Univerzitet u Sarajevu
Elektrotehnički fakultet Sarajevo
Odsjek za računarstvo i informatiku

Dokumentacija implementacije

Selma Ličina 19148
Selma Ljuhar 19177
Emir Kalajdžija 19059
Kerim Halilović 19215

Biblioteke

Koristili smo biblioteke sa linkova :

- jeffmer/micropython-ili9341: Micropython Driver for ILI9341 display (github.com)
- rdagger/micropython-ili9341: MicroPython ILI9341Display & XPT2046 Touch Screen Driver (github.com)

Za korištenje ovih biblioteka na razvojni sistem picoETF prebacili smo fajlove :

- ili934xnew.py
- glcdfont.py
- tt14.py
- tt24.py
- tt32.py
- ili9341.py

Kako bi se postigla bolja performansa pri iscrtavanju automobila, unutar fajla ili9341.py dodana je nova metoda "draw_car" koja omogućava brže izvršavanje u usporedbi s uključivanjem istog koda unutar glavnog programa.

```
def draw_car(self,car_position):
    # auto
    self.fill_rectangle(car_position*48+8, 320-48, 32, 48, color565(255,0,0))
    # lijevi far
    self.fill_rectangle(car_position*48+11, 320-45, 8, 2, color565(255,255,51))
    # desni far
    self.fill_rectangle(car_position*48+29, 320-45, 8, 2, color565(255,255,51))
    # gornja lijeva guma
    self.fill_rectangle(car_position*48+3, 320-41, 5, 14, color565(0, 0, 0))
    # gornja desna guma
    self.fill_rectangle(car_position*48+40, 320-41, 5, 14, color565(0, 0, 0))
    # donja lijeva guma
    self.fill_rectangle(car_position*48+3, 320-14, 5, 14, color565(0, 0, 0))
    # donja desna guma
    self.fill_rectangle(car_position*48+40, 320-14, 5, 14, color565(0, 0, 0))
```

Također dodana je i funkcija draw_vlines koja nam pomaže pri crtanju traka na cesti

```
def draw_vlines(self, coords, color):
        for i in range(0, len(coords)):
            x1, y1 = coords[i]
            self.draw_vline(x1, y1, 20, color)
```

Funkcija za joystick

Ova funkcija čita trenutno stanje joysticka i vraća informacije o njegovoj poziciji.

Omogućava korištenje joysticka za upravljanje igrom i za biranje ponuđenih opcija na ekranu.

```
direction = 0               # 1 2 3 4 (Gore , dole ,lijevo ,desno)

def joystickDirections():
    global direction
    x_value = X_JOYSTICK.read_u16()
    y_value = Y_JOYSTICK.read_u16()
    if(y_value<2000):
        direction=1         #gore
    elif(y_value>50000):
        direction=2         #dole
    if(x_value<2000):
        direction=3         #lijevo
    elif(x_value>50000):
        direction=4         #desno
```

Funkcije za dizajniranje prikaza na ekranu

Za postizanje željenog izgleda igre na ekranu, korišteno je nekoliko jednostavnih metoda za crtanje dobijenih iz biblioteka za displej , neke od tih metoda su :

- erase
- set_color
- fill_rectangle
- draw_rectangle
- set_font
- set_pos
- print
- …

Nazivi naših funkcija :

```
# Dizajn pocetnog ekrana
def makeHomeScreenDesign():···

# Dizajn ekrana tokom igre
def makeGameScreenDesign():···

# Dizajn ekrana koji se prikaze nakon zavrsetka igre u singleplayer-u
def makeGameOverScreenDesign():···

# Dizajn ekrana pri slanju zahtjeva za multiplayer
def makeSentRequestScreenDesign():···

# Dizajn ekrana pri primaju zahtjeva za multipalyer igru
def makeRecivedRequestScreenDesign():···

# Dizajn ekrana za ukucavanje pin-a
def makePinScreenDesign():···

# Dizajn ekrana koji se prikaze nakon zavrsetka igre u multiplayer-u
def makeMultiplayerResultScreenDesign():···

# Dizajn ekrana koji se prikaze igracu koji prvi zavrsi igru u multiplayer-u
def waitingForOtherPlyerScreenDesign():···
```

Jedan primjer naše takve funkcije je :

```
# Dizajn pocetnog home ekrana
def makeHomeScreenDesign():
    display.erase()
    display.set_color(WHITE,BLACK)
    display.fill_rectangle(0,0,SCR_HEIGHT,SCR_WIDTH,BLACK)
    display.fill_rectangle(0,0,SCR_WIDTH,110,BLUE)
    display2.draw_rectangle(10,120,220,SCR_HEIGHT-50,BLUE)
    display.set_font(tt32)
    display.set_color(WHITE,BLUE)
    display.set_pos(20,50)
    display.print('Obstacle Dash')
    display.set_color(WHITE,BLACK)
    display.set_pos(35,160)
    display.print('Singleplayer')
    display.set_color(WHITE,BLACK)
    display.set_pos(35,220)
    display.print('Multiplayer')
```

Funkcije za kontrolisanje prikaza na ekranu

Uz funkcije za izgled, implementirane su i funkcije koje kontroliraju prikaze na ekranu. Ove funkcije omogućavaju upravljanje prikazima i prelazak između različitih ekrana.

```
> def homeScreen():···

> def gameScreen():···

> def gameOverScreen():···

> def sentRequestScreen():···

> def recivedRequestScreen():···

> def pinScreen():···

> def multiplayerResultScreen():···
```

Na ekranima home , gameOver i recivedRequest date su korisniku opcije za biranje. Korisnik se uz pomoć joysticka može pomjerati po tim opcijama, a klikom na joystick bira opciju na kojoj se nalazi. Uz pomoć metode draw_rectangle (ili9321.py) crtamo okvir oko opcije na kojoj se korisnik trenutno nalazi.

Za prikaz objasnjene implementacije uzet cemo homeScreen()

```
def homeScreen():
    global direction           # pozicija joysticka
    makeHomeScreenDesign()
    single_player = True
    direction=0
    display2.draw_rectangle(25,150,180,45,RED)  # okvir oko opcije singleplayer
    while True:
        mqtt_conn.check_msg()
        joystickDirections()
        if(direction== 1 and single_player==False):      # joystick - gore
            single_player=True
            display2.draw_rectangle(25,210,180,45,BLACK) # brisanje okvira oko opcije multiplayer
            display2.draw_rectangle(25,150,180,45,RED)   # okvir oko opcije singleplayer
        elif(direction==2 and single_player==True):      # joystick - dole
            single_player=False
            display2.draw_rectangle(25,150,180,45,BLACK) # brisanje okvira oko singleplayer
            display2.draw_rectangle(25,210,180,45,RED)   # okvir oko opcije multiplayer
```

```
        if TASTER_JOYSTICK.value()==0:                  # klik na taster za biranje opcije
            break
        time.sleep(0.2)

    if single_player==True:                             # odabrana opcija singleplayer
        while True:
            if TASTER_JOYSTICK.value()==1:
                break
        gameScreen()
    else:
        sentRequestScreen()                             # slanje zahtjeva za multiplayer
```

Unutar funkcije gameScreen() deklarišemo timer koji periodično poziva funkciju za crtanje prepreka. Ukoliko je tokom igre kliknut joystick igra se pauzira a ponovnim klikom igra nastavlja.

```
def gameScreen():
    global game_over,multiplayer
    makeGameScreenDesign()
    game_over=False
    while True:
        if TASTER_JOYSTICK.value()==1 :
            break
    drawing_timer=Timer(period=300,mode=Timer.PERIODIC,callback=drawObstacle)
    pause=False
    while True:
        if game_over:
            drawing_timer.deinit()
            break
        if TASTER_JOYSTICK.value()==0 :
            drawing_timer.deinit()
            while True:
                if TASTER_JOYSTICK.value()==1 :
                    break
            while True:
                if TASTER_JOYSTICK.value()==0 :
                    drawing_timer.init(period=300,mode=Timer.PERIODIC,callback=drawObstacle)
                    while True:
                        if TASTER_JOYSTICK.value()==1 :
                            break
                    break
    if multiplayer:
        multiplayerResultScreen()
    else:
        gameOverScreen()
```

Funkcije koje se pozivaju tokom igre

Tokom igre pozivaju se funkcije za crtanje auta, prepreka, linija na cesti i ispisivanje trenutnog rezultata.

Objašnjenje rada prepreka

Prepreka može biti na jednoj od 5 pozicija (cesta ima 5 traka)

```
obstacle_pos=[0,3,1,4,2]
x_o=0
```

Niz obstacle_pos ima izmješane brojeve pozicija prepreka a x_0 prati trenutnu poziciju prepreke, odnosno obstacle_dash[x_0]. Ova pozicija predstavlja u kojoj će se traci prepreka nalaziti. Pozicija se mijenja nakon što prepreka dostigne dno ekrana, odnosno kada auto prođe tu prepreku.

Prepreka se pomijera u jednoj traci na način da se crta nova kocka a briše samo gornji dio stare prepreke da bi dali iluziju pomjerajuće prepreke.

Prepreke imaju brzine koje pratimo uz pomoć atributa :

```
speed=[30,45,54,90]
level=1
score=0
```

Nakon svakih 5 pređenih prepreka brzina se povećava, to pratimo preko rezultata (score). To smo implementirali na način da se povećava udaljenost između prepreka pri pomjeranju, odnosno između stare i nove pozicije.

Kod funkcije za crtanje prepreke:

```
def drawObstacle(p):
    global score
    printScore(score)
    global x_o, y_o, obstacle_pos, car_position, game_over, level, speed
    if obstacle_pos[x_o]==car_position and y_o+50>272:
        game_over=True
        return
    display.fill_rectangle(obstacle_pos[x_o]*48+4,y_o,40,50, WHITE)
    if(y_o>=speed[level-1]):
        display.fill_rectangle(obstacle_pos[x_o]*48+4,y_o-speed[level-1],40,speed[level-
1],GREY)

    y_o+=speed[level-1]

    if(y_o+50>320):
        x_o+=1
```

```
        if(x_o==5):
            x_o=0
        y_o=0
        score+=1
        if score>9:
            level=4
        elif score>6:
            level=3
        elif score>3:
            level=2
        display.fill_rectangle(obstacle_pos[x_o-1]*48+4,270,40,50, GREY)
    drawCar()
    if obstacle_pos[x_o]==car_position and y_o+50>272:
        game_over=True
```

Objašnjenje rada auta

Auto crtamo uz pomoć metode koju smo dodali u biblioteku (objašnjeno na strani 2).

Nakon svakog pomjeranja auta crtamo auto na novoj poziciji a brišemo sa stare pozicije. Staru poziciju pratimo uz pomoć atributa last_pos.

Kod funkcije za crtanje auta:

```
def drawCar():
    drawLines()
    global direction,car_position,last_position
    joystickDirections()
    if(direction==3 and car_position!=0):
        car_position-=1  # lijevo
        direction=0
    elif(direction==4 and car_position!=4):
        car_position+=1  # desno
        direction=0
    if(car_position!=last_position):
        display.fill_rectangle(last_position*48+3, 320-48, 42, 48,GREY)
        last_position=car_position
    display2.draw_car(car_position)
```

Objašnjenje crtanja traka na cesti

Korištenjem metoda iz biblioteke omogućava nam crtanje linija. Unutar naše funkcije drawLines nalazi se niz "xevi" koji predstavlja pozicije na kojima će se nalaziti trake (x-osa). Također imamo atribut "i" koji će se mijenjati nakon svakog poziva ove funkcije i on će se dodavati na poziciju trake gledajući y-osu. Uz pomoć ovog atributa dat ćemo iluziju pomjeranja traka. Prije svakog iscrtavanja traka brisat ćemo stare trake.

Kod za crtanje traka:

```
def drawLines():
    global i
    xevi=[47,48,95,96,143,144,191,192]
    for x in xevi:
        display2.draw_vline(x, 0, 319, GREY)
        coords = [[x,0],[x, 0+i], [x, 40+i],  [x, 80+i],  [x, 120+i],[x, 160+i], [x, 200+i],
[x, 240+i]]
        if i==0 or i==10 or i==20:
            display2.draw_vline(x, 279+i, 20, WHITE)
        elif i==30:
            display2.draw_vline(x, 0, 10, WHITE)
            display2.draw_vline(x, 309, 10, WHITE)
        display2.draw_vlines(coords, WHITE)
    if i==0:
        i=10
    elif i==10:
        i=20
    elif i==20:
        i=30
    elif i==30:
        i=0
```

Multiplayer

Multiplayer smo implementirali uz pomoć MQTT brokera.

Teme igrača 1

```
# Definisanje tema
TEMASUBZAHTJEV='obstacledash/player2'
TEMAPUBZAHTJEV='obstacledash/player1'
TEMASUBZAHTJEVPRIHVACEN='obstacledash/player2prihvacen'
TEMASUBZAHTJEVODBIJEN='obstacledash/player2odbijen'
TEMAPUBZAHTJEVPRIHVACEN='obstacledash/player1prihvacen'
TEMAPUBZAHTJEVODBIJEN='obstacledash/player1odbijen'
TEMASUBPIN='obstacledash/player2pin'
TEMAPUBPIN='obstacledash/player1pin'
TEMASUBEND='obstacledash/player2end'
TEMAPUBEND='obstacledash/player1end'
```

Teme igrača 2

```
# Definisanje tema
TEMASUBZAHTJEV='obstacledash/player1'
TEMAPUBZAHTJEV='obstacledash/player2'
TEMASUBZAHTJEVPRIHVACEN='obstacledash/player1prihvacen'
TEMASUBZAHTJEVODBIJEN='obstacledash/player1odbijen'
TEMAPUBZAHTJEVPRIHVACEN='obstacledash/player2prihvacen'
TEMAPUBZAHTJEVODBIJEN='obstacledash/player2odbijen'
TEMASUBPIN='obstacledash/player1pin'
TEMAPUBPIN='obstacledash/player2pin'
TEMASUBEND='obstacledash/player1end'
TEMAPUBEND='obstacledash/player2end'
```

Kao što se vidi oni će objavljivati i primati poruke sa suprotnih tema, odnosno igrač 1 objavljuje poruke na teme na koje je igrač 2 subrscriban i obrnuto.

Ovako izgleda funkcija koja se poziva pri primanju poruke (kod igrača 2):

```
#Funkcija koja se poziva kada dobijemo poruku na subscribanu temu
def sub(t,p):
    global request_answerd, request_accepted,player1_pin,multiplayer_score
    print("Message arrived: ",t," value: ",p)
    if t==b'obstacledash/player1':
        recivedRequestScreen()
    elif t==b'obstacledash/player1prihvacen':
        request_accepted=True
        request_answerd=True
    elif t==b'obstacledash/player1odbijen':
        request_accepted=False
```

```
        request_answerd=True
    elif t==b'obstacledash/player1pin':
        player1_pin=p
    elif t==b'obstacledash/player1end':
        multiplayer_score = int(p)
```

Kao što vidimo imamo 5 vrsta interakcija između igrača.

Da bi se započela igra jedan igrač treba da klikne na multiplayer opciju u homeScren-u. Klikom na tu opciju igrač publish-a poruku na TEMAPUBZAHTJEV i otvara mu se ekran za čekanje drugog igrača.

```
message="Zahtjev za igru"
mqtt_conn.publish(TEMAPUBZAHTJEV,message)
```

Drugom igraču stiže poruka i otvara mu se ekran u kojem mu je ponuđeno da prihvati ili odbije zahtjev.

```
 if request_accepted:
        message="Zahtjev prihvacen"
        mqtt_conn.publish(TEMAPUBZAHTJEVPRIHVACEN,message)
        request_accepted = False
        pinScreen()
    else:
        message="Zahtjev odbijen"
        mqtt_conn.publish(TEMAPUBZAHTJEVODBIJEN,message)
        homeScreen()
```

Ukoliko prihvati zahtjev objavljuje poruku na temu TEMAPUBZAHTJEVPRIHVACEN i otvara mu se ekran za ukucavanje pina. Ukoliko odbije također se šalje poruka na temu TEMAPUBZAHTJEVODBIJEN I otvara mu se homeScreen.

Igrač koji je poslao zahtjev preko sub(t,p) funkcije čita da li je zahtjev prihvaćen ili odbijen u zavisnosti sa koje teme je stigla poruka.

Unutar funkcije za ukucavanje pina deklarisali smo 4 prekida za 4 tastera. Svaki od tastera ima svoju funkciju koja se pozove ako se klikne na taj taster, također unutar tih funkcija uz pomoć debouncing-a smo omogućili da se ispravno unese broj.

```
# Funkcija koja ispisuje ukucani pin na ekran
> def ispisi():···

# Funkcija tastra
> def t1(p):···

> def t2(p):···

> def t3(p):···

> def t4(p):···
```

Nakon ukucavanja pin-a klikom na joystick igrač šalje svoj pin drugom igraču preko teme TEMAPUBPIN i čeka primanje pin-a od drugog igrača. Ukoliko se pin koji primi od drugog igrača poklapa sa njegovim onda se igračima otvara ekran za igranje igre. To smo implementirali na način:

```
while True:
        mqtt_conn.check_msg()
        if TASTER_JOYSTICK.value()==0:
            print("b'"+str(my_pin)+"'")
            while True:
                if TASTER_JOYSTICK.value()==1:
                    break
            message=str(my_pin)
            mqtt_conn.publish(TEMAPUBPIN,message)
            break

    while True:
        mqtt_conn.check_msg()
        if player1_pin!="":
            print(player1_pin)
            break
    pinCheck = "b'{}'".format(my_pin)
    if str(player1_pin) in pinCheck:
        multiplayer = True
        player1_pin=""
        gameScreen()
    else:
        homeScreen()
```

Kada jedan igrač ne uspije izbjeći prepreku on svoj score šalje na temu TEMAPUBSEND. I otvara mu se ekran za čekanje dok drugi igrač ne završi.

```
    message=str(score)
    mqtt_conn.publish(TEMAPUBEND,message)
    waitingForOtherPlyerScreenDesign()
```

Kada i drugi igrač završi i pošalje svoj score oni porede svoje rezltate i u zavisnosti od toga na ekran im se ispisuje WINNER, YOU LOST ili TIE.