STEMKIT 4SCHOOLS

SEMAFOARE CU PITON ȘI GPIO

PLANUL DE LECTIE 2

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission.

Cuprins

1. Semafoare cu Python și GPIO

1.1 Informatii generale

1.1.1 Scurta descriere

În acest plan de lecție vom învăța cum să construim semafoare în lumea reală și să le controlăm prin GPIO și limbajul de programare Python. Planul de lecție implică crearea circuitului folosind pinii GPIO ai Raspberry Pi și a componentelor electronice, precum și dezvoltarea unui program în Python care va controla secvența semafoarelor.

1.1.2 Obiectivele invatarii

Principalele obiective de învățare ale acestui plan de lecție sunt:

- Înțelegerea conceptului și a conținutului teoriei cozilor, a ingineriei traficului și a teoriei blocajelor.
- Familiarizarea cu componentele electronice și crearea de circuite și programe pentru a interacționa cu pinii GPIO ai Raspberry Pi.
- Înțelegerea structurilor de bază ale limbajului de programare Python (sintaxă, funcții, bucle, instrucțiuni if etc.).

1.1.3 Links catre curriculum

Domeniile, subdomeniile, subiectele / subiectele la care acest plan de lecție poate fi legat, sunt:

- Matematică: teoria cozilor, linii de așteptare, cozi.
- Inginerie: blocaje, sisteme de servicii, cozi.
- Informatică / Informatică: unitate de procesare și periferice, interfețe, limbaj de programare și structuri principale, codare.

1.1.4 Materiale solicitate

Pentru acest plan de lecție (și pentru fiecare grup de studenți), pe lângă consola STEMKIT și Raspberry Pi, vom avea nevoie de:

- 6 x fire jumper Male-to-Female
- 1 x fir jumper Male-to-Male
- 1 x Breadboard
- 1 x Push button /buton
- 1 x Buzzer
- 3 x LED (roșu, chihlimbar și verde)
- 3 x 220 Ohm Rezistențe

1.1.5 Durata

Durata acestui plan de lecție este estimată la aproximativ 45-60 de minute, adică o oră de clasă.

1.2 Planul Lectiei

Planul de lecție este împărțit în trei etape, care sunt introducerea, pregătirea și încheierea.

1.2.1 Introducere

Teoria cozilor este studiul matematic al liniilor de așteptare sau a cozilor. O coadă este un container de obiecte care sunt inserate și eliminate în conformitate cu principiul primul-în-primul-ieșire (FIFO). Adăugări noi sunt la linia făcută în spatele cozii, în timp ce eliminarea are loc în partea din față.

Coada este, de asemenea, studiul comportamentului traficului în apropierea unei anumite secțiuni în care cererea depășește capacitatea. Cozile pot fi văzute în multe situații: urcarea într-un autobuz / avion, blocaje pe autostradă, cumpărături, ieșirea din clasă la sfârșitul unei prelegeri sau așteptarea unui hamburger la biroul local.

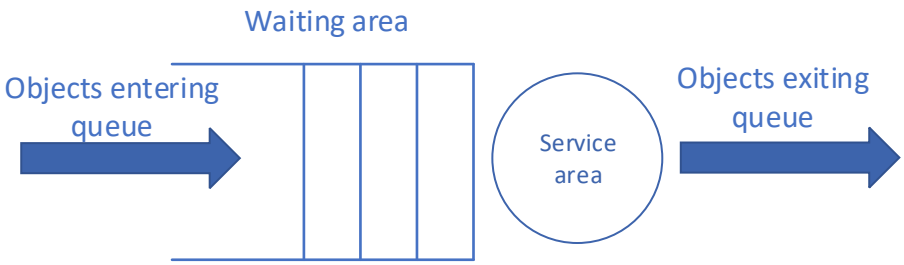

FIGURA 1 DIAGRAMA COZII SIMPLE (PRIMUL-ÎN-PRIMUL-OUT)

Există diferite distribuții statistice care explică o coadă. Distribuția de sosire ar putea fi deterministă (aleatorie) sau aleatorie (de exemplu, Poisson). Distribuția serviciilor este fie deterministă, fie aleatorie. În plus, există diferite modele de servicii:

- First In First Out (FIFO)
- Last In First Out (LIFO)
- Prioritate (ocolește coada)

În ingineria traficului, așteptarea poate avea loc la semafoare, semne de oprire, blocaje sau orice constrângere a fluxului bazată pe proiectare sau pe trafic. Atunci când nu sunt tratate corespunzător, cozile pot provoca congestii severe de rețea, făcându-le astfel ceva important de studiat și înțeles. Cel mai comun model de serviciu care explică o coadă la un semafor este FIFO. Cu toate acestea, în anumite circumstanțe, modelul de serviciu se schimbă (de exemplu, atunci când traficul este controlat de un polițist rutier).

În teoria cozilor, folosim următoarea notație:

- Rata de sosire = λ
- Rata de plecare = μ
- Rata de utilizare ρ = λ / μ

Înțelegând gradul de saturație al unei cozi, luăm în considerare următoarele:

- Suprasaturat: λ> μ
- Nesaturat: λ <μ
- Saturați: λ = μ

O coadă de trafic este explicată folosind formula lui Little:

Formula lui Little: $E(n) = \lambda E(v)$

Formula arată că dimensiunea medie a cozii (măsurată în vehicule) este egală cu rata de sosire (vehicule pe unitate de timp) înmulțită cu timpul mediu de așteptare (atât timpul de întârziere în coadă, cât și timpul de serviciu) (în unități de timp). Acest rezultat este independent de anumite distribuții de sosire și este un principiu fundamental important, care nu a fost dovedit până în 1961.

În cazul nostru, vom folosi teoria cozilor pentru a identifica semnalizarea optimă a semaforului (în unități de timp - secunde), astfel încât rata de sosire λ este egală sau aproape egală cu rata de plecare μ.

Acum, că am dobândit înțelegerea de bază a cozilor, să procesăm pregătirea experimentului nostru.

1.2.2 Pregartire

1.2.2.1 Crearea circuitului

Primul lucru pe care trebuie să-l facem este să creăm un circuit și să-l conectăm la pinii GPIO ai Raspberry Pi. Opriți Raspberry și deconectați cablurile atașate la acesta. Pentru circuitul nostru avem nevoie de o placă de testare, LED-uri (roșu, chihlimbar, verde), rezistențe, fire jumper, un buzzer și un buton.

Trebuie să așezăm toate componentele pe panou și să le conectăm la pinii GPIO corespunzători de pe Raspberry Pi. Înțelegând cum este conectată fiecare componentă, trebuie să avem în vedere următoarele:

- Un buton necesită 1 pin de împământare și 1 pin GPIO
- Un LED necesită 1 pin de masă și 1 pin GPIO, cu un rezistor de limitare a curentului.
- Un buzzer necesită 1 pin de masă și 1 pin GPIO. Componentele pot împărți un pin de împământare. Folosim panoul pentru a activa acest lucru.

Circuitul complet este în schema care urmează.

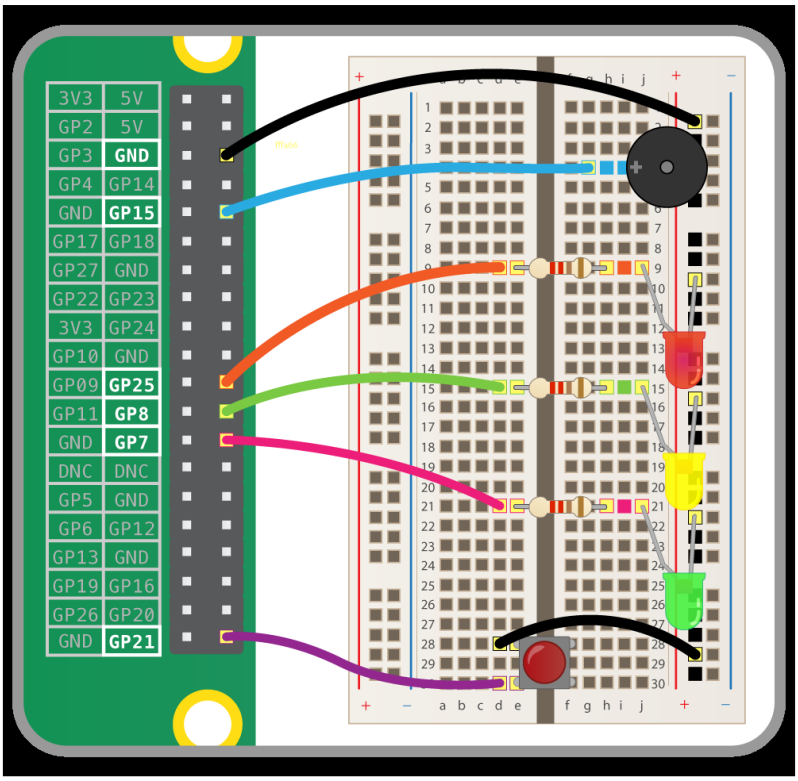

FIGURA 1 SCHEMA DIAGRAMA CIRCUITULUI DE SEMNURI CONECTATE LA PIN-URILE GPIO. SURSA: PROIECTE.RASPBERRYPI.ORG

Rețineți că rândul de-a lungul părții lungi a panoului este conectat la un pin de masă de pe Raspberry Pi, astfel încât toate componentele din acel rând sunt conectate la masă.

În ceea ce privește pinii GPIO, folosim următoarele:

- Buton la 21
- LED roșu la 25
- LED chihlimbar la 8
- LED verde la 7
- Buzzer la 15

După ce am finalizat circuitul, putem porni Raspberry Pi și putem începe dezvoltarea programului nostru în Python.

1.2.2.2 Programarea Python pentru circuit

Trebuie să creăm un program care să controleze circuitul pe care l-am creat în pasul anterior.

Pe scurt, în programul nostru importăm mai întâi modulele Python de care avem nevoie, apoi setăm pinii GPIO pentru LED-uri, buton și buzzer.

Folosim modulul ***gpiozero*** deoarece se bazează pe RPi.GPIO și are reguli de codare mult mai simple.

```
##############################################################################
# import libraries and set gpio numbering mode
from gpiozero import Button, TrafficLights, Buzzer
from time import sleep

# Button interface for controlling the push button
button = Button(21)
#TrafficLights interface for simulating an actual traffic light
lights = TrafficLights(25, 8, 7)
# Buzzer interface for controlling the buzzer
buzzer = Buzzer(15)

while True:
# Sequence for green light
    lights.green.on()
    button.wait_for_press()          #like in pedestrian crossing
    sleep(3)                         #delay to execute the code

#Sequence for amber light
    lights.green.off()
    lights.amber.on()
    sleep(3)

#Sequence for red light
    lights.red.on()
    buzzer.beep()
    lights.amber.off()
    sleep(15)

    lights.amber.on()
    buzzer.off()
    sleep(1)

    lights.red.off()
    lights.amber.off()
```

În GPIO Zero, creăm obiecte pentru fiecare componentă utilizată. Fiecare interfață componentă trebuie importată din modulul gpiozero și trebuie creată o instanță pe numărul pinului GPIO la care este conectat.

Interfața *TrafficLights* preia trei numere GPIO, unul pentru fiecare pin: roșu, chihlimbar, verde (în această ordine). Putem folosi pornit, oprit și clipit, toate acestea controlând toate cele trei lumini simultan.

Interfața *Buzzer* are un număr GPIO și poate fi controlată prin: pornit, oprit și bip. Folosind numărul dintre paranteze, putem manipula timpul de zumzet.

Programul de mai sus creează următoarea succesiune de semafoare:

- Verde aprins
- Chihlimbar aprins
- Roșu aprins
- Roșu și chihlimbar activat
- Verde aprins

Butonul simulează o trecere de pietoni. Când un „pieton" lovește butonul, luminile se mută în roșu (cu o oarecare întârziere) și acordă pietonilor ceva timp pentru a traversa strada înainte de a readuce luminile în verde. Adăugarea unui buzzer îmbunătățește simularea, deoarece într-o traversare reală există un buzzer în beneficiul pietonilor cu deficiențe de vedere.

Verificați dacă programul dvs. funcționează. Apăsați butonul de câteva ori și vedeți cum funcționează semafoarele. Încercați să schimbați ora pentru lumină verde și lumină roșie.

1.2.3 Concluzii

Am reușit să creăm un sistem de semafoare și o trecere de pietoni folosind programarea Python și GPIO-ul Raspberry. În această fază recapitulăm ce am făcut și cum, care au fost pașii principali și discutăm dificultățile întâmpinate. Ca exercițiu de urmărire, discutați modalități de optimizare a scripturilor utilizate pentru acest experiment.

1.3 Referinte sau Resurse

Iată câteva referințe utile și resurse suplimentare legate de acest plan de lecție.

- https://projects.raspberrypi.org/en/projects/physical-computing/10
- https://gpiozero.readthedocs.io/en/stable/
- https://www.cs.cmu.edu/~adamchik/15-121/lectures/Stacks%20and%20Queues/Stacks%20and%20Queues.html
- https://en.wikipedia.org/wiki/Queueing_theory
- https://en.wikibooks.org/wiki/Fundamentals_of_Transportation/Queueing#cite_ref-1
- https://en.wikipedia.org/wiki/Traffic_engineering_(transportation)