# Dokumentacija implementacije

## 1. Uvod

Ovaj dokument pruža tehničko objašnjenje implementacije verzije igre "Keep Talking and Nobody Explodes" na  Pi Pico W platformi. Igra uključuje nekoliko mini igara koje igrači moraju riješiti koristeći različite hardverske komponente.

## 2. Arhitektura Sistema

Arhitektura sistema uključuje dva Pi Pico W koji upravljaju različitim hardverskim komponentama. Komunikacija između komponenata i kontrolera ostvaruje se pomoću GPIO pinova, a između dva Pi Pica preko MQTT veze.

### 2.1 Dijagram Arhitekture

Arhitektura sistema se sastoji od sljedećih komponenti:

- **2 Pi Pico W**: Glavni kontroler sistema.
- **9 Buttona**: Korišteni za različite unose u igri.
- **2 LED DOT Matrix Displeja**: Prikazivanje informacija vezanih za igru.
- **Potenciometar**: Korišten za unos analognih vrijednosti.
- **Enkoder**: Korišten za unos rotacionih vrijednosti.
- **Matrica Tastatura**: Korištena za unos različitih sekvenci.
- **Sedmosegmenti displej:** Prikaz brojača datog reotacionim enkoderom.
- **4 LED diode**: Signalizacija različitih stanja igre.

## 3. Komponente Sistema

Pi Pico W master je srce sistema, zadužen za upravljanje svim komponentama i izvršavanje logike igre. Pi Pico W slave služi za dodavanje još potrebnih pinova.

### 3.1 Buttoni

Četiri buttona su povezani na GPIO pinove Pico W i koriste se za unos različitih komandi u igri. Potrebni za igre:
Button press-1
Lavirint-4
Simon says-3
Morse kod-1

### 3.2 LED DOT Matrix Displeji

Dva LED DOT Matrix displeja su povezana na Pico W i koriste se za prikazivanje različitih informacija i stanja igre. Potrebne za igru:
Button press-1
Lavirint-1

### 3.3 Sedmosegmentni Displej

Sedmosegmentni displej je korišten za prikazivanje brojeva i povezan je na GPIO pinove. Potrebne za igru Decode.

### 3.4 Potenciometar

Potenciometar je povezan na analogni ulaz Pi Pico W i koristi se za unos analognih vrijednosti u igri. Potrebno za igru PotenciometarGame

### 3.5 Enkoder

Rotacioni enkoder je povezan na Pi Pico W i koristi se za unos rotacionih vrijednosti. Potrebno za igr Decode.

### 3.6 Matrična Tastatura

Matrična tastatura je povezana na GPIO pinove i koristi se za unos sekvenci. Potrebno za igru Pin kod.

### 3.7 LED diode

LED diode su povezane na GPIO pinove i koriste se za signalizaciju različitih stanja igre. Potrebno za igre:
Simon says-3
Morse-1

### 3.8 RGB Dioda

RGB Dioda se koristi za prikaz stanja igre. Konkretno postoji 8 mogućih igara koje se jedinstveno RGB diodom mogu razaznati i adekvatno se može konsultirati priručnik da se bomba deaktivira.

## 4. Logika i Algoritmi

Glavna logika se nalazi unutar naših main programa. Pošto imamo dva Pi Pico W uređaja oni će međusobno komunicirati da znaju ukoliko je već jedna igra u toku ili ne. Razlog korištenja dva Pico W jeste taj što za implementaciju datog projekta potrebno nam je bilo više pinova nego što se nalazi na smo jednom.

**NAPOMENA:**

**Morse kod i Potenciometar igrica su izbaćeni jer je teško uočiti kada je igrač napravio grešku. Također je Wiretcutter igrica izbaćena jer je bilo potrebno previse pinova za implementaciju.**

### 4.1 Button

Ovaj kod implementira igru na mikrokontroleru koristeći MAX7219 LED matriks displej, taster i timer. U igri se na LED displeju prikazuje određeni broj tačkica, a korisnikov zadatak je da pritisne i drži taster onoliko sekundi koliko je tačkica prikazano. Ako korisnik pritisne taster i drži ga ispravno, sve LED tačkice se upale. Ako korisnik pogriješi u brojanju ili predugo čeka, na displeju se prikazuje znak "X". Igra koristi timer za periodično ažuriranje prikaza tačkica i provjerava korisnikov unos kako bi se utvrdilo je li tačan ili ne.

### 4.2 Decoder

Ovaj kod koristi dekoder za upravljanje brojačem koji prikazuje vrijednost na 7-segmentnom displeju. Inicijalizacija uključuje definisanje ulaznih i izlaznih pinova, postavljanje tajmera za

redovno osvježavanje prikaza brojeva, i podešavanje prekida za rukovanje ulaznim signalima sa dekodera (CLK i DT pinovi). Kada se okine prekid na CLK ili DT pinu, vrijednost brojača se ažurira: CLK prekid povećava broj, dok DT prekid smanjuje broj, u zavisnosti od stanja drugog pina. Kada korisnik pritisne taster (SW pin), provjerava se da li trenutna vrijednost brojača odgovara zadatom rješenju. Ako je tako, igra se završava sa uspjehom; u suprotnom, broj grešaka se povećava. Timer periodično osvježava prikaz na 7-segmentnom displeju, prikazujući trenutnu vrijednost brojača. Cilj igre je da korisnik pomoću dekodera postigne tačan broj i potvrdi ga pritiskom na taster.

### 4.3 Lavirint

Ovaj kod implementira igru lavirinta na mikrokontroleru koristeći MAX7219 LED matrični displej, taster dugmiće i tajmere. Inicijalizacija postavlja ulazne i izlazne pinove za taster dugmiće i SPI za komunikaciju sa MAX7219. Takođe se inicijalizuje i tajmer za periodično osvježavanje prikaza igrača na displeju. Generisanje lavirinta vrši se nasumično, kreirajući staze i zidove unutar 8x8 matrice. Postoje predefinisane početne i završne tačke lavirinta. Cilj igre je da korisnik upravlja igračem pomoću tastera kako bi stigao do završne tačke lavirinta bez sudara sa zidovima. Kada igrač stigne do završne tačke, igra se završava uspešno. Klasa ima metode za prikaz igrača na displeju, reakciju na pritisak tastera za kretanje u četiri moguća pravca (gore, dole, lijevo, desno), provjeru da li je igrač stigao do završne tačke, kao i metod za završetak igre kada se završna tačka dostigne. Dodatno, postoji statička metoda za kreiranje direktne putanje od početne do završne tačke lavirinta, kao i metoda za ispis trenutnog stanja lavirinta u konzoli. Ovaj kod omogućava interaktivnu igru na mikrokontroleru, koristeći jednostavne komponente poput LED matričnog displeja i tastera, demonstrirajući upotrebu periferija i logike programiranja za igre.

### 4.4 Pin kod

Klasa `MatricnaGame` omogućava interaktivnu igru sa matricom tastera na mikrokontroleru. U igri korisnik treba da unese određenu šifru koristeći matricu tastera, a cilj je da unese tačnu šifru prije nego što istekne vrijeme ili da ne napravi previše grešaka. Metoda `generisiSifruMatricna` postavlja trenutnu šifru koju korisnik treba da unese, na osnovu trenutnog stanja (state) koje se prosljeđuje. Metoda `postaviSifru` je postavljena kao reakcija na prekid (IRQ) koji se aktivira kada korisnik pritisne taster na matrici. Ona obrađuje unos korisnika, provjerava da li je unijeta šifra tačna ili greška, i na osnovu toga upravlja tokom igre. Konstruktor `__init__` inicijalizuje sve potrebne parametre za igru: pinove za izlaz (izlazMatricna) koji kontrolišu redoslijed aktivnih redova matrice, pinove za ulaz (ulazMatricna) koji čitaju stanje tastera, i postavlja početno stanje igre na osnovu proslijeđenog stanja. Dodatno, postoje metode `solved`, `get_sifra` i `get_strikes` koje vraćaju informacije o trenutnom stanju igre (da li je igra rešena, trenutna šifra, broj grešaka). Ovaj kod demonstrira upotrebu tajmera, obrade prekida, rad sa digitalnim ulazima i izlazima na mikrokontroleru kako bi se implementirala jednostavna igra sa matricom tastera.

### 4.5 Morse kod

Ovaj kod implementira sistem za prikazivanje Morseovog koda pomoću LED diode i analognog senzora. Glavna klasa je `Morse`, koja pruža funkcionalnost za dekodiranje i prikazivanje unaprijed definiranih poruka u Morseovom kodu na LED diodi. Klasa sadrži statičke definicije za tablicu Morseovih znakova (`morse`) i definira različite Morseove poruke s pripadajućim frekvencijama i dužinama impulsa (`code_book`). U metodi `__init__`, kada se instancira objekt klase `Morse`, definira se željena poruka prema indeksu `state` u `code_book`, te se inicijaliziraju pinovi za LED diodu i analogni senzor. Poruka se zatim konvertira u niz impulsa (svjetla ili tame) koji će se prikazivati na LED diodi prema definiranom Morseovom

kodu. Objekt prati stanje modulacije pomoću tri različita vrijemenska okidača (`Timer`): `timer` za periodično prikazivanje Morseovih impulsa na LED diodi, `sampler` za uzorkovanje analognog signala s frekvencijom definiranom u `rate`, te `comparator` za usporedbu uzorkovanog signala s ciljnim obrascem (definiranim u `target`). Metoda `solved()` provjerava je li modulacija riješena usporedbom sadržaja `buffer`a (uzorkovani analogni signal) s ciljnim obrascem `target`. Ako je modulacija riješena (tj. ako je uzorkovani signal dovoljno blizak ciljnom obrascu), deaktiviraju se svi timski okidači (`timer`, `sampler`, `comparator`) i postavlja se zastavica `self.solved` na `True`. Ovaj sistem omogućuje strojno dekodiranje i prikazivanje Morseovog koda, koristeći analogne senzore za detekciju svjetlosnih uzoraka, čime se simulira komunikacija putem Morseovog koda preko svjetla.

### 4.6 Potenciometar

Ovaj kod implementira jednostavnu igru s potenciometrom na mikrokontroleru. Cilj igre je detektirati određenu vrijednost potenciometra koristeći ADC interfejs za pretvorbu analognog signala u digitalni oblik. Klasa `Potenciometar` ima konstruktor koji inicijalizira igru postavljanjem ciljane vrijednosti potenciometra, koja je unaprijed definirana u listi `potencimetar_values`. Ovisno o odabranom `state` parametru, odabire se odgovarajuća ciljna vrijednost. Igra koristi ADC objekt za čitanje trenutne vrijednosti s potenciometra, koji je spojen na određeni pin na mikrokontroleru. Timer (`timerPot`) se koristi za periodično izvršavanje funkcije `PotenciometarGame`, koja provjerava trenutnu vrijednost potenciometra. U metodi `PotenciometarGame`, trenutna vrijednost potenciometra se uspoređuje s ciljnom vrijednosti `self.sifra`. Ako je apsolutna razlika između očitane vrijednosti i ciljne vrijednosti manja od zadane tolerancije (`eps`), igra se riješava. Timer se zaustavlja, a zastavica `self.solved` postavlja na `True`, što označava da je igra završena uspješno. Metoda `solved` omogućuje vanjskom kodu da provjeri je li igra riješena, dok `get_sifra` vraća ciljanu vrijednost potenciometra za referencu ili prikaz korisniku. Ovaj sistem omogućuje jednostavno i automatizirano praćenje i rješavanje igre s potenciometrom, što može biti korisno u edukativnim ili demonstracijskim svrhama za učenje o radu s analognim senzorima i digitalnim sučeljima mikrokontrolera.

### 4.7 Simon says

Ovaj kod implementira igru "Simon Says" na mikrokontroleru, simulirajući igru u kojoj korisnik mora ponoviti uzorak paljenja i gašenja LED-ica prema predefiniranoj sekvenci. Klasa `SimonSays` ima konstruktor koji inicijalizira igru s dva lista pinova: `button_pins` za tastere koji se koriste za unos, i `led_pins` za LED-ice koje će se paliti i gasiti. Također prima listu `code` koja definira sekvencu koju korisnik mora ponoviti. LED-ice i tasteri inicijaliziraju se kao Pin objekti s odgovarajućim ulaznim (`Pin.IN`) I izlaznim (`Pin.OUT`) postavkama. `code` lista čuva sekvencu koju treba pogoditi. Za svaki taster postavlja se prekid (`irq`) koji reagira na uzlaznu (rising) promjenu signala, pozivajući `button_handler` metodu za obradu pritiska tastera. Implementirana je i debouncing logika kako bi se izbjegli lažni prekidi prilikom pritiska.
Periodični tajmeri (`Timer`) se koriste za upravljanje igrom:

- `show_timer` periodično pokreće funkciju `turn_on_led` kako bi se prikazala sekvencija svjetlosnih signala korisniku.

- `prikazujSekvencu` tajmer se koristi za periodično prikazivanje cijele sekvence LED-ica, s pauzama između sekvenci.

Metoda `turn_on_led` upravlja redoslijedom paljenja i gašenja LED-ica prema definiranoj sekvenci. Svaka LED-ica se pali i gasi prema trenutnom indeksu u `code` listi, omogućujući vizualno prikazivanje sekvence korisniku. Metoda `checkInput` provjerava unos korisnika nakon što je završena prikazana sekvencija. Ako korisnik ispravno ponovi sekvencu, ispisuje igra se završava i sve LED-ice se upale. Ako se pogriješi, korisniku se omogućuje ponovni pokušaj, a broj pogrešaka se povećava. Metoda `ShowFailure` koristi se za vizualni efekt u slučaju pogrešnog unosa, gdje se LED-ice brzo pale i gase kako bi signalizirale grešku prije ponovnog pokušaja. Ovaj kod pruža interaktivno iskustvo igre "Simon Says" na mikrokontroleru, koristeći LED-ice i tastere za unos, što omogućuje praktičnu primjenu u obuci ili demonstracijama za rad s digitalnim ulazima i izlazima.

### 4.8 Wirecutter

Klasa `WireCutter` simulira sistem za rezanje žica na mikrokontroleru. Inicijalizira se s izlaznim i ulaznim pinovima te listom pinova koji se smiju rezati. Parametar `inOrder` određuje redoslijed rezanja žica. Svi pinovi se inicijaliziraju kao objekti `Pin` za odgovarajuće ulazne ili izlazne funkcije. Izlazni pinovi se inicijalno postavljaju na `on()`. Tajmer `timer` periodično poziva funkciju `CheckWires` svakih 100 ms, koja provjerava stanje žica i detektira događaje na ulaznim pinovima. Svaki ulazni pin ima postavljen prekid (`irq`) na padajući rub signala, što aktivira metodu `CheckWires` za daljnju provjeru. Tajmer `checker` također periodično poziva funkciju `checkSolved` svakih 100 ms, koja provjerava jesu li sve žice označene u `valid_to_cut` listi već rezane. Metoda `CheckWires` upravlja logikom rezanja žica: provjerava redoslijed i ispravnost te bilježi pogreške ako žice nisu rezane prema očekivanjima. Metoda `checkSolved` provjerava jesu li sve žice iz `valid_to_cut` već rezane i postavlja zastavicu `solved` na `True` ako jesu. Ovaj kod pruža simulaciju upravljanja i praćenja rezanja žica na mikrokontroleru, koristeći digitalne ulaze i izlaze te periodično provjeravanje stanja sistema.

### 4.9 Pico W slave

Pico W slave uređaj inicijalizira globalne varijable za praćenje stanja igre, spremnosti (`main_ready`), trenutnog stanja (`state`) i izvršavanja igre (`run`). Definira i pinove za različite komponente: tipke i LED diode za igru Simon Says, matricu tastatura, potenciometar, pinove za Morseov kod, pin za igru s tasterom i pinove za igru s žicama. Pomoćna funkcija `subscribe` obrađuje MQTT poruke: postavlja `main_ready` kada je uređaj spreman, ažurira `state` kada stignu poruke o stanju igre, te zaustavlja igru na poruke o završetku igre ("win" ili "lose"). Funkcija `check` periodično provjerava stanje modula u igri, broji riješene module i prati greške. Informacije o broju grešaka i riješenim modulima šalje putem MQTT-a. Nakon uspostave Wi-Fi veze i MQTT klijenta, program čeka da glavni uređaj označi spremnost (`main_ready`). U glavnoj petlji čeka da glavni uređaj pokrene igru slanjem odgovarajuće MQTT poruke (`run`). Moduli igre se inicijaliziraju kao objekti različitih klasa (SimonSays, MatrixGame, Potenciometar, Morse, Button, Wires) s odgovarajućim parametrima. Glavni tajmer je postavljen da periodično poziva funkciju `check` kako bi pratio stanje igre tokom trajanja. U glavnoj petlji programa ispisuje se "game is running..." i čeka se signal o trajanju igre putem MQTT-a.

### 4.10 Pico W master

Pico W master uređaj koordinira upravljanje stanjem igre, komunikaciju putem MQTT-a i procese donošenja odluka. Ovisi o Pico W slave uređaju za dodatne GPIO pinove kako bi se komuniciralo s različitim komponentama igre. Slave uređaj rukuje specifičnim interakcijama u igri i šalje svoje stanje i napredak natrag master uređaju putem MQTT-a. Ova podjela zadataka omogućava distribuiranu obradu i efikasno upravljanje složenom logikom igre preko više uređaja. Pico W master uređaj inicijalizira globalne varijable za upravljanje stanjem igre i komunikaciju s drugim uređajima putem MQTT protokola. Navode se i pinovi koje Pi Pico W koristi za povezivanje s komponentama poput labirintskih tipki, displeja, enkodera i segmentnog displeja. Pomoćne funkcije uključuju `subscribe`, koja prati MQTT poruke za ažuriranje globalnih varijabli o prisustvu drugih igrača i stanju igre. Funkcija `rgb_randomiser` generira slučajne RGB vrijednosti za vizualne efekte ili indikacije u igri. Funkcija `explode` aktivira se kada igra završi zbog prekoračenja dozvoljenog broja grešaka ili rješenja svih modula. Zaustavlja glavni tajmer i šalje poruku "lose" putem MQTT-a. Funkcija `check` periodično provjerava riješene module i broj grešaka za donošenje odluke o završetku igre (pobjeda ili poraz). Funkcija `publish_state` objavljuje trenutno stanje igre putem MQTT-a, uključujući riješene module i maksimalan broj dopuštenih grešaka. Nakon postavljanja Wi-Fi veze i MQTT klijenta, program čeka registraciju slave uređaja i aplikacije na telefonu prije početka glavne igre. U glavnoj petlji igre, program inicijalizira početno stanje i maksimalan broj grešaka, periodično provjerava stanje igre sve dok se ne završi.

## 5. Komunikacija

Za potrebe komunikacije smo koristili lokalno dignut *mosquitto MQTT broker* koji je bio podešen za velike keepalive vrijednosti. MQTT koristimo za multipoint komunikaciju između oba raspberry pico W i telefona na kom je pokrenuta aplikacija za monitoring bombe. Svaka tema ima specifičnu namjenu i asocirane aktere koji je slušaju i koji na nju šalju poruke.

| Tema | Publisher | Subscribers | Opis |
|---|---|---|---|
| katane/main_ready | Pico W Master | Aplikacija, Pico W Slave | Master Pico je spreman da se igra započne. |
| katane/slave_present | Pico W Slave | Pico W Master | Slave Pico javlja da je prisutan, ovako Master može znati kad je sistem spreman. |
| katane/app_present | Aplikacija | Pico W Master | Aplikacija javlja da je prisutna, ovako Master može znati kad je sistem |

| | | | |
|---|---|---|---|
| | | | spreman. |
| katane/game_start | Aplikacija | Pico W Master | Korisnik želi započeti igru. |
| katane/game_state | Pico W Master | Aplikacija, Pico W Slave | Master javlja randomizirano stanje i broj dozvoljenih pokušaja. Slave na osnovu stanja pokreće igru, dok aplikacija prikazuje maksimalni broj grešaka. |
| katane/slave_solved | Pico W Slave | Pico W Master | Slave javlja broj riješenih modula na njemu |
| katane/slave_strike | Pico W Slave | Pico W Master | Slave javlja broj greški na njemu |
| katane/strikes | Pico W Master | Aplikacija | Master javlja aplikaciji ukupni broj greški na bombi. |
| katane/solved | Pico W Master | Aplikacija | Master javlja aplikaciji ukupni broj riješenih modula |
| katane/time | Pico W Master | Aplikacija | Master javlja aplikaciji ukupni broj sekundi proteklih na bombi. |
| katane/game_over | Pico W Master | Aplikacija, Pico W Slave | Master Pico javlja da je igra završena, Slave Pico završava sve module i aplikacija prikazuje adekvatni rezultat igre. |

## 6. Zaključak

Ova dokumentacija pruža pregled ključnih tehničkih aspekata implementacije verzije igre "Keep Talking and Nobody Explodes" na  Pi Pico W platformi. Detaljniji uvidi su dostupni u izvornom kodu projekta.