SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR 1

MIKROKONTROLERSKI SUSTAV ZA GENERIRANJE, FILTRIRANJE I MJERENJE TRAJANJA DIGITALNIH SENZORSKIH SIGNALA

Jakov Begović

Mentor: doc. dr. sc. Dinko Oletić

Zagreb, veljača, 2025.

Mikrokontrolerski sustav za generiranje, filtriranje i mjerenje trajanja digitalnih senzorskih signala

Jakov Begović

Sažetak

Projekt vođen od strane doc. dr. sc. Dinka Oletića zvan "Elektronički senzorski sustavi s niskom potrošnjom energije za detekciju akustičkih događaja" istražuje koncept višekanalnog senzorskog sustava građenog od piezoelektričnih mikrorezonatora [1]. Mikrorezonatori su podešeni za detekciju akustičnih događaja u uskom frekvencijskom području. Zajedno pokrivaju željeni frekvencijski spektar.

Takav sustav izbjegava potrebu za frekvencijskom analizom prikupljenog signala. Na taj se način ubrzava proces prikupljanja podataka i umanjuje potrošnja energije kod obrade istih.

Senzorski sustav zahtjeva višekanalnu obradu signala. Analogni signal piezoelektričnog senzora se prvo digitalizira, koristeći trajanje za prijenos informacije o karakteru akustičnog događaja. To je trajanje potrebno očitati. Signale trajanja ispod određene vrijednosti je potrebno interpretirati kao šum.

Rad se temelji na 2 mikroupravljača: PSoC 5LP i ESP32. PSoC 5LP se koristi za filtriranje i mjerenje trajanja, a ESP32 za generiranje digitalnog signala proizvoljnog trajanja. Sustav zahtjeva korištenje maksimalne količine vanjskih digitalnih blokova PSoC 5LP-a. Tako se vrijeme rada mikroprocesora maksimalno umanjuje i cjelokupni sustav troši minimalnu količinu energije.

Signali kraći od 1 mikrosekunde se odbacuju Glitch Filter komponentom PSoC 5LP-a. Komponenta Timer istog mikroupravljača služi očitanju trajanja filtriranog digitalnog signala. Mikroprocesor se aktivira prekidom, što ostvaruje mogućnost spavanja mikroprocesora do potrebe za obradom, pohranom ili prijenosom podatka.

Komponenta "Remote Control Transciever" (akr. RMT) ESP32 mikroupravljača se koristi u svrhu generiranja digitalnih signala. S pomoću njih se sustav testira.

Ključne riječi: PSoC 5LP; ESP32; digitalni sustav; generiranje; filtriranje; mjerenje; niska potrošnja; digitalni blok;

Sadržaj

1. Uvod

1.1. O projektu

Projekt vođen od strane doc. dr. sc. Dinka Oletića zvan "Elektronički senzorski sustavi s niskom potrošnjom energije za detekciju akustičkih događaja" istražuje "novi koncept senzorskog sustava na čipu za detekciju akustičkih događaja niskih snaga, koji sadrži kontaktni akustični mikroelektromehanički (MEMS) senzor dizajniran kao potpuno pasivni niz frekvencijski uskopojasnih i selektivnih piezoelektričnih mikrorezonatora, u kombinaciji s višekanalnim mikroelektroničkim krugom niske potrošnje za očitavanje i obradu signala na čipu" [1].

Takav višekanalni MEMS senzor bi se sastojao od više piezoelektričnih elemenata. Individualni piezoelektrični element bi detektirao akustične valove u uskom frekvencijskom spektru. Korist takvog sustava bi bilo izbjegavanje potrebe za frekvencijskom analizom signala nakon njegovog prikupljanja.

1.2. Područja primjene višekanalnog MEMS senzora

Ovakvi senzori se mogu koristiti u bilo kojem kontekstu detekcije akustičnih ili ultrazvučnih događaja, kao što su "nadzor infrastrukture, preventivno održavanje strojeva, detekcija curenja, kavitacije, u implantabilnim medicinskim slušnim uređajima i slično" [1].

1.3. Sustav obrade signala

Signal iz jednog piezoelektričnog elementa bi se pretvorio u digitalni signal istog trajanja. Trajanje digitaliziranog signala iz senzora može biti od par desetaka nanosekundi do par stotina mikrosekundi. Ukoliko bi trajanje tog digitalnog signala bilo ispod proizvoljnog, bio bi odbačen kao šum. Trajnje filtriranih signala bi trebalo biti izmjereno i pohranjeno u trajnu memoriju.

Ovakav sustav bi morao biti višekanalan. Jedan bi se kanal koristio za obradu signala iz jednog piezoelektričnog senzora. Osim toga, maksimalna količina obrade signala bi trebala biti odrađena od strane analognog i digitalnog sklopovlja, isključujući mikroprocesor. Na taj bi se način potrošnja mikroprocesora mogla maksimalno umanjiti. Odnosno, mikroprocesor bi maksimalnu količinu vremena mogao provesti u stanju mirovanja.

1.4. Pohranjivanje podataka i vanjsko sučelje

Sustav bi morao moći trajno pohraniti podatke prikupljanje sa senzora. Ti bi se podaci stavili u kontekst koristeći atribute kao što su datum i vrijeme prikupljanja signala, trajanje signala, kanal prijenosa i slično. Također, trebalo bi postojati sučelje za prijenos prikupljenih podataka na drugi mikroprocesor ili drugu vrstu trajne pohrane.

1.5. Opseg seminarskog rada

Ovaj seminarski rad razmatra razvoj digitalnog sklopovlja i programskog koda za:

- filtriranje digitalnog ulaznog signala
  - odbacivanje signala kraćeg od proizvoljnog vremena trajanja
- mjerenje trajanja digitalnog ulaznog signala
  - koristeći digitalno sklopovlje van mikroprocesorske jedinice
- generiranje digitalnog signala proizvoljnog trajanja
  - u svrhu testiranja razvijenog sustava

2. Metodologija

2.1. Korištena tehnologija

U svrhu razvoja ovog seminarskog rada korištene su razvojne pločice PSoC 5LP i Dasduino CONNECTPLUS [2] [3]. Uz njih je dodano 5 vanjskih gumbova koji služe upravljanju sustavom i nekoliko žica koje povezuju njihove ulazno-izlazne pin-ove.

U svrhu razvoja programskog koda i postavljanja digitalnih blokova za PSoC 5LP korišten je računalni program PSoC Creator u verziji 4.2 [4]. U svrhu razvoja programskog koda za Dasduino CONNECTPLUS korišten je računalni program Visual Studio Code s ekstenzijom PlatformIO [5] [6].

U svrhu nadzora digitalnih signala korišten je digitalni osciloskop.

U svrhu čitanja serijske komunikacije PSoC 5LP-a i računala korišten je računalni program Tera Term [7].

2.1.1. PSoC 5LP

PSoC 5LP je programabilni sustav na čipu proizvođača Cypress Semiconductor Corporation, koja je od 2020. godine u vlasništvu Infineon Technologies AG.

Programabilni sustav objedinjuje Arm Cortex-M3 procesorsku jedinicu, 24-bitni digitalni filter blok (akr. DFB), 24 univerzalna digitalna bloka (akr. UDB), kontroler za direktni pristup memoriji (engl. *direct memory access controller*), analogne komponente, integrirane sustave za korisnička sučelja i drugo [2]. Korist tog programabilnog sustava na čipu je dostupnost raznih analognih i digitalnih komponenti na istom čipu na kojem se nalazi mikroprocesorska jedinica. Na taj se način smanjuje vrijeme i trošak prototipiranja.

U ovom radu je korištena jedna od njegovih verzija: CY8C5888LTI-LP097.

2.1.2. Dasduino CONNECTPLUS

Dasduino CONNECTPLUS je razvojna pločica koja sadrži ESP32 mikroupravljač u verziji ESP32-WROVER-E. Razvojna pločica je razvijena i proizvedena u Hrvatskoj [3].

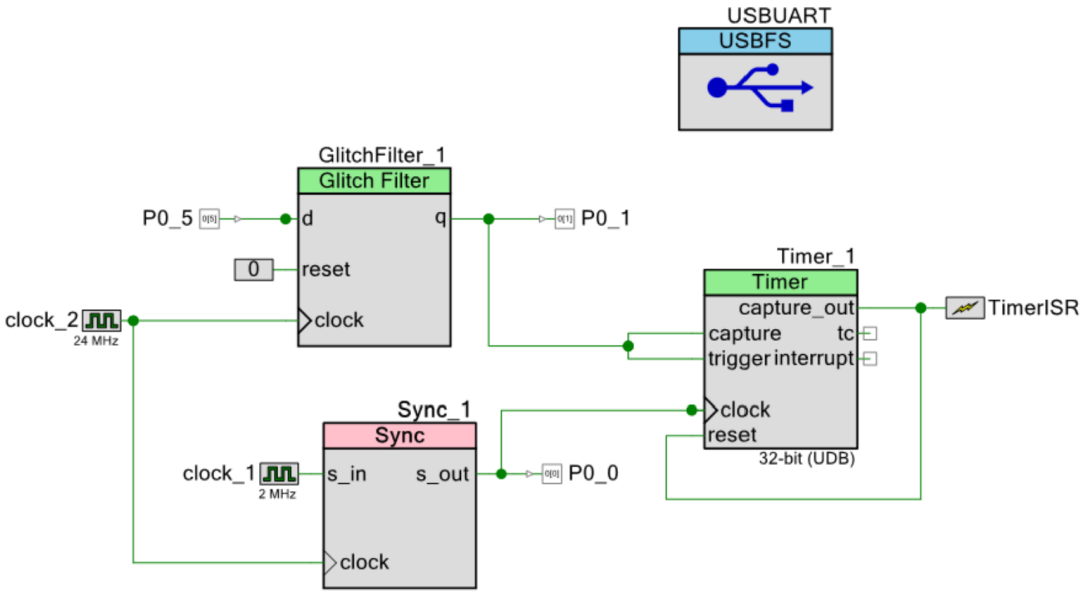
U ovom radu je korišten u svrhu generiranja digitalnih impulsa trajanja od 250 nanosekundi do 64 mikrosekunde. Ti se digitalni impulsi koriste za testiranje filtriranja digitalnog impulsa i mjerenja njegovog trajanja.

2.1.2.1. ESP32-WROVER-E

ESP32-WROVER-E proizvođača Espressif je mikroupravljačka jedinica koja sadrži Tensilica Xtensa 32-bit LX6 mikroprocesor. On radi na frekvencijama do 240 MHz. Mikroprocesor sadrži 2 procesorske jezgre. Jedna je uobičajeno korištena za Bluetooth i WiFi komunikaciju, no programer može upravljati raspodjelom zadataka između jezgri [8]. Programiranje ESP32 čipova potpomognuto je nizom biblioteka razvijenim od strane Espressif-a i okupljene zajednice. Te su biblioteke u jednoj svojoj verziji prilagođene kako bi odgovarale Arduino aplikacijskom programskom sučelju (akr. API) [9]. U ovom su radu korištene biblioteke `Arduino.h` i `driver/rmt.h`.

2.2. Sheme spajanja

2.2.1. Shema povezivanja digitalnih komponenti PSoC 5LP-a



Slika 2.1. Shema povezivanja digitalnih komponenti PSoC 5LP-a

2.2.2. Digitalni ulazi i izlazi PSoC 5LP-a

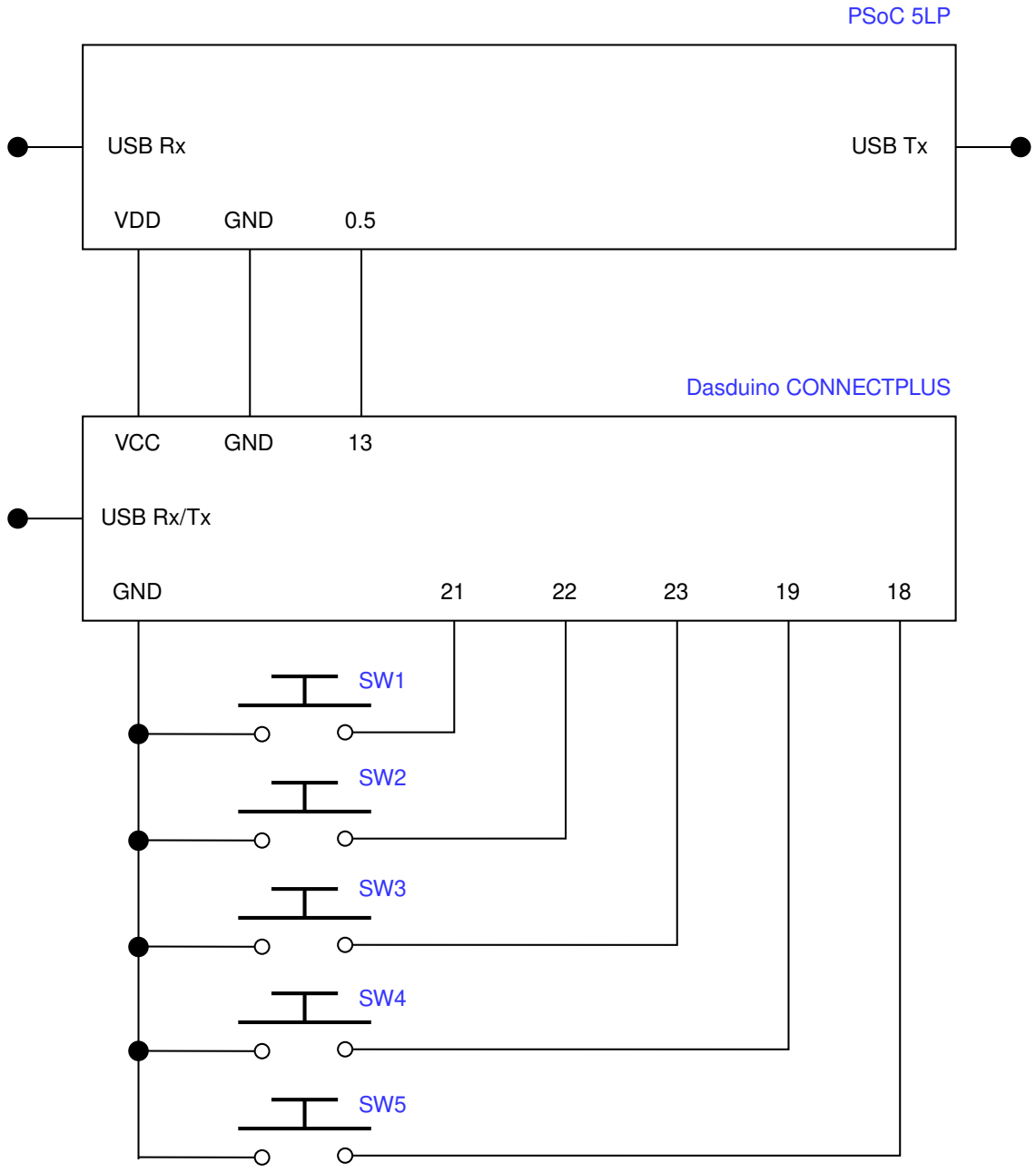
`P0_5` predstavlja digitalni ulaz, a `P0_1` i `P0_0` digitalne izlaze PSoC 5LP-a.

Ulaz `P0_5` služi kao ulaz digitalnog signala kojeg je potrebno filtrirati i čije je trajanje potrebno izmjeriti. Implementiran je kao digitalni ulaz s "pull-down" otpornikom.

Izlaz `P0_1` predstavlja filtrirani digitalni signal iz ulaza `P0_5`. Služi za vizualizaciju i testiranje rada sustava. Implementiran je kao digitalni izlaz tipa "strong drive". Takva implementacija najvjerodostojnije prikazuje stanje na digitalnom izlazu komponenti.

Izlaz `P0_0` predstavlja sat frekvencije 2 MHz imena "clock_1" sinkroniziran s obzirom na sat frekvencije 24 MHz imena "clock_2". Služi za vizualizaciju i testiranje rada sustava. Implementiran je kao digitalni izlaz tipa "strong drive".

2.2.3. Shema povezivanja PSoC 5LP-a i Dasduino CONNECTPLUS-a



Slika 2.2. Shema povezivanja PSoC 5LP-a i Dasduino CONNECTPLUS-a

Shema zanemaruje nekorištene elemente sustava. USB Tx izlazom PSoC 5LP-a upravlja USBFS komponenta prikazana na slici 2.1.

2.3. Mjerenje trajanja signala

U svrhu mjerenja trajanja digitalnog signala korištena je Timer komponenta PSoC 5LP-a. Odabrana je jer je namijenjena svrsi generiranja ili bilježenja trajanja (intervala između dva događaja), dok je Counter komponenta prilagođenija brojenju događaja [10].

2.3.1. Timer komponenta

Timer komponenta PSoC 5LP mikroupravljača sadrži i osnovne i napredne funkcije mjerača vremena. One su: dekrementiranje vrijednosti pohranjene u brojaču, pohranjivanje te vrijednosti na zahtjev, generiranje prekida i ostalo. Komponenta sadrži digitalne ulaze i izlaze za upravljanje komponentom.

"count" je cjelobrojna vrijednost koja se dekrementira u Timer-u. U početnom stanju i nakon resetiranja iznosi vrijednost periode. Dekrementira se na svaki impuls sata ukoliko je Timer komponenta omogućena i ukoliko je dekrementiranje omogućeno.

Timer komponenta PSoC 5LP-a je u ovom seminarskom radu konfigurirana s parametrima:

- implementacija: UDB
- rezolucija: 32-bitna
- perioda: 4294967296
- frekvencija sata: 2 MHz
- enable mode: isključivo softverski
- trigger mode: rastući brid
- capture mode: padajući brid
- run mode: neprekidan (engl. *continuous*)
- interrupts: niti jedna opcija odabrana
- ulazi i izlazi povezani kao što je prikazano na slici 2.1.

2.3.1.1. Implementacija

Komponenta dolazi u dvije implementacije: Fixed Function (akr. FF) i Universal Digital Block (akr. UDB). U ovom se radu koristi UDB implementacija jer nudi maksimalnu rezoluciju od 32 bita naspram maksimalnih 16 bita FF implementacije. Uz to, UDB implementacija nudi veću fleksibilnost konfiguracije. Najveći nedostatak korištenja UDB implementacije je zauzimanje jednog od 20 UDB blokova [11].

2.3.1.2. Rezolucija i maksimalna perioda

U ovom je radu korištena 32-bitna rezolucija. Ona određuje maksimalnu veličinu periode. Perioda predstavlja maksimalan broj otkucaja do prekoračenja opsega cijelog broja [12]. U ovom je radu postavljena na $2^{32} = 4294967296$. Vrijednost count se nakon resetiranja postavlja na $2^{32} - 1 = 4294967295$. Broj 2^{32} uzima u obzir i nulu.

Iznos maksimalne periode je relevantan jer je Timer komponenta u ovom mikroupravljaču izvedena kao dekrementirajuća. Očitanje vrijednosti "count" će davati broj otkucaja sata za vrijeme trajanja digitalnog signala, označeno kao n [10].

$$n = 2^{32} - 1 - count \tag{2.1}$$

2.3.1.3. Frekvencija sata

U slučaju da je Timer komponenta omogućena i da je omogućeno dekrementiranje vrijednosti "count", ona će se dekrementirati na svaki otkucaj sata.

Frekvencija sata stoga određuje mjernu jedinicu vrijednosti n, koja se dobiva formulom 2.1.

Frekvencija sata je u ovom seminarskom radu postavljena na 2 MHz. Perioda sata je: $1/2MHz = 0,5\mu s$. Dogode se 2 dekrementacije vrijednosti "count" unutar 1 mikrosekunde. Vrijednost n moramo podijeliti s 2 kako bi dobili trajanje digitalnog signala u mikrosekundama.

Jednadžba za vrijeme trajanja digitalnog signala t_{SIG} je

$$t_{SIG}[s] = \frac{2^{32} - 1 - count}{f} \tag{2.2}$$

$$t_{SIG}[\mu s] = \frac{2^{32} - 1 - count}{f} \cdot 10^6 \tag{2.3}$$

Frekvencija sata u ovoj implementaciji iznosi 2 MHz kako bi bila dovoljno točna u mjerenju duljine trajanja digitalnog signala.

Frekvencija od 1 MHz uzrokuje maksimalnu grešku očitanja u iznosu od ± 0,99 µs. Frekvencija sata od 2 MHz umanjuje maksimalnu grešku očitanja na ± 0,49 µs. Promjena ove točnosti je relevantna jer se trajanje ulaznog digitalnog signala mjeri u mikrosekundama.

S druge strane, za periodu 2^{32} i frekvenciju sata od 2 MHz maksimalno trajanje digitalnog signala $t_{SIG_{MAX}}[s]$ iznosi

$$t_{SIG_{MAX}} = 2^{32} \cdot \frac{1}{2 \cdot 10^{-6}} = 2147,483648s \tag{2.4}$$

Frekvencija se po potrebi može povećati. Tada bi faktor pretvorbe n u t_{SIG} bio promijenjen, kao što je i prijašnje pokazano. Također, umanjilo bi se maksimalno dopušteno trajanje digitalnog signala.

Kod povećanja frekvencije rada Timer-a na 4 MHz, maksimalno trajanje digitalnog signala $t_{SIG_{MAX}}[s]$ bi iznosilo

$$t_{SIG_{MAX}} = 2^{32} \cdot \frac{1}{4 \cdot 10^{-6}} = 1073.741824s \tag{2.5}$$

te bi umanjilo maksimalnu grešku na ± 0,24 µs.

2.3.1.4. Enable mode

Enable mode se odnosi na omogućavanje Timer komponente. Nad njom se ne mogu vršiti naredbe sve dok nije omogućena. Postoje 3 opcije za omogućavanje: isključivo

hardverski, isključivo softverski, hardverski i softverski.

U ovom radu se koristi isključivo softverski način. Timer se u kodu omogućava pozivom funkcije `Timer_1_Start()` unutar `main()` funkcije.

2.3.1.5. Trigger mode

Trigger Mode uzrokuje odgađanje brojanja Timer-a dok se ne detektira odgovarajući rub na "trigger" ulazu u komponentu. Aktiviranje "trigger" ulaza ne dekrementira "count".

U ovoj je konfiguraciji odabrana opcija "Rising Edge". To znači da, iako je Timer komponenta omogućena, "count" se neće mijenjati sve dok se ne detektira rastući brid na "trigger" ulazu.

Na "trigger" ulaz je doveden digitalni signal kojem se treba očitati trajanje. On će aktivirati rastući brid. Do tog trenutka se "count" neće moći dekrementirati, neovisno o drugim ulazima ili stanjima. Ako je Timer omogućen kao što je opisano u poglavlju 2.3.1.4., nakon aktiviranja "trigger" ulaza će se vrijednost "count" dekrementirati sve do naredbe "capture".

2.3.1.6. Capture mode

Izraz "capture" odnosi se na naredbu pohranjivanja trenutne "count" vrijednosti u zasebni registar. U slučaju UDB implementacije, moguće je pohraniti više vrijednosti bez njihova brisanja. U službenoj se dokumentaciji taj odjeljak memorije naziva "FIFO" zbog redoslijeda iščitavanja pohranjenih vrijednosti.

Ta naredba se može pozvati hardverski ili softverski. U ovom radu se ona poziva hardverski, padajućim bridom na ulaz "capture".

Kao što je vidljivo na slici 2.1., na ulaz "trigger" i "capture" se dovodi isti signal. Signal čije se trajanja pokušava izmjeriti. "trigger" se aktivira na rastući brid, a "capture" na padajući. Na taj način ostvarujemo sljedeće: "count" se krene dekremetirati na početku signala te se njegova vrijednost pohranjuje u FIFO na kraju signala.

Kako bi se pretvorila u duljinu trajanja digitalnog signala ta se vrijednost mora provesti kroz formulu 2.3.

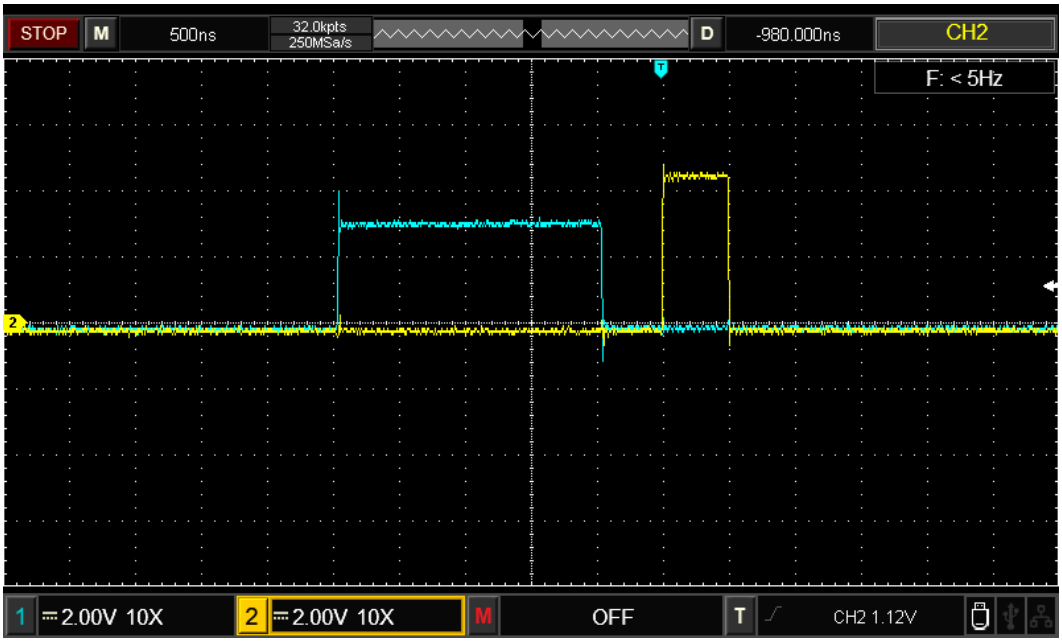
2.3.1.7. Run mode

Parametar "Run mode" utječe na ponašanje Timera u slučaju da "count" vrijednost dođe do 0.

U ovoj je konfiguraciji odabrana opcija "neprekidan" (engl. *continuous*). U toj konfiguraciji vrijednost "count" se neprekidno umanjuje.

2.3.1.8. Prekid

Prekid mikrokontrolera se aktivira na prijenos vrijednosti "count" u FIFO. Odnosno, na rastući brid izlaza "capture_out". Mikroprocesor može biti u stanju mirovanja do tog trenutka. Na taj način se umanjuje snaga koju sklop troši.



Slika 2.3. Aktiviranje prekida (crveni impuls) 0,5 µs nakon padajućeg brida ulaznog signala (plavi impuls)

Na prekid se u programskom kodu poziva funkcija prekida, kao što je definirano naredbom `TimerISR_StartEx( InterruptHandler )`. Defincija funkcije `InterruptHandler` je sljedeća:

```
CY_ISR( InterruptHandler )
{
    capturedValue = timerPeriod - Timer_1_ReadCapture() +
       glitchFilterDelay;
```

```
    capturedValue = capturedValue / 2;

    Timer_1_ReadStatusRegister();
}
```

gdje je `timerPeriod` jednak $2^{32} - 1$, `Timer_1_ReadCapture()` jednak "count" vrijednosti pohranjenoj u FIFO i `glitchFilterDelay` jednak broju otkucaja sata Timera unutar jedne mikrosekunde. `capturedValue` se dijeli s 2 kako bi se dobivena vrijednost pretvorila u mikrosekunde. Vrijednost 2 u ovom slučaju mijenja izraz f iz formule 2.2. Korist funkcije `Timer_1_ReadStatusRegister()` je resetiranje vrijednosti u statusnom registru Timer-a [10].

Na navedeni kod se u sljedećim iteracijama može povezati pohranjivanje podataka o digitalnom signalu, prijenos informacije i slično.

2.3.1.9. Napredne opcije prekida

Postoje napredne opcije za generiranje prekida, a neke od njih su:

- kada vrijednost "count" zauzme vrijednost 0 (engl. *terminal count*, akr. TC)
- kada se FIFO ispuni
- kada se pozove naredba "capture"

U ovoj konfiguraciji niti jedna od opcija nije odabrana. Timer će generirati prekid kroz svoj "capture_out" izlaz iako niti jedna od navedenih opcija nije odabrana.

U svrhu izbjegavanja pogrešaka, moguće je omogućiti prekid na TC. Taj događaj se tada može bilježiti pomoću softvera. Formula za trajanje ulaznog digitalnog signala bi tada postala

$$t_{SIG}[s] = \frac{n_{TC} \cdot 2^{32} + 2^{32} - 1 - count}{f} \tag{2.6}$$

gdje je n_{TC} broj puta kada je "count" vrijednost iznosila 0. Maksimalno trajanje ulaznog digitalnog signala $t_{SIG_{MAX}}$ bi tada prestalo biti ograničeno periodom.

2.3.1.10. Ulazi i izlazi

Dostupni ulazi i izlazi komponente ovise o njenoj konfiguraciji.

Kao što je prikazano na slici 2.1., u ovoj se implementaciji na "capture" i "trigger" ulaze se dovodi digitalni signal čije je trajanje potrebno izmjeriti. Na "clock" ulaz se dovodi sat frekvencije 2 MHz. "capture_out" izlaz generira prekid mikroprocesora u trenutku prijenosa vrijednosti "count" u FIFO [10]. Isti signal resetira Timer komponentu. Postavlja vrijednost "count" na $2^{32} - 1$ kao posljedica rastućeg brida na ulazu "reset".

2.4. Filtriranje signala

2.4.1. Glitch Filter komponenta

Glitch Filter komponenta eliminira neželjene digitalne impulse na ulazu. Filtrira ih po zadanim parametrima trajanja signala i zaobilaska komponente. Ona zamjenjuje softversko odbacivanje neželjenih digitalnih signala, što umanjuje ukupnu energetsku potrošnju mikrokontrolera.

2.4.1.1. Princip rada

Komponenta propušta visoki signal (1) samo kada prethodnih N uzoraka iznose 1, a niski signal (0) samo kada prethodnih N uzoraka iznose 0. Izlaz je u protivnom nepromijenjen s obzirom na trenutno stanje [13].

Opcije zaobilaska su:

- logička 0
  - logička 0 se pušta na izlaz bez uzorkovanja
- logička 1
  - logička 1 se pušta na izlaz bez uzorkovanja
- bez zaobilaska
  - i logička 0 i logička 1 se propuštaju na izlaz samo nakon zadovoljenih N uzoraka

Vrijednost broja N, frekvencija rada sklopa i opcije zaobilaska su proizvoljne, što dovodi do visoke fleksibilnosti sklopa.

2.4.1.2. Ulazi i izlazi

Kao što je prikazano na slici broj 2.1., na ulaz "d" se dovodi signal na ulazu pin-a PSoC-a broj 0.5. To je ulazni signal kojeg treba filtrirati i čije je trajanje potrebno izmjeriti. Izlaz "q" daje filtrirani signal.

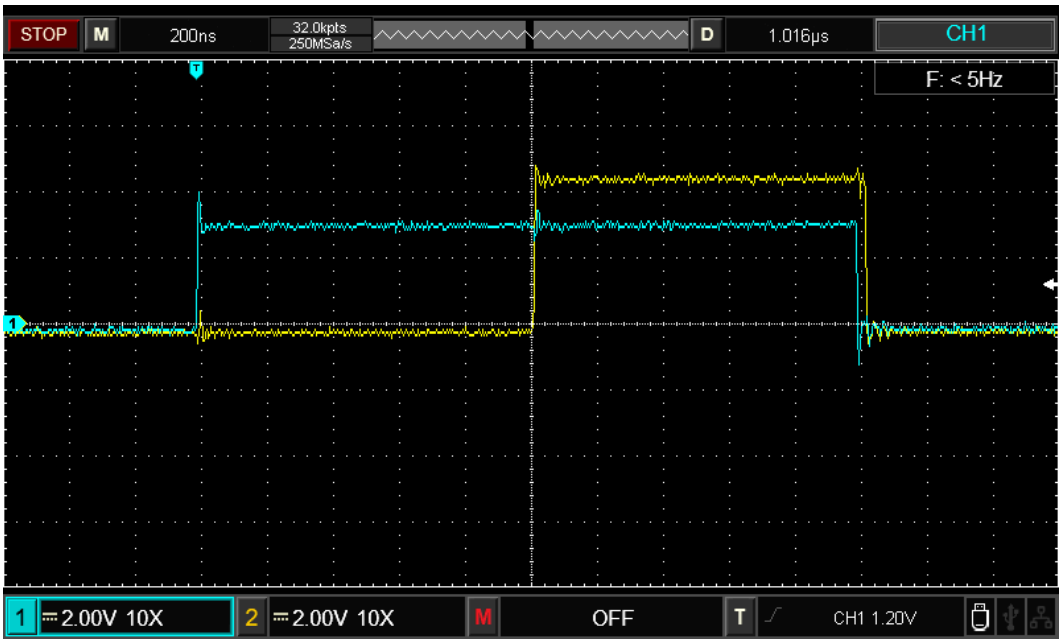
Ulaz "reset" povezan je na logičku 0 kako ne bi došlo do neželjenog resetiranja komponente.

Na ulaz "clock" doveden je sat frekvencije 24 MHz.

2.4.1.3. Konfiguracija

Broj uzoraka za ispunjenje N postavljen je na 24. Samim time, svi digitalni signali trajanja kraćeg od $24/24MHz = 1\mu s$ će biti zanemareni.

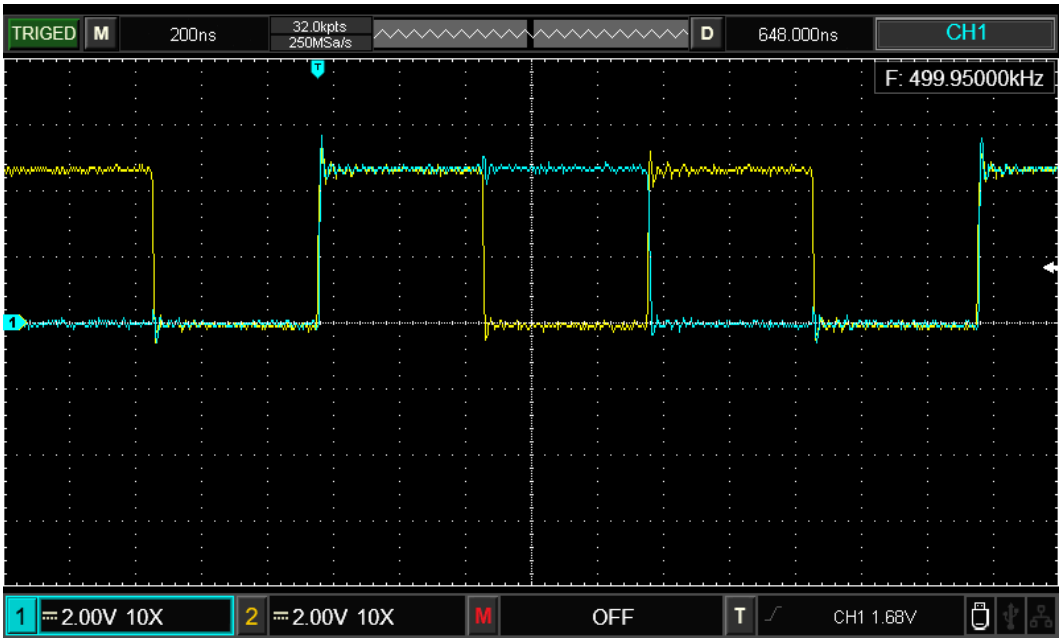
Za zaobilazak je odabrana opcija "logička 0". Na taj će se način osigurati detektiranje prekida signala i prijelaza u drugi signal. To znači da će se ulazni signal skratiti u trajanju za 1 μs, kao što je i prikazano na slici 2.4. Zbog toga je potrebno pribrojiti tu vrijednost očitanom trajanju digitalnog signala u softveru. To je prikazano u programskom kodu poglavlja broj 2.3.1.8. varijablom `glitchFilterDelay`.



Slika 2.4. Skraćivanje vremena trajanja ulaznog signala za 1 µs kao posljedica Glitch Filter komponente

2.4.2. Sync komponenta

Sync komponenta sinkronizira ulazni signal na rastući brid sata. Koristi se kod prijelaza signala iz domene jedne frekvencije sata u domenu druge frekvencije sata [14].



Slika 2.5. Izlaz Sync komponente koja sinkronizira sat frekvencije 1 MHz prema satu frekvencije 2 MHz

U ovoj se implementaciji koristi kako bi satovi Glitch Filter i Timer komponente bili

sinkronizirani. Na taj se način izbjegavaju nepredvidive i neželjene greške povezane sa sinkronizacijom komponenata.

2.5. Generiranje signala

Generator signala koristi testiranju razvijenog sustava. Postoji više zahtjeva nad funkcionalnosti tog generatora:

- generiranje digitalnih signala raznih trajanja
- generirani signal je iste naponske razine kao napajanje PSoC 5LP-a
- trajanje signala seže od ispod mikrosekunde do par mikrosekundi
- točnost trajanja generiranog signala je unutar par nanosekundi
- nema zahtjeva vezanog uz energetsku potrošnju generatora signala

U ovom seminarskom radu će generator signala biti izveden pomoću ESP32 mikrokontrolera. Točnije, njegove verzije ESP32-WROVER-E ugrađene u razvojnu pločicu Dasduino CONNECTPLUS. Razvojna pločica je razvijena i proizvedena u Hrvatskoj.

Taj je mikrokontroler odabran zbog širine fonda hardverskih i softverskih alata za upravljanje digitalnim izlazima, jednostavnosti i dostupnosti.

Za njega je razvijen programski kod koji upravlja njegovim digitalnim izlazima. Također, pin-ovi broj 21, 22, 23, 19 i 18 su mu preko gumbova povezani na GND pin. Shema povezivanja Dasduina i PSoC 5LP mikroupravljača, te navedenih pin-ova prikazana je na slici broj 2.2..

Struktura ESP32-WROVER-E mikrokontrolera je opisana u poglavlju broj 2.1.2.1..

2.5.1. Korištene biblioteke

U radu su za programiranje Dasduina korištene biblioteke `Arduino.h` i `driver/rmt.h`.

2.5.1.1. Biblioteka Arduino.h

Kao što je opisano u poglavlju broj 2.1.2.1., biblioteke za programiranje ESP32 mikrokontrolera su u jednoj svojoj verziji prilagođene kako bi odgovarale Arduino aplikacijskom programskom sučelju (akr. API) [9]. To ih čini jednostavnima za shvatiti i koristiti.

2.5.1.2. Remote Control Transceiver i popratna biblioteka

Biblioteka "driver/rmt.h" upravlja perifernim uređajem ESP32 mikroupravljača zvanim "Remote Control Transceiver". Prema [15], "RMT (Remote Control Transceiver) periferni uređaj je dizajniran da djeluje kao infracrveni primopredajnik. Međutim, zbog fleksibilnosti formata podataka, RMT se može proširiti na svestrani primopredajnik opće namjene, koji odašilje ili prima mnoge druge vrste signala."

Frekvenciju sata mikroupravljača je u ovoj implementaciji postavljena na 80 MHz. Tada je minimalno trajanje impulsa RMT uređaja $1/80MHz = 12,5ns$.

2.5.2. Struktura rješenja

Programsko rješenje se može podijeliti na par odjeljaka:

1. deklaracije globalnih varijabli
2. definicija funkcije za postavljanje RMT kanala
3. definicije funkcija za okidanje željenih izlaznih impulsa
4. definicija setup() funkcije
5. definicija loop() funkcije

2.5.2.1. Deklaracije globalnih varijabli

Deklarirane su sljedeće globalne varijable:

```
const int b1PIN = 21;
const int b2PIN = 22;
const int b3PIN = 23;
const int b4PIN = 19;
const int b5PIN = 18;
```

```
const int signalPIN = 13;
const int clockDivider = 20;

rmt_item32_t pulseItem1;
rmt_item32_t pulseItem2;
rmt_item32_t pulseItem3;
rmt_item32_t pulseItem4;
rmt_item32_t pulseItem5;

uint32_t lastCheck = 0;
```

Varijable sa sufiksom `PIN` predstavljaju definicije broja pin-a. Varijabla `clockDivider` predstavlja faktor s kojim se dijeli frekvencija sata od 80 MHz. Tada ta vrijednost postaje minimalno trajanje RMT impulsa. Varijable `pulseItem` predstavljaju slogove (engl. struct) koji sadrže informacije o RMT impulsu. Varijabla `lastCheck` koristi odgađanju provjere digitalnih stanja na pin-ovima povezanim s gumbovima. Koristi se u `main()` funkciji.

2.5.2.2. Definicija funkcije za postavljanje RMT izlaznih impulsa

Definicija funkcije za postavljanje RMT kanala je:

```
void setupRMTChannel( rmt_channel_t channel, gpio_num_t gpio_num )
{
    rmt_config_t rmtConfig;
    rmtConfig.rmt_mode = RMT_MODE_TX;
    rmtConfig.channel = channel;
    rmtConfig.gpio_num = gpio_num;
    rmtConfig.mem_block_num = 1;
    rmtConfig.tx_config.loop_en = false;
    rmtConfig.tx_config.carrier_en = false;
    rmtConfig.tx_config.idle_output_en = true;
    rmtConfig.tx_config.idle_level = RMT_IDLE_LEVEL_LOW;
    rmtConfig.clk_div = clockDivider;

    rmt_config( &rmtConfig );
    rmt_driver_install( channel, 0, 0 );
}
```

`RMT_MODE_TX` postavlja RMT kanal u funkciju prijenosa signala. `channel` se odnosi na redni broj kanala za kojeg se svojstva definiraju. Oni sežu od 0 do 7. `gpio_num` se odnosi na broj pin-a na kojem je taj kanal aktivan [15]. `mem_block_num` se odnosi na broj blokova memorija koji se rezerviraju. `RMT_IDLE_LEVEL_LOW` postavlja izlazni signal na logičku 0 u zadanom stanju.

Varijabla `clockDivider` je ona definirana u prethodnom poglavlju.

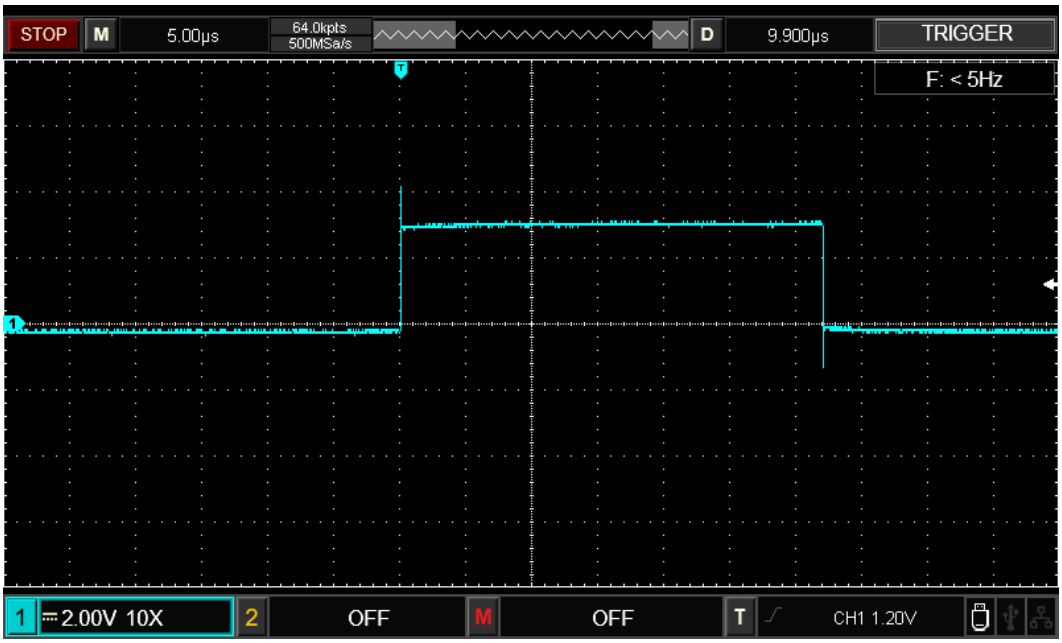
2.5.2.3. Definicije funkcija za okidanje željenih izlaznih impulsa

Definicije funkcija za okidanje željenih izlaznih impulsa su sljedeće:

```
void handleInterrupt1()
{
    Serial.println( "b1␣interrupted" );
    rmt_write_items( RMT_CHANNEL_0, &pulseItem1, 1, true );
}

void handleInterrupt2()
{
    Serial.println( "b2␣interrupted" );
    rmt_write_items( RMT_CHANNEL_0, &pulseItem2, 1, true );
}

void handleInterrupt3()
{
    Serial.println( "b3␣interrupted" );
    rmt_write_items( RMT_CHANNEL_0, &pulseItem3, 1, true );
}

void handleInterrupt4()
{
    Serial.println( "b4␣interrupted" );
    rmt_write_items( RMT_CHANNEL_0, &pulseItem4, 1, true );
}

void handleInterrupt5()
{
    Serial.println( "b5␣interrupted" );
```

```
    rmt_write_items( RMT_CHANNEL_0, &pulseItem5, 1, false );
}
```

Funkcija `Serial.println()` služi serijskoj komunikaciji između računala i mikroupravljača. Koristi se za nadzor rada komponente i uklanjanje grešaka. Funkcija `rmt_write_items()` odašilje signal definiran slogom `pulseItem` putem kanala `RMT_CHANNEL_0`, s jednim brojem puta s čekanjem da mikroupravljač završi sa slanjem.



Slika 2.6. Impuls trajanja 32 µs generiran RMT komponentom

2.5.2.4. Definicija setup() funkcije

`setup()` funkcija se okine jednom nakon resetiranja mikroupravljača. Koristi postavljanju izvršne okoline. Ona je u ovom radu definirana kao:

```
setup()
{
    Serial.begin( 115200 );
    Serial.println( "Hello" );

    pinMode( b1PIN, INPUT_PULLUP );
    pinMode( b2PIN, INPUT_PULLUP );
    pinMode( b3PIN, INPUT_PULLUP );
    pinMode( b4PIN, INPUT_PULLUP );
    pinMode( b5PIN, INPUT_PULLUP );
```

```
    setupRMTChannel( RMT_CHANNEL_0, ( gpio_num_t )signalPIN );

    pulseItem1.level0 = 1;
    pulseItem1.duration0 = 2;
    pulseItem1.level1 = 0;
    pulseItem1.duration1 = 2;

    pulseItem2.level0 = 1;
    pulseItem2.duration0 = 8;
    pulseItem2.level1 = 0;
    pulseItem2.duration1 = 2;
}
```

Posljednje 4 naredbe se ponavljaju još 3 puta, za varijable `pulseItem3`, `pulseItem4` i `pulseItem5` s vrijednostima za `duration0` redom 32, 128 i 256. Iz ove funkcije je moguće vidjeti da su pin-ovi povezani na gumbove povezani s naponom razine VCC preko "pull-up" otpornika. Postava "INPUT" nam omogućava čitanje njegovih digitalnih stanja. Ukoliko je digitalno stanje takvog pin-a logička 0, gumb povezan s njime je pritisnut. Također je vidljivo kako nije potrebno definirati tip pin-a za `signalPIN`, nego se on prosljeđuje u funkciju `setupRMTChannel`. Naredba `( gpio_num_t )` mijenja njegov tip iz cjelobrojnog zbog zahtjeva `rmt.h` biblioteke. Varijable `pulseItem` su slogovi tipa `rmt_item32_t`. Komponenta `level0` sloga tog tipa definira početnu logičku razinu impulsa. Komponenta `duration0` sloga tog tipa definira trajanje te logičke razine. Komponenta `level1` sloga tog tipa definira sljedeću logičku razinu impulsa. Komponenta `duration1` sloga tog tipa definira trajanje te logičke razine. Te 4 komponente zajedno definiraju redoslijed logičkih razina i njihovo trajanje kod digitalnog impulsa [15].

2.5.2.5. Definicija loop() funkcije

Funkcija loop() je ona koja se izvršava u beskonačnoj petlji. Ona je definirana kao:

```
void loop()
{
    if( ( ( millis() - lastCheck ) > 2500 ) )
    {
        if( !digitalRead( b1PIN ) )
```

```
        {
            handleInterrupt1();
        }
        else if( !digitalRead( b2PIN ) )
        {
            handleInterrupt2();
        }
        else if( !digitalRead( b3PIN ) )
        {
            handleInterrupt3();
        }
        else if( !digitalRead( b4PIN ) )
        {
            handleInterrupt4();
        }
        else if( !digitalRead( b5PIN ) )
        {
            handleInterrupt5();
        }

        lastCheck = millis();
    }
}
```

Funkcija `millis()` vraća broj milisekundi od resetiranja mikroprocesora [16]. Uvjet `( millis() - lastCheck ) > 2500` osigurava da je prošlo barem 2500 milisekundi od slanja posljednjeg digitalnog impulsa. Funkcija `digitalRead()` očitava digitalno stanje na pin-u proslijeđenog broja. Pomoću uskličnika se vraćena digitalna vrijednost negira. Tako dobivamo zadovoljenje uvjeta ako je gumb povezan s pin-om pritisnut jer tada digitalno stanje na njegovom ulazu iznosi 0. Zadnja naredba postavlja vrijednost varijable `lastCheck` na vremenski trenutak tik nakon slanja impulsa PSoC 5LP-u.

2.6. Serijska komunikacija od mikroupravljača do računala

U svrhu nadzora i testiranja sustava korišteni su kanali serijske komunikacije između korištenih mikrokontrolera i računala.

2.6.1. Serijska komunikacija PSoC 5LP-a

Serijska komunikacija od PSoC 5LP-a do računala se vršila putem USB priključka na pločici. Isti je prikazan na slici broj 2.2. pod oznakom "USB Tx". Kako bi se ta komunikacija omogućila potrebno je dodati Full Speed USB (akr. USBFS) komponentu u shemu komponenti PSoC 5LP-a, kao što je i prikazano na 2.1. [17].

U `main()` funkciji programskog koda PSoC 5LP-a se konfigurira rad USBFS komponente:

```
char str[ 100 ];

USBUART_Start( 0, USBUART_5V_OPERATION );
while( USBUART_GetConfiguration() == 0 ) {}
```

Nakon toga se po potrebi i zahtjevu mogu slati poruke putem USB priključka:

```
sprintf( str, "Poruka␣s␣cjelobrojnom␣vrijednosti:␣␣%d\r\n", 20 );
USBUART_PutString( str );
```

One se u računalu mogu dohvaćati na razne načine, a jedan od njih je korištenje računalne aplikacije Tera Term [7].

2.6.2. Serijska komunikacija Dasduina

Dasduino vrši svoju serijsku komunikaciju putem istog USB priključka putem kojeg ga se programira. Isto je i prikazano na slici broj 2.2. pod oznakom "USB Rx/Tx". Kako bi se ta komunikacija omogućila potrebno je u postavkama okruženja definirati parametar `monitor_speed`, koji je u ovom radu postavljena na 115200. Tada je potrebno u `setup()` funkciji inicijalizirati serijsku komunikaciju i definirati brzinu prijenosa:

```
Serial.begin( 115200 );
```

Nakon toga je moguće prenositi podatke kao na primjeru:

```
Serial.print( "Poruka␣s␣cjelobrojnom␣vrijednosti:␣");
Serial.println( 20 );
```

Poruke se dohvaćaju unutar terminala računalnog programa Visual Studio Code koristeći PlatformIO ekstenziju [5] [6].

3. Zaključak

U ovom je seminarskom radu razrađen sustav za generiranje digitalnih signala u trajanju od par stotina nanosekundi do par desetina mikrosekundi, filtriranje signala na samo one s trajanjem većim od 1 mikrosekunde i mjerenje trajanja filtriranih signala s greškom od $\pm$ 0,49 mikrosekundi.

Komponenta mikroupravljača PSoC 5LP-a Glitch Filter je korištena za filtriranje signala, komponenta Timer istog mikroupravljača je korištena za mjerenje trajanja filtriranog signala i u mikroprocesoru istog čipa je izveden sustav za dohvaćanje izmjerene vrijednosti i za njezino pretvaranje u mikrosekunde.

Komponenta mikroupravljača ESP32 zvana Remote Control Transciever je korištena za generiranje digitalnih signala u svrhu testiranja razvijenog sustava. Na ESP32 mikroupravljač su povezana 5 gumba s kojima se može birati trajanje odašiljanog signala.

Za oba je mikroupravljača uspostavljena mogućnost serijske komunikacije s računalom.

Sljedeći koraci razvoja digitalnog sustava uključuju obogaćivanje prikupljenih podataka rednim brojem kanala na kojem je signal generiran i bilježenje vremenskog trenutka prikupljana. Potrebno je razviti sustav pohranjivanja takvih obogaćenih podataka i serijsko sučelje za dohvaćanje istih.

Nadalje, potrebno je razviti analognu komponentu sustava i razraditi pristup prijelasku iz analogne u digitalnu domenu. Jedan od pristupa pretvorbi oscilacija piezoelektričnih elemenata u digitalne signale je pojačanje signala, generiranje envelope tog signala i korištenje komparatora za provjeru njene prisutnosti. Na taj bi se način oscilirajući signal iz piezoelektričnog elementa uspješno pretvorio u digitalni.

Literatura

[1] D. Oletić, J. Kundrata, B. Okorn, E. Emanović, i M. Katalenić, "Low-power electronic sensor systems for acoustic event detection (AEMEMS)", Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva (FER), srpanj 2023., pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://www.fer.unizg.hr/liss/aemems?@=30zho

[2] Infineon Technologies, "32-bit PSoC 5 LP – Arm Cortex-M3 Microcontroller", pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/32-bit-psoc-5-lp-arm-cortex-m3/

[3] Soldered Electronics, "Dasduino CONNECTPLUS", pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://soldered.com/product/dasduino-connectplus/?srsltid=AfmBOorjQuVffYampldao3zoQK7i5NKHzhNuJwiz42OXHY2qGDVIWkgq

[4] Infineon Technologies, "PSoC Creator - Software Development Kit", pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://www.infineon.com/cms/en/design-support/tools/sdk/psoc-software/psoc-creator/

[5] Microsoft, "Visual Studio Code", pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://code.visualstudio.com/

[6] PlatformIO, "PlatformIO - The Next Generation IDE for Embedded Development", pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://platformio.org/

[7] Tera Term Project, "Tera Term - Open-Source Terminal Emulator", pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://teratermproject.github.io/index-en.html

[8] Espressif Systems, “ESP32-WROVER-E / ESP32-WROVER-IE Datasheet”, pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf

[9] ——, “Arduino-ESP32 Documentation - Getting Started”, pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://docs.espressif.com/projects/arduino-esp32/en/latest/getting_started.html

[10] Infineon Technologies, “Infineon Component Timer V2.80 - Software Module Datasheets”, pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://www.infineon.com/dgdl/Infineon-Component_Timer_V2.80-Software%20Module%20Datasheets-v02_08-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ea72961273d

[11] ——, “PSoC 5LP CY8C58LP Family Datasheet v15.00”, pristupljeno: 2025-02-16. [Mrežno]. Adresa: https://www.infineon.com/dgdl/Infineon-PSoC_5LP_CY8C58LP_Family_Datasheet_Programmable_System-on-Chip_(PSoC_)-DataSheet-v15_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0ec547013ab9

[12] Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave, “Zlouporaba ranjivosti računalnih sustava - prekoračenje kapaciteta cijelog broja”, pristupljeno: 2025-02-17. [Mrežno]. Adresa: http://sigurnost.zemris.fer.hr/ns/malware/2007_zelanto/integer.html

[13] Infineon Technologies, “Infineon Component Glitch Filter V2.0 Software Module Datasheets”, pristupljeno: 2025-02-17. [Mrežno]. Adresa: https://www.infineon.com/dgdl/Infineon-Component_Glitch_Filter_V2.0-Software%20Module%20Datasheets-v02_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0e886222183c

[14] ——, “Infineon Component Sync V1.0 - Software Module Datasheets”, pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://www.infineon.com/dgdl/Infineon-Component_Sync_V1.0-Software%20Module%20Datasheets-v01_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0e91a8d51df8

[15] Espressif Systems, “ESP-IDF API Reference: RMT (Remote Control)”, pristupljeno: 2025-02-15. [Mrežno]. Adresa: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/rmt.html

[16] Arduino Documentation, “millis() | Arduino Documentation”, pristupljeno: 2025-02-18. [Mrežno]. Adresa: https://docs.arduino.cc/language-reference/en/functions/time/millis/

[17] Infineon Technologies, “Infineon Component USBFS V3.2 - Software Module Datasheets”, pristupljeno: 2025-02-18. [Mrežno]. Adresa: https://www.infineon.com/dgdl/Infineon-Component_USBFS_V3.2-Software%20Module%20Datasheets-v03_02-EN.pdf?fileId=8ac78c8c7d0d8da4017d0e818d2f1332