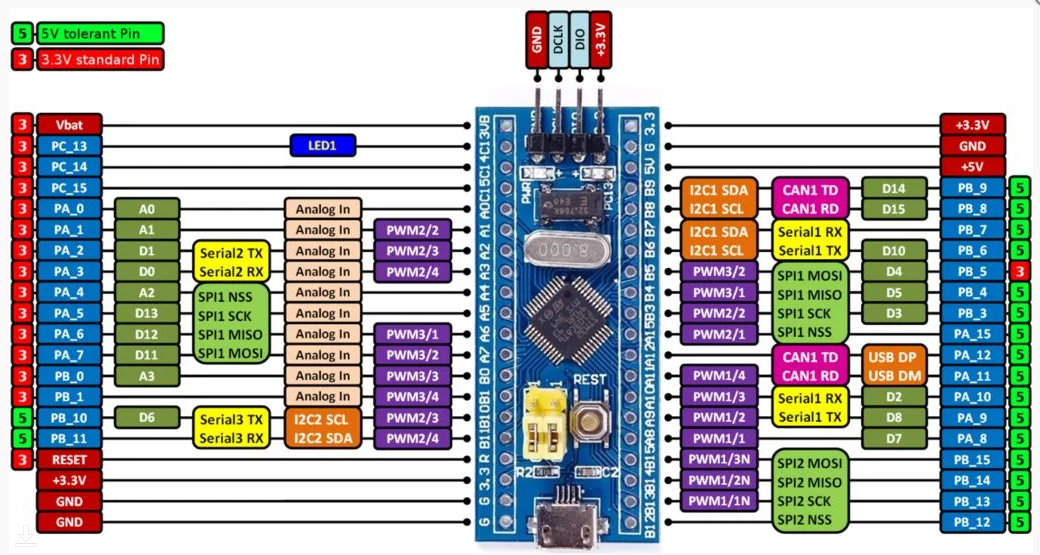
## Laboratorijska vježba 2:

## Ulazno-izlazni portovi opće namjene (GPIO)

## Ulazno-izlazni portovi opće namjene

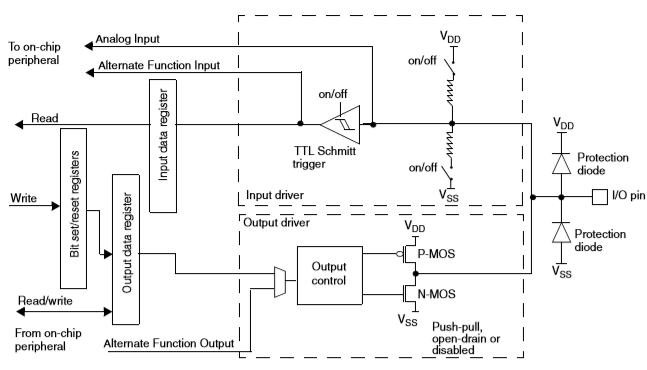
Fizička veza mikrokontrolera s okruženjem ostvaruje se spajanjem na nožice (engl. pin) mikrokontrolera. Svaka nožica mikrokontrolera može obavljati jednu ili više funkcija i te su funkcije specifične za određeni mikrokontroler. Osnovna funkcija koja se može ostvariti korištenjem jedne nožice prijenos je jednog bita podatka prema van ili prema unutra, gledano sa strane mikrokontrolera. Ukoliko je smjer prijenosa od mikrokontrolera prema van, nožicu preko koje se taj prijenos obavlja nazivamo izlaznom nožicom (engl. output pin), a ukoliko je smjer prijenosa prema mikrokontroleru nazivamo je ulaznom nožicom (engl. input pin). Obično je 8, 12, 16 ili 32 nožica grupirano u tzv. ulazno-izlazna sučelja opće namjene (engl. **G**eneral **P**urpose **I**nput-**O**utput port, **GPIO[[1]](#footnote-1)**). Time je omogućen istovremeni prijenos većeg broja bitova odjednom. Mikrokontroler STM32F103 koristi **16-bitne GPIO portove** koji se u STM32 referentnom priručniku nazivaju **GPIOA, GPIOB i GPIOC**, dok su na slici u nastavku označeni kao **PA, PB** i **PC**. Svaka nožica porta označena je indeksom, npr. PC\_13 predstavlja nožicu 13 porta C.



Prije korištenja određene nožice, potrebnu je istu konfigurirati, tj. definirati njezin način rada i namjenu. To se postiže upisom odgovarajuće vrijednosti u za to namijenjene memorijski mapirane registre. Korištenjem biblioteka taj se proces znatno pojednostavljuje, jer upotrebom struktura podataka i funkcija definiranih i implementiranih u bibliotekama nije više potrebno poznavati adrese i sadržaje konfiguracijskih registara. Ujedno je u velikoj mjeri umanjena mogućnost korištenja krivih adresa registara, ili krivih pozicija u registrima. Stoga će u nastavku biti prikazano korištenje GPIO portova putem biblioteke *Standard Peripheral Library* (SPL) koja je sastavni dio Atollic TrueSTUDIO paketa.

### 1.1 Ulazno-izlazne nožice (engl. I/O pins)

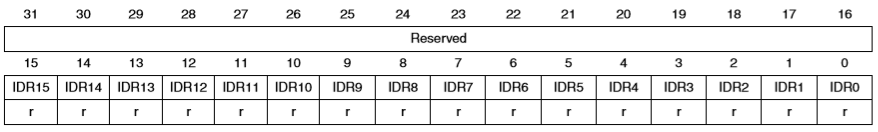
Slijedeća slika prikazuje shemu jedne nožice mikrokontrolera, tj. prikazani je sklop pridijeljen svakoj pojedinoj nožici mikrokontrolera.



Kako je u uvodnom dijelu navedeno, 16 nožica mikrokontrolera grupirano je u port. **Svakom portu pridjeljeni su konfiguracijski, ulazni, izlazni i statusni registri**. Upisom u konfiguracijski registar definira se način rada svake nožice odreženog porta. Čitanjem ulaznog registra čita se logičko stanje (‘0’ ili ‘1’) svih nožica porta. Upisom u izlazni registar, postavlja se željeno logičko stanje na izlazne nožice porta. Mikrokontoler STM32F103 koristi slijeće nazive za navedene registre:

* **IDR**( Input Data Register, ulazni podatkovni registar),
* **ODR**(Output Data Register, izlazni podatkovni registar),
* **BSRR** (Bit Set/Reset Register, bit orijentirani set/reset registar)
* **BRR** (Bit Reset Register, bit orijentirani reset registar)
* Konfiguracijske registre **CRH**, **CRL** i **LCKR** koji nisu prikazani na slici

Svaka je nožica u svakom od navedenih registara prestavljena jednim ili više bitova. Uzmimo za primjer register IDR, koji je slijedećeg sadržaja:



Kao i svi ostali registri mikrokontrolera, registar IDR je 32-bitni. 16 bitova manje težine registra označenih redom IDR0-IDR15 spojeno je s 16 ulaznih nožica dotičnog porta. Svaka nožicu predstavljena je jednim bitom, koji sadrži stanje dotične nožice (‘0’ ili ‘1’) u trenutku čitanja registra. Gornjih 16 bitova registra je neiskorišteno (označeno kao Reserved) jer su portovi 16-bitni. Detaljnije informacije mogu se potražiti u SMT32F103 referentnom priručniku, gdje su detaljno opisani svi registri.

### 1.2 Konfiguriranje i korištenje ulazno-izlaznih nožica

GPIO port modeliran je u Standard Peripheral Library (SPL) biblioteci, u datoteci stm32f10x.h, strukturom **GPIO\_TypeDef** na slijedeći način:

**typedef** **struct**

{

\_\_IO uint32\_t CRL; // Configuration Register Low

\_\_IO uint32\_t CRH; // Configuration Register Hi

\_\_IO uint32\_t IDR; // Input Data Register

\_\_IO uint32\_t ODR; // Output Data Register

\_\_IO uint32\_t BSRR; // Bit Set/Reset Register

\_\_IO uint32\_t BRR; // Bit Reset Register

\_\_IO uint32\_t LCKR; // Lock Register

} GPIO\_TypeDef;

Struktura GPIO\_TypeDef dakle sadrži sve registre GPIO porta. Nadalje, u SPL biblioteci definirane su i memorijske adese sva tri GPIO porta mikrokontrolera:

#define GPIOA ((GPIO\_TypeDef \*) 0x40010800)

#define GPIOB ((GPIO\_TypeDef \*) 0x40010C00)

#define GPIOC ((GPIO\_TypeDef \*) 0x40011000)

Ukratko, GPIOA makro definiran je kao pokazivač na strukturu tipa GPIO\_TypeDef koja se nalazi na memorijskoj adresi 0x40010800. Upravo se na toj adresi nalazi CRL registar porta A. Na slijedećoj adresi (0x40010804), koja je za 4 veća adrese CRL registra jer je riječ o 32-bitnim registrima, nalazi se registar CRH porta A i tako redom. Navedenom sintaksom jezika C omogućen je pristup pojedinačnim GPIO registrima kao članovima structure GPIO\_TypeDef.

Želimo li npr., nožice 0, 1 i 4 porta PB postaviti u stanje 1, a ostale nožice istog porta postaviti u stanje 0 upisat ćemo vrijednost 10011 (heksadecimalno 0x13) u ODR registar porta PB, na slijedeći način:

GPIOB->ODR = 0x13;

Postoji i drugi, većini korisnika jednostavniji, način upisa ili čitanja registara. U SPL-u, u datoteci stm32f10x\_gpio.h, nalaze se prototipovi funkcija kojima se to postiže. Najznačajnije su funkcije GPIO\_Write, GPIO\_WriteBit, GPIO\_ReadInputData i GPIO\_ReadInputDataBit. Prototipovi navedenih funkcija su slijedeći:

void **GPIO\_WriteBit**(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin, BitAction BitVal)

uint8\_t **GPIO\_ReadINputDataBit**(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

void **GPIO\_Write**(GPIO\_TypeDef\* GPIOx, uint16\_t PortVal)

uint16\_t **GPIO\_ReadInputData**(GPIO\_TypeDef\* GPIOx)

Prve dvije funkcije služe za upis odnosno čitanje jednog bita podatka, tj. upis na jednu nožicu, ili čitanje s jedne nožice porta. Zadnje dvije funkcije omogućuju pisanje i čitanje svih nožica određenog porta odjednom. Kako bi ih mogli korisno upotrijebiti potrebno je znati i značenje parametara korištenih u funkcijama:

* **GPIOx** = pokazivač na port. Moguće vrijednosti: GPIOA, GPIOB ili GPIOC
* **GPIO\_Pin** = označava nožicu porta. Moguće vrijednosti: GPIO\_Pin\_0 - GPIO\_Pin\_15
* **BitVal** = 1-bitna vrijednost koja se upisuje na nožicu porta. Moguće Vrijednosti: Bit\_SET (1) ili Bit\_RESET (0)
* **PortVal** = 16-bitna vrijednost koja se upisuje na port. Moguće vrijednosti:

0x0000-0xffff (65535 decimalno)

Upisa vrijednosti 0x13 na port B, možemo dakle ostvariti i na slijedeći način:

**GPIO\_Write**(GPIOB, 0x13);

Nožicu je prije njezinog korištenja u programu potrebno konfigurirati. Ukoliko se to ne učini, ona će obavljati predefiniranu (engl. default) funkciju[[2]](#footnote-2). Nožica se može postaviti u jedan od slijedećih načina rada (u zagradi je pored engleskog naziva navedena i u SPL-u definirana konstanta kojom se nožica postavlja u navedeni način rada):

* Analogni ulaz (analog input, GPIO\_Mode\_AIN)
* Digitalni ulaz s pull-up otpornikom (input pull-up, GPIO\_Mode\_IPU)
* Digitalni ulaz s pull-down otpornikom (input pull-down, GPIO\_Mode\_IPD)
* Plutajući digitalni ulaz (input floating, GPIO\_Mode\_IN\_FLOATING)
* Digitalni izlaz s obje aktivne razine (output push-pull, GPIO\_Mode\_Out\_PP)
* Digitalni izlaz s otvorenim odvodom (output open drain, GPIO\_Mode\_Out\_OD)
* Alternativna funkcija s obje aktivne razine (alternative function push-pull, GPIO\_Mode\_AF\_PP)
* Alternativna funkcija s otvorenim odvodom (alternative function open drain GPIO\_Mode\_AF\_OD)

Konfiguriranje se obavlja na način da se prvo popune članovi **GPIO\_InitTypeDef** strukture nakon čega se adresa popunjene strukture proslijedi kao parameter pri pozivu funkcije **GPIO\_Init**. Struktura GPIO\_InitTypeDef definirana je na slijedeći način:

**typedef** **struct**

{

uint16\_t GPIO\_Pin;

GPIOSpeed\_TypeDef GPIO\_Speed;

GPIOMode\_TypeDef GPIO\_Mode;

}GPIO\_InitTypeDef;

Značenje članova strukture je slijedeće:

* **GPIO\_Pin** = oznažava nožicu porta. Moguće vrijednosti: GPIO\_Pin\_0 - GPIO\_Pin\_15
* **GPIO\_Speed** = oznažava maksimalnu brzinu porasta (engl. slew rate) signala na nožici ukoliko je nožica konfigurirana kao izlazna. Nema utjecaja ukoliko je nožica konfigurirana kao ulazna. Moguće vrijednosti: GPIO\_Speed\_2MHz, GPIO\_Speed\_10MHz ili GPIO\_Speed\_50MHz.
* **GPIO\_Mode** = oznažava način korištenja nožice. Moguće vrijednosti: GPIO\_Mode\_AIN, GPIO\_Mode\_FLOATING, GPIO\_Mode\_IPD, GPIO\_Mode\_IPU, GPIO\_Mode\_Out\_OD, GPIO\_Mode\_Out\_PP, GPIO\_Mode\_AF\_OD, GPIO\_Mode\_AF\_PP

Funkcija GPIO\_Init ima slijedeću signaturu:

void **GPIO\_Init**(GPIO\_TypeDef\* GPIOx, GPIO\_InitTypeDef\* InitStruct);

Značenje parametara funkcije je slijedeće:

* **GPIOx** = pokazivač na port. Moguće vrijednosti: GPIOA, GPIOB ili GPIOC
* **InitStruct** = pokazivač na GPIO\_InitTypeDef strukturu

Ukoliko npr. želimo konfigurirati nožicu 13 porta C kao izlaznu, možemo to učiniti na slijedeći način:

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Pin = GPIO\_Pin\_13;

GPIO\_InitStruct.GPIO\_Mode = *GPIO\_Mode\_Out\_PP*;

GPIO\_InitStruct.GPIO\_Speed = *GPIO\_Speed\_2MHz*;

GPIO\_Init(GPIOC, &GPIO\_InitStruct);

### 1.3 Dovođenje clocka perifernim sklopovima

Važno je još napomenuti da se pri dizajniranju STM32 mikrokontolera vodilo računa o njegovoj energetskoj učinkovitosti, tj. o minimiziranju potrošnje električne energije, što je naročito dolazi do izražaja u baterijski napajanim sklopovima. Iz tog su razloga svi periferni sklopovi mikrokontrolera (GPIOA, GPIOB, GPIOC, ADC, UART, Timeri, … ), izuzev FLASH i RAM memorije, odspojeni od signala vremenskog vođenja (engl. clock). Perifernim sklopovima koje koristimo u programu potrebno je stoga za njihov normalan rad u programu spojiti na clock. Za tu su svrhu u SPL-u, u datoteci stm32f10x\_rcc.h definirane funkcije kojima se to ostvaruje:

**void RCC\_APB1PeriphClockCmd(uint32\_t RCC\_APB1Periph, FunctionalState NewState);**

**void RCC\_APB2PeriphClockCmd(uint32\_t RCC\_APB2Periph, FunctionalState NewState);**

Značenje parametara korištenih u funkcijama:

* **APB1Periph** = identifikator perifernog sklopa na APB1 sabirnici.

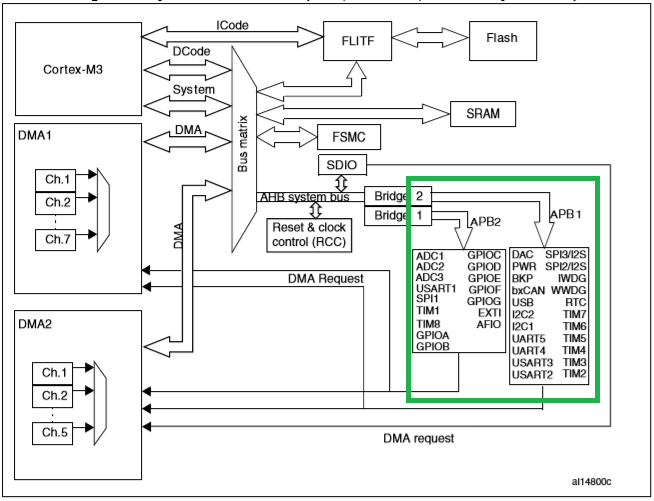
Moguće vrijednosti: RCC\_APB1Periph\_TIM2, RCC\_APB1Periph\_TIM3, RCC\_APB1Periph\_TIM4,RCC\_APB1Periph\_TIM5, RCC\_APB1Periph\_TIM6, RCC\_APB1Periph\_TIM7,RCC\_APB1Periph\_WWDG, RCC\_APB1Periph\_SPI2, RCC\_APB1Periph\_SPI3,RCC\_APB1Periph\_USART2, RCC\_APB1Periph\_USART3, RCC\_APB1Periph\_USART4, RCC\_APB1Periph\_USART5, RCC\_APB1Periph\_I2C1, RCC\_APB1Periph\_I2C2, RCC\_APB1Periph\_USB, RCC\_APB1Periph\_CAN1, RCC\_APB1Periph\_BKP, RCC\_APB1Periph\_PWR, RCC\_APB1Periph\_DAC, RCC\_APB1Periph\_CEC,RCC\_APB1Periph\_TIM12, RCC\_APB1Periph\_TIM13, RCC\_APB1Periph\_TIM14

* **APB2Periph** = identifikator perifernog sklopa na APB2 sabirnici.

Moguće vrijednosti: RCC\_APB2Periph\_AFIO, RCC\_APB2Periph\_GPIOA, RCC\_APB2Periph\_GPIOB, RCC\_APB2Periph\_GPIOC, RCC\_APB2Periph\_GPIOD, RCC\_APB2Periph\_GPIOE, RCC\_APB2Periph\_GPIOF, RCC\_APB2Periph\_GPIOG, RCC\_APB2Periph\_ADC1, RCC\_APB2Periph\_ADC2, RCC\_APB2Periph\_TIM1, RCC\_APB2Periph\_SPI1, RCC\_APB2Periph\_TIM8, RCC\_APB2Periph\_USART1, RCC\_APB2Periph\_ADC3, RCC\_APB2Periph\_TIM15, RCC\_APB2Periph\_TIM16, RCC\_APB2Periph\_TIM17, RCC\_APB2Periph\_TIM9, RCC\_APB2Periph\_TIM10, RCC\_APB2Periph\_TIM11

* **NewState** = zastavica (engl. flag) koja omogućuje ili onemogućuje dovođenje signala takta perifernom sklopu. Moguće vrijednosti: ENABLE i DISABLE

Razlog postojanja dvije funkcije je što STM32F103 mikrokontroler sadrži dvije sabirnice (engl. bus) na koje su spojeni periferni sklopovi: APB1 i APB2. Potrebno je dakle znati na kojoj od ove dvije sabirnice se nalazi određeni periferni sklop kako bismo pozvali odgovarajuću funkciju. U tome može pomoći blok shema mikrokontrolera.



Koristimo li u programu npr. periferne sklopove GPIOA i SPI2, dovođenje clocka omogućit ćemo pozivom slijedećih funkcija:

**void RCC\_APB2PeriphClockCmd(**RCC\_APB2Periph\_GPIOA**, ENABLE);**

**void RCC\_APB1PeriphClockCmd(**RCC\_APB1Periph\_SPI2**, ENABLE);**

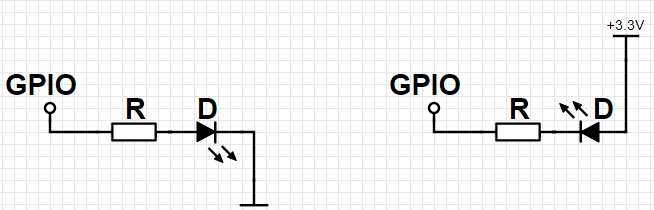
U nastavku će biti prikazan način korištenja GPIO portova kao digitalnih ulaza i izlaza, dok su drugi načini rada predmet narednih vježbi.

### 1.4 Električko povezivanje GPIO portova s vanjskim elementima

Pri spajanju mikrokontolera s vanjskim elementima, mora se voditi računa o njihovim statičkim i dinamičkim, električkim i mehaničkim svojstvima. Neke od navedenih aspekata razmotrit ćemo u nastavku, na primjeru spajanja svijetleće diode (engl. LED) i tipkala.(engl. pushbutton) na GPIO port mikrokontrolera

#### 1.4.1 Spajanje svijetleće diode na GPIO port

Svijetleću diodu ne bi bilo dobro direktno spojiti na nožicu mikrokontrolera iz slijedećeg razloga. Za pružanje svog svjetlosnog ugođaja LED zahtjeva jakost struje od približno 10 mA, a pritom je pad napona na diodi približno 1.5 V[[3]](#footnote-3). Napon na izlaznoj nožici mikrokontrolera, kad je nožica konfigurirana kao push-pull izlaz te je u stanju logičke jedinice (‘1’) iznosi 3.3V. Kada bi dakle taj iznos napona doveli direktno na anodu,diode, kroz LED bi potekla vrlo velika struja, ograničena samo malim unutrašnjim otporima izlaznog sklopa mikrokontrolera i svijetleće diode, što bi rezultiralo pregaranjem diode ili izlaznog sklopa mikrokontrolera. Kako se to ne bi dogodilo potrebno je oganičiti iznos struje na navedenih 10 mA. To se postiže spajanjem otpornika u seriju sa svijetlećom diodom na jedan od slijedećih načina:



Ukoliko LED spajamo na prvi način (lijevo), ona će svijetliti kad se na GPIO port upiše logička jedinica (‘1’), dok će pri spajaju prema slici desno LED svijetliti kad se na GPIO port upiše ‘0’. Kod mikrokontrolera koji su prethodili SMT32 nerijetko su GPIO portovi mogli podnijeti veću jakost struje u slučaju kada struja ulazi u port (engl. sink currect, slika desno) nego kad struja iz porta izlazi (engl. source current, slika lijevo) pa se češće koristio sklop na slici desno. Kod STM32 mikrokontrolera nema razlike u ograničenju s obzirom na smjer struje, jer je maksimalna dozvoljena jakost struje 25 mA u oba smjera.

Vrijednost otpornika R određuje se pomoću Ohmovog zakona:

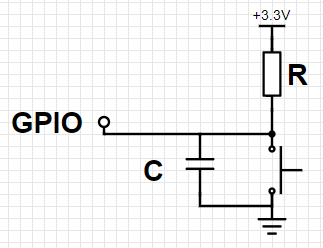
**R = (Vcc – Ud) / Id**

Uvrstimo li vrijednosti struja i napona, dobit ćemo vrijednost R = 180 Ohma (3.3 - 1.5) V / 0.01A. Dakle spajanjem diodi u seriju otpornika 180 Ohma, diodom će, kada je propusno polarizirana, teći struja jakosti 10 mA.

#### 1.4.2 Spajanje tipkala na GPIO port

Pri spajanju tipkala na GPIO port mora se voditi računa o tzv. istitravanju kontaka (engl. contact bouncing). Radi se o neželjnoj prijelaznoj nuspojavi koja se manifestira višestrukim uzastopnim spajanjem i odspajanjem kontakataa, a javlja se pri svakom pritisku ili otpuštanju tipkala. Navedena nuspojava, ukoliko ne bi bila na neki način umanjena ili uklonjena, mogla bi program navesti na pogrešan zaključak da je korisnik više puta pritisnuo tipkalo i ovisno o tome program bi mogao izvšiti neku drugu funkciju, ili bi istu funkciju izvršio više puta.

Nuspojava se može riješiti programskim putem na način da se više puta uzastopce pročita stanje tipkala i tek kada se stanje stabiliziralo, a to se utvrđuje tako što je više puta uzastopno pročitana ista vrijednost, vrijednost se smatra stvarnim stanjem. U praksi se koristi i sklopovsko filtriranje električkog istitravanja na način da se, u najjednostavijoj varijanti, paralelno tipkalu spoji kondenzator, Bez ulaženja u dublju analizu dinamike prijelazne pojave, najčešće se koristi keramički kondenzator kapaciteta 100 nF, koji su kombinaciji s pull-up ili pull-down otpornikom (10 – 100 kOhma) čini niskopropusni filtar vremenske konstante reda desetak milisekundi.



### 1.5 Primjeri i zadatci

#### 1.5.1 Primjer 1: GPIO digitalni ulaz i izlaz

Sastavi sklop i napiši program koji će stanje tipkala spojenog na port A, nožica 0 (GPIOA, GPIO\_Pin\_0) prikazati svijetlećom diodom spojenom na port B, na nožicu 1 (GPIOB, GPIO\_Pin\_1). Kada je tipkalo pritisnuto LED treba svijetliti, a u suprotnom LED treba biti ugašen. Nakon što je sklop spojen, prepiši, prevedi i izvrši program u nastavku te provjeri rad sklopa.

**#include** <stddef.h>

**#include** "stm32f10x.h"

**#include** "stm32f10x\_rcc.h"

**#include** "stm32f10x\_gpio.h"

// prototipovi funkcija za inicijalizaciju

**void** **initTipkalo**(**void**);

**void** **initLED**(**void**);

**int** **main**() {

// varijabla za pohranu stanja tipkala

uint8\_t stanje;

// inicijalizacija ulaznog i izlaznog porta

initTipkalo();

initLED();

// citanje stanja tipkala i azuriranje stanja LED

**while**(1) {

stanje = GPIO\_ReadInputDataBit(GPIOA, GPIO\_Pin\_0);

GPIO\_WriteBit(GPIOB, GPIO\_Pin\_1, stanje);

}

}

// funkcija za inicijalizaciju ulaznog porta na koji je spojeno tipkalo

**void** **initTipkalo**(**void**) {

GPIO\_InitTypeDef gpioInitStruct;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, *ENABLE*);

GPIO\_StructInit(&gpioInitStruct);

gpioInitStruct.GPIO\_Pin = GPIO\_Pin\_0;

// tipkalo je tako spojeno da se na ulazu cita ‘1’ kad nije stisnuto

gpioInitStruct.GPIO\_Mode = *GPIO\_Mode\_IPU*;

GPIO\_Init(GPIOA, &gpioInitStruct);

}

// funkcija za inicijalizaciju izlaznog porta na koji je spojena LED

**void** **initLED**(**void**) {

GPIO\_InitTypeDef gpioInitStruct;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, *ENABLE*);

GPIO\_StructInit(&gpioInitStruct);

gpioInitStruct.GPIO\_Pin = GPIO\_Pin\_1;

gpioInitStruct.GPIO\_Mode = *GPIO\_Mode\_Out\_PP*;

gpioInitStruct.GPIO\_Speed = *GPIO\_Speed\_2MHz*;

// pocetno stanje na izlazu je 1 (LED je spojena tako da tada ne svijetli)

GPIO\_WriteBit(GPIOB, GPIO\_Pin\_1, *Bit\_SET*);

GPIO\_Init(GPIOB, &gpioInitStruct);

}

#### 1.5.2 Zadatak 1: GPIO digitalni izlaz i pauza

Sastavi sklop i napiši program koji će svakih pola sekunde mijenjati stanje na svijetlećoj diodi spojenoj na port B, nožica 1. Napomena: za pauzu u trajanju od pola sekunde implementiraj funkciju void pauza(int brojac) koja neće raditi ništa korisno već samo trošiti procesorsko vrijeme. Eksperimentirati s vrijednošću argumenta brojac dok se ne postigne željeno trajanje pauze.

1. U nastavku ću umjesto naziva *sučelje* koristiti naziv *GPIO port*, ili samo *port*, jer se znatno češće koriste, barem zasada. [↑](#footnote-ref-1)
2. Svaka nožica ima svoju predefiniranu funkciju kojom započinje rad nakon dovođenja napajanja ili reseta. [↑](#footnote-ref-2)
3. Jakost struje kroz svijetleću diodu i pad napona ovisni su vrsti diode. [↑](#footnote-ref-3)