



# Dokumentacija implementacije VENDING MACHINE

Ime i prezime	Index
Nedim Džajić	19003
Nihad Baberović	18850
Lejla Heleg	19197
Aida Zametica	19227

Predmetni profesor: prof. dr Samim Konjicija

Asistent: mag. ing. Selimir Gajip

**UNIVERZITET U SARAJEVU**  
**Elektrotehnički fakultet u Sarajevu**  
**Odsjek - Računarstvo i informatika**  
**Juni 2. 2023**



## 1 | Main

Ovu dokumentaciju ču započeti predstavljanjem glavnog programa koji se izvrša na Raspberry Pi Picu. U njemu su implementirane glavne funkcionalnosti vending machine. U main-u smo uvezli klase Bank (za rad s novcem), Display (za prikazivanje poruka na zaslonu), Keypad (za unos korisničkog koda), machine (za upravljanje I/O pinovima) i time (za upravljanjem vremenom) neophodne za ispravno funkcionisanje sveukupnog sistema.

```
1 from Bank import Bank
2 from Display import Display
3 from Keypad import Keypad
4 import machine
5 from machine import Pin, Timer
6
7 import time
8
9 # initial money in the machine
10 INIT_BALANCE10kf = 0
11 INIT_BALANCE2KM = 0
12
13 bank = Bank(INIT_BALANCE10kf, INIT_BALANCE2KM)
14 display = Display(bank)
15 keypad = Keypad(display)
```

Inicijalizaciju vršimo definiranjem početnog stanja automata (inicijalni iznos kovanica u automatu) i stvaramo objekte za banku, zaslon i tastaturu.

Dalje definisat ćemo kako funkcioniše detekcija novca. Postavljaju se I/O pinovi za senzore koji detektiraju umetnuti novac (10kf i 2KM). Kroz funkciju moneyDetector(p), provjerava se da li se dogodila promjena stanja senzora i dodaje se odgovarajući iznos novca u banku.

```
16
17 # Pin numbers for sensors detecting inserted coins
18 PIN_10kf = 9
19 PIN_2KM = 10
20
21 detector10kf = Pin(PIN_10kf, Pin.IN)
22 detector2KM = Pin(PIN_2KM, Pin.IN)
23
24 lastDetection10kf = int(0)
25 lastDetection2KM = int(0)
26 |
27
28 def moneyDetector(p):
29     global lastDetection10kf, lastDetection2KM
30     if detector10kf.value() != lastDetection10kf:
31         bank.addToBank(0.10)
32         display.refresh()
33         lastDetection10kf = detector10kf.value()
34         time.sleep(2)
35     elif detector2KM.value() != lastDetection2KM:
36         bank.addToBank(2.00)
37         display.refresh()
38         lastDetection2KM = detector2KM.value()
39         time.sleep(2)
40
41 watchForCoins = Timer( callback = moneyDetector,
42                         mode = Timer.PERIODIC,
43                         period = 1
44                     )
```

Pomoću Timer objekta watchForCoins periodički se poziva funkcija moneyDetector(p) kako bi se stalno nadziralo umetanje kovanica.



Postavljanjem I/O pinova za solenoide kontrolišemo vraćanje novca (10kf i 2KM). Kroz funkciju moneyReturner(coins10kf, coins2KM), upravljamo solenoidima kako bi se povratili odgovarajući iznosi novca.

```
47 # Pin numbers for solenoids and estimated times for 1 and 0
48 SOL10kf_PIN = 4
49 SOL2KM_PIN = 5
50 WAIT_SOL10kf_ON = 0.15
51 WAIT_SOL10kf_OFF = 1
52 WAIT_SOL2KM_ON = 0.17
53 WAIT_SOL2KM_OFF = 1
54
55 returner10kf = Pin(SOL10kf_PIN, Pin.OUT)
56 returner2KM = Pin(SOL2KM_PIN, Pin.OUT)
57
58 # returning money handler, controlling solenoids here
59 def moneyReturner(coins10kf, coins2KM):
60     # global bank
61     for _ in range(coins10kf):
62         returner10kf.on()
63         time.sleep(WAIT_SOL10kf_ON)
64         returner10kf.off()
65         time.sleep(WAIT_SOL10kf_OFF)
66     for _ in range(coins2KM):
67         returner2KM.on()
68         time.sleep(WAIT_SOL2KM_ON)
69         returner2KM.off()
70         time.sleep(WAIT_SOL2KM_OFF)
71     lastDetected10kf = 3
72     lastDetected2KM = 3
```

Konačno smo i definirali kodove proizvoda s njihovim cijenama pomoću rječnika "products" i postavili smo output pinove za LED-ove koji signaliziraju kupnju proizvoda. Kroz funkciju signalisePurchase(product), odgovarajući LED se pali na zadato vrijeme kako bi simulirao kupnju određjenog proizvoda.

```
74 # product codes with their prices, stored in dictionary where key = code, value = price"
75 products = {
76     "A01": 1.80,
77     "A02": 2.00,
78     "A03": 2.10,
79     "-" : 0
80 }
81
82 # Pin numbers for leds signalising product purchase
83 LED_PROD1 = 14
84 LED_PROD2 = 13
85 LED_PROD3 = 12
86 # objects for controlling leds, stored in dictionary where key = code, value = pin
87 leds = {
88     "A01": Pin(LED_PROD1, Pin.OUT),
89     "A02": Pin(LED_PROD2, Pin.OUT),
90     "A03": Pin(LED_PROD3, Pin.OUT),
91     "A__": Pin(13, Pin.OUT)
92 }
93
94 # time for led to be on
95 LED_WAIT_TIME = 5
96 def signalisePurchase(product):
97     leds[product].on()
98     time.sleep(LED_WAIT_TIME)
99     leds[product].off()
```



U beskonačnoj petlji čeka se unos korisničkog koda preko tastature. Ako je unesen kod ispravan i ako korisnik ima dovoljno novca na računu, izvršava se kupovina proizvoda. Nakon kupovine, ukoliko kusur postoji on se vraća korisniku, a bank klasa se restartira za sljedeću kupovinu.

```
102 while True:
103     display.refresh()
104     code = keypad.input()
105     while not code in products:
106         display.showMessage("Netacan kod")
107         time.sleep(3)
108         display.showMessage("Odaberite artikl:")
109         code = keypad.input()
110     if products[code] > bank.sessionMoney():
111         display.showMessage("Nedovoljno novca.")
112         code = "-"
113         time.sleep(3)
114         display.showMessage("Odaberite artikl:")
115         print("code:" + code + ", money = " + str(bank.sessionMoney()) + ",prod:" + str(products[code]))
116         coins2Return = bank.calculateChange(products[code])
117         print("(" + str(coins2Return[0]) + "," + str(coins2Return[1]))
118
119         signalisePurchase(code)
120         moneyReturner(coins2Return[0], coins2Return[1])
121         time.sleep(2)
122         bank.sessionStart()
123         display.refresh()
124         time.sleep(4)
...nc
```

To je ukratko opis funkcionalnosti i metoda koje se koriste u ovom glavnem programu za naš projekat.

## 2 | Bank

Nastaviti ćemo sa daljom dokumentacijom implementacije opisujući ukratko klasu Bank. Ova klasa se koristi za praćenje stanja korisničkog kredita, upravljanje dodavanjem novca i izračun kusura korisniku u automatu za prodaju proizvoda.

Pri inicijalizaciji, klasa prima početne iznose novčića od 10kf i 2KM, koje predstavljaju početno novačno stanje automata.

Dodavanje novca vrši se putem metode "addToBank". Ova metoda prima ulazni iznos novca i ažurira novčano stanje automata. Ako je unesen iznos 10kf, povećava se broj novčića od 10kf, dok se za iznos od 2KM povećava broj novčića od 2KM. Također, ova metoda osvježava stanje sesije dodavanjem unesenog iznosa.

```
class Bank:
    def __init__(self, initialBalance10kf, initialBalance2KM):
        self.balance10kf = initialBalance10kf
        self.balance2KM = initialBalance2KM
        self.session = 0.00

    def sessionStart(self):
        self.session = 0

    def addToBank(self, money):
        if money == 0.10:
            self.balance10kf += 1
        else:
            self.balance2KM += 1
        self.session += money
```



Obračun povrata novca korisnicima obavlja se putem metode "calculateChange". Ova metoda prima cijenu proizvoda i izračunava razliku izmedju stanja sesije i cijene proizvoda. Na temelju te razlike, određuje se broj novčića od 10kf i 2KM koji se trebaju vratiti korisniku. Metoda osigurava da je povrat novca precizan koristeći zaokruživanje na najbliži cijeli broj.

```
17     def calculateChange(self, productPrice):
18
19         change = int((self.session - productPrice) * 100 + 0.5)
20         print("Change:" + str(change))
21         if change == 0:
22             return (0,0)
23         elif change < 0:
24             return self.calculateChange(0.00)
25
26
27         coins2KM = int(change // 200)
28         print("Change 2KM:" + str(coins2KM))
29         change -= coins2KM * 200
30         print("Change:" + str(change))
31         coins10kf = int(change // 10)
32         print("Change 10 kf:" + str(coins10kf))
33
34         self.balance2KM -= coins2KM
35         self.balance10kf -= coins10kf
36         self.session -= productPrice
37
38         return (coins10kf, coins2KM)
39
40     def sessionMoney(self):
41         return self.session
```

### 3 | Keypad

Klasa "Keypad" omogućava interakciju korisnika sa automatom putem tastature, omogućavajući unos i obradu korisničkih simbola.

Konstruktor klase "Keypad" prima objekat "display" koji se koristi za prikazivanje unesenih simbola.

Metoda "input" se koristi za unos simbola. Prilikom unosa, korisnik može unositi maksimalno do tri simbola. Uneseni simboli se prikazuju na displeju putem objekta "display".

Metoda radi na principu očitavanja stanja tastera u matrici tastature. Kada korisnik pritisne određeni taster, metoda će očitati simbol sa te pozicije i dodati ga u trenutni unos. Nakon unosa svakog simbola, prikazuje se ažurirani unos na displeju.

Kada korisnik unese tri simbola, unos se resetuje i korisnik može ponovo unositi simbole. Kada korisnik pritisne taster "#" na tastaturi, unos se završava (kupovina finalizira) i metoda vraća unesene simbole kao string.

Tastatura se sastoji od četiri reda i četiri kolone, pri čemu se svaki taster identificuje odgovarajućim simbolom u matrici "symbol".



```
1  from machine import Pin
2  import time
3  from Display import Display
4
5  row = [Pin(21,Pin.OUT),Pin(22,Pin.OUT),Pin(26,Pin.OUT),Pin(27,Pin.OUT)]
6  col = [Pin(0, Pin.IN),Pin(1, Pin.IN),Pin(2, Pin.IN),Pin(3, Pin.IN)]
7  symbol = [[1,2,3,'A'],[4,5,6,'B'],
8    [7,8,9,'C'],[*,'0','#','D']]
9
10 class Keypad:
11     def __init__(self, display):
12         self.display = display
13
14     def input(self):
15         # handle user input here, communication with display via display attribute
16         code = ['_','_','_'] # Use a list to store the input code
17         numOfSymbols = 0 # counts number of inputed symbols
18
19         self.display.showInput(code)
20         while True:
21             # Use < instead of != for the symbol count
22             for i in range(4):
23                 row[i].value(1)
24                 for j in range(4):
25                     if col[j].value() == 1:
26                         while col[j].value() == 1:
27                             pass
28                         if symbol[i][j] == '#':
29                             return ''.join(code)
30                         if numOfSymbols == 0:
31                             code = ['_','_','_']
32                             code[numOfSymbols] = symbol[i][j]
33                             numOfSymbols += 1
34                         if numOfSymbols == 3:
35                             numOfSymbols = 0
36                         self.display.showInput(code)
37                     row[i].value(0)
```



## 4 | Displej

Klasa "Display" omogućava prikazivanje informacija na displeju i komunikaciju sa korisnikom u sklopu funkcionalnosti automata.

Konstruktor klase "Display" prima objekat "bank" koji predstavlja stanje novca automata. Metoda "refresh" koristi se za osvježavanje prikaza na displeju. Prikazuje se poruka, trenutni kredit i uneseni kod. Metoda "showMessage" koristi se za prikazivanje poruke na displeju. Metoda "showInput" koristi se za prikazivanje unesenog koda na displeju.

```
1  from machine import Pin
2  import time
3  from Display import Display
4
5  row = [Pin(21,Pin.OUT),Pin(22,Pin.OUT),Pin(26,Pin.OUT),Pin(27,Pin.OUT)]
6  col = [Pin(0, Pin.IN),Pin(1, Pin.IN),Pin(2, Pin.IN),Pin(3, Pin.IN)]
7  symbol = [['1','2','3','A'],['4','5','6','B'],
8            ['7','8','9','C'],['*','0','#','D']]
9
10 class Keypad:
11     def __init__(self, display):
12         self.display = display
13
14     def input(self):
15         # handle user input here, communication with display via display attribute
16         code = ['_','_','_'] # Use a list to store the input code
17         numOfSymbols = 0 # counts number of inputed symbols
18
19         self.display.showInput(code)
20         while True:
21             # Use < instead of != for the symbol count
22             for i in range(4):
23                 row[i].value(1)
24                 for j in range(4):
25                     if col[j].value() == 1:
26                         while col[j].value() == 1:
27                             pass
28                         if symbol[i][j] == '#':
29                             return "".join(code)
30                         if numOfSymbols == 0:
31                             code = ['_','_','_']
32                         code[numOfSymbols] = symbol[i][j]
33                         numOfSymbols += 1
34                         if numOfSymbols == 3:
35                             numOfSymbols = 0
36                         self.display.showInput(code)
37                     row[i].value(0)
```

## 5 | Maketa

Za kraj ćemo se osvrnuti i na najbitniji dio naše implementacije, a to je hardverski aspekt kreiranja makete. Vending mašine smo napravili od kartona, papira, i raznoraznih materijala pristupnih svakom pojedincu. Potrudili smo se da kreiramo nešto što od postojećih materijala i da damo svrhu čak i nekim neobičnim predmetima poput plastičnih kašika koji su nam služili kao senzori za detektovanje ubačaja kovanica. Naš projekat je zahtijevao preciznost u izradi, kako bi se postigla funkcionalnost i estetski dojam.

Kroz proces izrade i eksperimentisanja, uspjeli smo stvoriti privlačan i funkcionalan prototip vending mašine. Integracija softvera i mehaničkih komponenti omogućila je da naša mašina bude sposobna za prikazivanje informacija na displeju, unos koda, obradu plaćanja kao i vraćanje kusura. Kroz kreativnu upotrebu dostupnih materijala, uspjeli smo stvoriti privlačan dizajn koji je istovremeno praktičan i efikasan korisniku.

