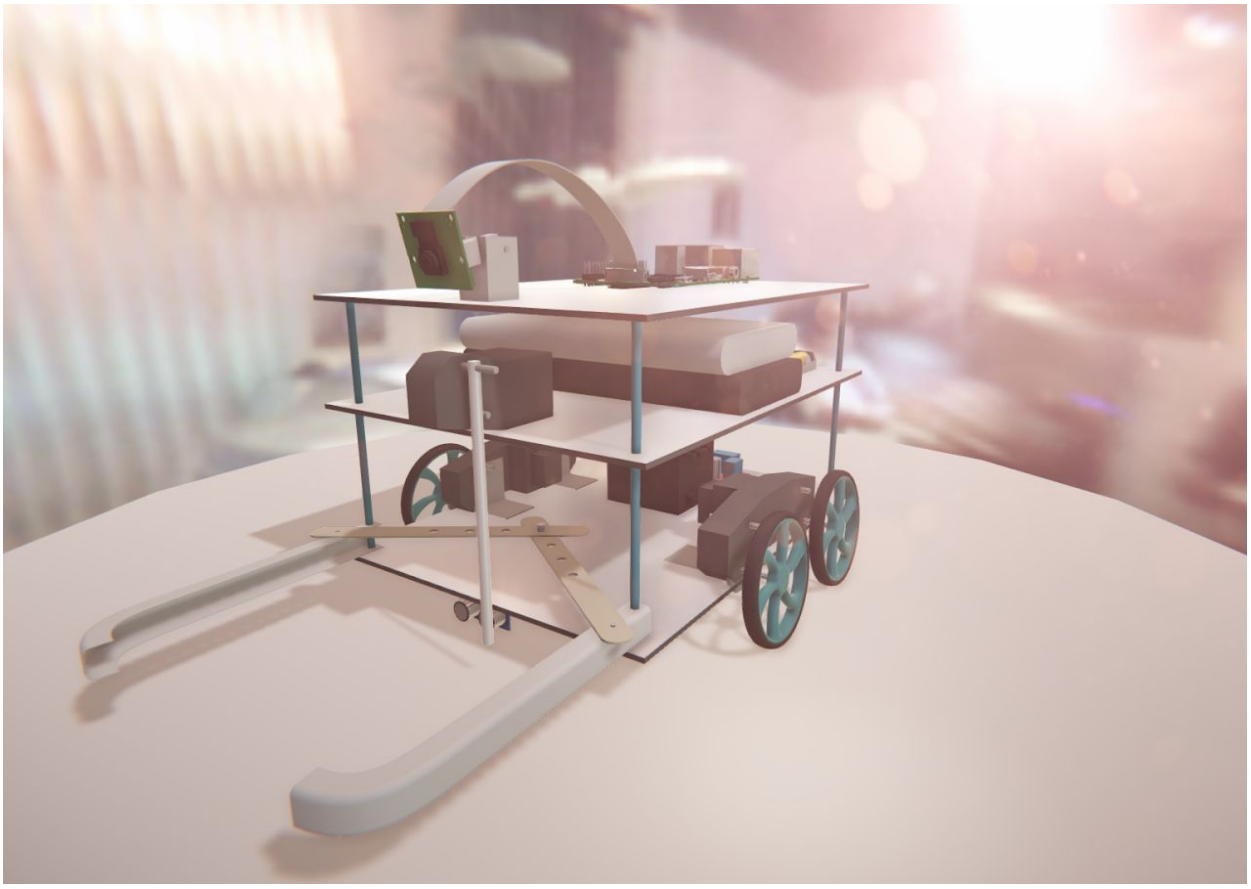
PiBall

پروژه درس معماری کامپیوتر،ربات فوتبالیست



استاد:دکتر جمشیدی

پاییز 97

بخش مهندسی کامپیوتر

دانشکده فنی مهندسی

بخش مهندسی کامپیوتر

محمدحسین علیزاده

غلامرضا دار

حمیدرضا رادفر

علی اسماعیل زاده

ساجده فاریابی

سارا جلالی

فهرست

مقدمه

ربات فوتبالیست PiBall رباتی قدرت گرفته از برد RaspBerry Pi و کتابخانه OpenCv برای پردازش تصویر می باشد که برنامه نویسی ان با زبان برنامه نویسی پایتون صورت گرفته است.

روند پیش روی ربات به این نحو است که ابتدا ربات باید توپ را تشخیص داده و با حرکت به سمت توپ، توپ را گرفته و به سمت دروازه شوت کند.

در این گزارش سعی شده است توضیح کاملی از نحوه انتخاب قطعات تا اسمبل کلی ربات داده شود.

معرفی کلی پروژه

پروژه PIBALL،رباتی فوتبالیست قدرت گرفته از برد RaspberryPi و زبان برنامه نویسی پایتون می باشد.

روندی که طی آن ربات طراحی و ساخته شده است به شکل زیر است.

که سعی شده طی فصل های آینده به طور کامل شرح داده شود

اهداف پروژه

هدف کلی این پروژه ارتباط بخش های مختلف سخت افزاری نظیر: موتور، چرخ ها ، کچر، شوتر،دوربین و... با یکدیگر به وسیله ی دستورات نرم افزاری می باشد که دستورات توسط بورد رزبری پای به سایر اجزا ارسال می شود.

این دستورات به صورت کدهایی به زبان پایتون در رزبری نوشته شده که تشخیص می دهد توپ کجا قرار دارد و برای رسیدن به آن باید کدام چرخ ها حرکت کنند و کچر چه زمانی باید شروع به بسته شدن و بازشدن کند و شوتر عمل نماید.

از دیگر اهداف این پروژه می توان به آشنایی با قطعات الکترونیکی نظیر:درایور موتور، سروو موتور، بورد رزبری پای و نحوه ی کار با آن ها، چگونگی دریافت تصاویر و پردازش تصویر، چگونگی اتصال موتورها به درایورموتور، نحوه اتصال سروو موتور به کچر و رزبری پای، نحوه کار با رزبری پای و چگونگی نصب سیستم عامل بر روی آن اشاره کرد.

چالش های پیش رو

در ابتدا باید مشخص کنیم که از رباتمان چه می خواهیم و انتظار انجام چه کارهایی را از آن داریم؟

یک ربات فوتبالیست ابتدا باید بتواند توپ مورد نظر را تشخیص دهد و به سمت آن حرکت کند و سپس توپ را بگیرد و آن را به سمت دروازه شوت کند.

ابتدا باید قطعاتی شامل : بورد،چرخ،دوربین،درایور موتور،موتور،منبع تغذیه را انتخاب خریداری کنیم.

پس از تهیه سخت افزار لازم باید سیستم عامل رسپین را بر روی برد رزبری پای نصب کنیم. کنیم.

از دیگر چالش های پیش رو میتوان به کامپایل و نصب کتابخانه OpenCv برای برد رزبری پای،تامین نیروی کافی برای قطعات الکتریکی و خرید قطعات مناسب اشاره کرد.

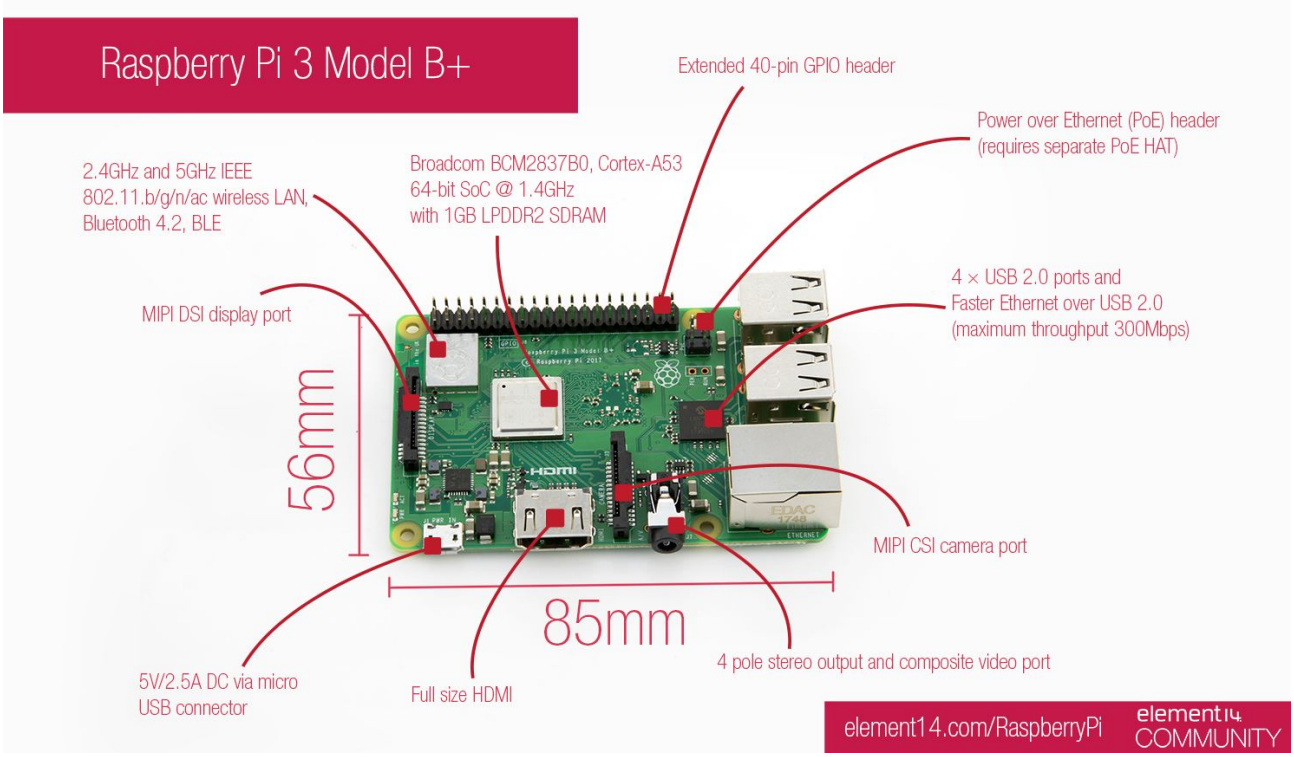
فصل ۱

انتخاب قطعات

بدون شک برای ساخت یک ربات خوب و کارآمد که توانایی انجام دستورات را در کمترین زمان و دقت بالا داشته باشد نیازمند انتخاب قطعات و سخت افزار مناسب هستیم . پس از بررسی ها و مقایسه انواع قطعات شامل برد ، درایور و موتور های مختلف به نتایجی رسیدیم که در ادامه به شرح هر یک از آنها می پردازیم.

لیست قطعات

1. Raspberry pi 3b+
2. Pi camera
3. Heatsink
4. L298 dc motor driver
5. Srf05 Ultrasonic
6. mg996r servo motor
7. Dc motor



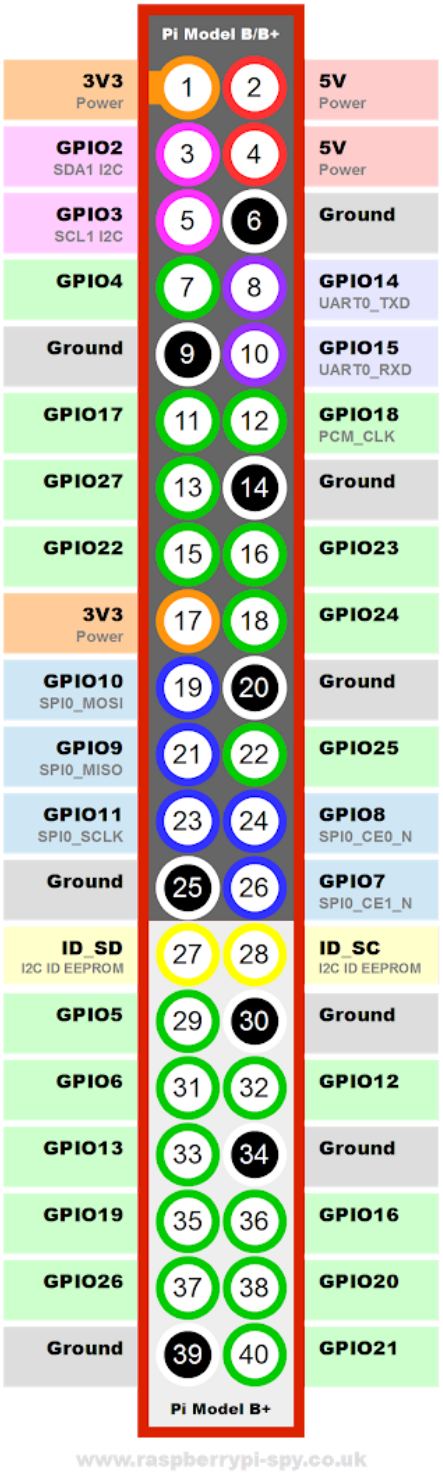
Raspberry pi 3b+

پس از بررسی انواع بورد های موجود در بازار و مقایسه آن ها و بنا به نیاز و هدفی که داشتیم باید بوردی را انتخاب می کردیم که توانایی پردازش اطلاعات در سرعت بالا را داشته باشد و بتوانیم با اطلاعات برنامه نویسی که از قبل داریم بورد را برنامه ریزی کنیم .

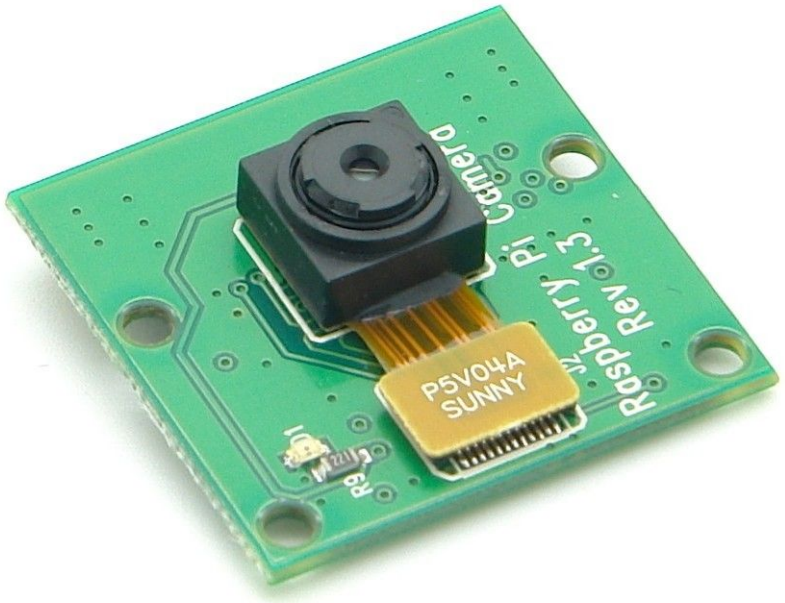
هدر بورد های 1768 ، 2368 و رزبری پای برای هدف ما مناسب تر بودند.

با تحقیق و پرسش از اطرافیان به این نتیجه رسیدیم که رزبری پای نسبت به دیگر بورد های موجود توان بیشتری در پردازش دارد و سهولت بیشتری در کار با GPIO ها و سخت افزار های جانبی دارد.

از دیگر مزیت های رزبری پای نسبت به دیگر بورد ها قابلیت پشتیبانی از سیستم عامل های لینوکس است



دیتاشیت GPIO های raspberry pi 3

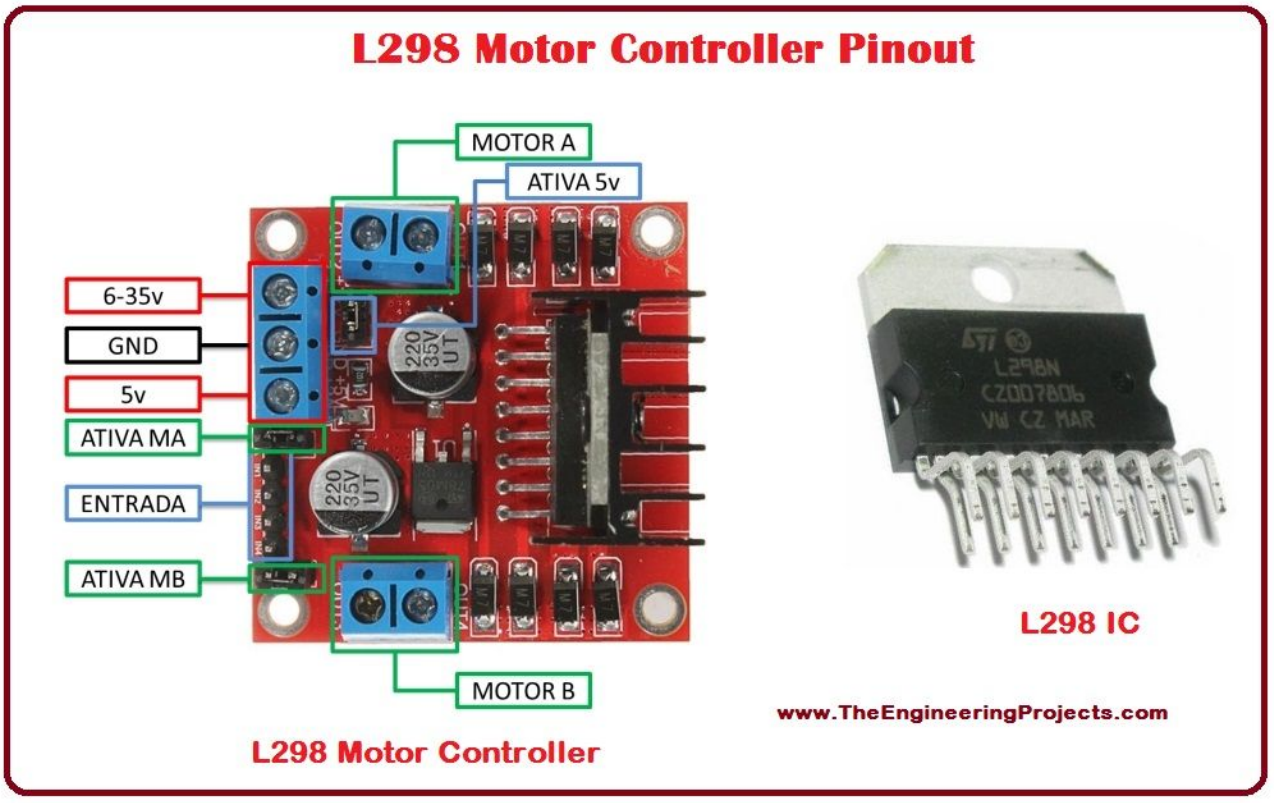


Pi camera

برای پردازش تصویر نیازمند دوربینی با کیفیت مناسب هستیم که آن را به بورد وصل کنیم برای این کار میتوان از دوربین های USB یا دوربین های رسمی مناسب بورد رزبری استفاده کرد . که پورت مناسب دوربین رزبری برای اتصال روی بورد قرار گرفته است.پس مناسب ترین انتخاب دوربین pi camera 5mp است

مشخصات دوربین

- Fully Compatible with Both the Model A and Model B Raspberry Pi
- 5MP Omnivision 5647 Camera Module
- Still Picture Resolution: 2592 x 1944
- Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
- 15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board
- Size: 20 x 25 x 9mm
- Weight 3g
- Fully Compatible with many Raspberry Pi cases



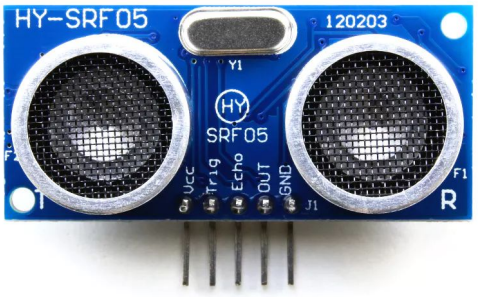
L298 Dc Motor Driver

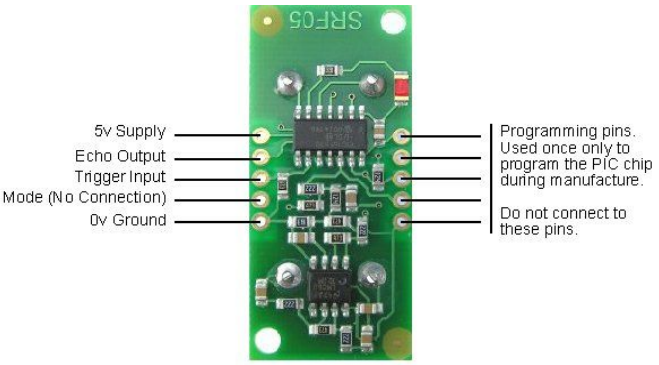
از این ماژول برای راه اندازی موتور های DC استفاده میشود. با استفاده از این ماژول میتوانید سرعت و جهت حرکت همزمان دو موتور DC را کنترل کنید.از یکی از ماژول ها برای کنترل دو موتور حرکتی و از یکی از آن ها برای کنترل شوتر استفاده کردیم.

ویژگی ها

مشخصات

1. L298N Motor driver IC
2. L2937 5V low-dropout voltage regulator
3. (channel motor driver (Dual H-Bridge 4
4. 2.5A continuous drive current per channel
5. 6-26VDC supply
6. 40mm (1.53") square footprint





Connections for 2-pin Trigger/Echo Mode (SRF04 compatible)

Srf05 Ultrasonic

ماژول التراسونیک ,ماژولی برای محاسبه مسافت بر اساس مدت زمان ارسال و دریافت موج تریگر است

مشخصات

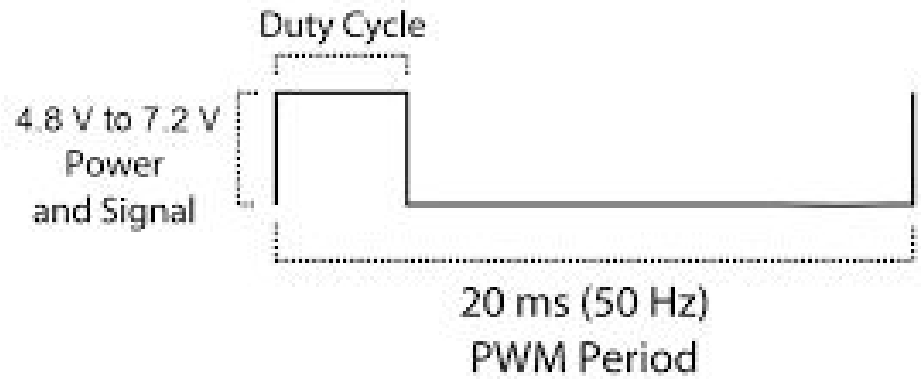
- Voltage: 4.5 to 5.5 VDC
- Sound frequency: 40 KHz
- Trigger Pin format: 10 uS pulse
- Detection distance: 2 to 450 cm

MG996 Servo Motor

MG996 یک سروو موتور با سایز استاندارد و گیربکس فلزی میباشد.

این موتور با دو بلبرینگ مناسب برای کار های قدرتی میباشد و سرعت چندانی ندارد. از این سروو در گریپر ربات استفاده کردیم.

انتخاب سروو برای گریپر به علت قدرت موتور و توانایی کنترل نسبتا دقیق ان است



فصل 2

تست ایزوله قطعات:

هدف از این بخش تست کردن تک تک قطعات و اطمینان از سلامت فیزیکی آنها بود.

همچنین رفع کردن باگها در این حالت بسیار ساده تر است

1.شووتر (Shooter)

برای شووتر از یک موتور DC گیربکس دار استفاده کردیم.و برای کنترل آن از درایور موتور L298 .

اتصالات GPIO را مشابه کد برقرار کردیم و قطعه را تست کردیم.

```
def Shoot(sec):
    # set gpio pins
    init()
    # Shoot
    gpio.output(24,True)
    gpio.output(25,False)
    time.sleep(sec)
    # Wait
    gpio.output(24,False)
    gpio.output(25,False)
    time.sleep(1)
    # Reset Shooter
    gpio.output(24,False)
    gpio.output(25,True)
    time.sleep(sec)

    gpio.cleanup()
```

2.کچر یا گریپر (Catcher, Gripper)

برای کچر از یک موتور Servo استفاده کردیم و برای کنترل آن مستقیما آن را به GPIO رزبری پای متصل کردیم. اتصالات GPIO را مشابه کد برقرار کردیم و قطعه را تست کردیم.

```
def SetAngle(angle,servo,pwm):
    duty = angle / 18 + 2.5
    GPIO.output(servo, True)
    pwm.ChangeDutyCycle(duty)
    sleep(1)
    GPIO.output(servo, False)
    pwm.ChangeDutyCycle(0)

def Catch(is_open=True):
    #Init
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(23, GPIO.OUT)
    pwm=GPIO.PWM(23, 50)
    pwm.start(0)
    if is_open:
    SetAngle(100,23,pwm)
    else:
    SetAngle(0,23,pwm)
    pwm.stop()
    GPIO.cleanup()
```

3.چرخ ها(Wheels / Motors)

برای کنترل چرخ ها از دو موتور DC گیربکس دار استفاده کردیم.و برای کنترل آنها از درایور موتور L298 .

اتصالات GPIO را مشابه کد برقرار کردیم و چرخ ها را تست کردیم.

```
def Move(sec,dir):
     init()
     gpio.output(18,dir)
     gpio.output(22,not dir)
     gpio.output(17,dir)
     gpio.output(27,not dir)
     time.sleep(sec)
     gpio.cleanup()

def Turn(sec,dir):
     init()
     gpio.output(18,dir)
     gpio.output(22,not dir)
     gpio.output(17,not dir)
     gpio.output(27,dir)
    time.sleep(sec)
     gpio.cleanup()
```

تابع **Move** برای حرکت ربات به سمت جلو یا عقب استفاده میشود که با استفاده از متغیر dir جهت را انتخاب میکنیم. و در این حالت موتور های چپ و راست در یک جهت شروع به حرکت میکنند.

تابع **Turn** برای چرخیدن ربات به طرفین استفاده میشود و جهت چرخش با استفاده از dir تعیین میشود. در این حالت موتور ها خلاف جهت هم میچرخند و عمل چرخش بوجود می آید .

4.رزبری پای

برای اطمینان از صحت رزبری پای و راه اندازی اولیه و نصب نرم افزارهای مورد نیاز . یکبار بورد را بدون هیچ وسیله اضافه ای راه اندازی کردیم و تنظیمات لازم را انجام دادیم . آموزش انجام دادن هر یک از این اعمال در بخش **ضمیمه** گزارش آمده است.

هر قطعه ای برای اتصال به رزبری پای از پین های GPIO استفاده میکند.

برای کار با پین های GPIO به طور کلی دو مرحله را پیش رو داریم.

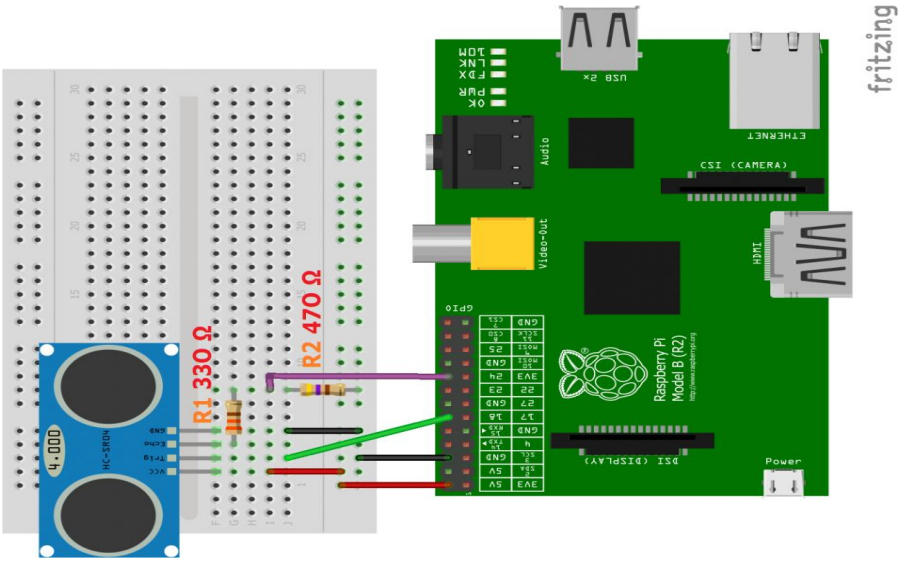
1. تنظیم کردن نوع پین مورد نظر به عنوان یک Input یا Output
2. فرستادن یا دریافت کردن اطلاعات از آن پین

Example:

```
GPIO.setup(23, GPIO.OUT)
GPIO.Output(23, True)
GPIO.Output(23, False)

GPIO.setup(17, GPIO.In)
Answer = GPIO.Input(17)
```

5.سنسور فاصله سنج (UltraSonic)



اتصالات مورد نیاز این سنسور را برقرار میکنیم و با کد زیر قطعه را تست میکنیم.

```
def distance():
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)
    StartTime = time.time()
    StopTime = time.time()
    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()
    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()
    TimeElapsed = StopTime - StartTime
    distance = (TimeElapsed * 34300) / 2
    return distance
```

فصل 3

برای پردازش تصویر ورودی ربات دو کتابخانه OpenCv و Pillow مطرح است که با مقایسه کارایی این دو کتابخانه و تابع هایی متعدد OpenCv تصمیم به استفاده از کتابخانه OpenCv شد

Opencv چیست؟

OpenCV یک کتابخانه متن باز- با لایسنس BSD برای توسعه دهندگان نرم افزارهای بصری و پردازش تصویر است که در سال 2000 توسط شرکت Intel پا به دنیای کامپیوتر نهاد.

در حال حاضر ورژن های مختلفی از این کتابخانه موجود می باشد که معروفترین آنها ورژن ۲.۴ و جدیدترین آنها ورژن 3.1 است که آخرین ورژن پایدار محسوب می شود.

این کتابخانه با زبان C/C++ نوشته شده است ولی تقریبا در تمام زبان های برنامه نویسی معروف از جمله Python قابل دسترسی است.

چه نرم افزارهایی از OpenCV استفاده می کنند ؟

نرم افزارهای بسیار زیادی از OpenCV استفاده می کنند اما با توجه به صفحه ی ویکی پدیا می توان به موارد زیر اشاره کرد.

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking
- Augmented reality

OpenCV بر روی چه سیستم عامل هایی اجرا می شود ؟

OpenCV در سیستم عامل های مختلف اجرا می شود که از مهمترین آنها می توان به این موارد اشاره کرد.

Desktop: Windows, Linux, OS X, FreeBSD, NetBSD, OpenBSD

Mobile: Android, iOS, Maemo, BlackBerry 10

برای پردازش تصویر لازم برای این ربات ابتدا باید مباحث مربوط به

Object Tracking را در این کتابخانه یادگرفت.

پس از مطالعه داکیومنشن Opencv متوجه این شدیم که روند ردیابی اشیا به صورت کلی بر دو اساس رنگ و شکل است.

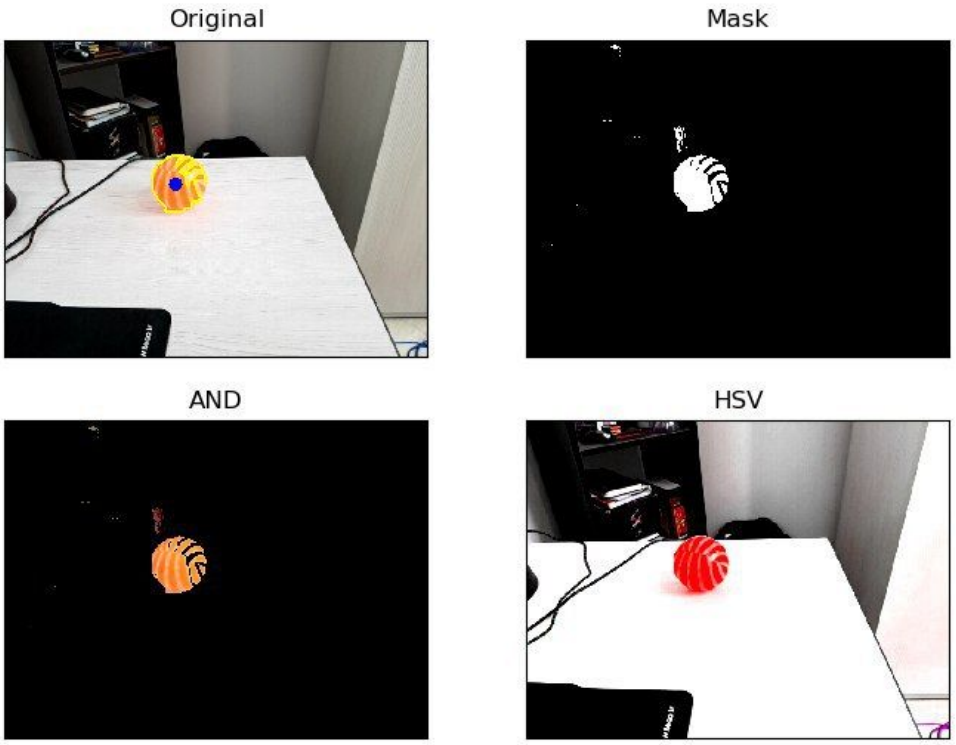
ما در این پروژه بر اساس نیاز از رنگ برای تشخیص شی استفاده کردیم

روند کلی به این شکل است که از دوربین تصویر را در هر فریم دریافت میکنیم و پردازش روی تصویر صورت میگیرد,انجام این کار روی هر فریم به ما پردازش

real time را میدهد.

برای هر فریم ابتدا با استفاده از تابع mask_generator با ورودی های تصویر(تصویر ورودی این تابع باید فرمت HSV داشته باشد) و دادن بازه رنگی دلخواه ماسکی تولید میشود که با ان میتوانیم تمام نقاطی که در تصویر در بازه رنگی ما هستن را مشخص کنیم.

با استفاده از ماسک تولید شده و تابع bitwise_and عکس ورودی را در خودش and میکنیم(شکل صفحه بعد).



حال با داشتن موقعیت توپ در هر لحظه می توانیم تصمیم گیری لازم را جهت کنترل ربات انجام دهیم

فصل 4

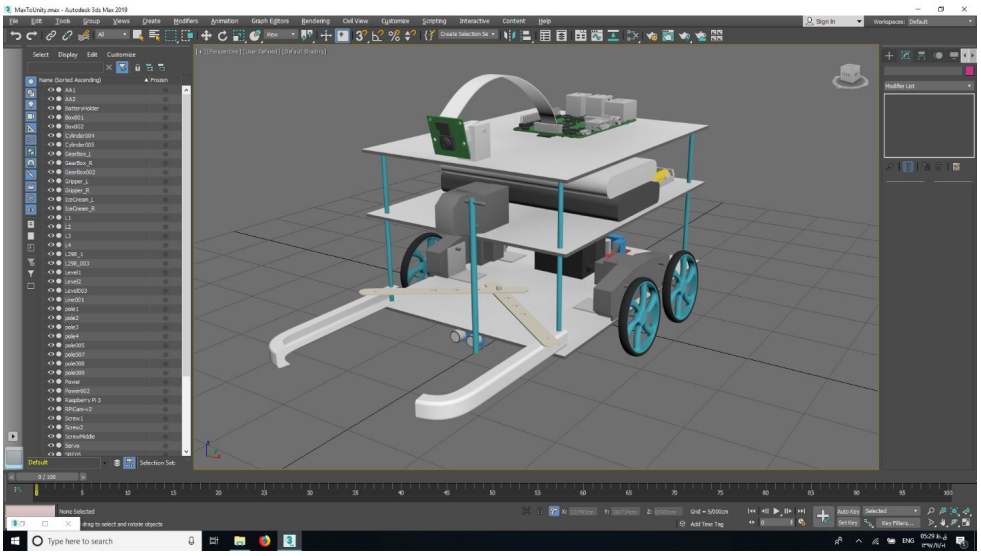
مدل سازی و طراحی شاسی

روندی که ما برای طراحی شاسی پیش گرفتیم به این صورت بود که ابتدا مدل 3D ربات را با اندازه های واقعی در نر افزار 3Dmax ساختیم تا بتوانیم قطعات مختلف سازنده ربات را به بهترین شکل درون طبقات قرار دهیم و اتصالات را برقرار کنیم.

با این کار مشکلاتی از قبیل جا نشدن قطعات کنار هم و کمبود فضا و ... را به حداقل رساندیم.

روند کلی ساخت بدنه به این شکل بود که :

- قطعات خریداری شده را یکی یکی مدلسازی کردیم
- مقیاس قطعات را با یکدیگر هماهنگ کردیم تا اندازه نسبی یکسانی داشته باشند
- بدنه اصلی را مدل سازی کردیم
- قطعات را یکی یکی با توجه به اتصالات آنها به یکدیگر روی طبقات قرار دادیم
- در محیط شبیه سازی شده مکانیزم ها را تست کردیم



مدل نهایی

معرفی مختصر نرم افزار 3Dmax

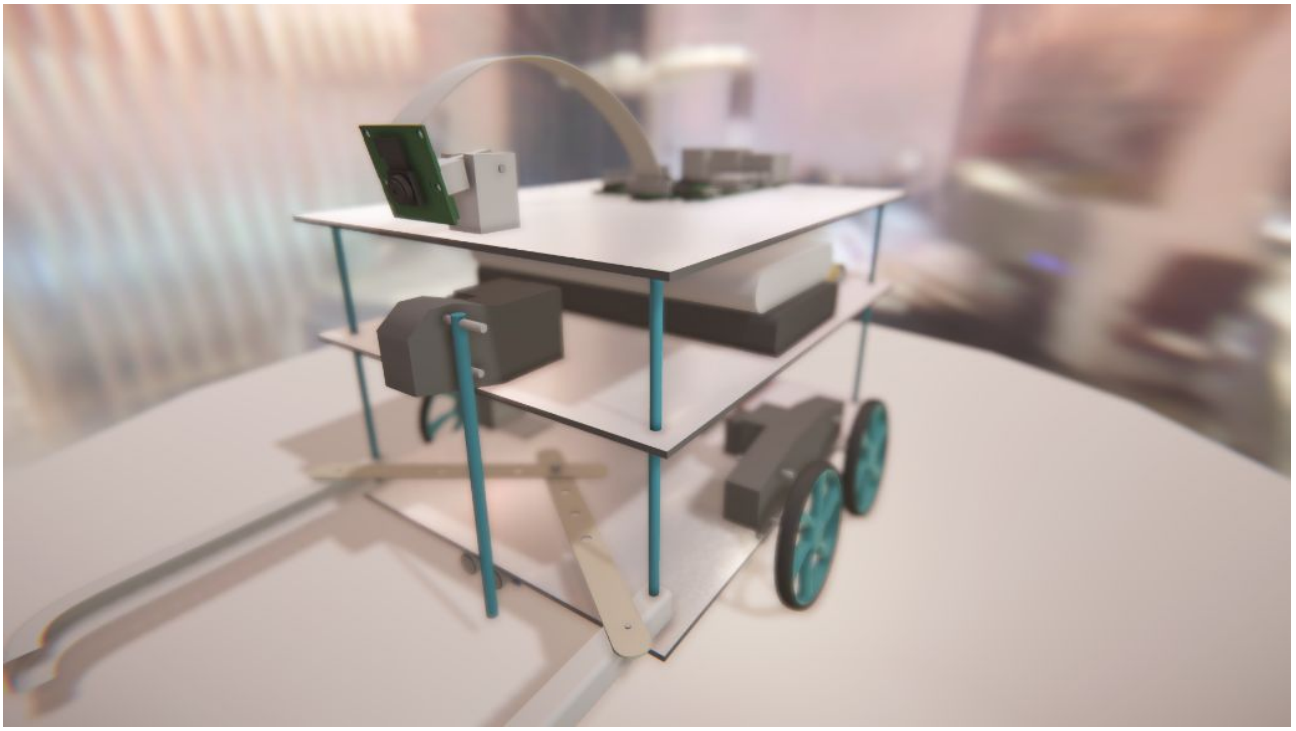
اتودسک تری‌دی‌اس مکس (به انگلیسی: Autodesk 3ds Max) که سابقاً استودیوی سه‌بعدی مکس (به انگلیسی: 3D Studio Max) نامیده می‌شد، یک برنامهٔ گرافیک سه‌بعدی رایانه حرفه‌ای است که برای ساخت پویانمایی‌ها، مدل‌ها، بازی‌ها و تصاویر سه‌بعدی استفاده می‌گردد. این نرم‌افزار توسط شرکت رسانه و سرگرمی اتودسک ساخته و منتشر شده‌است. برنامه قابلیت‌های مدل‌سازی و ساختار قابل انعطاف افزایه داشته و می‌تواند بر روی سکوهای رایانش مایکروسافت ویندوز اجرا شود. تری‌دی‌اس مکس مرتب توسط توسعه‌دهندگان بازی‌های ویدئویی، بسیاری از استودیوهای تبلیغات تلویزیونی و استودیوهای مجسمسازی معماری مورد استفاده قرار می‌گیرد. از این برنامه همچنین در جلوه‌های ویژه سینمایی و مجسمسازی اولیه فیلم‌ها استفاده می‌شود.

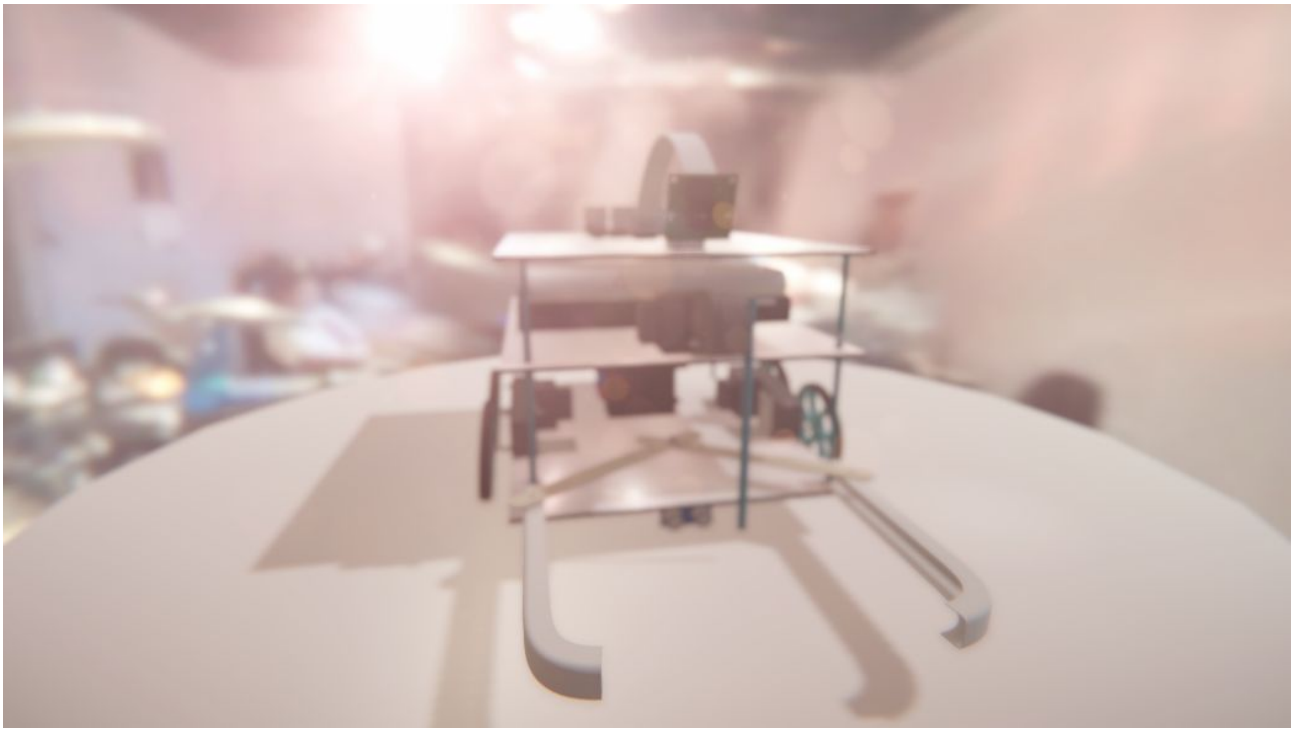
معرفی مختصر نرم افزار یونیتی

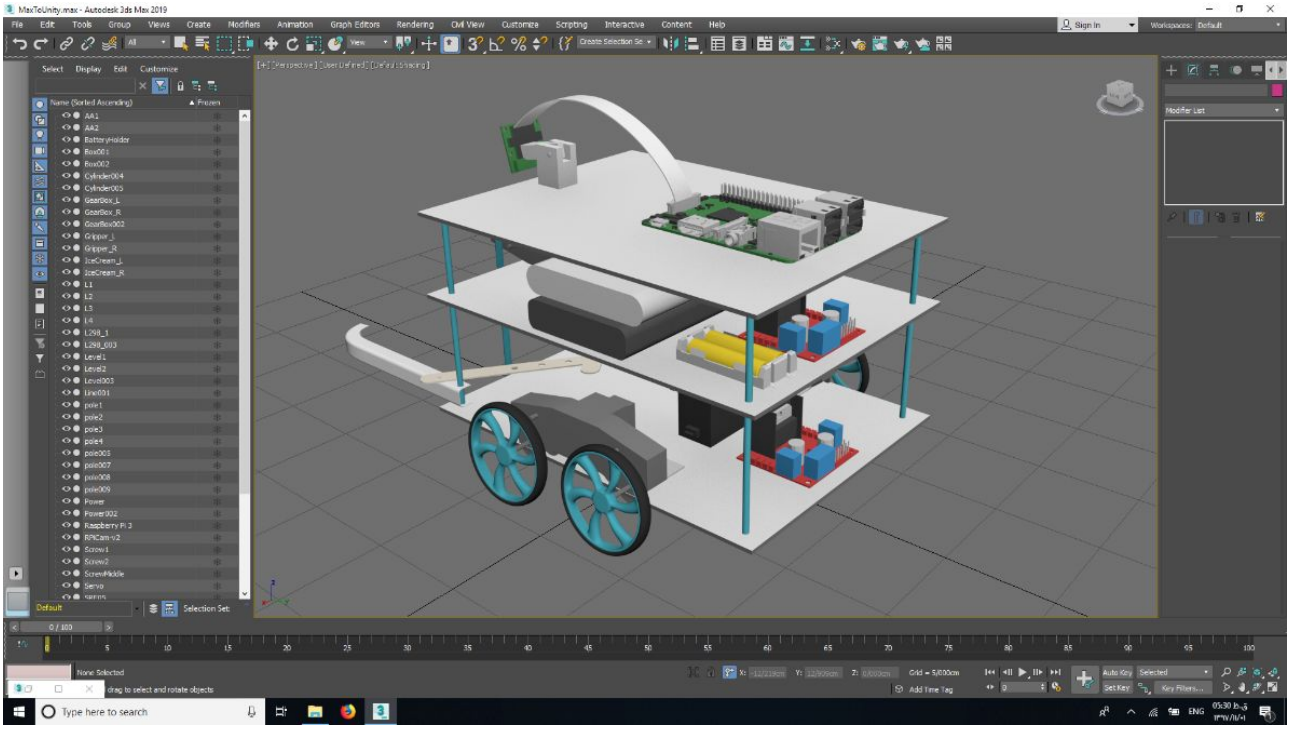
یونیتی (به انگلیسی: Unity) یک موتور بازی چند سکویی است که توسط فناوری‌های یونیتی (Unity Technologies) ساخته شده است و در ساخت بازی ویدئویی برای کامپیوترهای شخصی، کنسول‌های بازی، دستگاه‌های همراه و وب‌سایت‌ها استفاده می‌شود. برای اولین بار در کنفرانس جهانی توسعه‌دهندگان اپل در سال ۲۰۰۵ برای سیستم عامل OS X معرفی شد و از آن زمان تا به حال بر روی بیست و یک سکوی دیگر توسعه یافته است.

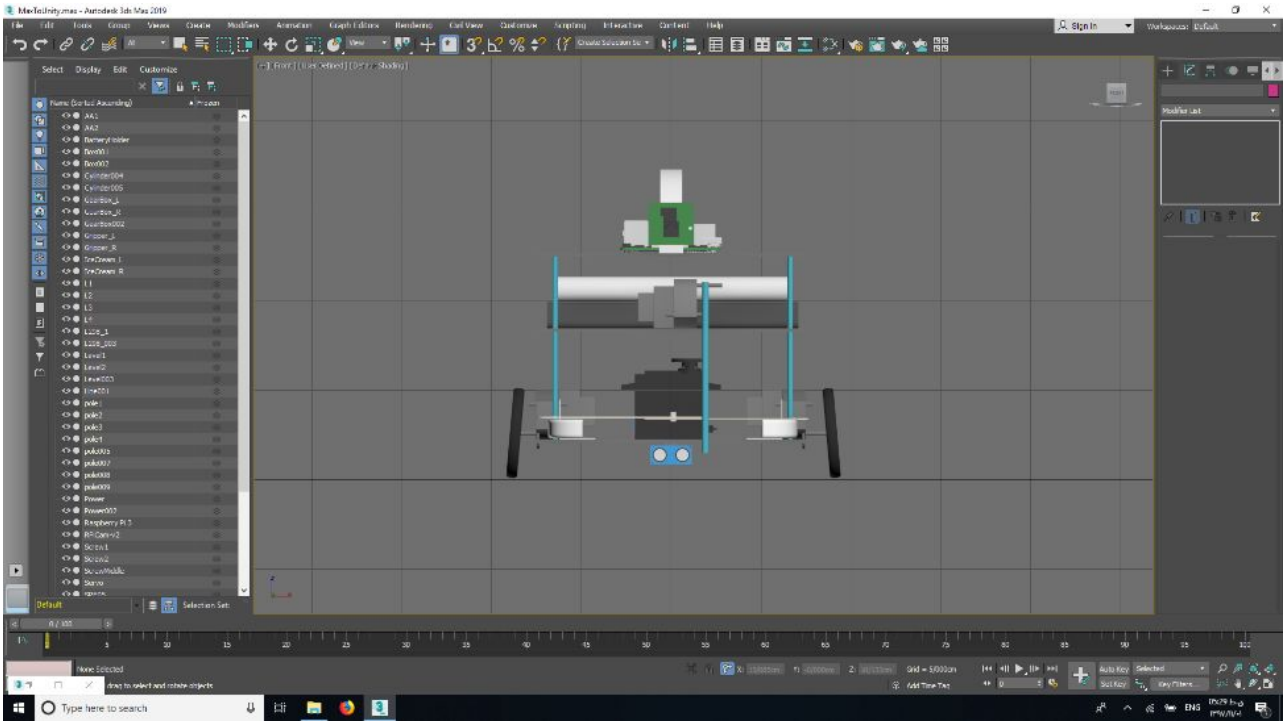
در مرحله بعدی برای شبیه سازی و تست ربات آن را به موتور Unity منتقل کردیم و یک خروجی قابل کنترل توسط کاربر برای درک نحوه کلی عملکرد ربات تهیه کردیم .

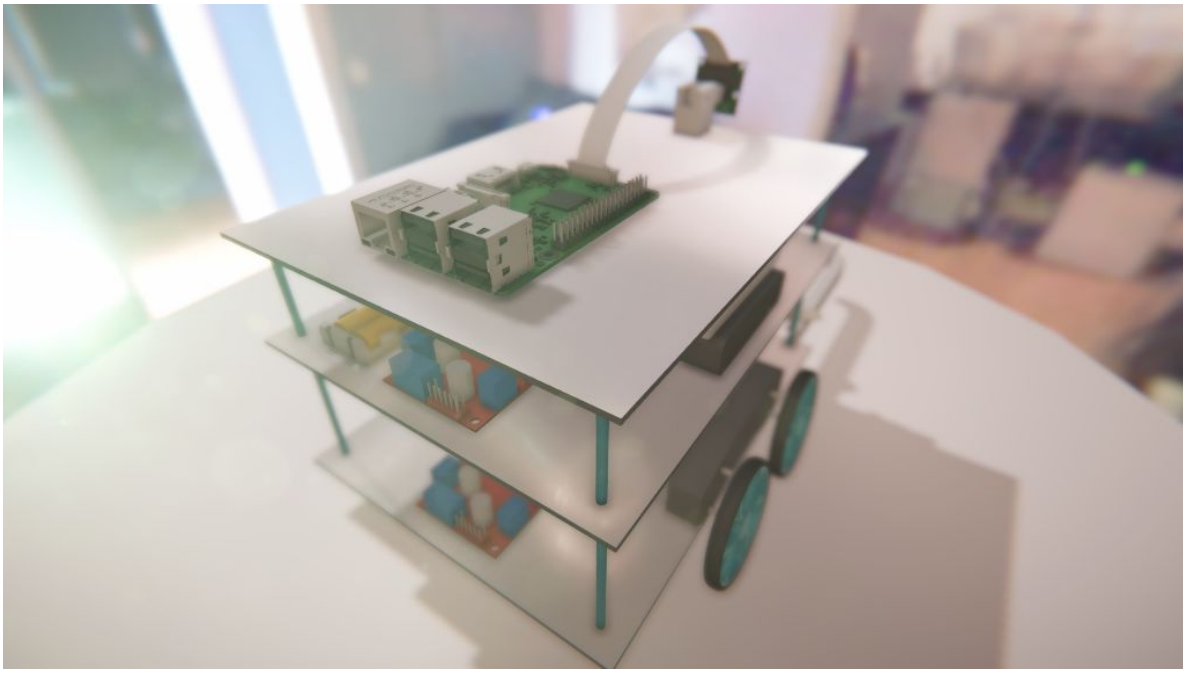
نمای 4 گانه از ربات

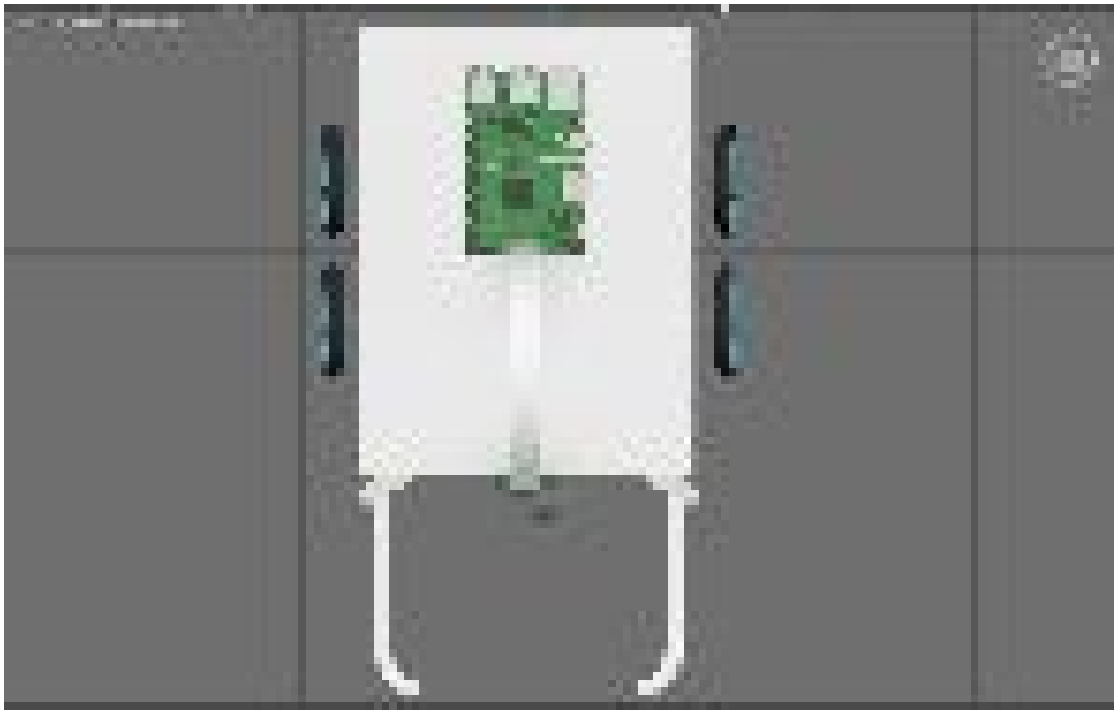


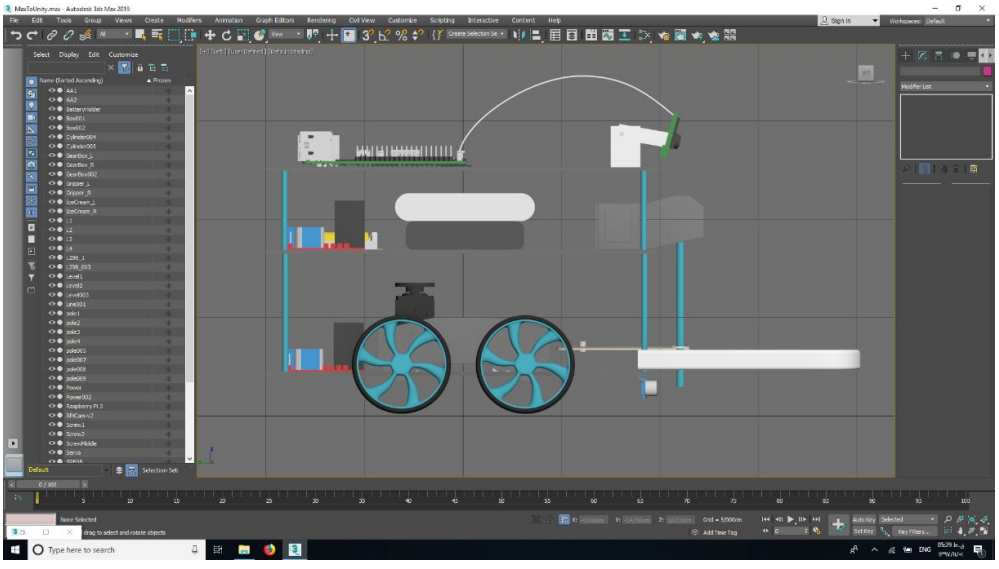












فصل 5

طراحی شاسی

س از طراحی نمونه کانسپت شاسی با استفاده از نرم افزار MAX 3D و در اختیار داشتن رندر های مختلف از آن نوبت به انتخاب متریال مناسب برای ساخت ربات رسید.

پلاستیک فشرده ، ورق های سقف کاذب ، ورق های MDF، ورق آلومینیومی با ضخامت های مختلف و ورق های کامپوزیت گزینه های انتخابی برای ساخت بدنه بودند . شاسی مناسب باید سبک باشد تا مجموع وزن ربات در پایان کار باعث پایین آمدن راندمان آن نشود .

برش پذیر باشد تا بتوانیم تمامی قطعات اعم از بورد ، درایور موتور موتور DC ، دوربین و منبع تغذیه (Bank Power) را روی آن به به وسیله پیچ نصب کنیم همچنین بتوانیم اتصال هایی در هر سطح برقرار کنیم تا طبقات مختلف به یکدیگر وصل شوند .

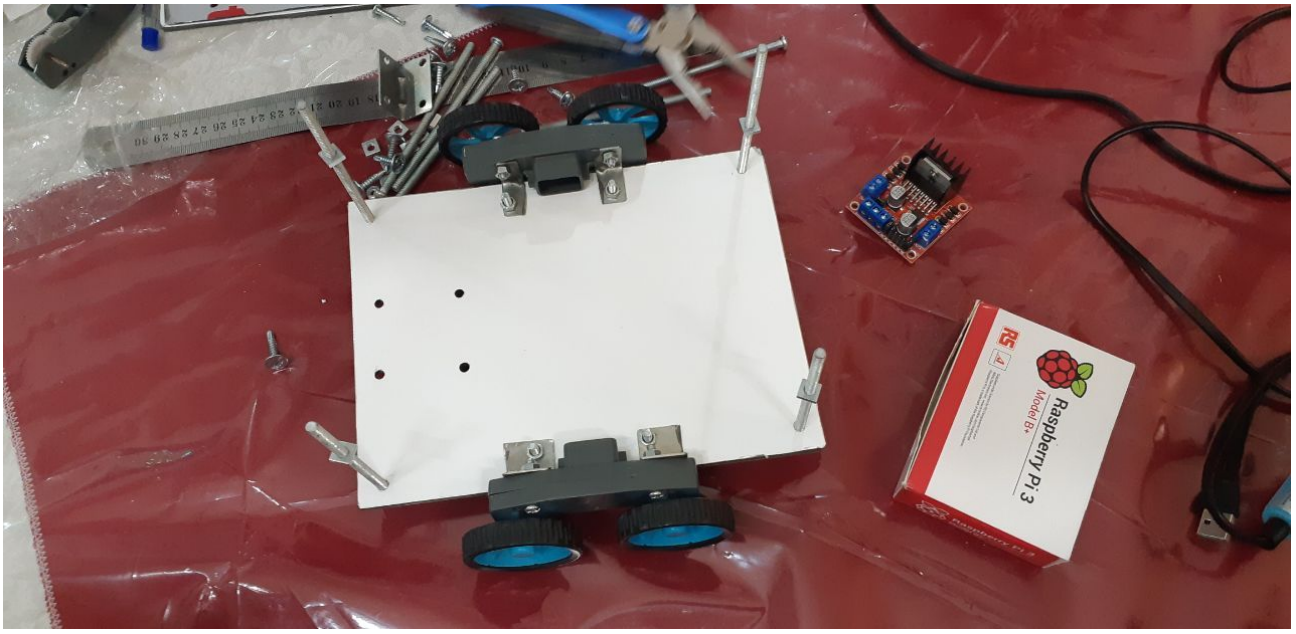
همچنین شاسی باید استحکام زیادی داشته باشد تا بعد از اسمبل قطغات روی آن تغییر شکل ندهد و بتواند حرکت کند .

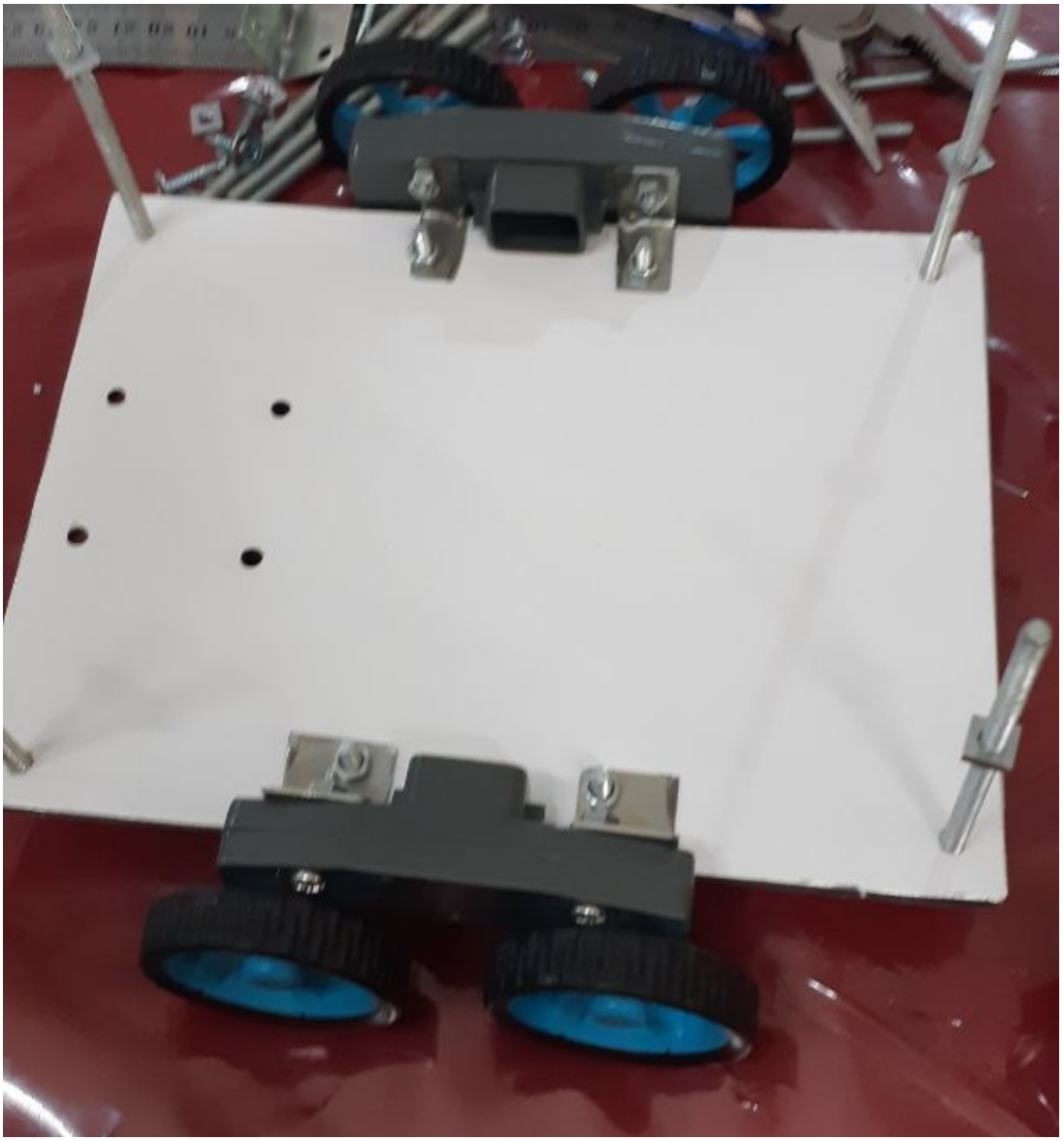
بدنه ربات از جنس ورقه آلومینیومی دوالیه انتخاب شد تا تمامی موارد بالا در آن صدق کنند .

بعد از خریداری ورق های آلومینیومی به اندازه لازم و پس از اندازه گیری خط برش ها ، با استفاده از دستگاه سنگ فرز سه تا از ورقه ها را برش دادیم و لبه ها را با سمباده برقی صاف کردیم تا شکل ظاهری مناسبی داشته باشند .

سپس سخت افزار ها را روی بدنه گذاشتیم و مکان جایگذاری آن ها را علامت گذاری کردیم .

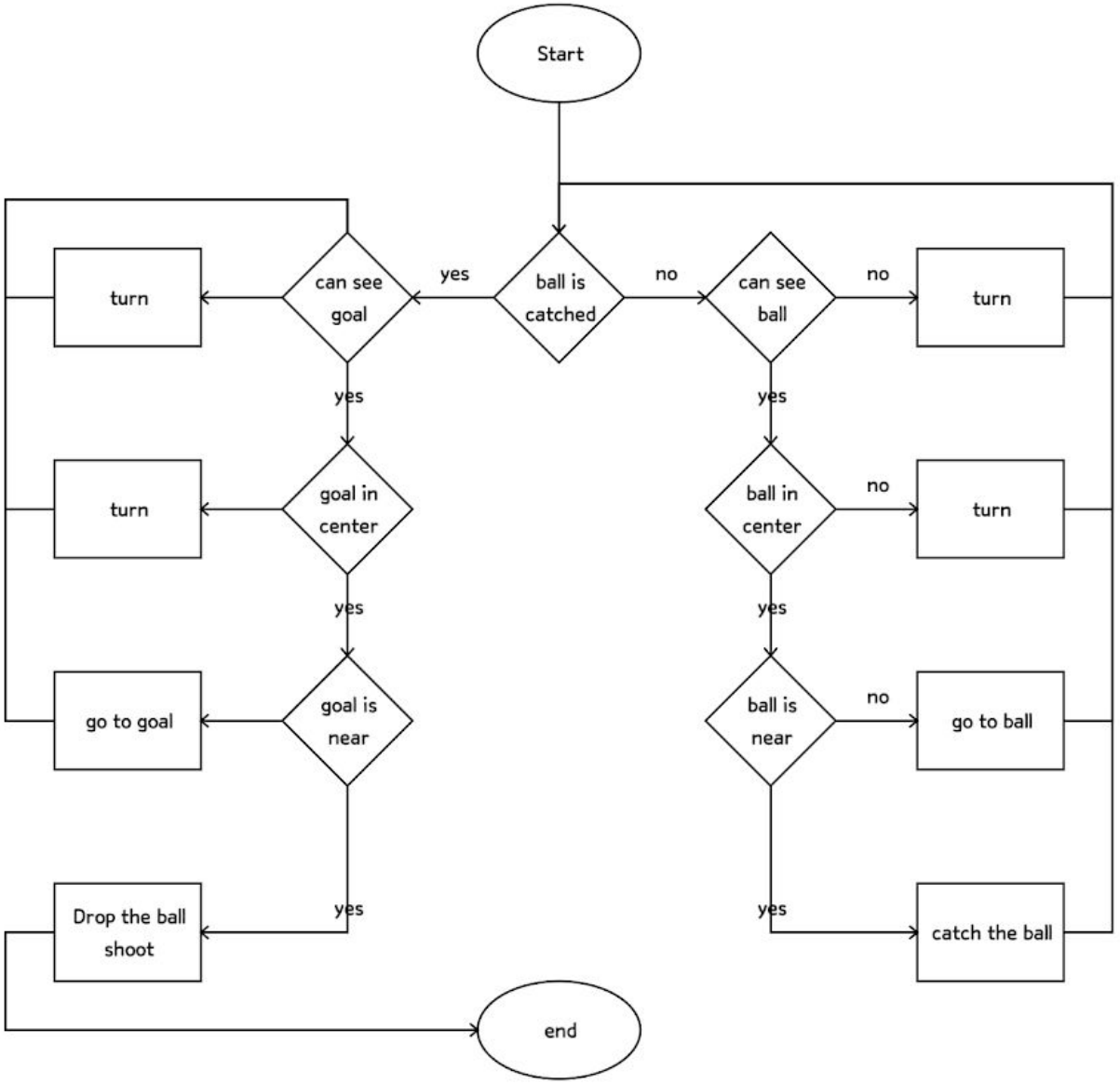
با استفاده از دریل مکان های علامت گذاری شده را سوراخ کردیم و طبقات را با پیچ به هم وصل کردیم. قطعات مختلف شامل بورد ، درایور ، موتور ها و دوربین با پیچ و بعضی از قطعات با چسب به بدنه متصل شدند .


Raspberry Pi 3
Model B+



فصل 6-اسمبل کد

فلوچارت و مسیر کد



کتابخانه ها

```
import time
# این کتابخانه مربوط به زمان است و از تابع اسلیپ ان استفاده شده است
import RPi.GPIO as gpio
# این کتابخانه استاندارد رزبری پای برای ارتباط با پینها است
import cv2
# کتابخانه opencv برای پردازش تصویر
import numpy as np
# کتابخانه numpy برای اعمال ریاضی روی آرایه ها
from  picamera.array import PiRGBArray
from picamera import PiCamera
#کتابخانه استاندارد برای ارتباط با دوربین pi camera
```

توابع

```
def init():
    gpio.setmode(gpio.BCM)
    gpio.setup(17,gpio.OUT)
    gpio.setup(27,gpio.OUT)
    gpio.setup(18,gpio.OUT)
    gpio.setup(24,gpio.OUT)
    gpio.setup(25,gpio.OUT)
    gpio.setup(22,gpio.OUT)
```

این تابع برای تنظیم پین های مربوط به موتور ها بوده و با استفاده از تابع setup مشخص میکنیم کدام gpio خروجی و کدام gpio ورودی است

```
def Move(sec,dir):
    init()
    gpio.output(18,dir)
    gpio.output(22,not dir)
    gpio.output(17,dir)
    gpio.output(27,not dir)
    time.sleep(sec)
    gpio.cleanup()
```

این تابع با ورودی sec و dir به مدت sec در جهت dir حرکت میکند

```
def Turn(sec,dir):
    init()
    gpio.output(18,dir)
    gpio.output(22,not dir)
    gpio.output(17,not dir)
    gpio.output(27,dir)
    time.sleep(sec)
    gpio.cleanup()
```

این تابع با ورودی sec و dir به مدت sec در جهت dir میچرخد

```
def Shoot(sec):
    # set gpio pins
    init()

    # Shoot
    gpio.output(24,True)
    gpio.output(25,False)
    time.sleep(sec)

    # Wait
    gpio.output(24,False)
    gpio.output(25,False)
    time.sleep(1)

    # Reset Shooter
    gpio.output(24,False)
    gpio.output(25,True)
    time.sleep(sec)

    gpio.cleanup()
```

این تابع شوتر را فعال میکند

```
def SetAngle(angle,servo,pwm):
    duty = angle / 18 + 2.5
    GPIO.output(servo, True)
    pwm.ChangeDutyCycle(duty)
    sleep(1)
    GPIO.output(servo, False)
    pwm.ChangeDutyCycle(0)

def Catch(is_open=True):
    #Init
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(23, GPIO.OUT)
    pwm=GPIO.PWM(23, 50)
    pwm.start(0)

    if is_open:
        SetAngle(100,23,pwm)
    else:
        SetAngle(0,23,pwm)
    pwm.stop()
    GPIO.cleanup()
```

از دو تابع بالا برای کنترل سروو موتور و در نتیجه کنترل کچر استفاده میشود

```
def img_adjustment(img, brightness, contrast):
    hsv_img= cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    brightness = 50
    contrast = 50
    hsv_img= np.int16(hsv_img)
    hsv_img= hsv_img* (contrast/127+1) - contrast + brightness
    hsv_img= np.clip(hsv_img, 0, 255)
    hsv_img= np.uint8(hsv_img)
    return hsv_img
```

این تابع برای اعمال تنظیمات نور و کنتراست روی تصویر میباشد و همچنین فرمت تصویر را به hsv تغییر میدهد

```
def mask_generator(img,hsv_img,lower_color,upper_color):
    lower = np.array(lower_color)
    upper = np.array(upper_color)
    mask = cv2.inRange(hsv_img, lower, upper)
    kernel = np.ones((5,5),np.uint8)
    mask = cv2.erode(mask,kernel,iterations = 1)
    color_mask=cv2.cvtColor(mask,cv2.COLOR_GRAY2BGR)
    return mask,color_mask
```

این تابع برای تولید ماسک بین دو رنج رنگی استفاده میشود و در آن ابتدا ماسک از طریق تابع inrange تولید میشود و سپس نویز های آن حذف میشود

```
def find_ball(img,mask,cnt=False):
    image, contours, hierarchy =
cv2.findContours(mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=lambda x: cv2.contourArea(x))
    if len(contours) >0:
        x,y,w,h = cv2.boundingRect(contours[-1])
        if cnt:
            cv2.drawContours(img, [contours[-1]], -1, (0,255,255), 2)
        else:
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
        M = cv2.moments(contours[-1])
        if M['m00'] != 0.0:
            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])
            cv2.circle(img,(cx,cy),10,(255,0,0),-1)
            return (cx,cy)
        return None
```

این تابع با استفاده از ماسک و تصویر اصلی تمام اشیا با رنگ مورد نظر را شناسایی و بزرگترین آن را مشخص میکند و x,y مرکز آن شی را برمیگرداند

```
def find_ball_direction(size,position,radius):
    if position is not None:
        x,y=position
        w,h=size
        if x < w / 2 - radius:
            return -1
        elif x > w / 2 + radius:
            return 1
        else:
            return 0
    else:
        return 2
```

این تابع با گرفتن موقعیت توپ مشخص میکند که آیا در مرکز تصویر از است یا سمت چپ یا راست

فصل 7

پردازش صدا

پردازش صدا در این پروژه کاربرد کنترل ربات از طریق صوت را دارد

برای پردازش صدا ابتدا باید نیاز های سخت افزاری را بررسی کنیم

ابتدا قصد اتصال هدست بلوتوث را داشته یم اما پس از تلاش های زیاد متوجه این موضوع شدیم که فقط از پروفایل A2DP پشتیبانی میکند که بر خلاف HSP و HFP، این پروفایل سیگنال‌های صدا را به صورت یک سویه ارسال می‌نماید و قابلیت اتصال میکروفون به صورت بلوتوث را ندارد.

راهکار دوم اتصال میکروفون از طریق جک 3.5 میلیمتری بود .که متاسفانه کارت صدای پیش فرض قابلیت اتصال میکروفون را نداشته و فقط خروجی صداست

سومین راهکار و راهکاری که اجرا شد استفاده از میکروفون لپ تاپ و ارسال صدا از طریق ssh میباشد.

فایل صدا را با استفاده از دستور scp کپی کرده و آدرس آن را به عنوان آرگومان ورودی فایل اجرایی پایتون میدهیم.

در فایل اجرایی ابتدا صدا توسط موتور تبدیل صدا به متن پردازش شده و متن آن بررسی میشود که آیا با هیچ یک از دستورات از پیش تعریف شده برابر است یا نه

و اگر برابر است دستورات آن اجرا شود.

فصل 8

تست و دیباگ

از اولین مشکلاتی که با آن رو به رو شدیم تنظیمات پیش فرض دوربین برای white balance بود که باعث میشد رنگ اجسام در شرایط نوری متفاوت متفاوت باشد

برای رفع این مشکل تنظیم خودکار white balance را با استفاده از قطعه کد زیر غیر فعال کردیم

```
camera.awb_mode='off'
```

و با استفاده از دو اسلایدر در ابتدای اجرای کد مقادیر red/green و red/blue را تنظیم میکنیم تا white balance با توجه به شرایط مختلف نوری ثابت باشد

از دیگر مشکلات میتوان به نویز موجود در ماسک اشاره کرد که باعث میشد گاهی ربات در تعقیب توپ دچار مشکل شود و به سمت و سوی اشتباهی برود برای رفع این مشکل در تابع mak generator کد زیر را استفاده کردیم

```
kernel = np.ones((5,5),np.uint8)
mask = cv2.erode(mask,kernel,iterations = 1)
```