Dokumentacija implementacije

# 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.

# 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 sledećih komponenti:

* **2 Pi Pico W**: Glavni kontroler sistema.
* **4 Buttona**: Korišćeni za različite unose u igri.
* **2 LED DOT Matrix Displeja**: Prikazivanje informacija vezanih za igru.
* **Sedmosegmentni Displej**: Prikaz preostalog vremena dok bomba ne eksplodira.
* **Potenciometar**: Korišten za unos analognih vrednosti.
* **Enkoder**: Korišten za unos rotacionih vrednosti.
* **Matrica Tastatura**: Korišćena za unos različitih sekvenci.
* **LED diode**: Signalizacija različitih stanja igre.

# Komponente Sistema

Pi Pico W je srce sistema, zadužen za upravljanje svim komponentama i izvršavanje logike igre.

## 3.2 Buttoni

Četiri buttona su povezani na GPIO pinove Pico W i koriste se za unos različitih komandi u igri.

## 3.3 LED DOT Matrix Displeji

Dva LED DOT Matrix displeja su povezana na Pico W i koriste se za prikazivanje različitih informacija i stanja igre.

## 3.4 Sedmosegmentni Displej

Sedmosegmentni displej je korišćen za prikazivanje brojeva i povezan je na GPIO pinove.

## 3.5 Potenciometar

Potenciometar je povezan na analogni ulaz Pi Pico W i koristi se za unos analognih vrednosti u igri.

## 3.6 Enkoder

Rotacioni enkoder je povezan na Pi Pico W i koristi se za unos rotacionih vrednosti.

## 3.7 Matrična Tastatura

Matrična tastatura je povezana na GPIO pinove i koristi se za unos sekvenci.

## 3.8 LED diode

LED diode su povezane na GPIO pinove i koriste se za signalizaciju različitih stanja igre.

# 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.

## Pico W slave

from machine import Pin

import time

import network

from umqtt.robust import MQTTClient

import ujson

from simonsays import SimonSays

from morse import Morse

from matricna import MatricnaGame

from potenciometar import Potenciometar

from button import Button

from wires import Wires

#slave

import network

import time

from machine import Pin

from umqtt.robust import MQTTClient

import ujson

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Globalne varijable\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

main\_ready = False

state = 0

run = False

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Pinouts\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

simon\_buttons = [8, 9, 12]

simon\_leds = [13, 14, 16]

pins\_izlazna = [0, 1, 2, 3] # za matricnu tastaturu

pins\_ulazna = [4, 5, 6, 7] # mislim da su ovo redovi po sjecanju

pin\_potenciometar = [28]

morse\_pins = [17, 27] # jedna ledica za morse kod prikaz i ulaz signala

button\_game\_pins = [9] # jedno dugme, button zauzima vec ss=15, sck=10, mosi=11!

wire\_pins = [18, 19, 20, 21, 22, 26] # sta god ali 6

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Pomocne funkcije\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

def subscribe(topic, msg):

global run, main\_ready, state, main\_timer

if topic == b'katane/main\_ready' and msg == b'1':

main\_ready = True

if topic == b'katane/game\_state':

parsed = ujson.loads(msg)

state = int(parsed["state"])

run = True

if topic == b'katane/game\_over' and msg in (b'win', b'lose'):

main\_timer.deinit()

run = False

def check(t):

global moduli\_pool, strikes, solved

strikes = 0

new\_pool = []

for m in moduli\_pool:

if not m.solved:

new\_pool.append(m)

else:

solved += 1

strikes += m.get\_strikes()

moduli\_pool = new\_pool

mqtt\_conn.publish(b'katane/slave\_strike', str(strikes).encode('ascii'))

if solved != 0:

mqtt\_conn.publish(b'katane/slave\_solved', str(solved).encode('ascii'))

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Spajanje na Wifi\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

network\_name = 'Wifi'

network\_password = 'password'

nic = network.WLAN(network.STA\_IF)

nic.activate(True)

nic.connect(network\_name, network\_password)

while not nic.isconnected():

print('Cekam vezu')

time.sleep(1)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Game setup\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

mqtt\_conn = MQTTClient(client\_id='slavepico', server='broker.hivemq.com',user='',password='',port=1883)

mqtt\_conn.set\_callback(subscribe)

mqtt\_conn.connect()

mqtt\_conn.subscribe(b"katane/#")

while not main\_ready:

mqtt\_conn.wait\_msg()

while not run:

mqtt\_conn.wait\_msg()

moduli\_pool = [SimonSays(state, simon\_buttons, simon\_leds),

MatricnaGame(state, pins\_izlazna, pins\_ulazna),

Potenciometar(state, pin\_potenciometar),

Morse(state, morse\_pins),

Button(state, button\_game\_pins),

Wires(state, wire\_pins)]

main\_timer = Timer(period=100, mode=Timer.PERIODIC, callback=check)

while run:

print('game is running...')

time.sleep(1)

U kodu iznad se nalazi main za slave Pico W na kojem vidimo da imamo šest igara našeg sistema gdje smo prvenstveno deklarisali neke globalne varijable koje će nam trebati za provjeru stanja Sistema tačnije da li se pokreće već neka igra ili ne, da li smo se uspjesno spojili na sve mreže i da li ovaj Pico radii li drugi. Nakon toga deklarisani su svi pinovi koje ćemo koristiti na slave-u. U kodu ispod ovoga smo deklarisali neke pomoćne funkcije koje će nam trebati za spajanje sa našom aplikacijom preko mqtt. Nakon toga se spajamo na WiFi i postavljamo i započinjemo igru.

## Pico W master

from machine import Pin

import time

import network

from umqtt.robust import MQTTClient

import ujson

from labirint import Labirint

from decoder import Decoder

#master

# spoji se na internet

import network

import time

from machine import Pin

from umqtt.robust import MQTTClient

import ujson

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Glob varijable\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

slave\_present = False

app\_present = False

game\_start = False

solved = 0

strike = 0

solved\_slave = 0

strike\_slave = 0

main\_timer = Timer(-1)

game\_running = False

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Pinouts\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

labirint\_pins = [21,22,26,27] # tasteri labirinta lijevo, gore, dole, desno

display\_pins = [0, 18,19,20] # labirint display: spi, SCK, MOSI, CSm

encoder\_pins = [0,1,2] # jasno clk, dt, sw

segmenti = [4,5,6,7] #

digits = [8,9,10,11,12,13,14,15] #

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Pomocne funkcije\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

def subscribe(topic, msg):

global slave\_present, app\_present, game\_running, game\_start, strike\_slave, solved\_slave

if topic == b'katane/slave\_present' and msg == b'1':

slave\_present = True

if topic == b'katane/app\_present' and msg == b'1':

app\_present = True

if topic == b'katane/game\_start' and msg == b'1':

game\_running = True

game\_start = True

if topic == b'katane/slave\_solved':

solved\_slave = int(msg)

if topic == b'katane/slave\_strike':

strike\_slave = int(msg)

def rgb\_randomiser():

return random.randint(0, 7), random.randint(0,3)

def explode(t):

global game\_running

mqtt\_conn.publish(b'katane/game\_over', b'lose')

main\_timer.deinit()

game\_running = False

def check(t):

global solved, strike, game\_running

solved = 0

for m in moduli:

solved += m.solved

strike += m.get\_strikes()

if strike + strike\_slave > max\_strike:

explode(t)

elif solved + solved\_slave == 7:

count\_down.deinit()

mqtt\_conn.publish(b'katane/game\_over', b'win')

main\_timer.deinit()

print('ggwp')

game\_running = False

def publish\_state(state, max\_strike):

json = '{ "state": ' + str(state) + ',"max\_strike": ' + str(max\_strike) + ' }'

mqtt\_conn.publish(b'katane/game\_state', json.encode('ascii'))

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Spajanje na Wifi\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

network\_name = 'Wifi'

network\_password = 'password'

nic = network.WLAN(network.STA\_IF)

nic.activate(True)

nic.connect(network\_name, network\_password)

while not nic.isconnected():

print('Cekam vezu')

time.sleep(1)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Game setup\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

mqtt\_conn = MQTTClient(client\_id='masterpico', server='broker.hivemq.com',user='',password='',port=1883)

mqtt\_conn.set\_callback(subscribe)

mqtt\_conn.connect()

mqtt\_conn.subscribe(b"katane/#")

while not slave\_present:

print('cekam slave da se javi')

mqtt\_conn.wait\_msg()

while not app\_present:

print('cekam aplikaciju sa telefona da se javi')

mqtt\_conn.wait\_msg()

mqtt\_conn.publish(b'katane/main\_ready', b'1') # javi telefonu da je main spreman da pocne

while not game\_start:

mqtt\_conn.wait\_msg()

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Game Run\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_#

state, max\_strike = rgb\_randomiser()

publish\_state(state, max\_strike)

count\_down = Timer(period=3 \* 1000 \* 60, mode=Timer.ONE\_SHOT, callback=explode)

moduli = [Labirint(state, labirint\_pins), Decoder(encoder\_pins, segmenti, digits, state)]

main\_timer.init(period=100, mode=Timer.PERIODIC, callback=check)

while game\_running:

print('igra u toku...')

time.sleep(0.5)

U kodu iznad se nalazi naš Pi Pico W master koji ima samo dvije vrste igara. Princip rada mastera je isti ko i slave samo što na masteru imamo Timer koji sve prati i na njemu se bilježi koje smo igre zavrsili koje nismo.

## Ostali programi

Pored main programa napravili smo 10 klasa za svaku igru koji omogućavaju da jednostavnim pozivom neke od klasa započne tu igru. Date klase su u odvojenim fajlovima pa ih je potrebno importovati kao biblioteke. (DA LI DA OBJAŠNJAVAM SVAKU IGRU NEĆU KODOVE STAVLJATI)

# 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.