گزارشکار پروژه درس معماری کامپیوتر

اساتید مربوطه:

دکتر وحید جمشیدی، دکتر عراقی پور

نام اعضای گروه:

امیرعلی میرزایی

مهدی برهانی

اسماعیل باقری

بابک فتحی

پاییز ۱۴۰۲

هدف انجام پروژه

ساخت یک ربات فوتبالیست که با دسته‌ی بازی کنترل شود و توپ را بگیرد و آن را حمل کند و در فاصله چند سانتی متری دروازه توپ را شوت کند. همچنین قابلیت تشخیص دروازه و توپ را داشته باشد.

سخت افزار مورد نیاز

۱. شاسی فلزی و پرچ

ما از ۷ عدد شاسی(۶ عدد خمیده و ۱ عدد تخت) استفاده کردیم.

۲. چرخ

۴ عدد چرخ شیار دار که با کش دور آنها پیچیده شده است و دو چرخ جلو هرزگرد و دو چرخ عقب به موتور متصل شده‌اند.

۳. موتور گیربکس دار

دو عدد موتور گیربکس دار که ما از موتور های ۶ ولتی و ۱/۵ آمپری استفاده کردیم.

۴. پیچ و اسپیسر

این مورد باعث می‌شود اجزای الکترونیکی با یک فاصله از روی شاسی های فلزی قرار بگیرند ولی اگر شما از شاسی پلاستیکی استفاده کنید نیاز به اسپیسر ندارید.

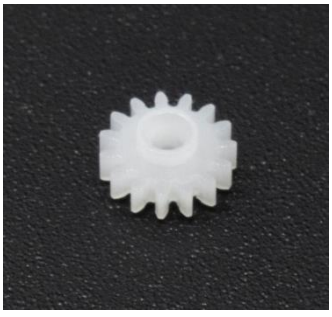
۵. برد Raspberry Pi 4

۶. سرو موتور (servo motor)

برای گرفتن توپ نیاز به یک موتور داریم که با یک سرعت ثابت یک زاویه‌ای را طی کند.

۷. شوتر

از چند عدد چرخ دنده و یک کیس مخصوص و یک آرمیچر برای شوت کردن استفاده کردیم.



۸. کیس Raspberry Pi

برای جلوگیری از اتصال کوتاه نکردن Raspberry Pi بهتر است از یک کیس استفاده کنیم.

۹. دسته پلی استیشن

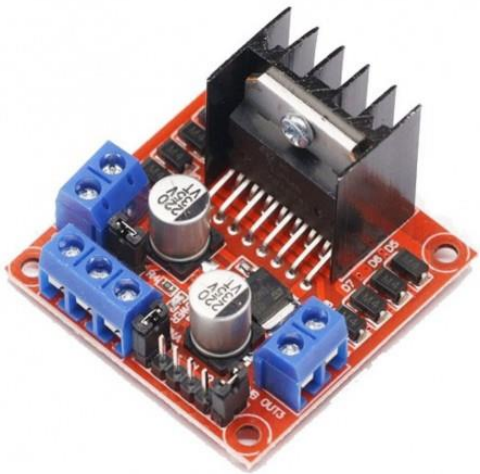
جهت کنترل ربات

۱۰. سیم

سیم جامپر دو سر نری، دو سر مادگی و نری به مادگی

۱۱. درایور L298

ما برای کنترل دو موتور ربات و همچنین شوتر نیاز به دو تا ماژول L298 داریم.

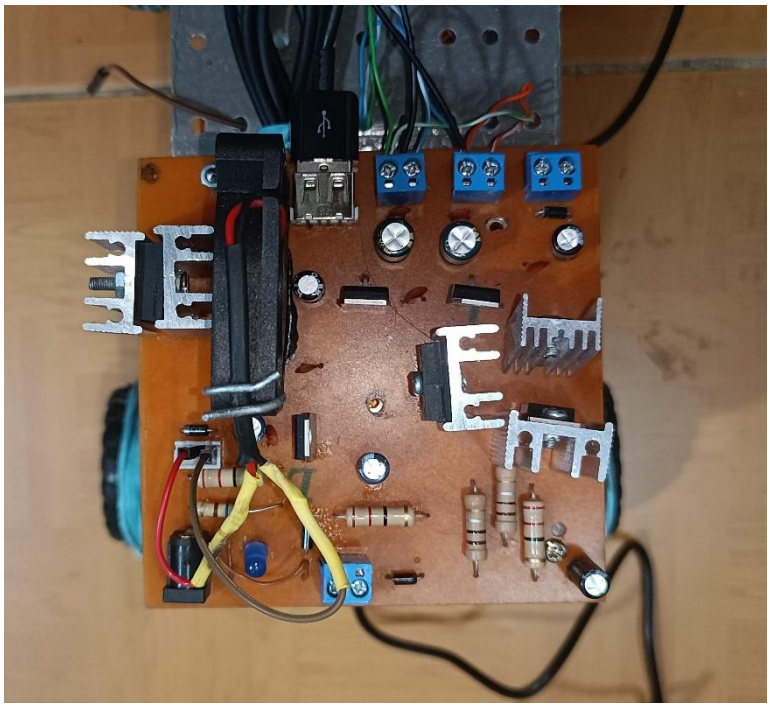


۱۲. آداپتور

ما نیاز به یک آداپتور ۱۲ ولت و ۳ آمپر داریم.

زیرا ورودی منبع تغذیه‌ی ما به همین مقدار می‌باشد.

۱۳. منبع تغذیه (PCB)



۱۴. دوربین

ما از دو عدد موبایل استفاده می‌کنیم که یکی نقش دوربین را ایفا می‌کند و تصویر جلوی ربات را ارسال می‌کند و دیگری تصویر را دریافت و به نمایش می‌گذارد، تا به واسطه آن ربات توسط گروه کنترل شود.

۱۵. SD card

یک عدد SD card ۸ الی ۳۲ گیگ

۱۶. سیستم عامل Raspberry Pi (Raspbian)

از طریق سایت رسمی Raspberry Pi قابل دانلود و نصب می‌باشد.

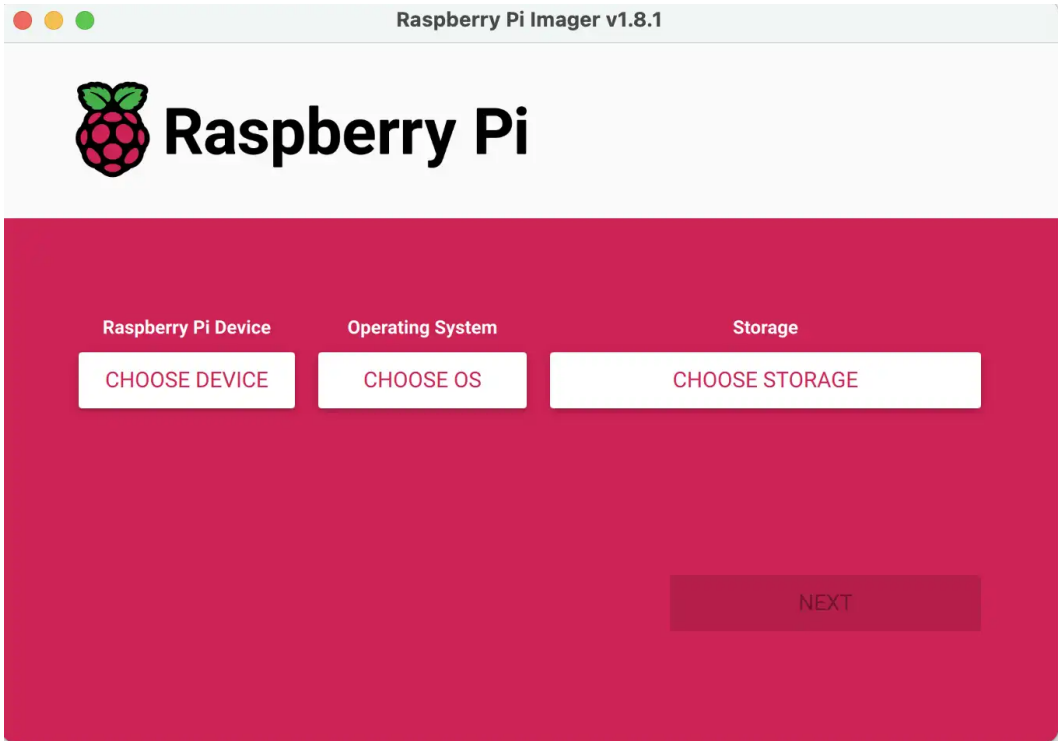
۱۷. نرم افزار VNC Viewer

۱۸. نرم افزار VNC Server

۱۹. یک عدد توپ

۲۰. Raspberry Pi Imager

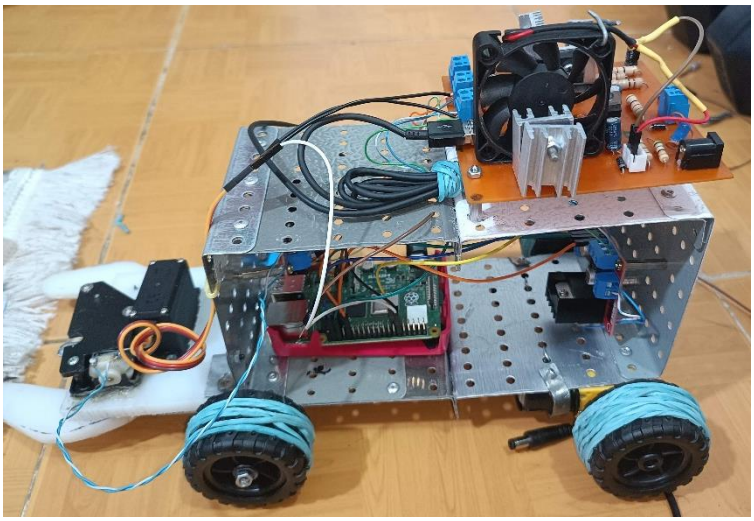
با استفاده از این نرم افزار می‌توانید به راحتی سیستم عامل دانلود شده را روی SD card نصب و راه اندازی کنید.

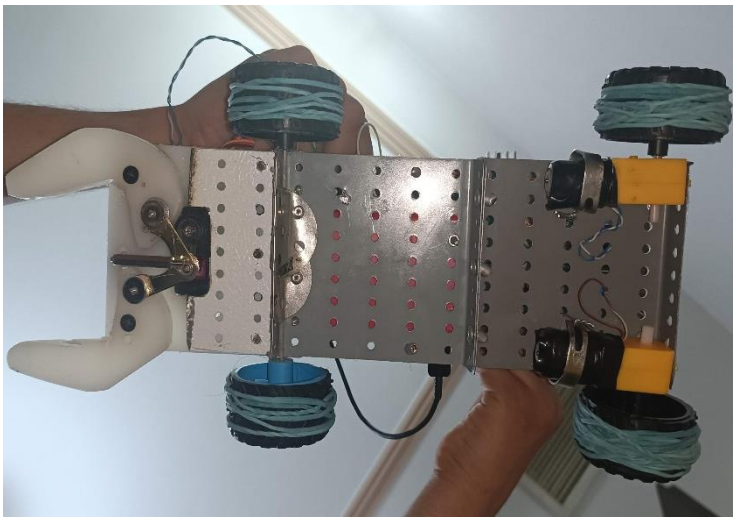


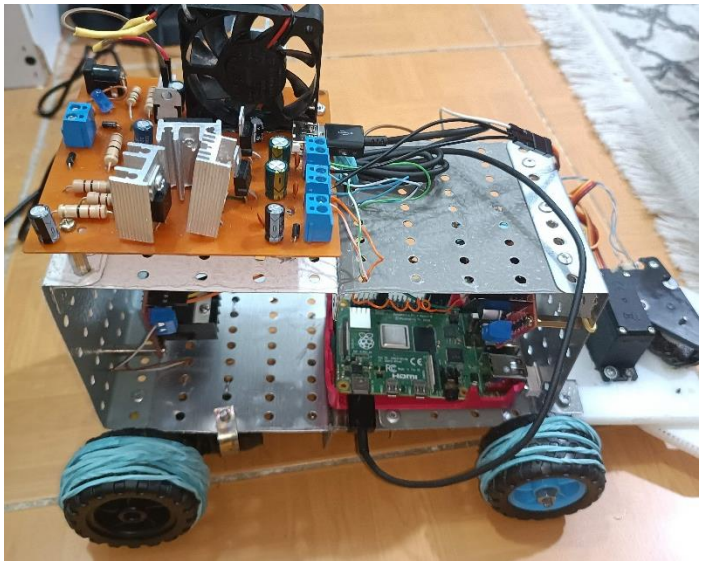
شرح گزارش ربات

اسمبل بدنه ربات:

ابتدا شاسی های فلزی را به هم دیگر پرچ می‌کنیم.



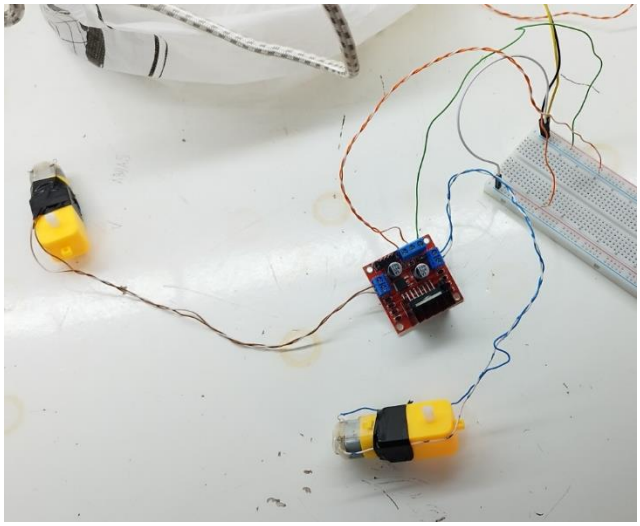




سیستم حرکتی ربات:

ربات برای حرکت نیاز به چرخ، موتور، درایور موتور، منبع تغذیه و برد Raspberry Pi دارد.

دو عدد موتور گیربکس دار را با استفاده از بست های مخصوص به بدنه ربات متصل کرده و سپس دو چرخ هرزگرد را به جلو متصل می‌کنیم.



سیم های دو سر موتور را به L298 متصل می‌شوند تا جهت چرخش موتورها را مشخص کند.

نکته: بعد از لحیم کردن سیم ها به موتورها ابتدا با وارنیش از اتصال سیم ها به بدنه جلوگیری کنید و بعد با چسب برق دور موتورها بپیچید.

درایو موتور:

درایو موتور یک قطعه الکترونیکی می‌باشد که بین میکرو کنترلر و چرخ ها تعامل برقرار می‌کند.

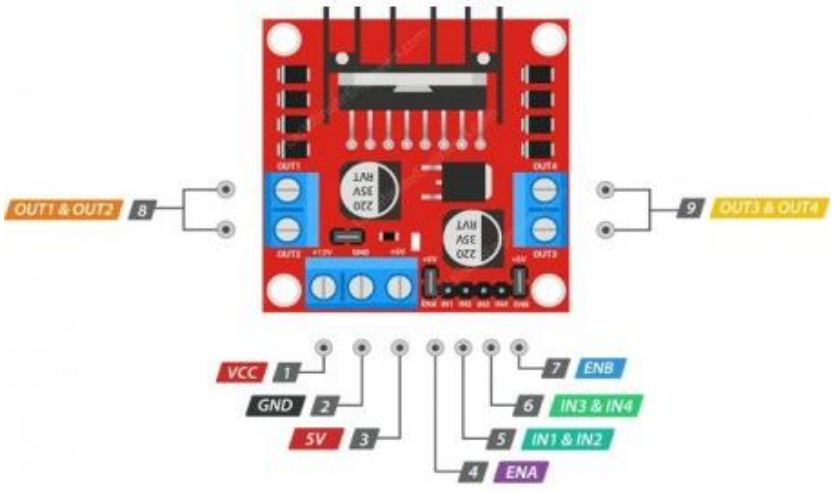
رزبری به صورت مستقیم با پین های کنترلی می‌تواند جهت

چرخ ها و سرعت مورد نیاز آنها را کنترل کند ولی محدودیتی که دارد تامین جریان مورد نیاز موتورها می‌باشد. اگر جریان عبوری از Raspberry Pi هم زیاد باشد، احتمال سوختنش وجود دارد.

از آنجایی که ما از ماژول L298 استفاده می‌کنیم، ۳ تا سیم ورودی دارد و ۴ تا خروجی. البته این ها برای انرژی هستند، چون ۶ پین کنترل هم دارد که جلوتر به آن می‌پردازیم.

ورودی اولی Vcc می‌باشد که ولتاژ مورد نیاز موتورها را فراهم می‌کند و دومی GND می‌باشدکه مستقیم به PCB متصل می‌شود.

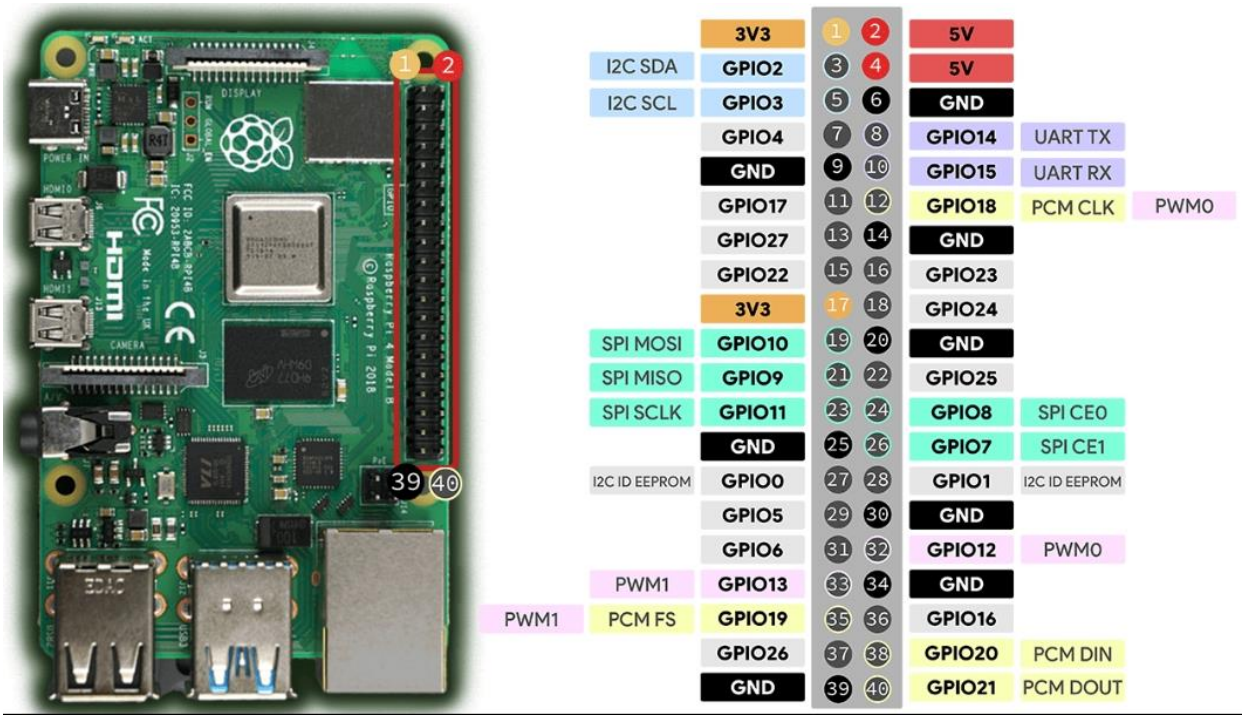
سومی هم ورودی 5v می‌باشد که مستقیم به PCB متصل می‌شود.



حالا باید موتورها را به حرکت دربیاوریم که سیستم کنترلی ما Raspberry Pi 4 می‌باشد.

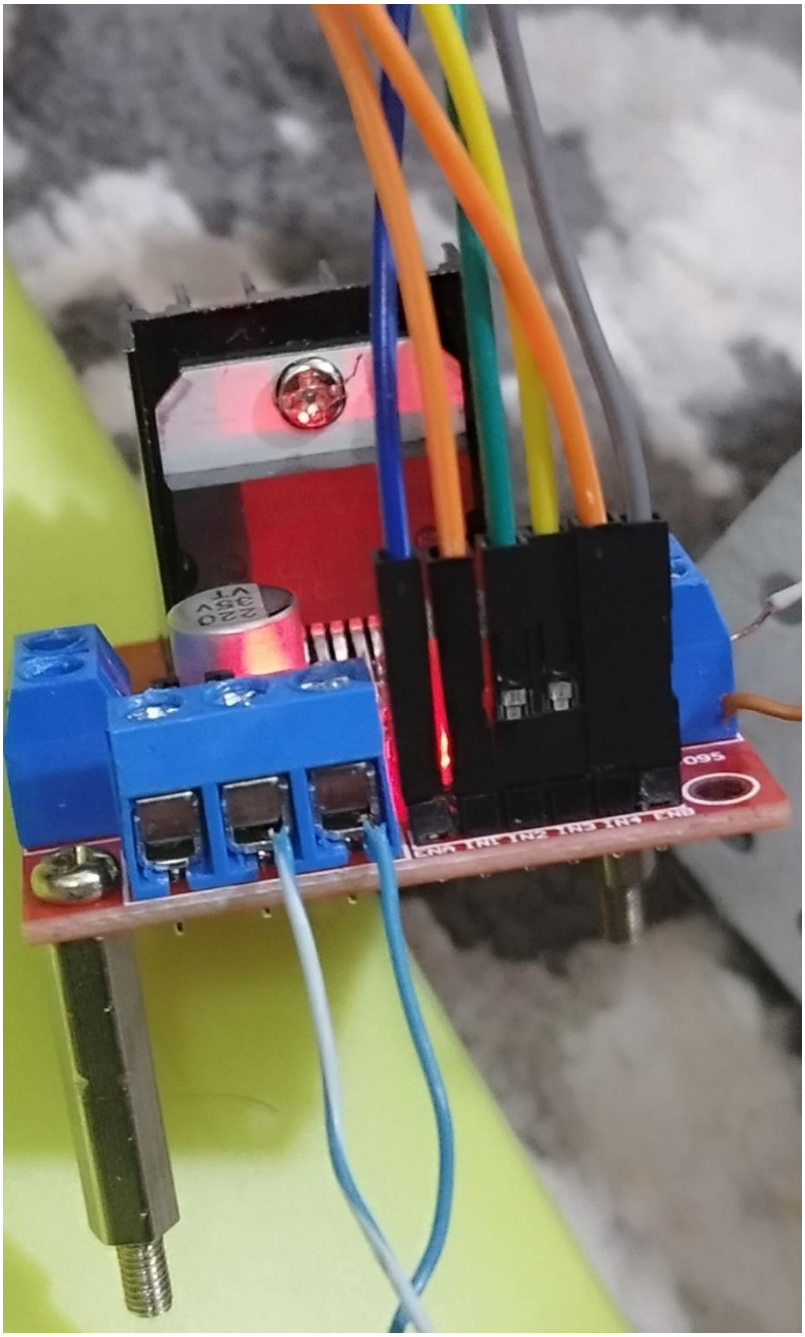
سیستم کنترلی (Raspberry Pi 4):

از منبع تغذیه یک سیم Type C به ورودی برق Raspberry Pi متصل می‌کنیم. روی برد Raspberry Pi ۴۰ عدد پین(Pin) داریم که شماره گذاری این پین ها توسط نرم افزار انجام می‌شود که دو روش PCM و Borad داریم که ما از PCM استفاده کردیم.



از آنجایی که ما باید سرعت موتورها را هم تنظیم کنیم، نیاز به ۶ عدد پین GPIO روی Raspberry Pi داریم و برای کچر(Cacher) یک پین GPIO از نوع PWM و برای شوتر(Shooter) یک پین GPIO معمولی نیاز داریم.

۶ عدد پورت را که دو تای آنها PWM هستند را باید به ENA و ENB متصل کنیم و باید ابتدا جامپرهای روی آنها را حذف کنیم و ۴ پین کنترلی دیگر را هم متصل کنیم.

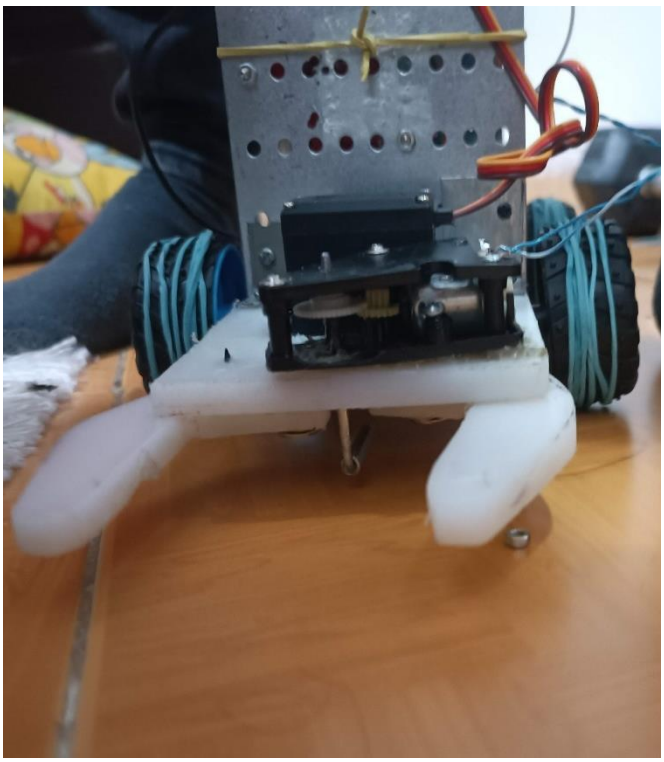


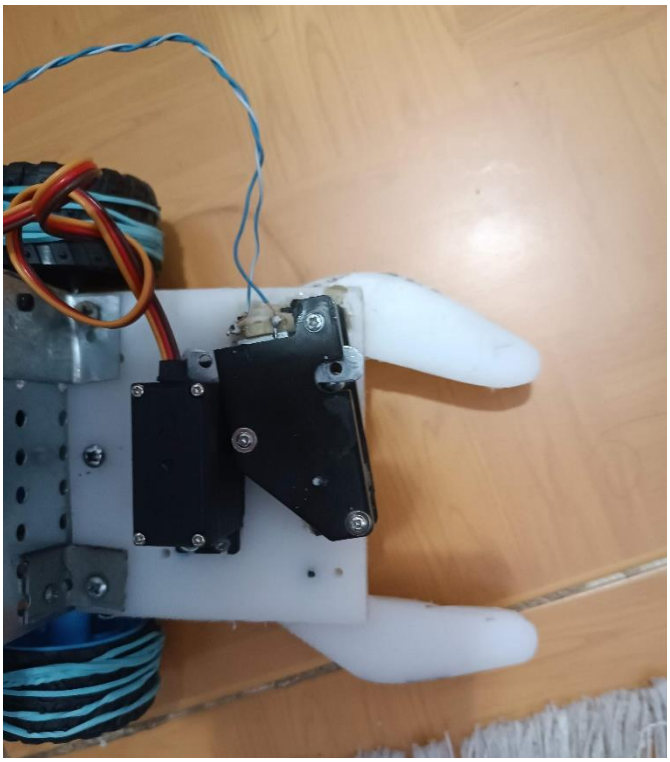
سیستم حرکتی ربات این گونه است که اگر GPIO های متصل به Raspberry Pi، IN2 و IN3 را فعال کنند، ربات به جلو حرکت می‌کند و اگر IN1 و IN4 را فعال کند، ربات به عقب حرکت می‌کند.

برای چرخش به چپ و راست هم به ترتیب (IN3, IN1) و (IN2, IN4) باید فعال شوند. فعال شدن به معنای True شدن و غیر فعال شدن به معنای False شدن پین GPIO روی برد Raspberry Pi می‌باشد.

برای کچر(Cacher) هم یک سیگنال PWM داریم که هر موقع زاویه خاصی از حرکت را اگر ارسال کند، توپ را می‌گیرد و با یک زاویه دیگر توپ را رها می‌کند.

برای شوتر(Shooter) هم فقط یک سیگنال GPIO معمولی داریم و سرعت حرکت با چرخ دنده ها کنترل شده است و نیاز به PWM نداریم و هر موقع GPIO 1 شود، این موتور شروع به حرکت می‌کند.





توضیح نرم افزار و کد ربات

راه اندازی Raspberry Pi:

ابتدا برای راه اندازی Raspberry Pi باید سیستم عامل آنرا روی SD card نصب کنید. برای این کار SD card خود را داخل SD card Reader قرار داده و به کامپیوتر خود وصل کنید.

سپس برنامه Raspberry Pi Imager را که پیشتر به آن اشاره شده بود را باز کنید.

در این قسمت کافیست سیستم عاملی که دانلود کرده‌اید را برای نصب انتخاب کنید.

بعد از انتخاب کارت حافظه در نرم افزار گزینه write را انتخاب کنید. این عملیات ممکن است چند دقیقه طول بکشد.

پس از آن کارت حافظه به رزبری پای خود متصل کنید. پس از این کار، دستگاه‌های جانبی مانند کیبورد، موس و مانیتور را هم به رزبری پای‌تان وصل کرده و درنهایت آن را روشن کنید.

پس از روشن کردن باید کشور و منطقه زمانی را انتخاب کنید.

نام کاربری و رمز را در صفحه ایجاد شده وارد کنید.

سپس رزبری پای را به اینترنت مورد نظر خود وصل کنید.

در نهایت رزبری پای خود را ری استارت کنید تا تنظیمات اعمال شود.

راه اندازی VNC VIEWER و VNC SERVER

ابتدا VNC Viewer را روی رزبری نصب کنید با استفاده از دستور زیر:

"Sudo apt-get install real-vnc-server"

سپس با دستور sudo apt-get update را اجرا کنید تا آپدیت شود و بعد vnc را از طریق دستور زیر فعال کنید.

"sudo raspi-config"

سپس VNC Viewer را روی کامپیوتر تان نصب کنید.

در مرحله بعد کافیست ip نشان داده شده در رزبری را در VNC Viewer وارد کنید و سپس نام کاربری و رمز عبور رزبری‌تان را وارد کنید تا به رزبری از طریق سیستم خودتان دسترسی داشته باشید.

کد های ربات:

همان طور که در تصویر زیر مشخص است به کتابخانه های زیر جهت کنترل ربات با دسته پلی استیشن ، دسترسی به دوربین و پین های رزبری پای و ... نیاز داریم:

```
import threading
import RPi.GPIO as GPIO
from pyPS4Controller.controller import Controller
import cv2
import imutils
import numpy as np
```

متغیرهای زیر نشان دهنده پین های مورد نظر ما برای کنترل ربات هستند که تعریف شده‌اند.

```


FORWARD_IN_1 = 27
FORWARD_IN_2 = 17
BACKWARD_IN_1 = 20
BACKWARD_IN_2 = 21
SHOOTER = 16


```

در کلاس MyController ما تمام پین های مورد نیاز را setup کردیم. مود GPIO را روی BCM تنظیم کردیم و تمام GPIO های از پیش تعیین شده را به همراه سرو(Servo) و کچر فعال کردیم.

```
15  class MyController(Controller):
16
17      def __init__(self, **kwargs):
18          Controller.__init__(self, **kwargs)
19          GPIO.setmode(GPIO.BCM)
20          GPIO.setwarnings(False)
21          ###########
22          GPIO.setup(FORWARD_IN_1, GPIO.OUT)
23          GPIO.setup(FORWARD_IN_2, GPIO.OUT)
24          GPIO.setup(BACKWARD_IN_1, GPIO.OUT)
25          GPIO.setup(BACKWARD_IN_2, GPIO.OUT)
26          GPIO.setup(SHOOTER, GPIO.OUT)
27
28          GPIO.setup(13, GPIO.OUT)
29          GPIO.setup(12, GPIO.OUT)
30          GPIO.setup(19, GPIO.OUT)
31
32          self.servo1 = GPIO.PWM(13, 50)
33          self.servo2 = GPIO.PWM(12, 50)
34          self.catcher = GPIO.PWM(19, 50)
35          self.servo1.start(0)
36          self.servo2.start(0)
37          self.catcher.start(0)
```

در ادامه کارکرد هر کدام از دکمه های دسته پلی استیشن را تعریف کردیم که به شرح زیر می‌باشد:

ضربدر(X): فعال کردن شوتر

مثلث(△): غیر فعال کردن شوتر

مربع(▭): بستن کچر

دایره(O): باز کردن کچر

```
40      def on_x_press(self):
41          GPIO.output(SHOOTER, True)
42
43      def on_triangle_press(self):
44          GPIO.output(SHOOTER, False)
45
46      def on_square_press(self):
47          self.catcher.ChangeDutyCycle(11)
48
49      def on_circle_press(self):
50          self.catcher.ChangeDutyCycle(8)
```

حرکت ربات:

آنالوگ سمت چپ دسته پلی استیشن یا L3 را برای حرکت ربات در نظر گرفته‌ایم. بسته به اینکه چقدر آنرا از حالت ثابت به هر جهت ممکن تکان دهیم سرعت ربات تغییر می‌کند. هر چه بیشتر از حالت ثابت تکان بخورد و فاصله بگیرد، سرعت حرکت بیشتر می‌شود. در تابعی که در تصویر پایین قابل مشاهده است، میزان این سرعت را تعیین کرده‌ایم که به چهار حالت تقسیم میشود:

```
52      def cycle_number_calculator(value):
53          res = 0
54          if value == 0:
55              res = 0
56          if 0 < value <= 2500:
57              res = 25
58          if 2500 < value <= 5000:
59              res = 50
60          if 5000 < value <= 12000:
61              res = 75
62          if 12000 < value:
63              res = 100
64          return res
```

بسته به این که به کدام جهت میخواهیم حرکت کنیم GPIO های مورد نیازمان را فعال میکنیم.

حرکت رو به جلو:

```
66      def on_L3_up(self, value):
67          value = abs(value)
68          cycle_number = cycle_number_calculator(value)
69          GPIO.output(FORWARD_IN_1, True)
70          GPIO.output(FORWARD_IN_2, True)
71          self.servo1.ChangeDutyCycle(cycle_number)
72          self.servo2.ChangeDutyCycle(cycle_number)
```

حرکت رو به عقب:

```
74      def on_L3_down(self, value):
75          value = abs(value)
76          cycle_number = cycle_number_calculator(value)
77          GPIO.output(BACKWARD_IN_1, True)
78          GPIO.output(BACKWARD_IN_2, True)
79          self.servo1.ChangeDutyCycle(cycle_number)
80          self.servo2.ChangeDutyCycle(cycle_number)
```

چرخش به راست:

```
82      def on_L3_right(self, value):
83          value = abs(value)
84          cycle_number = cycle_number_calculator(value)
85          GPIO.output(FORWARD_IN_1, True)
86          GPIO.output(BACKWARD_IN_2, True)
87          self.servo1.ChangeDutyCycle(cycle_number)
88          self.servo2.ChangeDutyCycle(cycle_number)
89
```

چرخش به چپ:

```
90      def on_L3_left(self, value):
91          value = abs(value)
92          cycle_number = cycle_number_calculator(value)
93          GPIO.output(FORWARD_IN_2, True)
94          GPIO.output(BACKWARD_IN_1, True)
95          self.servo1.ChangeDutyCycle(cycle_number)
96          self.servo2.ChangeDutyCycle(cycle_number)
```

بعد از اینکه آنالوگ را به حالت اولیه برگردانیم، تمامی پین های فعال شده را غیر فعال می‌کنیم تا ربات از حرکت بایستد:

```
98      def on_L3_x_at_rest(self):
99          GPIO.output(FORWARD_IN_1, False)
100         GPIO.output(FORWARD_IN_2, False)
101         GPIO.output(BACKWARD_IN_1, False)
102         GPIO.output(BACKWARD_IN_2, False)
103
104     def on_L3_y_at_rest(self):
105         GPIO.output(FORWARD_IN_1, False)
106         GPIO.output(FORWARD_IN_2, False)
107         GPIO.output(BACKWARD_IN_1, False)
108         GPIO.output(BACKWARD_IN_2, False)
```

در کلاس بعدی توابع تشخیص توپ و دروازه را نوشته‌ایم.

تشخیص توپ:

```python
class BallDetector:
    def __init__(self, lower_ball, upper_ball, upper_gate, lower_gate):
        self.LOWER_HSV = lower_ball
        self.UPPER_HSV = upper_ball
        self.UPPER_HSV_GATE = upper_gate
        self.LOWER_HSV_GATE = lower_gate

    def detect_ball(self, frame):
        output = None
        frame = imutils.resize(frame, width=600)
        blurred = cv2.GaussianBlur(frame, (17, 17), 0)
        hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv, self.LOWER_HSV, self.UPPER_HSV)
        mask = cv2.erode(mask, None, iterations=0)
        mask = cv2.dilate(mask, None, iterations=0)
        cnts = cv2.findContours(mask.copy(),
                                cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        center = None

        if len(cnts) > 0:
            c = max(cnts, key=cv2.contourArea)
            ((x, y), radius) = cv2.minEnclosingCircle(c)
            M = cv2.moments(c)
            center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
            output = center
            # print(center)
            if radius > 10 and radius < 170:
                cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)
                cv2.circle(frame, center, 5, (0, 0, 255), -1)
        cv2.imshow("Frame", frame)
        return output
```

در تابع detect_ball رنگ و مرز توپ در محیط اطراف مورد نظر قرار می‌گیرد. برای آنکه توپ هم در محیط کم نور و هم در محیط پر نور قابل تشخیص باشد، باید مرز پایین و بالا در محیط رنگ HSV را برای توپ مشخص کنیم.

بنابراین در هر فریم به دنبال یک جسم دایره شکل با رنگی در بازه مشخص می‌گردیم.

اگر وجود داشته باشد یک دایره ی محیطی حول آن محدوده پیکسل میاندازیم و سپس مرکز دایره را پیدا میکنیم و در خروجی نشان می‌دهیم.

تشخیص دروازه:

تابع detect_gate به شیوه مشابه تشخیص داده می‌شود و در صورت وجود دور آن کادری مشخص می‌شود و مرکز آن درتصویر خروجی نشان داده می‌شود:

```python
    def detect_gate(self, frame):
        output = None

        frame = imutils.resize(frame, width=600)
        blurred = cv2.GaussianBlur(frame, (17, 17), 0)
        hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv, self.LOWER_HSV_GATE, self.UPPER_HSV_GATE)
        mask = cv2.erode(mask, None, iterations=0)
        mask = cv2.dilate(mask, None, iterations=0)
        cnts = cv2.findContours(mask.copy(),
                                cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(cnts)
        if len(cnts) > 0:
            c = max(cnts, key=cv2.contourArea)
            rect = cv2.minAreaRect(c)
            box = cv2.boxPoints(rect)
            box = np.int0(box)
            cv2.drawContours(frame, [box], 0, (0, 255, 0), 2)  # Draw rectangle around the gate
            output = rect[0]  # Return the center of the gate
        cv2.imshow("Frame", frame)
        return output
```

تابع اصلی در تصویر زیر قابل مشاهده می‌باشد:

```
if __name__ == "__main__":
    controller = MyController(interface="/dev/input/js0", connecting_using_ds4drv=False)

    def my_command():
        controller.listen()

    command_thread = threading.Thread(target=my_command)
    command_thread.start()
    ball_detector = BallDetector((13, 99, 132), (24, 255, 255),
                                 (107, 159, 192), (94, 89, 100))
    video = cv2.VideoCapture(0)
    address = 'http://192.168.1.102:8080/video'
    video.open(address)

    while True:
        ret, frame = video.read()
        if not ret:
            break

        print("searching for ball")
        output = ball_detector.detect_ball(frame=frame)
        print(output)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    video.release()
    cv2.destroyAllWindows()
```

در اینجا برای آنکه میخواستیم به طور همزمان کنترل ربات را در اختیار داشته باشیم و هم پردازش تصویر همراه آن اجرا شود، با استفاده از تابع my_command و پاس دادن آن به Command_thread خواندن دستورات از ورودی دسته پلی استیشن شروع به کار می‌کند تا ربات بتواند حرکت کند و دستورات بعد از آن به طور موازی نیز اجرا شوند.

با استفاده از برنامه IP Webcam تصویر گرفته شده در موبایل متصل به ربات را می‌توانیم در صفحه نمایش خود مشاهده کنیم.

تنها کافیست که ip address را به برنامه بدهیم تا در پنجره‌ای فیلم گرفته شده که جلوی ربات را نشان می‌دهد، داشته باشیم.