Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

DOCUMENTAȚIE ROBOTICĂ

Nume : Pintilie Georgian Vasile

Olaru Alexandru

Programul de studii : Automatica

Grupa : 4LF412 4LF413

An : 2022-2023

Cuprins

1. Introducere

Ideea noastra a pornit de la creearea unei mașini pe care să o putem controla cu ajutorul unui controller de PS2.

Aceasta idee este una simplă, se poate realiza și de acasă cu ajutorul unui kit de robot cu două roti, o baterie portabilă, o plăcuță Raspberry Pico și câteva linii de cod scrise in Python.

2. Arhitectură

Pentru început, aveți nevoie de un kit pentru șasiu (2.1).

FIg.2. 1 Kit șasiu 2WD

După care, aveți nevoie de o plăcuță Raspberry Pico (2.2), o baterie portabilă(2.3).

FIg.2. 2 Raspberry Pico

FIg.2. 3Raspberry Pico

Pentru a controla motoarele este nevoie de un driver pentru motoare în punte H, dubla,L298N (2.4).

FIg.2. 4 Punte H dubla L298N

Pentru a trimite comenzi robotului, vom folosi un controller de PS2(2.5).

FIg.2. 5Controller PS2

3. Realizarea părții hardware

Ca prim pas, am asamblat partea caroseria și am montat cele două motoare, împreună cu cele două roți, ghidându-ne dupa instinct. (3.1).

Fig3. 1

În figura (3.2) se pot vedea toate porturile folosite de pe plăcuța Raspberry Pico.

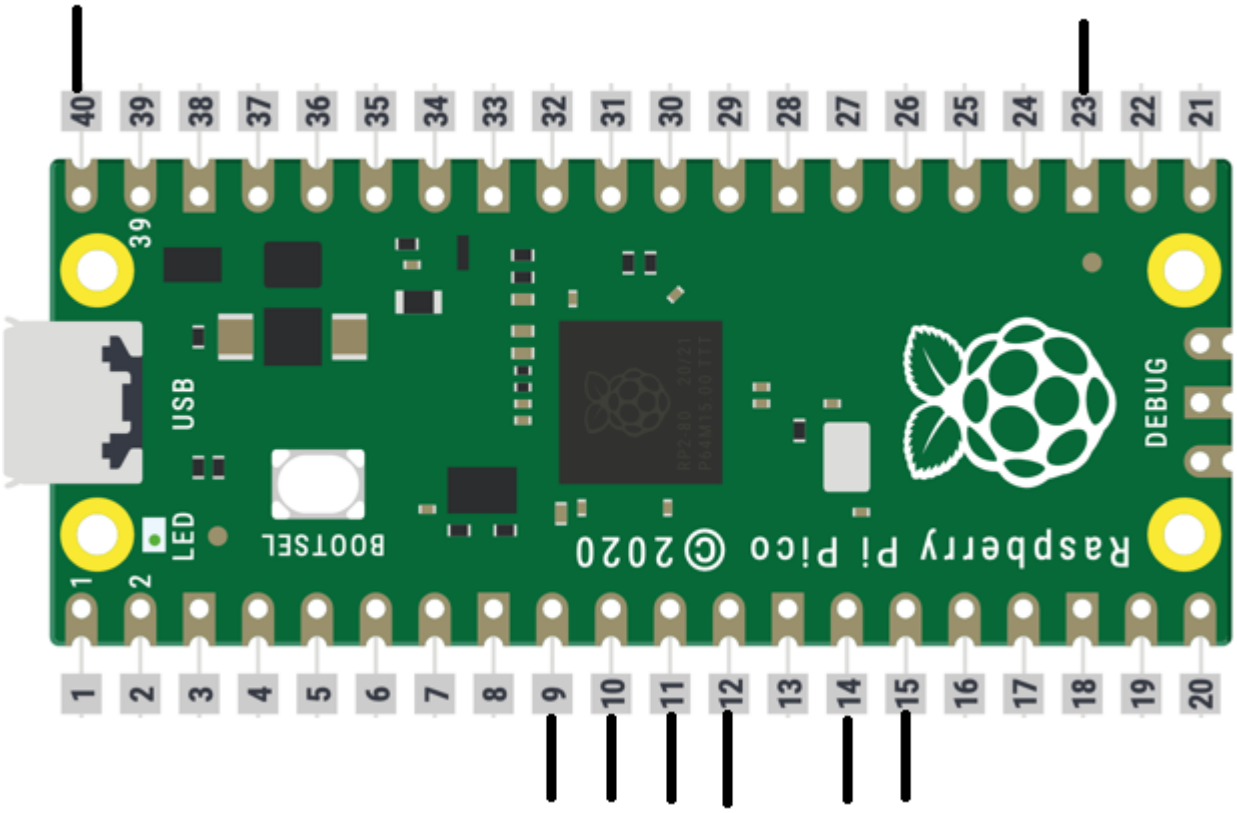

Fig3. 2

Porturile 8 și 9 se conecteaza la driverele pentru motoare. Driverul este pozitionat în centru, între cele două motoare. Porturile 1 și 2 sunt conectate la porturile 40 si 23 de pe Raspberry, iar porturile 4 5 6 7sunt conectate la porturile 9 10 11 12 14 15. Conexiunea pentru un singur driver se poate vedea in imaginea(3.4).

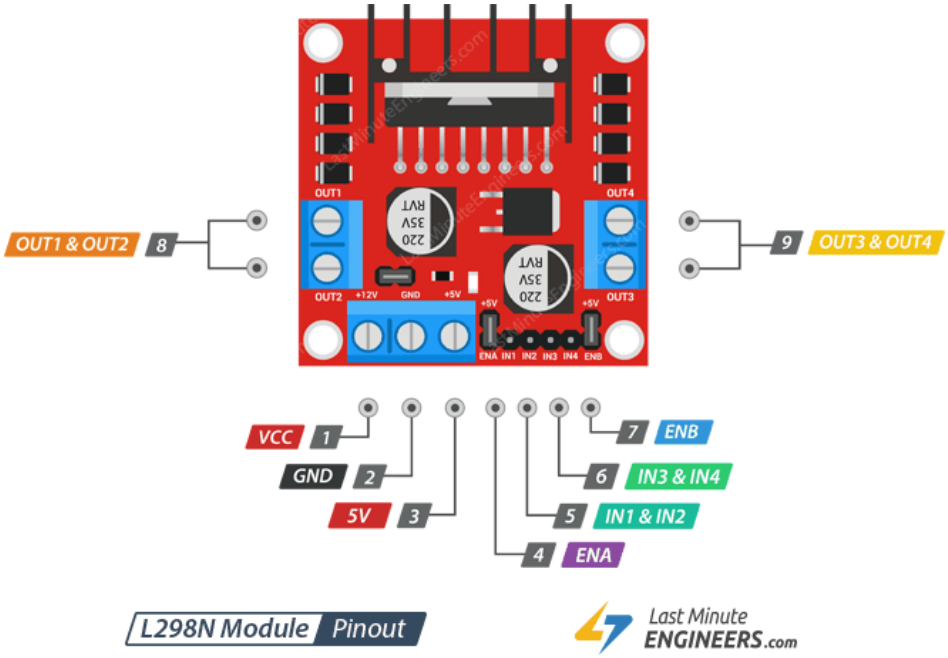

Fig3. 4

Porturile 4 și 5 sunt conectate la +5V si GND de pe Raspberry, porturile 1, 2, 6 și 7 sunt conectate la porturile 4, 5, 6 și 7, restul fiind nefolosite.

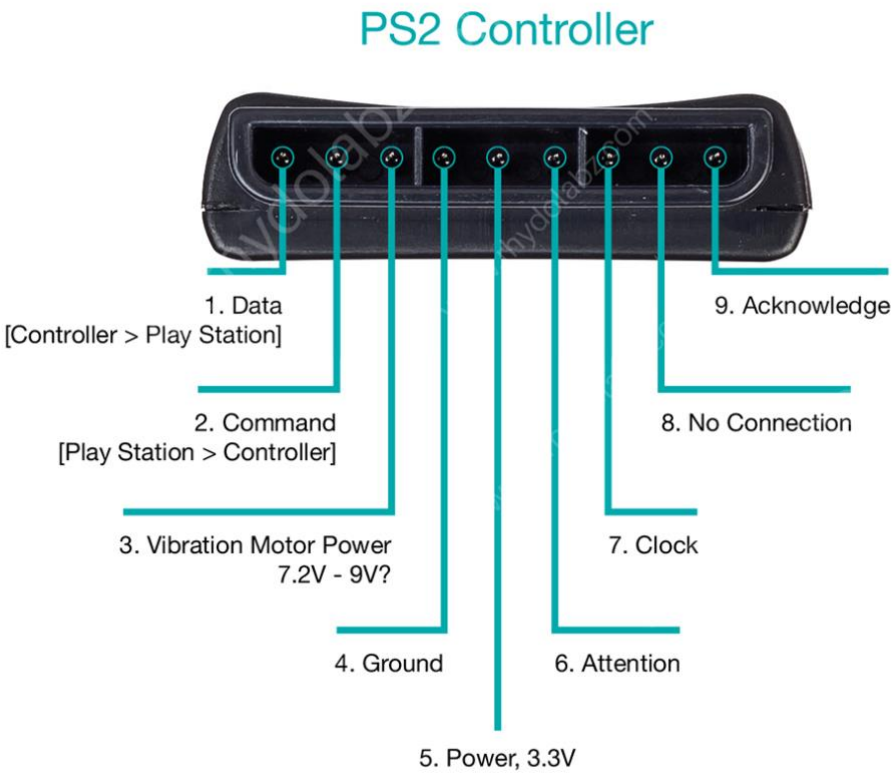

Fig3. 5

4. Realizarea părții software

În urmatoarele figuri se vor putea vedea fragmente din cod.
Am folosit o librărie pentru controllerul de PS2 portată de pe Arduino.

```
397 import time
398 import board
399 import digitalio
400 import pwmio
401
402 def main():
403
404     # Select desired pins according to your wiring
405     PS2_DAT = 2
406     PS2_CMD = 3
407     PS2_SEL = 4
408     PS2_CLK = 5
409
410     pressures = True
411     rumble = False
412
413     # Init PS2 controller.
414     ps2x = PS2X()
415
416     st = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT, pressures, rumble)
417     if st != 0:
418         exit(-1);
```

```
420     # Init L298N motor driver.
421     in1 = digitalio.DigitalInOut(board.GP6)
422     in1.direction = digitalio.Direction.OUTPUT
423     in1.value = True
424
425     in2 = digitalio.DigitalInOut(board.GP7)
426     in2.direction = digitalio.Direction.OUTPUT
427     in2.value = False
428
429     in3 = digitalio.DigitalInOut(board.GP8)
430     in3.direction = digitalio.Direction.OUTPUT
431     in3.value = False
432
433     in4 = digitalio.DigitalInOut(board.GP9)
434     in4.direction = digitalio.Direction.OUTPUT
435     in4.value = True
```

```
437     # Enable PWM output
438     ena = pwmio.PWMOut(board.GP10)
439     deada = True
440
441     enb = pwmio.PWMOut(board.GP11)
442     deadb = True
443
444     vibrate1 = 0  # for left motor
445     vibrate2 = 0  # for right motor .
```

```
447 while True:
448     # Pump PS2 controller
449     ps2x.read_gamepad(vibrate1, vibrate2);
450
451     # Read PS2 controller joysticks
452     jstick = [ps2x.Analog(PSS_LY), ps2x.Analog(PSS_LX), ps2x.Analog(PSS_RX), ps2x.Analog(PSS_RY)]
453
454     # Controll motors base on joystick values.
455     if jstick[0] < 120:
456         in3.value = True;
457         in4.value = False;
458         n = 120 - jstick[0];
459         enb.duty_cycle = cap(n * 500);
460         deadb = False
461     elif jstick[0] > 134:
462         in3.value = False;
463         in4.value = True;
464         n = jstick[0] - 134
465         enb.duty_cycle = cap(n * 500)
466         deadb = False
467     elif not deadb:
468         in3.value = False
469         in4.value = False
470         enb.duty_cycle = 0;
471         deadb = True
```

```
473         if jstick[3] < 120:
474             in2.value = True;
475             in1.value = False;
476             n = 120 - jstick[3];
477             ena.duty_cycle = cap(n * 500);
478             deada = False
479         elif jstick[3] > 134:
480             in2.value = False;
481             in1.value = True;
482             n = jstick[3] - 134
483             ena.duty_cycle = cap(n * 500)
484             deada = False
485         elif not deada:
486             in2.value = False
487             in1.value = False
488             ena.duty_cycle = 0;
489             deada = True
490 
491         time.sleep(1000 / 30)
492 
493 main()
```

5. Ansamblu final

În imaginile de mai jos, se poate vedea asamblarea finală a proiectului.

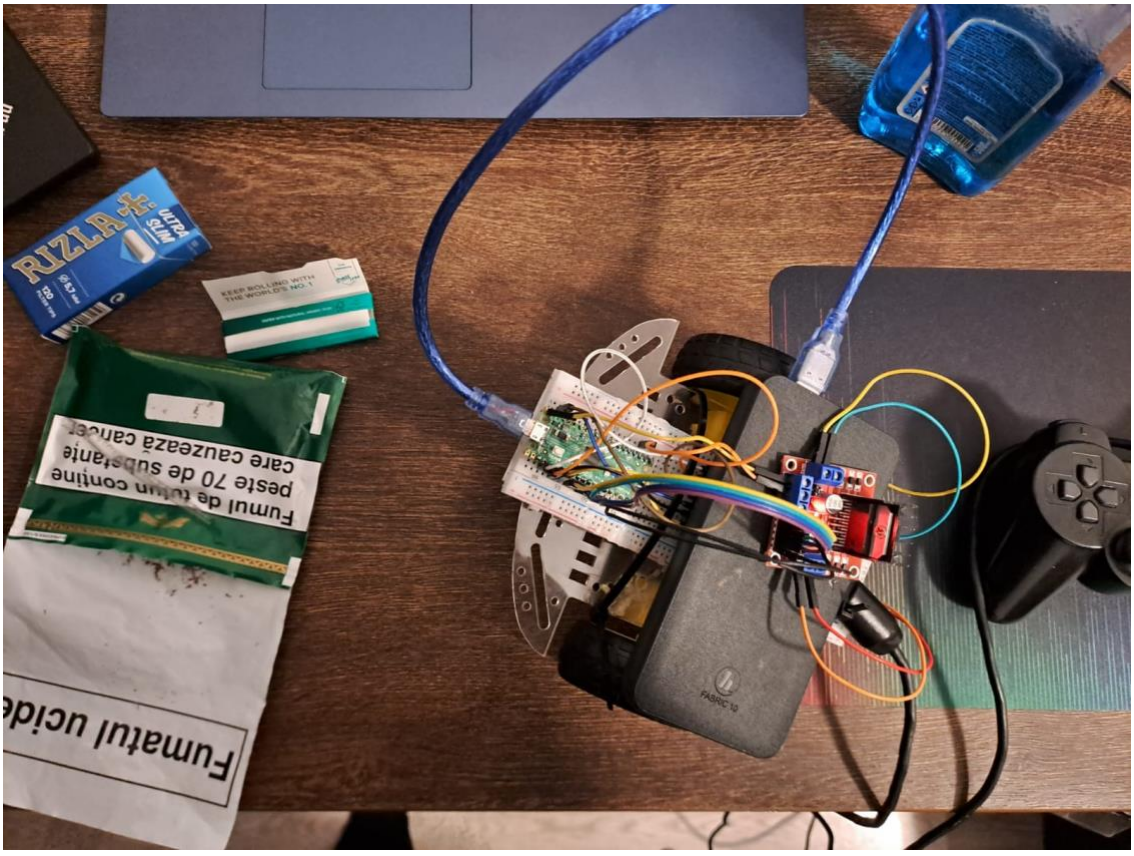

Fig.5. 1

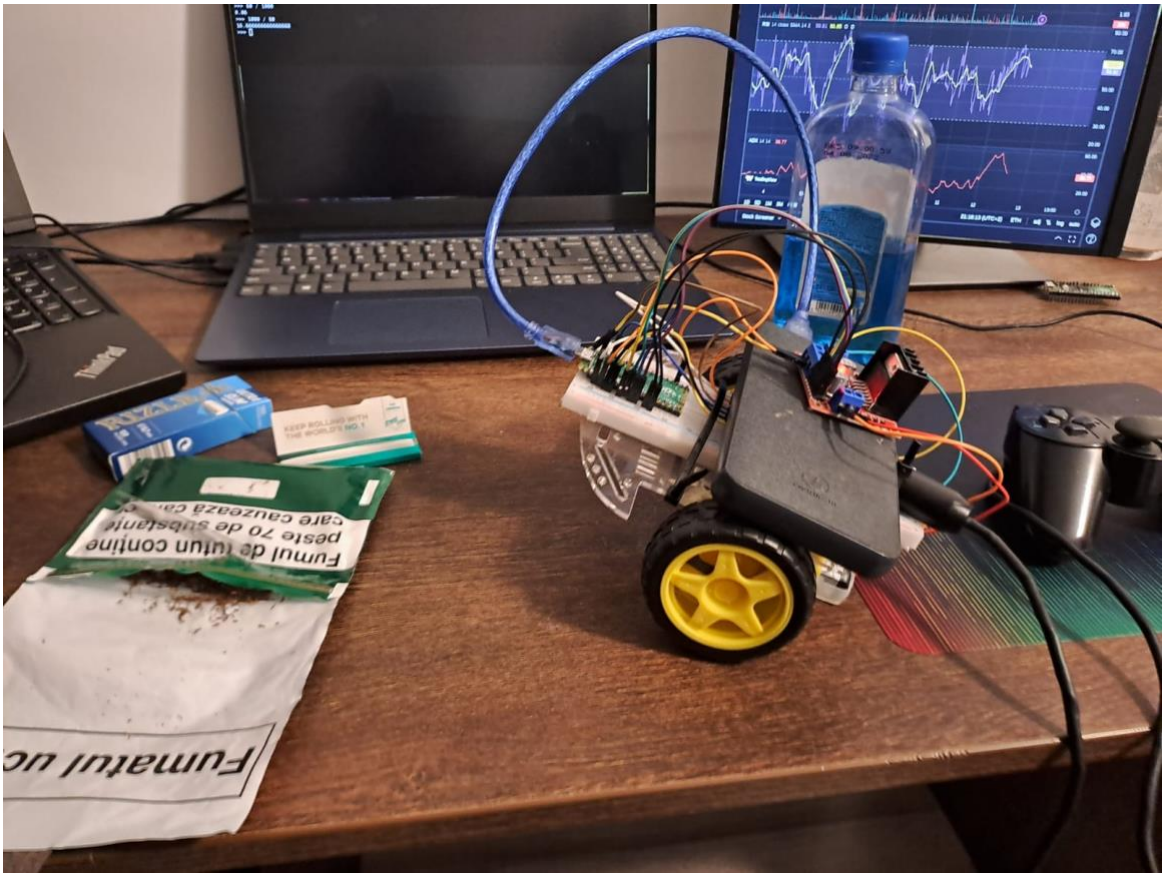

Fig.5.2

6. Anexe

1. https://github.com/madsci1016/Arduino-PS2X