

JU MJEŠOVITA SREDNJA ELEKTROTEHNIČKA ŠKOLSKA GODINA

ŠKOLA TUZLA 2020./2021.

MATURSKI RAD

*Predmet:* Mikroracunari

*Tema:* Timer

MENTOR: UČENIK:

Muamer , dipl. ing. el.teh. Amel Smajic

Razred: IVT1

TUZLA, maj, 2021.god.

Contents

[1. Uvod 3](#_Toc72346460)

[2. Timeri 4](#_Toc72346461)

[2.1.Elektronski ( Digitalni) timeri 4](#_Toc72346462)

[3.0 STM32D103C8 (Mikrokontroler) 5](#_Toc72346463)

[3.1 Detaljni opis (Tablica) 6](#_Toc72346464)

[3.2 ARM® Cortex ™ -M3 jezgra s ugrađenim Flashom i SRAM-om 8](#_Toc72346465)

[3.3 CRC (ciklična provjera redundancije) jedinica za izračunavanje 8](#_Toc72346466)

[3.4 Ugnježđeni vektorski kontroler prekida (NVIC) 8](#_Toc72346467)

[3.5 Vanjski kontroler prekida / događaja (EXTI) 9](#_Toc72346468)

[3.6 Satovi i pokretanje 9](#_Toc72346469)

[3.7 Boot modusi 9](#_Toc72346470)

[3. 8 Šeme napajanja 10](#_Toc72346471)

[3.9 Režimi male snage 10](#_Toc72346472)

[3.10 RTC (sat u stvarnom vremenu) i rezervni registri 11](#_Toc72346473)

[4.0 Pinout Mikrokontrolera STM32F103C8 12](#_Toc72346474)

[4. Zaključak 13](#_Toc72346475)

[5. Literatura 14](#_Toc72346476)

# 1. Uvod

Dobar dan aj sam amel smajic i u sljedecem prilogu cete vidjeti moj maturski rad.

Odabrao sam da odradim maturski iz mikroracunara sa temom : Timer , jer mi se cinlo jako interesantnom i pogodnom prilikom da naucim nesto novo, i izvanrednu priliku da iskusim rad sa 3D printerom . Za maturski rad koristimo 3 glavne elektricne komponente ty : Standardno „Tipkalo “, prog . Oled(64x32) ekran te naravno „STM32f103C8“ mikrokontroler Za izradu programa koristili smo gotovu biblioteku za pisanje na datom ekranu , licno sam kreirao biblioteku za „Tipkalo “ kao i ostatak program. Detalje o datim komponetama cu priloziti u ostatku dokumenta.

Pored 3 glavne elektronske komponente imamo i jednako bitno 3d kuciste koje sam kreirao u Onshape (Beplatnom online 3D editoru) te otprintao na skolskom 3D printeru .

Nakon vise poteskoca sa podesavanjem mikrokontrolera i razumjevanjem nacina funkcionisanja matador ploce (Sto sam trebo davno nauciti) uspjeo sam sve spojiti i odraditi kod , Nakon nekoliko pokusaja uspjeo sam da sredim i kuciste , te se ono u ovome trenutku nalazi ispred vas.

Tipkalo koje se sadrzi od 4 tugmeta ima sljedece komande

Start setin timer

Dodaj vrijeme (1min)

Oduzmi vrijeme (1min)

Pocni odbrojavanje

To bi terebalo da je dovoljno za uvod , idemo na glavni dio.2. Timeri

Timer je specijalizirani tip sata koji se koristi za mjerenje određenim vremenskim intervalima. Tajmeri se mogu podijeliti u dvije glavne vrste. Tajmer koji broji prema gore od nule za mjerenje proteklog vremena često se naziva štopericom , dok se uređaj koji odbrojava od određenog vremenskog intervala , prema nuli obično naziva tajmerom. Najjednostavnij primjer ove vrste je pješčani sat . Tajmeri radnih metoda imaju dvije glavne grupe: hardverski i softverski tajmeri.

Većina tajmera daje naznaku da je vremenski interval koji je postavljen istekao.

U nasem slucaju ta naznaka je blinkanje ekrana i moze se prekinuti pritiskom na bilo koje dugme tipkala.

Timere inace dijelimo i na Hardverske (mehanicke) , Elektromehaničke i Elektricke timere , buduci da smo u elektrotehnickoj skoli jednostavno je zakljuciti da se govori o elektrickom timeru.

## 2.1.Elektronski ( Digitalni) timeri

Digitalni tajmeri prate vrijeme; za pokretanje radnje, za pokretanje mjerenja vremena pokrenute radnjom ili za oboje. Neki proizvodi se mogu programirati, dok drugi mogu biti fiksirani u određeno interno vrijeme i funkciju. Pored broja i vrste funkcija, ovi se uređaji razlikuju u pogledu postavki vremenskog raspona.

# 3.0 STM32D103C8 (Mikrokontroler)

STM32 je porodica 32-bitni mikrokontroler integriranih krugova po STMicroelectronics . Čipovi STM32 grupirani su u povezane serije koje se temelje na istoj 32-bitnoj ARM procesorskoj jezgri, poput Cortex-M33F , Cortex-M7F , Cortex-M4F , Cortex-M3 , Cortex-M0 + ili Cortex-M0 . Interno se svaki mikrokontroler sastoji od jezgre procesora, statičke RAM-a , flash memorije, interfejsa za otklanjanje grešaka i raznih perifernih uređaja.

Dizajn jezgre ARM ima brojne mogućnosti koje se mogu konfigurirati, a ST odabira pojedinačnu konfiguraciju za svaki dizajn. ST pričvršćuje vlastitu perifernu opremu na jezgru prije nego što dizajn pretvori u silicijsku matricu. Sljedeće tablice rezimiraju porodice mikrokontrolera STM32.

Serija STM32 F1 bila je prva grupa STM32 mikrokontrolera zasnovana na ARM Cortex-M3 jezgri i smatrala je svojim glavnim ARM mikrokontrolerima. F1-serija se vremenom razvijala povećavanjem brzine procesora, veličinom interne memorije, raznovrsnim perifernim uređajima. Postoji pet F1 linija: povezivost (STM32F105 / 107), performanse (STM32F103), USB pristup (STM32F102), pristup (STM32F101), vrijednost (STM32F100).

Iz te grupacije mi koristimo mikrokontroler pod specificnim anzivom STM32F103C8 koji se razlikuje od ostalih mikrokontroleraa ,ali razlika je prilicno mala , dovoljno mala da bi se isti kod mogao koristiti za skoro sve STM32 mikrokontrolere sa minimalnim modifikacijama.

Najvaznije podatke iz DATASHEATA za ovaj mikrokontroler cu izdvojiti u ovaj dokument , ali prilazem i cijeli datasheat uz dokumentaciju kako bi bio siguran.

Mikrokontroler se nalazi na 45 x 18 mm koji je prilicno slican plocici arduino nano.

Velicina samog mikrokontrolera varira od 6x6 - 14x14 te je prilicno kompaktna i moze se koristiti u digitalnim prirucnim satovime itd.

Radni napon mikrokontrolera je 2.0-3.6 V, radni napon je od -40 i +85 oC , preko kompjutera komuniocira sa USB 2.0 – Full speed interfacom . Sadrzi flas memoriju od 64 – 128 kb 3 timera sa generalnom upotrebom (koji se nemogu kontrolirati) i 1 timer sa advencet conroler interfacom (Koje mozemo kontrolirati regularnim kodom). Sadrzi ukupno 37 brzih I/O pinova , ali u ovome nano izvodu mozemo da koristimo samo 33 ok kojih 15 iz GPIOA , 15 GPIOB i 3 iz GPIOC .

3.1 Detaljni opis (Tablica)

|  |  |  |  |
| --- | --- | --- | --- |
| Periferije | | Stm32F103C8 | |
| Flash - Kbytes | | 64 | 128 |
| SRAM - Kbytes | | 20 | |
| Timeri | General-purpose | 3 | |
| Advanced-control | 1 | |
| Kommunikacija | SPI | 2 | |
| I 2 C | 2 | |
| USART | 3 | |
| USB | 1 | |
| CAN | 1 | |
| GPIOs | | 37 | |
| 12-bit synchronized ADC Number of channels | | 2  10 kanala | |
| CPU frequency | | 72 MHz | |
| Operating voltage | | 2.0 to 3.6 V | |
| Operating temperatures | | –40 to +85 °C /  –40 to +105 °C | |
| Packages | | LQFP48,  UFQFPN48 | |

Porodica performansi srednje gustine STM32F103xx uključuje visoko efikasnu ARM Cortex ™ -M3 32-bitnu RISC jezgru koja radi na frekvenciji od 72 MHz, ugrađene memorije velike brzine (Flash memorija do 128 Kbytes i SRAM do 20 Kbytes),

i širok raspon poboljšanih U / I i perifernih uređaja povezanih s dvije APB magistrale. Sve

uređaji nude dva 12-bitna ADC-a, tri 16-bitna tajmera opće namjene i jedan PWM tajmer, kao

kao i standardni i napredni komunikacijski interfejsi: do dva I2C i SPI, tri

USART-ovi, USB i CAN.

Uređaji rade od 2,0 do 3,6 V napajanja. Dostupni su u modelima od –40 do

Temperaturni opseg od +85 ° C i prošireni opseg temperature od –40 do +105 ° C. A

sveobuhvatan skup režima za uštedu energije omogućava dizajn aplikacija male snage. Porodica performansi srednje gustine STM32F103xx uključuje uređaje u šest različitih tipovi paketa: od 36 do 100 pinova.

Ovisno o odabranom uređaju, različiti setovi Tablica 1.0 Statisticki opis mikrokontrolera

periferne uređaje su uključene, opis u nastavku daje pregled cijelog asortimana

periferne uređaje predložene u ovoj porodici.

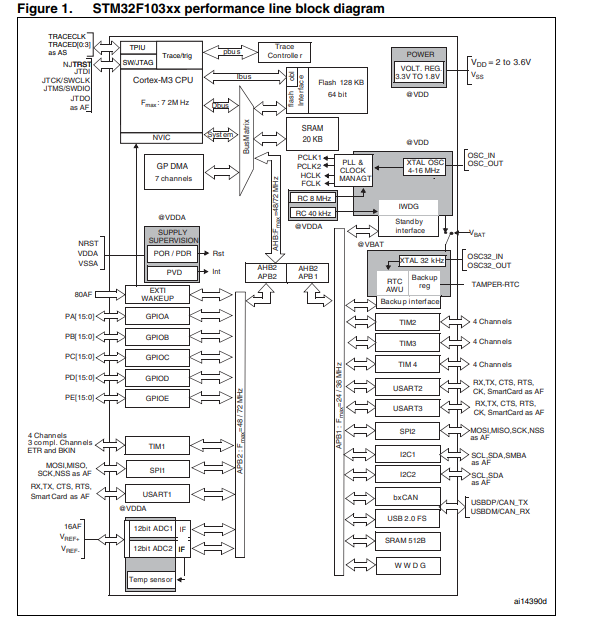
Ove karakteristike čine STM32F103xx linijski mikrokontroler performansi srednje gustine

porodica pogodna za širok spektar aplikacija kao što su motorni pogoni, upravljanje aplikacijama,

medicinska i ručna oprema, PC i igraće periferne jedinice, GPS platforme, industrijske

aplikacije, PLC-ovi, pretvarači, pisači, skeneri, alarmni sistemi, video portafoni i

HVAC

****

## 3.2 ARM® Cortex ™ -M3 jezgra s ugrađenim Flashom i SRAM-om

ARM Cortex ™ -M3 procesor je najnovija generacija ARM procesora za ugrađene sisteme. Razvijen je kako bi pružio jeftinu platformu koja zadovoljava potrebe implementacije MCU-a, sa smanjenim brojem pinova i niskom potrošnjom energije, istovremeno pružajući izvanredne računske performanse i napredni sistemski odgovor na prekide. 32-bitni RISC procesor ARM Cortex ™ -M3 odlikuje se izuzetnom efikasnošću koda, pružajući visoke performanse koje se očekuju od ARM jezgre u veličini memorije koja je obično povezana sa 8 i 16-bitnim uređajima. Obitelj linija performansi STM32F103xx koja ima ugrađenu ARM jezgru, stoga je kompatibilna sa svim ARM alatima i softverom. Slika 1 prikazuje opći blok dijagram porodice uređaja

## 3.3 CRC (ciklična provjera redundancije) jedinica za izračunavanje

Jedinica za izračunavanje CRC (ciklična provjera redundancije) koristi se za dobivanje CRC koda iz 32-bita

riječ podataka i fiksni polinom generatora.

Između ostalih aplikacija, tehnike zasnovane na CRC-u koriste se za provjeru prijenosa podataka ili

integritet skladišta. U opsegu standarda EN / IEC 60335-1, nude sredstva za

provjera integriteta Flash memorije. CRC jedinica za izračunavanje pomaže u izračunavanju potpisa

softver za vrijeme izvođenja, koji se uspoređuje s referentnim potpisom generiranim u vrijeme veze i pohranjenim na datoj memorijskoj lokaciji.

## 3.4 Ugnježđeni vektorski kontroler prekida (NVIC)

Linija performansi STM32F103xx ugrađuje ugniježđeni vektorski kontroler prekida u mogućnosti

obrađuju do 43 maskirana kanala prekida (ne uključujući 16 linija prekida od

Cortex ™ -M3) i 16 nivoa prioriteta.

● Usko povezan NVIC daje obradu prekida sa malim kašnjenjem

● Adresa tabele vektora unosa prekida proslijeđena je izravno u jezgru

● Usko povezan NVIC interfejs jezgre

● Omogućuje ranu obradu prekida

● Obrada prekida višeg prioriteta koji kasne u dolasku

● Podrška za ulančavanje repa

● Stanje procesora se automatski sprema

● Unos prekida vraćen na prekidnom izlazu bez režijskih uputstava

## 3.5 Vanjski kontroler prekida / događaja (EXTI)

Vanjski kontroler prekida / događaja sastoji se od 19 linija detektora ivica koje se koriste za generiranje

zahtjevi za prekid / događaj. Svaka linija se može samostalno konfigurirati za odabir okidača

događaja (rastuća ivica, padajuća ivica, oboje) i mogu se maskirati nezavisno. Registar na čekanju

održava status zahtjeva za prekidom. EXTI može prepoznati vanjsku liniju pomoću a

širina impulsa kraća od internog takta APB2. Može se povezati do 80 GPIO-a

na 16 vanjskih linija prekida

## 3.6 Satovi i pokretanje

Izbor sistemskog sata vrši se prilikom pokretanja, međutim interni je RC 8 MHz oscilator

odabran kao zadani CPU sat pri resetovanju. Može se odabrati vanjski sat od 4-16 MHz, u

u kom slučaju se nadgleda zbog neuspjeha. Ako se otkrije kvar, sistem se automatski prebacuje

natrag na interni RC oscilator. Softverski prekid se generira ako je omogućen. Slično tome, pun

upravljanje prekidima unosa PLL sata je dostupno po potrebi (na primjer na

kvar indirektno korištenog vanjskog kristala, rezonatora ili oscilatora).

Nekoliko prescalera omogućava konfiguraciju AHB frekvencije, APB velike brzine

(APB2) i domene male brzine APB (APB1). Maksimalna frekvencija AHB i

APB domeni velike brzine su 72 MHz. Maksimalno dozvoljena frekvencija male brzine

APB domena je 36 MHz. Pogledajte sliku 2 za detalje o stablu sata.

## 3.7 Boot modusi

Pri pokretanju, boot pin-ovi koriste se za odabir jedne od tri opcije pokretanja:

● Dizanje iz korisničkog blica

● Dizanje iz sistemske memorije

● Dizanje sa ugrađenog SRAM-a

Učitavač pokretača nalazi se u sistemskoj memoriji. Koristi se za reprogramiranje Flash memorije pomoću

koristeći USART1. Za dodatne detalje pogledajte AN2606.

## 3. 8 Šeme napajanja

● VDD = 2,0 do 3,6 V: vanjsko napajanje za I / O i unutarnji regulator.

Isporučuje se izvana kroz VDD pinove.

● VSSA, VDDA = 2,0 do 3,6 V: vanjska analogna napajanja za ADC, reset blokovi, RC

i PLL (minimalni napon koji se mora primijeniti na VDDA je 2,4 V kada se koristi ADC).

VDDA i VSSA moraju biti povezani na VDD i VSS.

● VBAT = 1,8 do 3,6 V: napajanje za RTC, oscilator eksternog takta 32 kHz i sigurnosna kopija

registrira (preko prekidača za napajanje) kada VDD nije prisutan.

Za više detalja o povezivanju pinova za napajanje pogledajte sliku 14: Šema napajanja

## 3.9 Režimi male snage

Linija performansi STM32F103xx podržava tri načina male snage kako bi se postiglo najbolje

kompromis između male potrošnje energije, kratkog vremena pokretanja i dostupnog buđenja

izvori:

● Režim spavanja

U stanju mirovanja zaustavljen je samo CPU. Sve periferne jedinice i dalje rade i mogu

probudite CPU kada dođe do prekida / događaja.

● Stop način rada

Stop način rada postiže najmanju potrošnju energije zadržavajući sadržaj

SRAM i registri. Svi satovi u domeni od 1,8 V su zaustavljeni, PLL, HSI RC

a HSE kristalni oscilatori su onemogućeni. Može se staviti i regulator napona

bilo u normalnom ili u režimu male snage.

Uređaj se može probuditi iz stop načina rada bilo kojom linijom EXTI. Linija EXTI

izvor može biti jedna od 16 vanjskih linija, PVD izlaz, RTC alarm ili USB

probudi se.

● Stanje pripravnosti

Režim mirovanja koristi se za postizanje najniže potrošnje energije. Unutrašnji

regulator napona je isključen tako da je isključena cijela domena od 1,8 V. The

PLL, HSI RC i HSE kristalni oscilatori su takođe isključeni. Nakon ulaska

U stanju pripravnosti, SRAM i sadržaj registra se gube, osim za registre u sigurnosnoj kopiji

domena i standby sklopovi.

Uređaj izlazi iz stanja pripravnosti kada se eksterno resetiranje (NRST pin), IWDG resetiranje, a

rastuća ivica na WKUP pinu ili se javlja RTC alarm.

## 3.10 RTC (sat u stvarnom vremenu) i rezervni registri

RTC i rezervni registri isporučuju se preko prekidača koji uključuje napajanje

VDD napajanje kada je prisutan ili putem VBAT pina. Registri sigurnosnih kopija su deset 16-bitnih

registri koji se koriste za spremanje 20 bajtova podataka korisničke aplikacije kada VDD napajanje nije prisutno.

Sat u stvarnom vremenu nudi set neprekidno brojača koji se mogu koristiti

odgovarajući softver koji pruža funkciju kalendara sata, a nudi i prekid alarma i a

periodični prekid. Takt mu je vanjski kristal, rezonator ili oscilator od 32,768 kHz,

interni RC oscilator male snage ili eksterni sat velike brzine podijeljen sa 128

interni RC male snage ima tipičnu frekvenciju od 40 kHz. RTC se može kalibrirati pomoću

vanjski izlaz od 512 Hz za kompenzaciju svakog prirodnog odstupanja kristala. RTC karakteristike

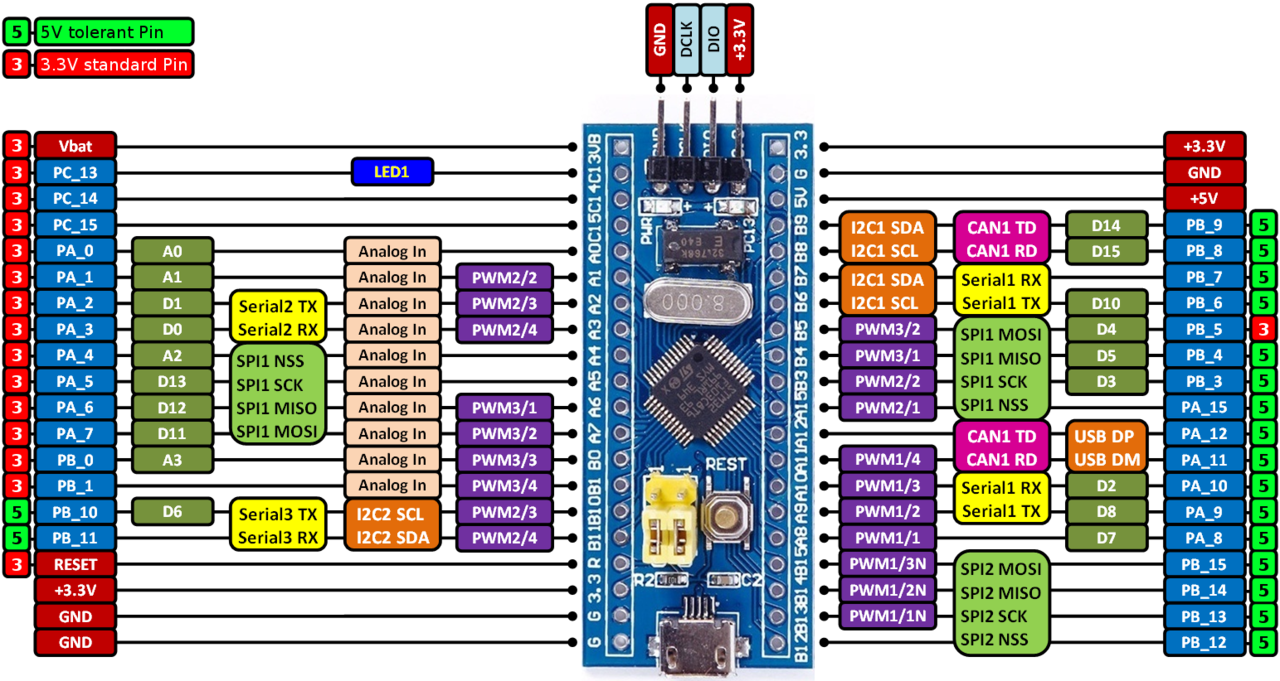
32-bitni programabilni brojač za dugoročno mjerenje pomoću registra Compare to

generirati alarm. 20-bitni prescaler koristi se za sat vremenske baze i prema zadanim je postavkama

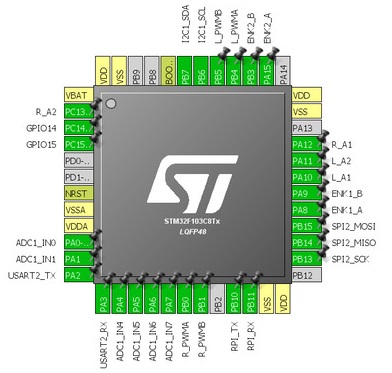
konfiguriran da generira vremensku bazu od 1 sekunde od takta na 32.768 kHz.

Tablice moraju biti označene brojevima, a iznad tablice mora doći tekst koji opisuje šta je u tablici prikazano, npr.,

# 3.11 Pinout Mikrokontrolera STM32F103C8



Pinout mikrokonteolera STM32F103C8

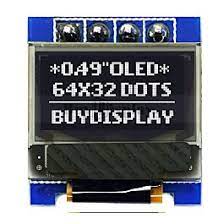


Na samom mikrokontroleru (Bez razvojnog okruzenja) nalazi se 48 pinova , 37 I/O pinova , 15 GPIOA .15 GPIOB ,7 GPIOC. Pinovi A1-A7 i B0, B1 se mogu koristit za analogni input. Nalaze se 3 VDD pina , 3 VSS pina, Vbat ,VSSA VDDA , VBAT , PD0 ,1 ,PB2, PB8, 9,12,13 , PA 14.

Ali vecina tih pinova na rayvojnom okruyenju je „Nedostupno“ , tako da cemo se fokusirati samo na one koji su nam dostupni , te samim time i bitni.

Prva stvar koja se moze primjetiti jeste da su svi pinovi iz GPIOA i GPIOB grupacije dostupni sa par pinova iz GPIOC IMAMO 3 Ground pina 2 3.3V naponska pina i 1 5v –tni naponski pin , 1 Vbat pin te reset pin.

# 4.0 OLED Display Module 64x32

Maleni grafički OLED od 0,49 × 64 × 32 OLED-a tvrtke PMD Way . To je kompaktan i koristan zaslon koji zahtijeva samo malo vremena da biste započeli rad s vašim Arduinom ili kompatibilnim mikrokontrolerom.

OLED je skraćenica od Organic Light Emitting Diode.

Ako pojednostavnimo terminologiju, recicemo da se radi o organskim spojevima koji svijetle ako kroz njih pustimo struju.

Za razliku od LED-a koji više-manje radi istu stvar, samo što koristi drukčiju tehnologiju, OLED je savitljiv, manji i tanji. Ovo znači da OLED rješava problem s LED-om, odnosno njegovom veličinom. Štoviše, OLED je toliko mali da se može koristiti za pojedine piksele te, što je najvažnije, crnu boju reproducira tako da doslovno prestane osvjetljavati pojedini piksel. Odatle, konkretno, dolazi i OLED-ova najpoznatija prednost. Naime jedan je od rijetkih digitalnih ekrana koji dostojanstveno pretstavlja crnu boju.

Korejski proizvođač LG trenutno je cijenjen po proizvodnji OLED TV panela. Sony i LG imaju dogovor koji Sonyju dozvoljava da LG-jeve OLED panele ugrađuje u svoje televizore.

Buduci da je OLED tehnologija jos uvjek u odredzenoj fazi razvoja cijena samih televizora je prilicno visoka. Ali ocigledna i najbolja primjena OLED tehnologije zapravo i nije TV industrija vec industrija pametnih tlefona , satova , radija i ostalih manjih proizvoda .

Stanje je takvo radi cinjenice da se sa OLED tehnologijom moze dobiti veca rezolucija cak i na manjim ekranima kao i debljina i savitljivost samih ekrava . OLED tehnologija je jedno od mogucih rijesenja za kreiranje fleksibilnih pametnih telefona.

## 4.1 Pinout OLED 64x32 Displeya

GND ---- GND (GND)  
5V/3.3V- Vcc (power supply, can be 3.3V or 5V)  
B6 ----- SCL (I2C bus clock)  
B7 ----- SDA (I2C bus data)

U primjeru iznad imamo nazive pinova displeya kao I njihove parove na mikrokontroleru.

Prva dva pina sluze za napajanje (VCC,GND) a sljedeca dva za komunikaciju sa kontrolerom.

Radi cinjenice da imamo veci ekran sa matricom komunikacija nije digitalna vec serijska.

Sam ekran se prodaje po cijeni od 3.5$ , dok je moguce dobiti popust u slucaju kupovine na vise.

# 5.0 Tuch – Pad (Tipkalo)

Interakcija korisnika sa danasnjim uredzajima se uglavnom svodi na dodir npr : pametni telefoni , satovi , tableti ,novij laptopi I notepadovi …

Tehnologij za izradu povrsina osjetljivih na dodir varira , ali u nasem slucaju koristimo kapacitivni senzor dodira.

Kapacitivni senzor dodira koji mi koristimo u ovakvoj postavci ima ulogu push-buttona (make) sa time da je pouzdani I pri kreiranju veze nemamo prelazno oscilatorno stanje tipkalo sa kojim radimo se sastoji od 4 serijska tipkala sa pojedinacnim data izlazima .

## 5.1 Raspored pinova Tuch - Pada

**Controler Tipkalo**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

VCC(3.3) 🡪 VCC

GND 🡪 GND

A0 🡪 OUT4 (Paus / set timer)

A1 🡪 OUT3 (Inkrement /++)

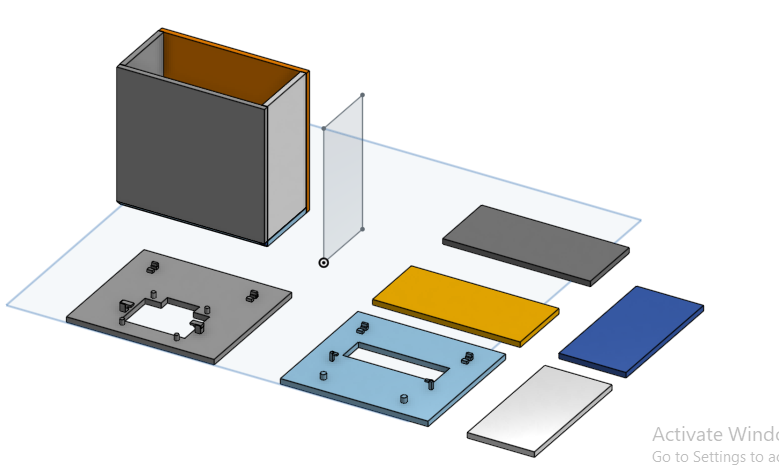
A2 🡪 OUT2(Dekremnt / - -)

A3 🡪 OUT1(Nastavi / zapocni odbrojavanje)

Iznad imamo prikazan redoslijed pinova na tipkalu sa desne strane I njihove odabrane parove na mikrokontroleru sa desne strane. Nacin rada samog tipkala je prilicno jednostavan , naime kada dodirnemo povrsinu koja je planirana za koristenje kao prekidac drasticno smanjimo otpornost vodova I dopustimo proticanje elektricne energije kroz isti , zatim mikrokontroler detektuje promjenu napona na digitalnom ulazu I uz pomoc koda promjeni stanje na ekranu (poveca ili smanji vrijeme) . Na slici je tipkalo spojeno na matador (radi jednostavnijeg razvijanja koda).

# on.PNG6.0 3D Kuciste (Onshape)

Onshape – je besplatni online 3D editor koji se koristi za kreiranje i editovanje razlicitih 3D modela. Najpopularnij je radi cinjenice da sluzi kao oblak za 3D kreacije i u slucaju podjeljenih i zajednickih dokumenata svi koji imaju pristup istome mogu u stvarnom vremenu vidjeti promjenu koju je njihov kolega ucinio na bilo kojem drugo kutku zemlje . U nasem slucaju koristen je za kreiranj 3D timerkog kucista . Potrudio sam se da dizajn bude sto jednostavnij i kompaktnij , ali i prakticnij. Samo kuciste iskoristili smo kao sretstvo za omogucavanje implementacije wire veza za nas projekat (Radi cinjenice da bi narudzivanje custom plocice zahtjevalo dodatn ovrijeme i financiski trosak) te smo uspjeli da isto kuciste otprintamo i sastavimo.



# efb41dec0aabd42ea1ce2a35416f181e.png7.0 Opis koda

Kod za ovaj program smo odradili u razvojnom okruzenju keill µ5 . To je program koji se koristi za embedet deweloping prilicno je mocan i cak i sa besplatnom verzijom se mnogo toga moze odradit. Jednostavan je za nauciti i postoji dosta pomocnih programa koji mogu pomoci oko kreiranja osnovnog koda (Baze koda) .

Sto se tice naseg koda konkretno , koristili smo gotovu biblioteku za OLED ekran , ali cemo osnovne dijelove te biblioteke opisati naravno , biblioteku za tipkalo kao i glavni kod sam sam kreirao te cu iz tih biblioteka takodjer objasniti najvaznije djelove u sljedecem dijelu tokumenta .

## 7.1 Okis biblioteke za tipkalo

Kod ove biblioteke imali smo 3 vazne naredbe koje smo koristili za ovaj program a to su :

1. **GPIO\_ClockCtrl(GPIOA,ENABLE);**
2. **GPIO\_PinMode(GPIOA,x,INPUT,FAST); (**x=0,1,2,3/ 4 linije za definiciju)
3. **A.x=GPIO\_DigitalRead(GPIOA,x);** (x=0,1,2,3/ 4 linije za citanje stanja)

1. PIO\_ClockCtrl(GPIO\_TypeDef \*pGPIOx, uint8\_t EnOrDi){

if(EnOrDi==ENABLE){

if(pGPIOx==GPIOA){RCC->APB2ENR |= RCC\_APB2ENR\_IOPAEN ;

}else if(pGPIOx==GPIOB){

RCC->APB2ENR |= RCC\_APB2ENR\_IOPBEN ;

}else if(pGPIOx==GPIOC){

RCC->APB2ENR |= RCC\_APB2ENR\_IOPCEN ;

}}

if(EnOrDi==DISABLE){

if(pGPIOx==GPIOA){RCC->APB2ENR &=~ RCC\_APB2ENR\_IOPAEN ;

}else if(pGPIOx==GPIOB){

RCC->APB2ENR &=~ RCC\_APB2ENR\_IOPBEN ;

}else if(pGPIOx==GPIOC){

RCC->APB2ENR &=~ RCC\_APB2ENR\_IOPCEN ;

}}

}

Dakle ova funkcija ima 2 arkumenta , prvi arkumenat jeste GPIOx I unosi se registar kojim zelimo da manipulisemo . Drugi argument je EnOrDi , tu govorimo funkcij dali treba da ukljuci ili iskljuci klok na registru.

Sam kod nam slusi kako bi provjerili dali korisnik zeli ukljuciti ili iskljuciti subjekat , te da provjeri o kojem subjektu je tacno rijec . To je odradjeno pomocu if/else naredbi.

2. void GPIO\_PinMode(GPIO\_TypeDef \*pGPIOx, uint16\_t pinNumber, uint16\_t pinMode, uint32\_t pinSpeed=FAST){

if(pinMode==INPUT){

pGPIOx->CRL &=~ (1<<pinNumber\*4);

pGPIOx->CRL &=~ (1<<(pinNumber\*4)+1);

pGPIOx->CRL &=~ (1<<(pinNumber\*4)+2);

pGPIOx->CRL |= (1<<(pinNumber\*4)+3);

}else if(pinMode==OUTPUT){

pGPIOx->CRL |= (0<<pinNumber\*4);

pGPIOx->CRL |= (1<<pinNumber\*4+1);

pGPIOx->CRL &=~(1<<pinNumber\*4+2);

pGPIOx->CRL &=~(1<<pinNumber\*4+3);}}

Ovaj kod sluzi za podesavanje statusa pina / definise dali je seljeni pini **INPUT ili OUTPUT**

(Ulaz ili Izlaz). Sama funkcija je malo komplikovanija I sadrzi 4 argumenta ukupno , s time da je 4-tvrti argument zapravo konstanta. Prvi argumenat kao I u prethodnom primjeru sluzi za definisanje GPIO registra kojime zelimo manipulisati , drugi nam govori o pinu koji zelimo manipulisati , a treci registar nam sluzi definisanje funkcije koji zelimo da dati pin obavlja . U heder filuz imamo definisane INPUT I OUTPUT vrijednosti tako da kod koristenja ove komande koristimo te “oznake” za jednostavniju definiciju.

Sto se tice samog koda kreiran je po definicij registara naime registri pocinju od indeksa 0 I to je 0-ti pin datog GPIOx , I sljedeca 4 bita sluze za definisanje istog . Tako da do zeljenog bita koristimo tako sto uzmemo broj pina pomnozimo ga sa 4 I za svaki sljedeci bit samo taj rezultat povecamo za 1. Za podesavanje pinova koristimo |= za 1 I $=~ za 0 , tako da za input stanje pinova treba biti 1000 , output 0011 .

3. bool GPIO\_DigitalRead(GPIO\_TypeDef \*pGPIOx,uint16\_t pinNumber){

bool ulaz=false;

if(pinNumber==0)

ulaz = pGPIOx->IDR & GPIO\_IDR\_IDR0;

Ova funkcija ima svrhu ocitavanje stanja na INPUT pinovima . Ovo naravno nije cijela funkcija ali se u biti ponavlja if statement sa promjenom pinova od 0 – 15.

Dakle ova funkcija je prilicno jednostavna ima 2 argumenta , bas kao I prva .

Argument 1 je GPIOx, argument 2 je broj pina.

Sam kod prvo deklarise bool vrijednost ulaz I prtpostavlja da je isti false .

Nakon toga pocinju if statmenti koji porede cijeli idr registar sa ranije definisanom vrijednoscu za pin u pitanju . Vrijednost rezultata se sprema u promjenjljivu ulaz I funkcija na kraju vrati ulaz na glavni program.

## 7.2 Opisivanje vaznih I razumljivih dijelova biblioteke za OLED

Biblioteka koju koristim u ovom primjeru kreirana je od strane nekih kolega iz slovenije („Tilen Majerle“) i prilagodjena za ovaj mikrokontroler od strane moga mentora.

Zastoji se iz vise filova i potrudicu se da ih sto bolje objasnim u sljedecem dijelu dokumenta.

### 7.2.1 Fondovi

Ovaj file sadrzi 3 velike funkcije koje sluze za definisanje fondova (16x26 ,11x18,7x10) i 4 pomcne funkcije koje sluze za primtanje trazenih karaktera na ekranu.

const uint16\_t Font7x10 [] = {

0x1000, 0x2800, 0x2800, 0x2800, 0x2800, 0x7C00, 0x4400, 0x4400, 0x0000, 0x0000, // A

Uprimjeru koda iznad imamo funkciju koja definise karakter A u 7x10 fondu.

Imamo 10 hexadecimalnih kodova koji kada bi se pretvorili u binarnu formu kao sto mozemo

0001000000000000 datom primjeru ispitsuje karakter . Svaki hex kod se prevodi u binarnu

0010100000000000 kombinaciju od 10 karaktera. Tako je odradjeno za sve ostale karaktere.

0010100000000000

0111110000000000

0100010000000000

FontDef\_t Font\_7x10 = {

7,

10,

Font7x10

};

Ovo je primjer jedne od pomocnih funkcija . Data funkcija sluzi za definisanje fonda , visine duzine i naziva. Postoje 2 skoroo identicne funkcije za 2 preostala fonda.

char\* FONTS\_GetStringSize(char\* str, FONTS\_SIZE\_t\* SizeStruct, FontDef\_t\* Font) {

/\* Fill settings \*/

SizeStruct->Height = Font->FontHeight;

SizeStruct->Length = Font->FontWidth \* strlen(str);

/\* Return pointer \*/

return str;}

Ovo je zadnja 7 funkcija u fond.cpp dokumentu i ona sluzi za odrdzivanje velicine simbola i pozicioniranje pointera na ekranu za jednostavnij ispis karaktera.

### 7.2.2 SSD1306.cpp

Ovo je sljedci dio biblioteke koji je bitan za razumjevanje u svrhu kreiranja applikacije

(moze se odraditi i bez razumjevanja). Ta biblioteka koristi mnogo korisnih funkcija , ali za nas program bitne su sam sljedece 3 :

1. SSD1306\_UpdateScreen
2. SSD1306\_GotoXY
3. SSD1306\_Puts

Podrudit cu se da sto bolje objasnim ove funkcije u sljedecem dijelu dokumenta.

1. void SSD1306\_UpdateScreen(void) {

uint8\_t m;

for (m = 0; m < 8; m++) {

SSD1306\_WRITECOMMAND(0xB0 + m);

SSD1306\_WRITECOMMAND(0x00);

SSD1306\_WRITECOMMAND(0x10);

/\* Write multi data \*/

ssd1306\_I2C\_WriteMulti(SSD1306\_I2C, SSD1306\_I2C\_ADDR, 0x40, &SSD1306\_Buffer[SSD1306\_WIDTH \* m], SSD1306\_WIDTH);}}

Po samom nazivu funkcij mozemo zakljuciti da je njezin zadatak ubdatovanje ekrana

I u sebi koristi seriju drugih funkcija koje su definisane u ostatku dokumenta (nemoramo razumjeti kako te funkcije rade kako bi mogli kreirati program , a objasnjnje bi ucinilo maturski jos vecim sto nije potrebno).

# 2. void SSD1306\_GotoXY(uint16\_t x, uint16\_t y) {

# /\* Set write pointers \*/

# SSD1306.CurrentX = x;

# SSD1306.CurrentY = y;}

# Funkcija iznad je takodjer jednostavna za objasniti , kosto joj samo ime kaze ona salje pointer na odredjenu lokaciju na ekranu o , ima 2 argumenta , kao I dvije linije koda koji nas dalje slaju na posebne funkcije za pozicioniranje po x I y osi.

Nije neophodno da razumijemo te funkcije za koristenje biblioteke I nacin pozicioniranje ne utjece drasticno na rezultate rada.

3. char SSD1306\_Puts(char\* str, FontDef\_t\* Font, SSD1306\_COLOR\_t color) {

/\* Write characters \*/

while (\*str) {

/\* Write character by character \*/

if (SSD1306\_Putc(\*str, Font, color) != \*str) {

/\* Return error \*/

return \*str;}

/\* Increase string pointer \*/

str++;}

/\* Everything OK, zero should be returned \*/

return \*str;}

Ovo je glavna funkcija koja se koristi za printanje karaktera iz fond cpp fiela na ekranu

To postize uz pomoc \* c++ pointera I druge funkcije koja je definisana negdje u biblioteci

Kao sto sam vec gore naveo nacin rada tih pratecih funkcija nije toliko bitan , ali uglavnom one rade zajedno tak ost F nm2 pozicionira pointer na zeljenu lokaciju , F nm 3 Ispise karaktere definisane u Fond fajlu , F nm 1 resatuje ekran kako bi se zeljeni rezultat ispisao na ekranu.

## 7.3 Opisivanje glavnog „main()“ koda

Sa svim bibliotekam odradjenim uspjeli smo da zadrzimo main prilicno cistim.

Sa samo dvijema dodatnim funkcijama i nekoliko globalnih variabli.

Funkcije je nisam odradio ja vec sam ih dobio od strane mentora te cu u sljedecem djelu koda objasniti funkciju koja je bitna i dovoljno jednostavna da je objasnim.

1. void testString(int x)

{

switch(x){

case 0:

SSD1306\_Puts("00", &Font\_16x26, SSD1306\_COLOR\_WHITE);

break;

Dakle ovo je funkcija koja nam je neophodna kako bismo mogli intager brojeve ispisati na ekran , buduci da orginalne funkcije smo primaju string argument za printanje . Funkcija je prilicno jednostavna , prima 1 int argument , te pomocu switch statmenta odredjuje o kojem se broju radi I printa taj zeljeni broj . Boduci da radimo sa timerom switc statmenti idu

Od 0 do 59 .

Iz main() koda izdvojicu najvaznije dijelove u dokument kako bi ih opisao I docarao .

1. int n=59,m=59, x,j;

j>=0? j--:j=n;

if(j==0){x>0?x--:x=m;}

Dakle ovaj dio koda je jednostavan I on sluzi za brojanje dakle dekrementiranje vrijednosti n-sekundi I m-minuta.Sa 2 brojaca (J & i).

2. if(a0==true)

{while(1){bool Pritisnuto,P2;

a1=GPIO\_DigitalRead(GPIOA,1);

a2=GPIO\_DigitalRead(GPIOA,2);

a3=GPIO\_DigitalRead(GPIOA,3);

if(a1==true){ if(!Pritisnuto)

x++;else if(59<x)x=0;

Pritisnuto=true;

}else {Pritisnuto=false;}

if(a2==true)

{if(x>0&&!P2)

x--;else if(x==0)x=59;

P2=true;

}else {P2=false;}

SSD1306\_GotoXY(18 ,35);

testString(x);

SSD1306\_UpdateScreen();

if(a3==true){break;}}}

Ovaj dio koda sluzi kako bi nakon sto se detektuje pritisak na prvom tuch buttonu detektuje 1 zaustavi timer , ali pocne nadgledati stanje na ostala 3 buttona .

Zatim pri pritisku na 2 buuton se vrijednost x-minute poveca za 1 , na 3 smanji za jedan , a na 4 timer nastavi odbrojavati od odabranog vremena prema 0.

Pored tih djelova postoje i if osiguravajuci statmenti goji ne dozvoljavaju vrijednost x da izace iz opsega (59-0) i takodjer sprjecavaju da se autoincrementira vise puta pri jednom pritisku , jer bi inace vrijednost porasla brzinom od 8MGHz.

Kao i dio koji ubdajtuje ekran sa novom vrijednoscu x-a kako bi korisnik znao dokle je stigao.

3. SSD1306\_GotoXY( 18,35);

testString(x);

SSD1306\_GotoXY( 54,35);

SSD1306\_Puts(":", &Font\_16x26, SSD1306\_COLOR\_WHITE);

SSD1306\_GotoXY(64,35);

testString(j);

SSD1306\_UpdateScreen

for(int i=0;i<10000;i++) //delay

{}

Ovaj dio sluzi za ispisivanje vremena za vrijeme odbrojavanja , te za pravljenje pauze od 1s izmedju svake promjene vrijednosti.

# 4. Zaključak

Odabrao sam Mikroracunare kao predmet iz kojeg cu raditi maturski iz vise razloga :

* Izazov za zavrsetak srednje - Kako bi sam sebi dokazao da zadnje 4 godine nisam protracio i da sa znanjem stecenim u ovoj skoli mogu krirati nesto sto je korisno za svakodnevni zivot.
* Kako bi primjeni sve svoj vjestine .
* Kako bi dobio priliku da radim sa 3D printerom.
* Kako bi izbjegao guzvu za koju sam znao da ce se stvoriti oko predmeta ciji su maturski iskljucivo softwer based.

I na kraju mogu reci da je maturski rad ispunio sva moja ocekivanj . Sve probleme sam uspjeo rijesiti uz pomoc mentora , pa cak i uz nenajavlljeno skracivanje roka za 8 dana uspjeo sam rijesiti maturski sa minimalnim sakasnjnjem.

Prilicno sam zadovoljan novim vjestinama koje sam morao nauciti za ovaj projekat tj :

- Kontrola OLED ekrana , kao i

- 3D manipuliranje

Prilicno sam zadovoljan sam sa postignutim rezultatima i vjerujem da je dovoljno barem za prolaznu ocjenu .

Naravno mogao sam ovo odraditit mnogo bolje , jedna od stvari za koju mi je jako zao sto nisam implementirao jeste slika analognog sata sa kazaljkam akoj govore vrijeme i blicanje cijelog ekrana pri isteku vremena . 5. Literatura

[1] wikipedija (en)

[2] STM32F103C8 DATASHEAT

[3] encyclopedia2.

[4] tronixstuff.com

[5] electronicshub.org

|  |
| --- |
| Mišljenje mentora o radu: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Predložena ocjena: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( \_\_ )

Ispitivač:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Članovi komisije:

1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Izdvojeno mišljenje:

Datum odbrane: \_\_\_. \_\_\_. \_\_\_\_\_\_\_.