datom u tablici 2.1.

Tablica 2.2 Korištenje CMSIS-CORE datoteka za podešavanje registara GPIO.

a) Isječak iz `stm32f4xx.h`

```
typedef struct
{
  __IO uint32_t MODER;    /*!< GPIO port mode register, Address offset: 0x00*/
  __IO uint32_t OTYPER;   /*!< GPIO port output type register, Address offset: 0x04*/
  __IO uint32_t OSPEEDR;  /*!< GPIO port output speed register, Address offset: 0x08*/
  __IO uint32_t PUPDR;    /*!< GPIO port pull-up/pull-down register, Address offset: 0x0C*/
  __IO uint32_t IDR;      /*!< GPIO port input data register, Address offset: 0x10*/
  __IO uint32_t ODR;      /*!< GPIO port output data register, Address offset: 0x14*/
  __IO uint16_t BSRRL;    /*!< GPIO port bit set/reset low register, Address offset: 0x18*/
  __IO uint16_t BSRRH;    /*!< GPIO port bit set/reset high register, Address offset: 0x1A*/
  __IO uint32_t LCKR;     /*!< GPIO port configuration lock register, Address offset: 0x1C*/
  __IO uint32_t AFR[2];   /*!< GPIO alternate function registers, Address offset: 0x20-0x24*/
} GPIO_TypeDef;
```

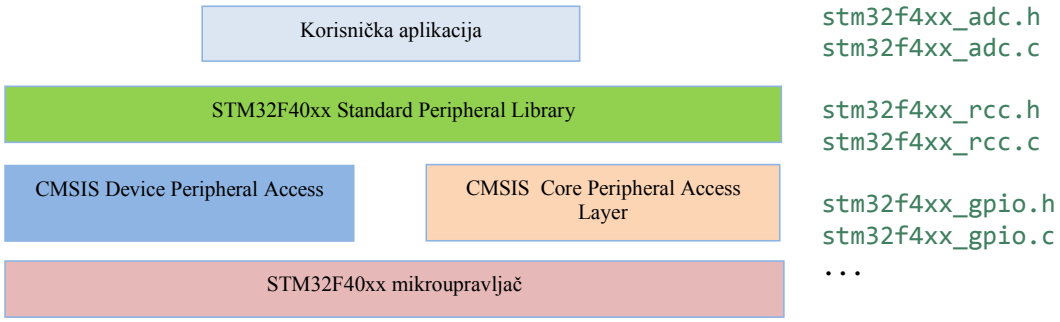
b) Primjer podešavanja konfiguracijskih registara pomoću CMSIS-Core datoteke stm32f4xx.h

```
#include "stm32f4xx.h"
#define MASK(x) (1U << (x))

RCC->AHB1ENR |= MASK(3);     //uključivanje takta za port D
GPIOD->MODER |= MASK(30);    //PD15 se koristi kao izlazni pin
GPIOD->ODR |= MASK(15);      //postavljanje PD15 u logičku "1"
```

II.1.3. Rad s registrima GPIO sučelja pomoću Standard Peripheral Library

Jedan od načina podešavanja i korištenja GPIO sučelja podrazumijeva korištenje gotovih biblioteka za rad s GPIO sučeljem. U slučaju STM32 serije mikroupravljača ova biblioteka naziva se Standard Peripheral Library (SPL) i zapravo je skup .h i c. datoteka (često se nazivaju i upravljački programi - driveri). Ove datoteke sadrže podatkovne strukture i funkcije za podešavanja i korištenje periferije mikroupravljača. Sama biblioteka je zapravo dodatni apstrakcijski sloj koji se zasniva na prethodno spomenutim CMSIS datotekama kao što prikazuje slika 2.2. Svaka periferija mikroupravljača ima odgovarajuću .h i .c datoteku kao npr. stm32f4xx_gpio.h i stm32f4xx_gpio.c koje sadrže podatkovne strukture i funkcije za rad s GPIO sučeljem. Uključenje odnosno isključenje datoteka pojedine periferije u projekt najčešće se obavlja unutar datoteke stm32f4xx_conf.h.



Sl. 2.2 STM32 Standard Peripheral Library.

Kako bi se koristilo GPIO sučelje pomoću SPL najprije je potrebno uključiti stm32f4xx_gpio.h datoteku u projekt. Naputci kako koristiti GPIO driver iz SPL nalaze se u datoteci stm32f4xx_gpio.c u obliku komentara naziva "How to use this driver". U tablici 2.3 dan je primjer korištenja SPL za rad s GPIO pri čemu se pin 15 porta D (PD15) koristi kao izlazni pin u push-pull konfiguraciji. GPIO_InitTypeDef je struktura koja se koristi za konfiguraciju GPIO pina. Zapisivanje konfiguracije obavlja se pozivanjem funkcije GPIO_Init. Za postavljanje GPIO pina u logičku "1" moguće je koristiti funkciju GPIO_SetBits.

Tablica 2.3 Podešavanje i korištenje GPIO pomoću SPL.

```
#include "stm32f4xx.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_gpio.h"

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);  //uključivanje takta za port D
GPIO_InitTypeDef GPIO_InitStructure;                   //inicijalizacijska struktura
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_15;             //pin 15 se konfigurira
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;          //koristi se kao izlazni pin
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;         //push-pull konfiguracija

GPIO_Init(GPIOD, &GPIO_InitStructure);   //zapisivanje u konfiguracijske registre porta D
GPIO_SetBits(GPIOD,GPIO_Pin_15);         //postavljanje PD15 u logičku "1"
```

II.2. Analogno digitalni pretvornik

STM32F407VG sadrži tri 12-bitna A/D pretvornika. Najznačajnije karakteristike A/D pretvornika:

- Podesiva rezolucija (12-bitna, 10-bitna, 8-bitna ili 6-bitna)
- Ukupno 19 multipleksiranih kanala koji omogućuju mjerenje analognih signala s 16 vanjskih izvora (GPIO pinovi), dva interna izvora te napona baterije
- Napajanje V_{DDA}: 2.4V do 3.6V (puna brzina) ili 1.8V do 3.6V (sporija brzina pretvorbe)

- Ulazni signal u rasponu $V_{SSA} \leq V_{IN} \leq V_{REF+}$ pri čemu je $1.8V \leq V_{REF+} \leq V_{DDA}$, a $V_{SSA}=V_{SS}$ (analog ground)
- Način konverzije: pojedinačna (engl. s*ingle)* i kontinuirana (engl. *continuous*)
- Mogućnost aktiviranja pretvorbe vanjskim okidačem

Postoji velik broj mogućnosti prilikom korištenja A/D pretvornika, kao što je podešavanje načina rada, broja kanala, vremena uzrokovanja i sl. Detalji se mogu pronaći u RM0090 Reference manual od stranice 388.

Rad s A/D pretvornikom podrazumijeva odgovarajuću konfiguraciju, pokretanje pretvorbe i dohvaćanje rezultata rezultata pretvorbe. Kao i prilikom rada s GPIO periferijom mikroupravljača, najprije je potrebno uključiti takt za A/D pretvornik (ADCCLK) postavljanjem odgovarajuće vrijednosti u RCC APB2 peripheral clock register (RCC_APB2ENR). Nadalje, postoji niz konifguracijskih registara kojima se definira način rada A/D pretvornika (npr. ADC_CR1, ADC_CR2, ...) te statusni registar (ADC_SR) koji daje informacije o trenutnom statusu A/D pretvorbe. Rezultat pretvorbe pohranjuje se u registar ADC regular data register (ADC_DR) pri čemu se u prvih 16 bitova nalazi rezultat pretvorbe. Dohvaćanje rezultata pretvorbe moguće je obaviti na tri načina: ispitivanjem statusa A/D pretvorbe prozivanjem (engl. *pooling*), preko prekidne rutine ili preko DMA (engl. *Direct Memory Access*). Ne podržavaju svi GPIO pinovi analogni način rada. U tablici 2.4 dan je popis 16 GPIO pinova koji su spojeni na odgovarajuće kanale A/D pretvornika (izdvojeno iz tablice 7 STM32F407 datasheet).

Tablica 2.4 GPIO pinovi koji podržavaju A/D pretvorbu.

GPIO pin	A/D pretvornik	Kanal
PC0	ADC1, ADC2, ADC3	10
PC1	ADC1, ADC2, ADC3	11
PC2	ADC1, ADC2, ADC3	12
PC3	ADC1, ADC2, ADC3	13
PA0	ADC1, ADC2, ADC3	0
PA1	ADC1, ADC2, ADC3	1
PA2	ADC1, ADC2, ADC3	2
PA3	ADC1, ADC2, ADC3	3
PA4	ADC1, ADC2	4
PA5	ADC1, ADC2	5
PA6	ADC1, ADC2	6
PA7	ADC1, ADC2	7
PC4	ADC1, ADC2	14
PC5	ADC1, ADC2	15
PB0	ADC1, ADC2	8
PB1	ADC1, ADC2	9

U tablici 2.5 dan je primjer konfiguracije i čitanja analogne vrijednosti na pinu PC1 pomoću SPL i to u pojedinačnom načinu rada pomoću prvog A/D pretvornika pri čemu se koristi 12-bitna rezolucija. Najprije se konfigurira GPIO pin koji se koristi kao analogni ulaz bez aktivnog internog *pull-up* ili *pull-down* otpornika. Nakon toga se aktivira takt za A/D pretvornik te se podešava A/D pretvornik pomoću `ADC_InitTypeDef` strukture i `ADC_init` funkcije. Omogućavanje rada A/D pretvornika obavlja se pomoću funkcije `ADC_Cmd`. Na kraju potrebno je povezati GPIO pin s odgovarajućim A/D pretvornikom tako da se odabere kanal na koji je spojen GPIO pin pomoću funkcije `ADC_RegularChannelConfig`. U ovom slučaju se analogna vrijednost napona na pinu PC1 pretvara u digitalnu vrijednost pomoću prvog A/D pretvornika na zahtjev. Najprije se poziva funkcija koje pokreće A/D pretvorbu te program zatim "čeka" dok se ne završi pretvroba kako bi dohvatio odgovarajuću digitalnu vrijednost. Čekanje programa se u ovom slučaju svodi na provjeravanje statusnog bita `ADC_FLAG_EOC` koji signalizira završetak A/D pretvorbe pomoću funkcije `ADC_GetFlagStatus`. Dohvaćanje rezultata pretvorbe obavlja se pomoću funkcije `ADC_GetConversionValue`.

Tablica 2.5 Podešavanje i korištenje GPIO kao analognog ulaza pomoću SPL

```
#include "stm32f4xx.h"

uint16_t result = 0;

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);  //uključivanje takta za port B

// pin PB1 se koristi u analognom načinu rada
GPIO_InitTypeDef GPIO_InitStructADC;
GPIO_InitStructADC.GPIO_Mode = GPIO_Mode_AN;
GPIO_InitStructADC.GPIO_Pin = GPIO_Pin_1;
GPIO_InitStructADC.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_Init(GPIOB, &GPIO_InitStructADC);
```

```
// omogućavanje takta za prvi A/D pretvornik
RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

// inicijalizacijska struktura za prvi A/D pretvornik
ADC_InitTypeDef ADC1_InitStruct;
// rezolucija A/D pretvorbe
ADC1_InitStruct.ADC_Resolution = ADC_Resolution_12b;
// pretvorba jednog kanala
ADC1_InitStruct.ADC_ScanConvMode = DISABLE;
// koristi se pojedinačni način rada
ADC1_InitStruct.ADC_ContinuousConvMode = DISABLE;
// ne koristi se vanjski okidač za pokretanje pretvorbe
ADC1_InitStruct.ADC_ExternalTrigConv = DISABLE;
ADC1_InitStruct.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_None;
// desno poravnanje rezultata u ADC_DR registru
ADC1_InitStruct.ADC_DataAlign = ADC_DataAlign_Right;
// broj pretvorbi
ADC1_InitStruct.ADC_NbrOfConversion = 1;

// zapisivanje postavki u konfiguracijske registre A/D pretvornika
ADC_Init(ADC1, &ADC1_InitStruct);
// pin PB1 je spojen na 9 kanal prvog A/D pretvornika
ADC_RegularChannelConfig(ADC1, ADC_Channel_9, 1, ADC_SampleTime_84Cycles);
// omogući rad prvog A/D pretvornika
ADC_Cmd(ADC1, ENABLE);

// pokreni A/D pretvorbu
ADC_SoftwareStartConv(ADC1);
// "čekaj" dok se ne završi pretvorba na prvom A/D pretvorniku
while (!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));
//dohvati rezultat pretvorbe s prvog A/D pretvornika
result = ADC_GetConversionValue(ADC1);
```

III. Priprema za vježbu

1. Ponovite operacije s bitovima. Znate li kako biste postavili odgovarajući bit nekog registra u 0 ili 1 bez utjecaja na ostale bitove registra?

2. Proučite GPIO konfiguracijske registre u RM0090 Reference manual. Koje sve registre trebate podesiti ako želite određeni pin porta D koristiti kao izlazni pin? Koje sve registre trebate podesiti ako želite određeni pin porta D koristiti kao ulazni pin? Što je potrebno učiniti ako želite aktivirati interni *pull-up* ili *pull-down* otpornik?

3. Za prethodni zadatak Pripreme za vježbu, možete li reći na kojim se adresama nalaze konfiguracijski registri?

4. Na koje pinove su spojeni LED i tipkalo na STM32F4 Discovery razvojnoj ploči?

5. Podsjetite se što je *bouncing* mehaničkog kontakta. Koje probleme uzrokuje kada se mehanički kontakti poput tipkala spajaju na mikroupravljač. Kako se takvi problemi mogu riješiti?

6. Kolika je rezolucija A/D pretvornika STM32F407VG mikroupravljača? Koliko je raspon ovog A/D pretvornika?

7. Koja je razlika između pojedinačnog i kontinuiranog načina rada A/D pretvornika?

IV. Rad na vježbi

1. Pokrenite Atollic True Studio. Potrebno je kreirati novi Projekt naziva *LV2* u kojem će se izraditi jednostavni program za STM32F4 Discovery razvojnu ploču. Odaberite *File-->New-->C project* te odaberite opciju *Embedded C Project.* Kako biste uspješno kreirali projekt potrebno je:
   a. dati odgovarajući naziv projektu (LV_2)
   b. odabrati raspoloživi hardware pod opcijom *Target* (točan naziv mikroupravljača pročitajte s raspoložive Discovery pločice)
   c. odabrati odgovarajući debug hardware (pročitajte s Discovery pločice)

Ostale postavke nije potrebno mijenjati. Pritiskom na tipku *Finish* nastat će novi projekt naziva "LV_2".

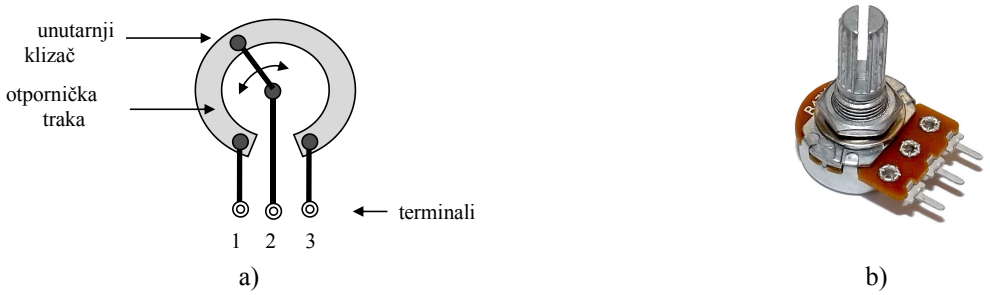
2. Napišite program koji će periodički uključivati i isključivati ugrađene LED koristeći datoteku `stm32f4xx.h`. Za potrebe definiranja vremenskog intervala između uključivanja i isključivanja LED koristite `for` ili `while` petlju u kojoj se inkrementira neka varijabla do određene vrijednosti. Prevedite program bez optimizacije (-O0) i sa stupnjem optimizacije -O1. Radi li program u oba slučaja ispravno? Ako ne radi, znate li koji je uzrok i kako riješiti problem?

3. Dopunite rješenje prethodnog zadatka tako da se pritiskom na ugrađeno tipkalo na razvojnoj ploči prekine odnosno započne periodičko uključivanje/isključivanje LED (tipkalo ima funkciju "*toggle"*). Radi li napisani program kako je zamišljeno?

4. Napišite rješenje prethodnog zadatka pomoću STM32F4xx Standard Peripheral Library.

5. Spojite pinove potenciometra na STM32F4 Discovery razvojnu ploču na način:
   Pin 1 --> 3.3V
   Pin 2 --> pin mikroupravljača koji podržava ADC
   Pin 3 --> GND

Izgled potenciometra i njegov unutrašnji prikaz da je na slici 2.3. Napišite program koji će na temelju vrijednosti napona na pinu 2 potenciometra uključivati odnosno isključivati ugrađene LED na način prikazan u tablici 2.6.



Sl. 2.3 Potenciometar a) unutrašnji prikaz, b) vanjski izgled.

Tablica 2.6 Vrijednosti napona na analognom pinu i uključene ugrađene LED.

Vrijednost napona na analognom pinu	Ugrađena LED
0% - 10% V_{REF}	-
10% - 25% V_{REF}	LD4
25% - 50% V_{REF}	LD4, LD3
50% - 75% V_{REF}	LD4, LD3, LD5
75% - 100% V_{REF}	LD4, LD3, LD5, LD6

V. Za one koji žele znati više

STMicroelectronics pruža još viši sloj apsktrakcije hardwera kroz tzv. *emebbeded software library* za određenu seriju STM32 mikroupravljača. Ova biblioteka sastoji se od: hardware abstraction layer (HAL) i Low-Layer (LL) API-a za STM32 periferiju te seta middleware komponenti (RTOS, USB, TCP/IP, Graphics, …). STM32CubeMX je grafički alat koji na temelju ove biblioteke može generirati inicijalizacijski C kod za mikroupravljač te tako znatno ubrzati postupak razvoja programske podrške za ugradbeni računalni sustav zasnovan na STM32 mikroupravljaču.