



دانشگاه صنعتی اصفهان

دانشکده مهندسی مکانیک

کنترل کوادروتور با استفاده از سیستم عامل ربات (ROS)

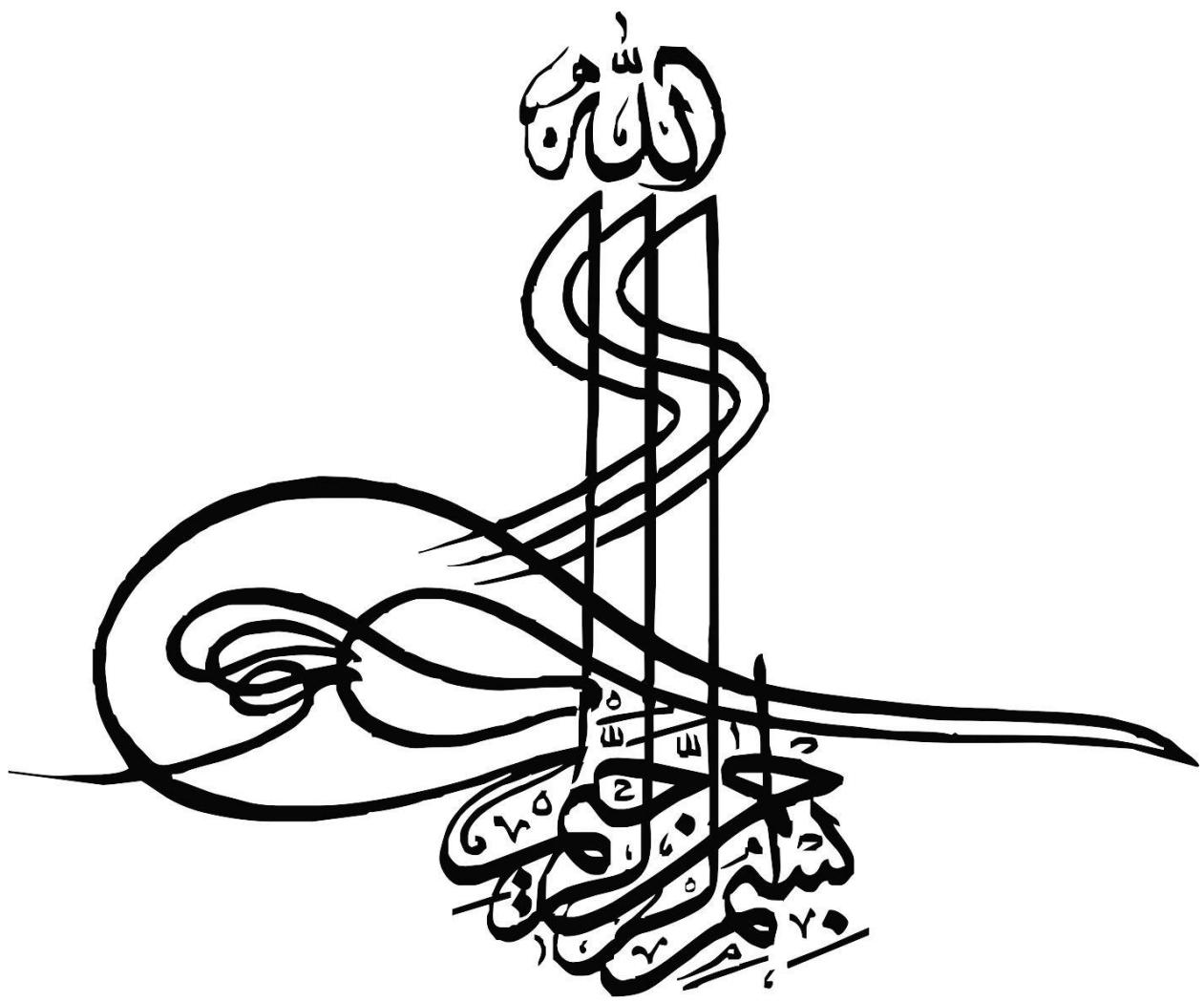
پایان نامه کارشناسی مهندسی مکانیک (گرایش مکاترونیک)

نبیل حیدر

استاد راهنما

دکتر مازیار پالهنج

دکتر محمد مشایخی



فهرست مطالب

1.....	1- فصل اول: مقدمه و تعاریف
1.....	1-1 هدف پژوهش
1.....	1-2 مروری بر کارهای انجام شده
2.....	2-1 مونتاژ کوادکوپتر
2.....	2-2 مطالعه مقالات و جستجو
2.....	2-3 کنترل کوادراتور با یک کنترلر رادیویی از راه دور (RC)
2.....	2-4 شبیه‌سازی با سیستم عامل ربات (ROS) و GAZEBO
3.....	2-5 ارتباط کوادراتور با سیستم عامل ربات (ROS)
3.....	3-1 نتائج به دست آمده
4.....	4-1 بخش‌های پایان‌نامه
5.....	2- فصل دوم: ساخت افزار و نرم‌افزار
5.....	2-1 ساخت افزار
5.....	5-1 فریم
6.....	5-2 ملخ‌ها
6.....	5-3 موتورها
7.....	5-4 کنترل کننده‌های الکترونیکی سرعت (ESC)
8.....	5-5 کنترل کننده پرواز (فلایت کنترلر)
9.....	5-6 رادیو کنترل (فرستنده و گیرنده)
10.....	5-7 مازول تлемتری رادیو Sik فرستنده و گیرنده
11.....	5-8 حسگر GPS
11.....	5-9 باتری
12.....	5-10 محافظ ملخ
14.....	2- نرم‌افزار
14.....	14-1 کنترل کننده پرواز
17.....	14-2 ایستگاه زمینی (Ground Station)
19.....	14-3 سیستم عامل
19.....	14-4 سیستم عامل ربات (ROS)
19.....	14-5 شبیه‌سازی
20.....	3- خلاصه و جمع بندی
21.....	3- فصل سوم: اتصال و شبیه‌سازی
21.....	1-3 شرح MAVLink
21.....	21-1 MAVLink چیست؟
21.....	21-2 فهمیدن پیام‌های MAVLink

23 شکل پیام 3-1-3
24 2-3 ارتباط با ایستگاه زمینی
24 3-2-1 نصب فیرمور یا سیستم عامل
25 2-2-3 ارتباط با کابل USB
26 3-2-3 ارتباط با تلمتری
26 4-2-3 کالیبراسیون
27 3-3 روشن کردن موتورها با استفاده از RC و Mission Planner
27 3-3-1 حالت های پرواز (Flight Modes)
28 2-3-3 روشن موتورها با استفاده از RC
29 3-3-3 روشن موتورها با استفاده از Mission Planner
30 4-3 شبیه سازی با GAZEBO و MAVProxy
30 1-4-3 MAVProxy چیست؟
31 2-4-3 شبیه ساز GAZEBO چیست؟
31 3-4-3 تنظیم محیط SITL و نصب GAZEBO
31 5-3 خلاصه و جمع بندی
32 4-4 فصل چهار: ارتباط با سیستم عامل ربات (ROS)
32 1-4 سیستم عامل ربات (ROS)
32 1-1-4 ROS چیست؟
32 2-1-4 چرا از سیستم عامل ربات استفاده می کنیم؟
33 3-1-4 معماری ROS
36 4-1-4 فضای کاری و بسته ROS چی هستند؟
37 2-4 Raspberry Pi چیست؟
38 3-4 بسته MAVROS
38 1-3-4 تعریف بسته MAVROS
39 2-3-4 ساختار بسته MAVROS
40 3-3-4 نصب بسته MAVROS
40 4-4 اتصال به ROS در SITL روی لپتاپ
41 4-4 ارتباط واقعی با ROS
41 1-5-4 ارتباط با Serial Port
42 2-5-4 ارتباط با کابل usb یا تله متري
43 4-4 اجرا برنامه ROS
44 4-4 موارد اضافي: ارتباط لپتاپ به رزپری پای توسط SSH
46 5-5-4 فصل پنجم: نتائج

48.....	پیوست
48.....	پیوست 1 : تنظیم محیط برای SITL و اجرای آن.....
49.....	پیوست 2 : نصب GAZEBO و اجرای آن.....
51.....	پیوست 3 : نصب بسته MAVROS روی لپتاپ:
51.....	پیوست 4 : نصب بسته MAVROS روی raspberry pi
53.....	پیوست 5 : اتصال به ROS در SITL روی لپتاپ.....
55.....	پیوست 6 : نوشتن و اجرا کردن برنامه ROS
60.....	مراجع

فهرست شکل‌ها

6	شکل 2-1 فریم F450 کوادروتور [2]
6	شکل 2-2 - ملخ 1045 کوادروتور [3]
7	شکل 2-3 - موتور براشلس [5] EMAX MT2216 810KV
7	شکل 2-4 - اسپید کنترل [7] EMAX SIMON SERIES 20A
9	شکل 2-5 - تاریخ بردهای آردوبایلت [11]
9	شکل 2-6 - برد [13] APM2.8
10	شکل 2-7 - کنترل رادیویی از راه دور (فرستنده) [14] JR DMSS XG6 2.4GHz
10	شکل 2-8 - گیرنده رادیو [15]
11	شکل 2-9 - مژوی فرستنده و گیرنده تله‌متري [18]
11	شکل 2-10 حسگر [19] GPS
12	شکل 2-11 باتری لیتیوم پلیمر TATTU [21]
12	شکل 2-12 باتری لیتیوم پلیمر GENS ACE [22]
13	شکل 2-13 طراحی محافظ در نرمافزار
13	شکل 2-14 شکل محافظ ملخ روی پهپاد
14	شکل 2-15 نقشه اتصالات قطعات با هم [25]
17	شکل 2-16 انواع فیرمورهای آردوبایلت [28]
18	شکل 2-17 نقشه بخش از دانشگاه صنعتی اصفهان در MISSION PLANNER
20	شکل 2-18-معماری SITL [35]
22	شکل 3-1- تقسیم بایت‌های پیام MAVLINK [38]
23	شکل 3-2- نحوه تبادل اطلاعات بین پهپاد و MISSION PLANNER توسط پیام‌های MAVLINK [39]
23	شکل 3-3- جدول مفصل پیام COMMAND_LONG [40]
24	شکل 4-3 صفحه نصب فیرمور در MISSION PLANNER
25	شکل 5-3 دکمه CONNECT موجود در بالای صفحه MISSION PLANNER
25	شکل 6-3 مرحله اتصال با کوادروتور (دربافت پارامترها)
26	شکل 7-3 دکمه CONNECT در نرم‌افزار MISSION PLANNER
26	شکل 8-3 MISSION PLANNER در حال اتصال با کوادروتور توسط تله‌متري
27	شکل 9-3 صفحه که شامل کالیبراسیون قطعات متنوع در MISSION PLANNER
28	شکل 10-3 سه محور اصلی کوادروتور: ROLL، PITCH و YAW [47]
29	شکل 11-3 حالت مسلح کردن موتورها در RC

29	شکل 12-3 تقسیم RC ELEVATOR، THROTTLE، RUDDER و AILERON در یک
30	شکل 13-3 قسمت MISSION PLANNER در ARM/DISARM برای روشن و خاموش کردن موتورها.
30	شکل 14-3- ایستگاه زمینی MAVPROXY در سیستم عامل لینوکس [31]
31	شکل 15-3- شبیه‌سازی کوادروپور در شبیه‌ساز GAZEBO
34	شکل 1-4- ارتباط دو گره با گره اصلی [55]
35	شکل 2-4 ارتباط ناشر و مشترک در ROS [55]
35	شکل 3-4- ارتباط ناشر و مشترک توسط تاپیک و تعیین اسم تاپیک و نوع پیام [56]
36	شکل 4-4 ارتباط سرور و کلاینت (با سرویس‌ها) [55]
36	شکل 4-5 ارتباط بین سرور و کلاینت (با اکشن‌ها) [55]
37	شکل 6-4 ساختار فضای کاری ROS [58]
38	شکل 7-4 برد RASPBERRY PI 3B [59]
39	شکل 8-4- نقشه مختصر روند ارتباط کنترلر پرواز با ROS و همچنین با حسگر خارجی [62]
41	شکل 9-4- روند اتصال کنترل کننده پرواز با RASPBERRY PI با استفاده از SERIAL PORT [65]
42	شکل 10-4- اتصال RASPBERRY PI 3B با پورت‌های TELEMETRY کنترل کننده پرواز [66]
42	شکل 11-4 گزینه فعال و غیر فعال بودن SERIAL [65]
44	شکل 12-4- گزینه‌ها داده شده با اجرای برنامه APM_CONTROL.PY
44	شکل 13-4- رفتار برنامه APM_CONTROL.PY با زدن عدد 7
44	شکل 14-4- رفتار برنامه APM_CONTROL.PY با نوشتن ^C
49	شکل 1-0- سمت چپ بالا: CONSOLE و سمت جپ پایین: ترمینال و سمت راست: نقشه
50	شکل 2-0 شبیه‌ساز GAZEBO با ARDUCOPTER مجازی در آن
55	شکل 3-0- لیست کامل اعداد حالت‌های پرواز [73]

فصل اول: مقدمه و تعاریف

1-1 هدف پروژه

هدف این پروژه ورود به دنیای ربات‌ها از درب عریض آن است. امروزه رشته رباتیک مانند هر رشته دیگری بخشی از علم عظیم هوش مصنوعی است و از بسیاری از منظرها برای شروع به کارگیری هوش مصنوعی در رباتیک، ابتدا باید مناسب‌ترین روش‌ها را برای انجام آن بیاموزیم. یکی از این روش‌ها سیستم عامل ربات است. به دلایل بسیاری، ROS به یکی از ابزارهای توسعه دهنده اصلی برای مهندسان نرم‌افزار رباتیک تبدیل شده است. به کاربران اجازه می‌دهد با زبان‌های سطح بالا کار و کدنویسی کنند و ربات‌های پیچیده را به روشهای ساده و سازماندهی شده بکار ببرند. درمورد سیستم عامل ربات بعداً بیشتر صحبت خواهد شد. و دومین موردی که هدف پروژه ما محسوب می‌شود، پهپادهای ساقیه یا ربات‌های پرنده چالش دنیای امروز هستند. آنها فناوری چالشی هستند که کشورهای برتر در تلاش برای تسخیر آن هستند، و ما می‌توانیم آنها را با همه زمینه‌های مختلف مانند نظامی، کشاورزی و فیلم سازی مرتبط بدانیم. کار در زمینه برنامه‌نویسی پهپادها، به ویژه با استفاده از ROS، نیازمند دانش خوب در پایتون یا C++ و گاهی اوقات هر دو است. از آنجایی که برخی از برنامه‌ها به هر دو زبان توسعه یافته‌اند و گاهی اوقات توسعه دهنده‌گان باید کدها را در کنند تا ویرایش‌های خود را در آن اضافه کنند. برنامه‌نویسی بخش بزرگی از هدف پروژه بود.

در نظر گرفتن فناوری هواپیماهای بدون سرنشین و به کارگیری فناوری رباتیک و هوش مصنوعی در آن، هدف اصلی است. این پایان‌نامه بخشی از این هدف می‌باشد.

2-1 مروری بر کارهای انجام شده

1-2-1 مونتاژ کوادکوپتر

در این پروژه از یک کوادروتور که در آزمایشگاه هوش مصنوعی در دانشکده برق و کامپیوتر وجود دارد استفاده گردید. مانند تمام پهپادها، این پهپاد نیز برای انجام کار خود به چندین نوع سختافزار و نرمافزار نیاز داشت که تفصیل این موضوع در فصل دوم: سختافزار و نرمافزار خواهیم گفت.

2-2-1 مطالعه مقالات و جستجو

برای انجام این پروژه لازم بود که منابع و مستندات گوناگونی مطالعه شوند. اطلاعات در مورد سختافزار استفاده شده در ربات، نحوه راهاندازی و کنترل ربات، ایستگاه زمینی، ROS، و شبیه‌سازها از جمله این موارد بودند.

3-2-1 کنترل کوادروتور با یک کنترلر رادیویی از راه دور (RC) و Mission Planner

هدف این بود که موتورها با استفاده از RC یا با استفاده از نرمافزار MissionPlanner (یک ایستگاه زمینی) روشن شوند. قبل از روشن کردن موتورها باید کالیبراسیون حسگرهای مختلف انجام شود. و برای بهترین نتیجه، کالیبراسیون باید در فضای باز انجام شود. موارد اصلی که باید کالیبره شوند عبارتند از: شتاب‌سنج، قطب نما، کنترل کننده‌های سرعت (ESC) و رادیو می‌باشند. شتاب‌سنج، قطب نما و رادیو با استفاده از نرم افزار Mission Planner کالیبره می‌شوند در حالی که ESC فقط با استفاده از RC قابل کالیبره شدن است. همه به جز کنترل کننده‌های سرعت به درستی کالیبره شدند. متاسفانه پس از انجام کالیبراسیون موتورها فعال نشدند. پس از ساعتها عیب‌یابی و تلاش برای رفع این مشکل به این نتیجه رسیده شد که مشکل از RC فرستنده و گیرنده و به طور خاص از کانال "Throttle" است. پس از تنظیم تنظیمات در فرستنده، کنترل کننده‌های سرعت روی مدولاسیون پهنه‌ای باند (PWM) رادیویی جدید کالیبره شدند و اکنون موتورها بدون هیچ مشکلی روشن می‌شوند. بعد از کالیبره شدن کنترلرهای سرعت، روشن موتورها توسط Mission Planner هم درست شد. ارتباط کوادروتور با این نرمافزار توسط سیم USB یا ماژول تله‌متري انجام می‌شود.

4-2-1 شبیه‌سازی با سیستم عامل ربات (ROS) و GAZEBO

همیشه بهتر است قبل از انجام کارهایی که ممکن است باعث برخی خرابی‌ها شود، شبیه‌سازی انجام شود. اگر این گزینه در دسترس بود پس ضروری است. GAZEBO یک نرمافزار شبیه‌سازی رباتیک است که ربات‌های دنیای واقعی را شبیه‌سازی می‌کند و می‌تواند به ROS متصل شود تا شبیه‌سازی واقعی‌تری داشته باشد. برای این کار از نرمافزار در حلقه یا SITL¹ استفاده شد. این ابزاری است که توسط سازنده اوتپایلت ارائه شده است که به کاربران امکان می‌دهد همه چیز را با استفاده از نرمافزارهای مختلف بدون استفاده از هیچ سختافزاری شبیه‌سازی کنند. در این مورد، برنامه‌های ROS را می‌توان توسعه داد و اگر قرار بود به خوبی کار کنند، آزمایش می‌شوند. یک بسته ROS ایجاد شد که برای کنترل این کوادروتور با استفاده از

¹ Software in the loop

دستورات ساده استفاده شد. برای نوشتن این بسته ROS از زبان پایتون استفاده شده است. این کد پایتون امکان برخاستن و فرود کوادروتور، تغییر حالت پرواز، روشن و خاموش موتورها و چاپ مختصات فعلی آن را فراهم می‌کند. قبل از استفاده از ROS در شبیه‌سازی، یک شبیه‌سازی ساده با استفاده از SITL و یک آیستگاه زمینی به نام MAVProxy انجام شد که همچنین به کاربران اجازه می‌دهد بدون استفاده از هیچ سخت‌افزاری پهپاد را کنترل و آزمایش کنند.

5-2-1 ارتباط کوادروتور با سیستم عامل ربات (ROS)

با آمدن به دنیای واقعی، احتمال بیشتری وجود دارد که با مشکلاتی روبرو شد. شبیه‌سازی ممکن است ساده باشد و بسیار خوب کار کند اما دنیای واقعی نظرات متفاوتی دارد. در شبیه‌سازی، تمامی نرم‌افزارها به روز هستند و نمی‌توان با مشکل ناسازگاری مواجه شد. از آنجایی که برده کنترلر پرواز قدیمی است و توسط نسخه‌های جدید نرم‌افزار پشتیبانی نمی‌شود، بسیاری از نرم‌افزارها باید حذف و نسخه‌های قدیمی‌تر نصب شوند. برای این کار از برده Raspberry Pi به عنوان کامپیوتر (جايگزين لپتاپ) استفاده شد و نرم‌افزار قدیمی روی آن دانلود شد تا آماده استفاده شود. برای اتصال کوادروتور به ROS، پیام‌های کوادروتور باید ROS و پیام‌های ROS برای کوادروتور قابل درک باشد. بسته ROS که زبان کوادروتور را به زبان ROS و بالعکس ترجمه می‌کند MAVROS [1] نام دارد. پس از تلاش‌های فراوان برای دانلود نسخه‌ای از MAVROS که با نسخه کنترل کننده پرواز و نسخه ROS سازگار است، اتصال با موفقیت برقرار شد و کد نوشته شده تست شد.

3-1 نتائج به دست آمده

دنیای امروز به سرعت در حال توسعه است و برای داشتن بهترین عملکرد، سریع‌ترین کار، و نتایج بهتر، همیشه بهتر است از سخت‌افزار پشتیبانی شده استفاده شود، مخصوصاً زمانی که هدفی که باید به آن رسید، هوش مصنوعی باشد. در داخل آزمایشگاه، با استفاده از یک rqt_graph فهرست گره‌های مختلف بسته MAVROS مورد تجزیه و تحلیل قرار گرفت و از برخی گره‌های ویژه برای نوشتن کد پایتون استفاده شد. همچنین با استفاده از خدمات MAVROS امکان نظارت بر وضعیت زنده پهپاد و خواندن لحظه به لحظه پیام‌ها و پارامترهای آن وجود داشت. با استفاده از بسته ROS نوشته شده، موتور روشن و خاموش می‌شد و مختصات پهپاد در هر لحظه چاپ می‌شد. برخاستن و فرود باید در بیرون آزمایش تست شوند. (تا لحظه نوشتن گزارش هنوز بیرون نشده است).

4-1 بخش‌های پایان‌نامه

فصل دوم نرم‌افزار و سخت‌افزار مختلف مورد استفاده در این پروژه را نشان می‌دهد. در این فصل به تفصیل در مورد کل کوادروتور از بیرون و درون صحبت خواهد شد.

فصل سوم در مورد اتصال پهپاد به Mission Planner و فرآیندهای کالیبراسیون و در مورد شبیه‌سازی با استفاده از GAZEBO، MAVProxy، SITL و MAVROS صحبت خواهد شد.

فصل چهارم مفاهیم ROS را نشان می‌دهد و به طور خلاصه در مورد بسته MAVROS و در نهایت اتصال پهپاد به ROS با استفاده از شبیه‌سازی و به صورت واقعی توضیح می‌دهد. در این فصل کل بسته نوشته شده برای کنترل کوادروتور توضیح خواهد شد. در پایان تمام نتایج نشان داده خواهد شد.

فصل آخر درمورد خلاصه مشکلاتی که با آن مواجه شده‌ایم و هم در مورد نتایج صحبت می‌شود.

فصل دوم: سخت‌افزار و نرم‌افزار

در این پژوهه از یک کوادراتور که در آزمایشگاه هوش مصنوعی در دانشکده برق و کامپیوتر وجود دارد و توسط دکتر پالهنج ساخته شده است استفاده گردید. مانند تمام پهپادها، این پهپاد نیز برای انجام کار خود به چندین نوع سخت‌افزار و نرم‌افزار نیاز دارد. در این فصل، بر روی سخت‌افزار و نرم‌افزار مورد استفاده برای ساخت این کوادراتور تمرکز خواهیم کرد.

1-2 سخت‌افزار

1-1-2 فریم^۲

از فریم کوادراتور F450 استفاده شده است (شکل 1-2). عرض 450 میلی متر و ارتفاع 55 میلی متر را دارد. وزن آن 295 گرم می‌باشد. بعضی از مشخصات این فریم [2]:

- ساخته شده از فیبر شیشه‌ای با کیفیت بالا و نایلون پلی آمید بادوام.

- بازوها برای جلوگیری و کاهش شکستگی تقویت شده‌اند.

- اتصالات PCB^۳ یکپارچه برای لحیم کاری مستقیم کنترل کننده‌های سرعت.

- بازوی رنگی برای جهت‌گیری تا در جهت درست پرواز نگه دارد.

- مونتاژ آسان دارد.

² Frame

³ Printed circuit board



شکل 2-1 فریم F450 کوادروتور [2]

2-1-2 ملخها

از ملخ 1045 استفاده شده است (شکل 2-2). به این معنی که طول 10 اینچ یا 254 میلی متر، و گام 4.5 اینچ را دارد. وزن آن 14 گرم است [3].



شکل 2-2 - ملخ 1045 کوادروتور [3]

3-1-2 موتورها

موتورهای استفاده شده موتورهای براشلس⁴ EMAX MT2216 810Kv (شکل 3-2)

توان: 228 وات

نیروی پرتاپ: 1070 گرم

جریان: 15.4 آمپر

ولتاژ: 14.8 ولت

قطر شافت: 4 میلی متر

وزن: 62 گرم

⁴ Brushless

اطلاعات بیشتر در مورد مشخصات این موتور در این مرجع [4] موجود است.



شکل 2-3-2 - موتور برشلس [5] EMAX MT2216 810KV

4-1-2 کنترل کننده‌های الکترونیکی سرعت (ESC)^۵

کنترل کننده‌های الکترونیکی سرعت که اسپیدکنترلر هم نامیده می‌شوند، یکی از مهم‌ترین قطعات کوادروتور است. کنترلر سرعت (ESC) یک مدار الکترونیکی است که سرعت موتور الکتریکی را کنترل و تنظیم می‌کند. همچنین ممکن است معکوس کردن موتور و ترمز دینامیکی را فراهم کند. این واحد مسئول تغییر سرعت موتورها بر اساس دستورات کاربر است که از طریق کنترل از راه دور یا جوی استیک می‌دهد. اغلب اوقات، مستقیماً این اسپیدکنترلرها برنامه‌ریزی نمی‌شوند، اما می‌توان آنها را از قبل، توسط کارخانه اصلی برنامه‌ریزی کرد یا با آپلود سیستم عامل یا فایرمور در کنترل کننده اصلی، آنها را برنامه‌ریزی می‌کند. یک سیگنال از کنترلر پرواز باعث می‌شود که ESC ولتاژ موتور را در صورت نیاز افزایش یا کاهش دهد و در نتیجه سرعت پروانه را تغییر دهد.

در پروژه از EMAX Simon Series 20A (شکل 2-4) استفاده شده است [6]. برای هر موتور یک اسپیدکنترلر را نیاز دارد.



شکل 2-4-2 - اسپیدکنترل EMAX Simon Series 20A [7]

⁵ Electronic Speed Controller

5-1-2 کنترل کننده پرواز (فلایت کنترل)

اگر از لحاظ فیزیکی بخواهیم در مورد کنترل کننده پروازها صحبت کنیم، چیزی بیش از یک مدار الکترونیکی نیست. می‌توان آن را با برد یک گوشی هوشمند مقایسه کرد. به عبارتی دیگر مغز کوادروتور به حساب می‌آید. یک جعبه کوچک پر از لوازم الکتریکی و نرمافزار هوشمند که هر کاری را که پهباد انجام می‌دهد، نظارت و کنترل می‌کند و مانند مغز موجودات مختلف، کنترل کننده‌های پرواز نیز از نظر اندازه و پیچیدگی متفاوت هستند. به همین دلیل مولتی روتورها (در این پروژه کوادروتور می‌باشد) به یک قطعه که نام آن را کنترل کننده پرواز می‌گذارند برای انجام کل کارهای کنترلی مورد نیاز نیازمندند.

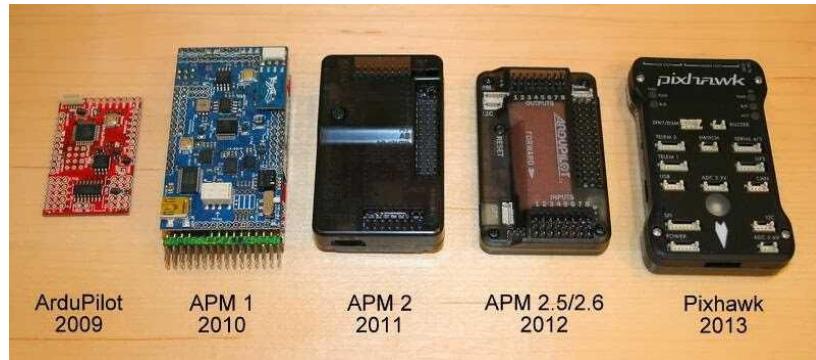
کنترل کننده پرواز به مجموعه‌ای از حسگرها اطلاعاتی مانند ارتفاع، جهت و سرعت کوادروتور را به کنترل کننده پرواز منتقل می‌کنند. به عنوان نمونه می‌توان به حسگرهای اینرسی که برای تعیین سرعت و شتاب زاویه‌ای به کار می‌روند، فشارسنج برای تعیین ارتفاع و حسگرهای فاصله هم برای تشخیص موائع هستند. همان‌طور که انسان‌ها حواسی برای درک اطرافمان دارند، این کوادروتورها هم این اطلاعات به دست‌آمده را بررسی و کنترل می‌کنند و برای کارایی بهتر آن‌ها را با هم ترکیب می‌کنند. یک کوادروتور حرفه‌ای خیلی سریع‌تر و بدقت بیشتر تشخیص‌های لازم را انجام می‌دهد [8].

تنها حس کردن آنچه در اطراف یک کوادروتور رخ می‌دهد کافی نیست. یک کنترل کننده پرواز حرکت این پهپاد را کنترل می‌کند. کوادروتور خیلی راحت می‌تواند بین چهار موتور به کارفته در آن تغییر سرعت ایجاد کند و همین امر سبب شتاب گرفتن آن می‌شود. برای آن که کوادروتور به سرعت موردنظر برسد، کنترل کننده پرواز داده‌های جمع‌آوری شده توسط حسگرها را برای محاسبه سرعت موردنظر، برای هر یک از چهار موتور استفاده می‌کند. محاسبه حرکات، ترکیب و فیلتر کردن اطلاعات حسی و تخمین ایمنی و دوام پرواز همه توسط یک الگوریتم و توسط کنترل کننده پرواز انجام می‌گیرد [8].

برای این پروژه و از آنجایی که چند سال پیش عملیات ساخت این پهپاد انجام شد، از برد کنترل پرواز APM2.8 [9] استفاده شده است (شکل 2-6). لازم به ذکر است که اکیداً استفاده از برد های پشتیبانی نشده پیشنهاد نمی‌شود زیرا هنگام عیب‌یابی، کاربر به دلیل ناسازگاری با مشکلات زیادی روبرو خواهد شد. بنابراین بهتر است از برد های جدیدتری مثلاً برد های Pixhawk [10] استفاده شود.

شرکت ArduPilot ساخت سخت‌افزار کنترل کننده پرواز را متوقف کرد و فقط روی نرمافزار تمرکز کرد [9]. آخرین بردی است که ArduPilot ساخته است (شکل 2-5). APM2.8 مبتنی بر آردوینو ۲۵۶۰ مگا^۶ است و از آنجایی که منبع باز است، می‌توان آن را دوباره برنامه ریزی کرد.

⁶ Arduino Mega



شکل 2-5-2 - تاریخ بردهای آردوپایلت [11]

پس از ساخت Pixhawk، دنیای کنترل کننده‌های پرواز توسط آن فتح شد و بهترین کنترل کننده پرواز اقتصادی در این زمینه شد [12]. و تا به حال کاملاً پشتیبانی شده است. نرمافزار Ardupilot همچنین می‌تواند در آپلود شود و مانند APM2.8 کار کند. پس چرا Pixhawk بهتر است؟ از آنجایی که با فیرمور ArduPilot جدید و به روز و سایر فیرمورهای AUTOPILOT که روزانه در حال توسعه هستند **Error!** سازگار است. علاوه بر امکانات سخت‌افزاری بیشتر که دارد. در مورد فیرمور ArduPilot در قسمت **Reference source not found.** صحبت خواهد شد.



شکل 2-6-2 - برد APM2.8

6-1-2 رادیو کنترل (فرستنده و گیرنده)

به صورت کلی رادیو کنترل‌ها ابزاری برای کنترل ربات‌ها و کوادراتورها به حساب می‌آیند. این دستگاه‌های کنترلی تنها از طریق امواج رادیویی و بی‌سیم کار می‌کنند و استفاده از آن‌ها با کمک فرکانس‌های رادیویی ممکن است. برای سینک شدن یا به اصطلاح ایجاد هماهنگی بین کوادراتور و RC تنها کافی است که منحصر به فردی که مخصوص دستگاه گیرنده است را در حین روشن کردن هر دو دستگاه فرستنده و گیرنده فعال شود و به این صورت مولتی‌روتر مورد نظر با RC مرتبط متصل می‌شود و هیچ تداخلی برای اتصال با سایر دستگاه‌ها نخواهد شد. در این پروژه از فرستنده RG831B 2.4GHz (شکل 7-2) و گیرنده DMSS XG6 JR (شکل 8-2) استفاده شده است.



[14] JR DMSS XG6 2.4GHz رادیویی از راه دور (فرستنده)



[15] گیرنده رادیو

7-1-2 مژول تله‌متري راديو SIK فرستنده و گيرنده

راديو تله‌متري SIK يکی از ساده‌ترین راه‌ها برای راهاندازی يک اتصال تله‌متري بین Autopilot و ايستگاه زمینی است. راديو تله‌متري SIK يک پلتفرم رادیویی منبع باز کوچک، سبک و ارزان است که عموماً مسافت بیش از 300 متر را امکان‌پذیر می‌کند (با استفاده از يک آنتن روی مژول زمین، يعني مژولی که به ايستگاه زمینی متصل است، برد را می‌توان تا چندین کيلومتر افزایش داد). اين راديو از فيرمور منبع باز استفاده می‌کند که به طور ویژه برای کار با بسته‌های MAVLink و ادغام با Copter، Mission Planner، Rover و Plane طراحی شده است (در مورد MAVLink در فصل سوم صحبت خواهد شد) فقط باید بدانيد که MAVLink پروتکلی برای برقراری ارتباط با پهپادها است [16]. اين مژول‌ها به پهپاد اجازه می‌دهند تا به ايستگاه زمینی متصل شود و به طور کامل کنترل شود. به عنوان مثال، يک ماموریت ویژه که شامل ايستگاه‌های مختلف می‌تواند با استفاده از اين مژول‌ها بدون نياز به هیچ‌گونه اتصال سيمی به پهپاد ارسال شود [17]. (شکل 2-9)



شکل 2-9- مازول فرستنده و گیرنده تله‌متري [18]

8-1-2 GPS حسگر

اسم حسگر GPS که در اين پروژه استفاده شده است NEO-M8N Flight Controller GPS Module می‌باشد (شکل 2-10). اين مازول هم قطب نما داخلی دارد.



شکل 2-10 GPS حسگر

برای خوانندگانی که با GPS و روش کار آن آشنایی ندارند، یک راهنمای کامل در مورد GPS و نحوه انتخاب بهترین گزینه برای پروژه‌ها وجود دارد [20].

9-1-2 باتری

بهترین انتخاب برای چنین پروژه‌هایی باتری‌های لیتیوم پلیمری است. آنها کارآمد هستند و ظرفیت بالای دارند. از دو نوع باتری استفاده شده است، يکی TATTU (شکل 2-11) و يکی Gens Ace (شکل 2-12). TATTU 3 سل و دارای 4200 میلی آمپر ساعت می‌باشد [21] و Gens Ace هم 3 سل و دارای 4000 میلی آمپر ساعت می‌باشد [22].



شکل 2-11 باتری لیتیوم پلیمر [21] TATTU



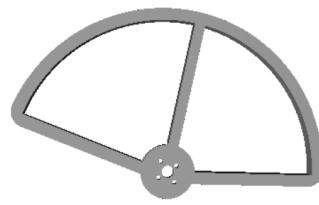
شکل 2-12 باتری لیتیوم پلیمر [22] Gens Ace

10-1-2 محافظ ملخ

با کمک نرم افزار CAD⁷, [23] Solidworks, برای هر موتور یک محافظ ملخ ساده طراحی شده است (شکل 2-13) و روی پهپاد نصب شد (شکل 2-14). این محافظها به محافظت از پهپاد در برابر آسیب در هنگام آزمایش در داخل آزمایشگاه کمک می‌کند و می‌تواند به کاربران حدی که دستشان می‌تواند به موتور نزدیک شود را نشان دهد. اگرچه طراحی بسیار ساده است، اما اینمی‌حرف اول را می‌زنند. با کمک یک کولیس، تمام اندازه‌گیری‌ها به طور دقیق انجام شد و اولین نمونه به صورت سه بعدی پرینت شد و روی پهپاد آزمایش شد. نتایج نشان داد که طول بهتر است بیشتر باشد پس طرح جدیدی ساخته شد و سپس بر روی پهپاد نصب شد. این طرح از طرح‌های بسیاری الهام گرفته شده است که توسط علاقمندان و سازندگان پهپاد ساخته شده‌اند. طرح‌ها به صورت سه بعدی پرینت شدند. جنس محافظها PLA⁸ [24] می‌باشد. آنها با توجه به طول پروانه و قرار گرفتن در قاب با موتورهای نصب شده طراحی شدند.

⁷ Computer Aided Design

⁸ Polylactic Acid

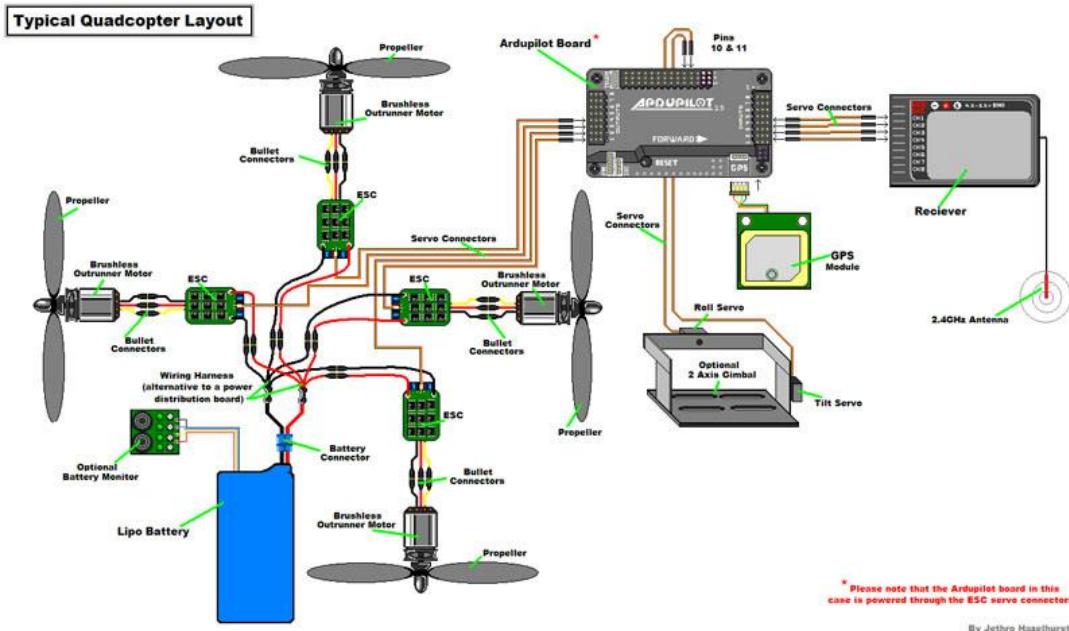


شکل 2-13 طراحی محافظ در نرم افزار



شکل 2-14 شکل محافظ ملخ روی پهپاد

با این سخت افزار اکنون می شود کوادروتور را مونتاژ کرد (شکل 152) و آن را به پرواز در آورد. البته از ابزارهای دیگری مخصوصاً هنگام اتصال به ROS استفاده شده است، بعداً در مورد آن در فصل چهار: ارتباط با سیستم عامل ربات (ROS) صحبت خواهد شد. به عنوان مثال از بسیاری از کابل‌ها، باتری‌های دوبل A و حتی یک Raspberry Pi استفاده شده است. Raspberry pi برای کمک به اتصال با ROS استفاده شد. از آنجایی که لپ تاپ مورد استفاده دارای نسخه جدید لینوکس است، از رزبری پای برای دانلود نسخه‌های قدیمی لینوکس و ROS استفاده شده است تا با برد APM2.8 سازگار باشد. اما اساساً تفاوتی وجود ندارد که از لپ تاپ، دسکتاپ یا رایانه همراه استفاده شود. نکته مهم این است که کامپیوتر باید ROS داشته باشد.



شکل 2-15- نقشه اتصالات قطعات با هم [25]

2-2 نرم افزار

در این بخش به شرح نرم افزار مورد استفاده در پروژه خواهیم پرداخت.

1-2-2 کنترل کننده پرواز

برای کنترلر پرواز، از فیرمور ArduCopter، نسخه 3.2.1 استفاده شده است. این آخرین نسخه و در عین حال سازگار با برد APM2.8 است. بسیاری از نسخه‌های جدیدتر وجود دارند که می‌توانند با Pixhawk یا هر برد جدیدتر دیگری استفاده شوند.

یک سیستم خلبان خودکار قابل اعتماد، همه کاره و منبع باز است که از انواع وسائل نقلیه پشتیبانی می‌کند: چند کوپتر⁹، هلیکوپتر، هواپیمای بال ثابت، قایق، زیردریایی، مریخ‌نورد¹⁰ و غیره. کد منبع توسط جامعه بزرگی از متخصصان و علاقه مندان ایجاد شده است [26].

ایجاد و استفاده از سیستم‌های وسیله نقلیه مطمئن، مستقل و بدون سرنشین را برای منافع صلح آمیز همگان امکان‌پذیر می‌کند. ArduPilot مجموعه‌ای جامع از ابزارها را فراهم می‌کند که تقریباً برای هر وسیله نقلیه و برنامه‌ای مناسب است. به عنوان یک پروژه منبع باز، به طور مداوم بر اساس بازخورد سریع جامعه بزرگی از کاربران در حال تکامل است. تیم توسعه با جامعه و شرکای تجاری کار می‌کند تا

⁹ Multi Copter

¹⁰ Rover

عملکردی را به ArduPilot اضافه کند که به نفع همه باشد. اگرچه ArduPilot هیچ سخت افزاری تولید نمی‌کند، سیستم عامل ArduPilot بر روی طیف گسترده‌ای از سخت افزارهای مختلف برای کنترل انواع وسایل نقلیه بدون سرنشین کار می‌کند. همراه با نرم‌افزار کنترل زمینی (ایستگاه زمینی)، وسایل نقلیه بدون سرنشین که ArduPilot را اجرا می‌کنند، می‌توانند عملکردهای پیشرفته‌ای از جمله برقراری ارتباط بی‌درنگ با اپراتورها داشته باشند. ArduPilot دارای یک انجمان آنلاین بزرگ است که به کاربران در مورد سؤالات، مشکلات و راه حل‌ها کمک می‌کند. اولین مخزن کد باز ArduPilot در سال 2009 ایجاد شد و از آن زمان تاکنون توسط تیمی از مهندسان حرفه‌ای مختلف، دانشگاهیان، دانشمندان کامپیوتر و سایر اعضای جامعه جهانی توسعه داده شده است. این اتوپایلت می‌تواند تقریباً هر سیستم وسیله نقلیه قابل تصویری را کنترل کند مانند هواپیماها، گلایدرها^{۱۱}، مولتی روتورها، هلیکوپترها، قایق‌های بادیانی، قایق‌های برقی، زیردریایی‌ها، وسایل نقلیه زمینی و حتی ربات‌های خود متعادل کننده^{۱۲}. ArduPilot که در بیش از ۱,۰۰۰,۰۰۰ وسیله نقلیه در سراسر جهان نصب شده است، و با ابزارهای پیشرفته ثبت داده، تجزیه و تحلیل و شبیه‌سازی، یک سیستم خلبان خودکار عمیقاً آزمایش شده و قابل اعتماد است. پایه کد منبع باز به این معنی است که به سرعت در حال تکامل است و همیشه در لبه پیشرفت فناوری قرار دارد. مجموعه نرم افزار ArduPilot در وسایل نقلیه از بسیاری از تولید کنندگان و به طور گسترده‌تر در سراسر صنعت سیستم‌های خودمختار جهانی^{۱۳} نصب شده است. همچنین برای آزمایش و توسعه توسط موسسات و شرکت‌های بزرگ مانند ناسا^{۱۴}، اینتل^{۱۵} و Boeing و همچنین مدارس و دانشگاه‌های بی‌شماری در سراسر جهان استفاده می‌شود. [26]

خلاصه موضوع بالا این است که آردوپایلت یا ArduPilot یک سخت‌افزار نیست، بلکه نرم‌افزار است. این نرم‌افزار در یک کنترل کننده پرواز مانند Pixhawk یا APM^{۱۶} گذاشته می‌شود. در مورد کنترل کننده‌های پرواز در قسمت کنترل کننده پرواز (فلایت کنترل) صحبت شد.

برای توضیح بیشتر، اینها برخی از انواع خودروهای پشتیبانی شده توسط ArduPilot که برای هر نوع، یک فیرمور خاصی وجود دارد. (شکل 2-16)

الف - انواع کپترهای^{۱۷}: کوادروپور، OctaCopter، HexaCopter، TriCopter، ... [27].

¹¹ Gliders

¹² Balance-Bots

¹³ Global Autonomous Systems Industry

¹⁴ NASA

¹⁵ Intel

¹⁶ ArduPilot Mega

¹⁷ Copters

¹⁸ QuadX8

ب- انواع هواپیماهای **Tri-Tilt-Wing QuadPlane**^{۱۹}، **هواپیما کواد**^{۲۰}، **Plank Flying Wing** :**(Planes)** .[27] ...**GyroCopter**,**Ornithopter**

ج- انواع روورها (Rovers): کل انواع ربات‌های چرخ‌دار و قایق‌ها [27].

بسته به نوع ربات مورد استفاده، باید یک فیرمور مخصوص به کنترلر این ربات آپلود شود. مثلاً اگر از هواپیما استفاده می‌شود، باید فیرمور **Plane** را آپلود شود. و اگر از مولتی‌کپتر استفاده می‌شود، باید فیرمور **Copter** آپلود شود. و همینطور برای **Rover** و زیردریابی^{۲۰}.

و اسّم دیگر هر یکی از انواع بالایی **ArduRover**، **ArduPlane** و **ArduCopter** می‌باشد. آپلود فیرمور تنها کافی نیست، بلکه باید تنظیمات بیشتری انجام شود که بعداً نشان داده می‌شوند.

مثلاً اگر سیستم عامل **Copter** را آپلود شود، خود کنترلر متوجه می‌شود که ربات از نوع **Copter** می‌باشد اما کدام نوع از **Copter** متوجه نمی‌شود، باید این را تعیین شود. این موضوع با استفاده از برنامه **Mission Planner** انجام می‌شود که خود این برنامه یک ایستگاه زمینی محسوب می‌شود. شرح ایستگاه‌های زمینی در قسمت ایستگاه زمینی (**Ground Station**) است.

^{۱۹} QuadPlane

^{۲۰} Submarine



شکل 2-16 انواع فیرمورهای آردوپایلت [28]

2-2-2 ایستگاه زمینی (Ground Station)

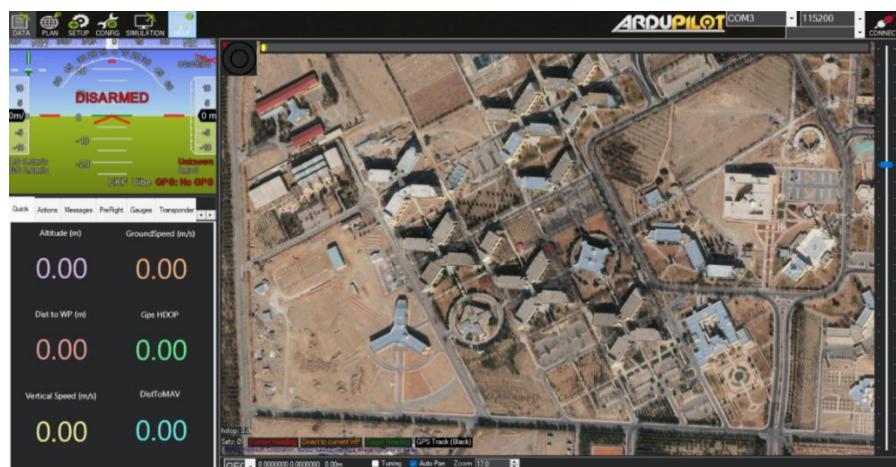
از نرم افزار **Ardupilot Mission Planner** [29] به عنوان ایستگاه زمینی اصلی استفاده شده است. آخرین نسخه که 1.3.77 است را نصب شده است. اگرچه از **APM Planner** [30] برای برخی از مسائل تست و **Mission Planner** [31] برای فرآیندهای شبیه‌سازی استفاده گردید، اما ایستگاه زمینی اصلی **MAVPROXY** مشابهی را که اگر با هواپیمای واقعی پرواز می‌کردید را نشان می‌دهد. یک ²¹GCS همچنین می‌تواند برای کنترل یک پهپاد در پرواز، آپلود دستورات ماموریت جدید و تنظیم پارامترهای پهپاد استفاده شود. همچنین اغلب برای نظارت بر جریان‌های ویدئویی زنده از دوربین‌های پهپاد استفاده می‌شود.

²¹ Ground Control Station

حداقل ده ایستگاه کنترل زمینی مختلف وجود دارد. روی دسکتاپ (Mission Planner) APM²²، QGroundControl²³، MAVProxy²⁴، Planner 2²⁵ یا Tower²⁶ برای تبلت/گوشی هوشمند Planner 3²⁷ وجود دارد که می‌توان از آنها برای SidePilot²⁸، AndroPilot²⁹، MAVPilot³⁰، DroidPlanner³¹ و ArduPilot³² (یعنی Copter³³، Plane³⁴ و Rover³⁵) استفاده کرد. [32]

Mission Planner که توسط مايكول اوبورن²² ساخته شده است، بسیار بیشتر از نام خود انجام می‌دهد. در اینجا برخی از ویژگی‌های این برنامه آورده شده است: [29]

- هر نقشه‌ای را می‌توان در Mission Planner مانند نقشه‌های گوگل²³، نقشه‌های بینگ²⁴ و بسیاری دیگر استفاده کرد. (شکل 2-17)
- تنها با استفاده از Mission Planner می‌شود پهپاد را به طور کامل کنترل کرد.
- بدون استفاده از هیچ سخت‌افزاری می‌توان از آن برای شبیه‌سازی استفاده کرد. (در قسمت SITL در قسمت شبیه‌سازی توضیح می‌شود)
- ماموریت‌ها و ایستگاه‌های بین راهی یا waypoints را می‌توان از Mission Planner به پهپاد فرستاد.



شکل 2-17- نقشه بخش از دانشگاه صنعتی اصفهان در Mission Planner

²² Micheal Oborne

²³ Google

²⁴ Bing

3-2-2 سیستم عامل

از سیستم عامل لینوکس استفاده شده است زیرا ROS فقط روی لینوکس کار می‌کند (در این پروژه از ویرایش 1.0 از ROS استفاده شده است، ویرایش 2.0 از ROS روی سیستم عامل ویندوز و مک هم کار می‌کند). نسخه لینوکس اوبونتو^{۲۵} ۲۰.۰۴ نصب شده است. این برای لپتاپ و چیزهای شبیه‌سازی است. پس از آن خواهیم دید که واجب بود که توزیع‌های قدیمی لینوکس و ROS را روی Raspberry Pi نصب کنیم تا اتصال به ROS به خوبی انجام شود.

اگر لپتاپ از ویندوز استفاده می‌کند، می‌توان از یک ماشین مجازی^{۲۶} برای نصب اوبونتو استفاده کرد. شرح روند نصب ماشین مجازی روی سیستم عامل ویندوز در این مرجع [33] موجود است.

4-2-2 سیستم عامل ربات (ROS)

در لپتاپ برای نصب ROS از توزیع Noetic استفاده شده است. البته روی لینوکس نصب می‌شود نه روی ویندوز. شرح روند نصب ROS روی سیستم عامل لینوکس (توزیع اوبونتو) در این مرجع [34] موجود است. در مورد سیستم عامل ربات در این قسمت معماری ROS صحبت می‌شود.

5-2-2 شبیه‌سازی

برای شبیه‌سازی از ابزاری به نام نرمافزار در حلقه (SITL^{۲۷}) استفاده شده است (شکل 18-2). شبیه‌ساز SITL (نرم افزار در حلقه) به کاربر امکان می‌دهد بدون هیچ سختافزاری، Copter، Plane یا Rover را اجرا کند [35].

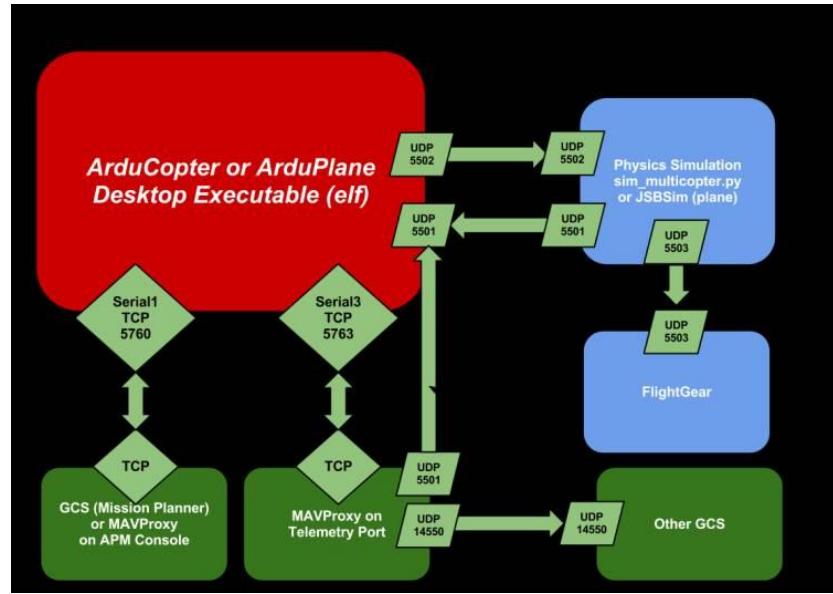
SITL به کاربر امکان می‌دهد ArduPilot را مستقیماً روی رایانه خود اجرا کند. SITL می‌تواند بر روی لینوکس (فقط معماری x86)، مک و ویندوز، یا در یک ماشین مجازی اجرا شود [36].

از SITL برای تست کدهای ROS که نوشته شده‌اند استفاده شده است، ابتدا با استفاده از شبیه‌سازی تست و سپس روی سختافزار واقعی آپلود می‌شود. می‌توان هر کدی را در SITL نوشت و توسعه داد و آن را به راحتی تست شود.

²⁵ Ubuntu

²⁶ Virtual Machine

²⁷ Software in the loop



[35] شکل ۲-۱۸- معماری SITL

۲- خلاصه و جمع بندی

در این فصل، انواع نرم‌افزار و سخت‌افزار مورد استفاده در این پروژه مورد بحث قرار داده شد. پس از خواندن این فصل، خواننده باید در مورد ابزارهای مختلف مورد استفاده برای ساخت کوادروتور اطلاعاتی تا حدی داشته باشد. در فصل بعدی، مراحل آماده‌سازی پهپاد برای پرواز به دو صورت، شبیه‌سازی و واقعی را مورد بحث قرار خواهیم داد.

فصل سوم: اتصال و شبیه‌سازی

در این فصل، در مورد اتصالات انجام شده بین پهپاد از یک طرف و ایستگاه زمینی و کنترل رادیویی از طرف دیگر صحبت خواهد شد. اما قبل از شروع ابتدا باید نحوه و شکل پیام‌هایی که از ایستگاه زمینی به پهپاد و بالعکس ارسال می‌شود، شناخته شود. صحبت در مورد MAVLink است.

1-3 شرح MAVLink

1-1-3 MAVLink چیست؟

MAVLink یا Micro Air Vehicle Link پروتکلی برای برقراری ارتباط بین وسیله نقلیه کوچک بدون سرنوشتین و ایستگاه زمینی است. MAVLink اولین بار در اوایل سال 2009 توسط لارنر مایر^{۲۸} تحت مجوز LGPL^{۲۹} منتشر شد. بیشتر برای ارتباط بین ایستگاه کنترل زمینی (GCS) و وسائل نقلیه بدون سرنوشتین و در ارتباطات بین سیستم فرعی یا subsystem وسیله نقلیه استفاده می‌شود. می‌توان از آن برای انتقال مکان GPS، جهت‌گیری و سرعت وسیله نقلیه استفاده کرد. [16]

بنابراین پهپاد پیام‌هایی را به ایستگاه زمینی ارسال می‌کند که به ایستگاه زمینی اجازه می‌دهد وضعیت پهپاد را نظارت کند. ایستگاه زمینی همچنین می‌تواند برای انجام برخی اقدامات مانند کنترل حرکت پهپاد، پیام‌هایی را به پهپاد ارسال کند. بنابراین پروتکل MAVLink اساساً مجموعه‌ای از پیام‌ها، و ساختارها و قالب‌های آنها و نحوه تبادل آنها بین پهپاد و ایستگاه زمینی را مشخص می‌کند.

2-1-3 MAVLink پیام‌های

پیام MAVLink (این پیام هم «msg» نامیده می‌شود) اساساً جریانی از بایت‌ها است که توسط ایستگاه زمینی مانند Mission Planner کدگذاری شده‌اند و از طریق سریال USB^{۳۰} یا Telemetry به APM ارسال می‌شوند. وقتی در مورد ساختار پیام MAVLink صحبت می‌شود، این ساختارها یک بسته را تشکیل می‌دهند. پس از کلمه بسته یا Packet استفاده می‌شود. [37]

هر بسته MAVLink دارای طول 17 بايت و ساختار آن به شرح زیر است: (شکل 13)

²⁸ Lorenz Meier

²⁹ Lesser General Public License

³⁰ Universal Serial Bus

طول پیام = 6 بایت سرآمد (Header) + 9 بایت محموله (Payload) + 2 بایت (Checksum)

شکل 3-1- تقسیم بایت‌های پیام [38] MAVLink

Byte Index	Content	Value	Explanation
0	Packet Start Sign (STX)	0xFE	Indicates start of a new packet
1	Payload Length (LEN)	0-255	Indicates length of the following payload
2	Packet sequence (SEQ)	0-255	Packet transfer sequence information for detecting packet loss
3	System ID (SYS)	1-255	ID of the sending system; Allows to identify multiple platforms on the same network
4	Component ID (COMP)	0-255	ID of the sending component; Allows to identify multiple components on the same platform
5	Message ID (MSG)	0-255	ID of the message; Define what payload means, and how to decode it
6 to (n+6)	Data (Payload)	0-255 (bytes)	Data of message; depends on the message ID
(n+7) to (n+8)	Checksum (CKA and CKB)	ITU X.25/SAE AS-4 hash of bytes 1 to (n+6); It includes MAVLINK_CRC_EXTRA parameter computed from message fields	

آنچه مورد توجه ماست این است:

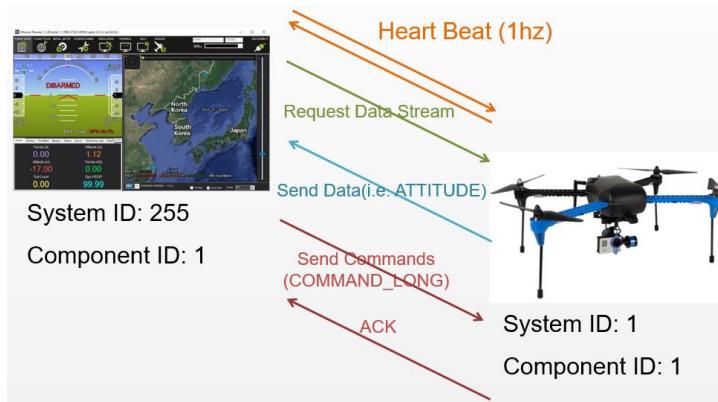
- شناسه سیستم (System ID) (معروف به منبع پیام): این منبع (یعنی Mission Planner) پیام را به APM از طریق تله‌متری بی‌سیم یا پورت USB ارسال می‌کند. نرم افزار APM یک بررسی منظم انجام می‌دهد تا بداند که این پیام برای خودش است.

- شناسه مؤلفه (Component ID) (معروف به زیرسیستم درون سیستم): هر زیرسیستم در سیستم اصلی در حال حاضر هیچ زیرسیستم وجود ندارد و واقعاً از آن استفاده نمی‌شود.

- شناسه پیام (Message ID): این مشخص می‌کند که بار یا Payload به چه معناست، و چگونه می‌توان آن را رمزگشایی کرد.

- بار (Payload): داده‌های پیام، بستگی به شناسه پیام است.

یک بسته داده است که حاوی تعداد ثابت بایت است (یعنی 17). APM بایتهای جریان (از هوا) را دریافت می‌کند، آن را به رابط سخت‌افزاری می‌رساند (به عنوان مثال از طریق سریال UART یا تله‌متری) و پیام را در نرم‌افزار رمزگشایی می‌کند. [37]



شکل 3-2- نحوه تبادل اطلاعات بین پهپاد و Mission Planner توسط پیام‌های [39] MAVLink

3-1-3 MAVLink پیام

در شکل 3-2- یک فلش قرمز رنگ از Mission Planner به پهپاد که دستورات به کوادرotor می‌فرستد وجود دارد. نوع این پیام **COMMAND_LONG** است[38]. به شرح این نوع پیام خواهیم پرداخت.

شکل 3-3- جدول مفصل پیام [40] COMMAND_LONG

Field Name	Type	Values	Description
target_system	uint8_t		System which should execute the command
target_component	uint8_t		Component which should execute the command, 0 for all components
command	uint16_t	MAV_CMD	Command ID (of command to send).
confirmation	uint8_t		0: First transmission of this command. 1-255: Confirmation transmissions (e.g. for kill command)
param1	float		Parameter 1 (for the specific command).
param2	float		Parameter 2 (for the specific command).
param3	float		Parameter 3 (for the specific command).
param4	float		Parameter 4 (for the specific command).
param5	float		Parameter 5 (for the specific command).
param6	float		Parameter 6 (for the specific command).
param7	float		Parameter 7 (for the specific command).

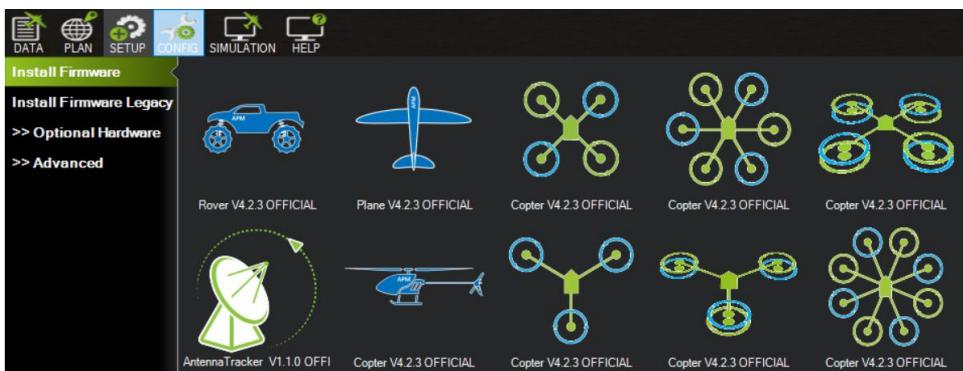
- قسمت target_system یعنی این پیام به کدام سیستم ارسال می‌شود. (یا کدام سیستم باید این پیام را اجرا کند). معمولاً این نوع پیام‌ها از ایستگاه زمینی به پهپاد ارسال می‌شوند.
- قسمت target_component یعنی کدام جزء از این سیستم پیام را اجرا می‌کند.
- قسمت command مهمترین قسمت در این نوع پیام می‌باشد. این قسمت مسؤول تعیین دستوری که به پهپاد ارسال می‌شود، می‌باشد. مثلاً اگر command عدد 21 باشد یعنی پهپاد به وضعیت LAND تبدیل می‌شود. و اگر 22 باشد یعنی پهپاد به وضعیت TAKE_OFF تبدیل می‌شود. این اعداد در قسمت MAVLink Commands (MAV_CMD) موجود در مرجع [40] تعیین می‌شوند. تقریباً 60 نوع command وجود دارد.
- قسمت confirmation اگر صفر باشد یعنی پیام برای اولین بار ارسال می‌شود، و اگر غیر از صفر باشد یعنی پیام به درستی ارسال شده است.
- هفتتا قسمت اخیر بستگی به نوع command استفاده شده در قسمت سوم می‌باشد. مثلاً اگر استفاده شده LAND command باشد، یعنی کل هفتتا پارامتر بی فایده می‌شوند. اگر باشد، فقط پارامتر اخیر که مسؤول تعیین ارتفاع یا altitude TAKE_OFF برای این حالت مفید است.

3-2 ارتباط با ایستگاه زمینی

همانطور که قبلاً گفته شد، Mission Planner به عنوان ایستگاه زمینی اصلی انتخاب شده است.

1-2-3 نصب فیرمور یا سیستم عامل

پس از اتصال برد APM به لپ‌تاپ با استفاده از سریال USB، با باز کردن Mission Planner و رفتن به گزینه SETUP، در این قسمت باید فریم ربات مورد نظر انتخاب شود و فیرمور نصب می‌شود. (شکل 3-4)

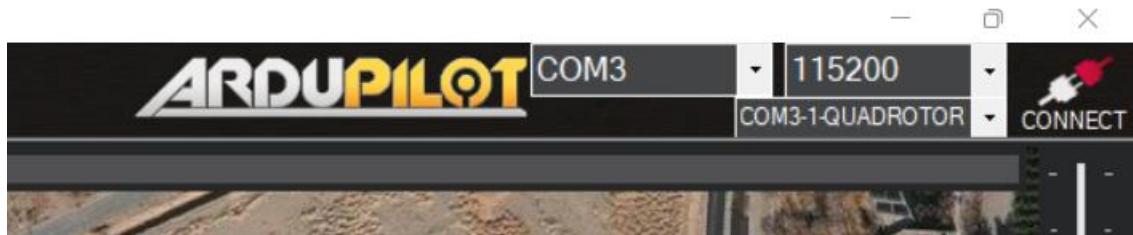


شکل 3-4 صفحه نصب فیرمور در Mission Planner

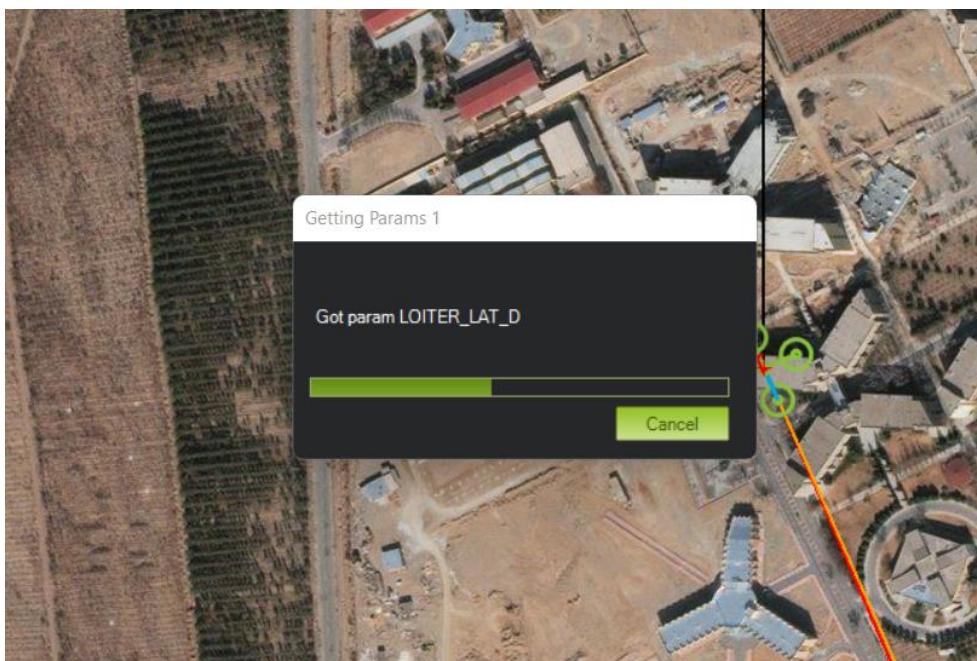
توجه داشته باشید که Mission Planner آخرین نسخه‌های سیستم عامل Ardupilot را ارائه می‌دهد. اگر سخت‌افزار مورد استفاده با آخرین نسخه فیرمور سازگار نیست، Mission Planner این کار را انجام می‌دهد. خود نرم‌افزار جدیدترین نسخه که با برد کنترل کننده پرواز سازگار است دانلود و نصب می‌کند. اگر یک نسخه معینی از فیرمور قرار نصب شود، باید به سایت آردوپایلت قسمت firmware download مراجعه شود [28].

2-2-3 ارتباط با کابل USB

پس از نصب فرمور، با استفاده از کابل micro usb، برد را به لپ‌تاپ متصل کرده و در Mission Planner بر روی دکمه اتصال (یا همان CONNECT) کلیک می‌شود (شکل 5-3). پیش‌فرض در حین اتصال با کابل baud rate 115200 است. نرخ باد یا همان baud rate نرخ انتقال داده می‌باشد. 115200 یعنی 115200 بیت در ثانیه منتقل می‌شود. بعد از زدن روی دکمه CONNECT، پارامترهای کنترل پرواز به Mission Planner منتقل می‌شوند (شکل 5-3).



شکل 5-3 دکمه CONNECT موجود در بالای صفحه Mission Planner



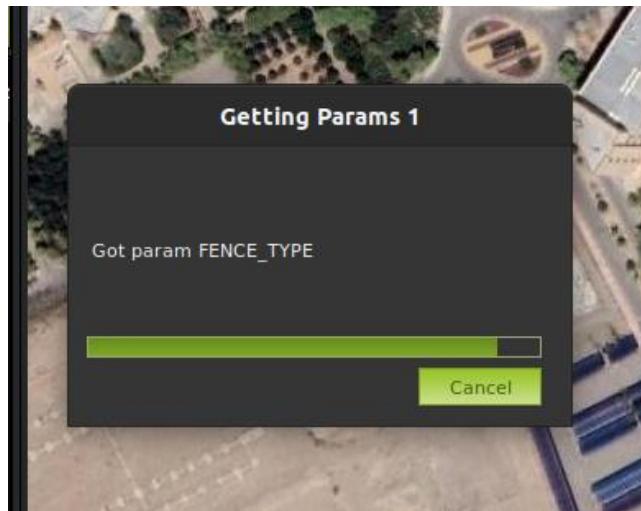
شکل 5-6 مرحله اتصال با کوادروتور (دریافت پارامترها)

3-2-3 ارتباط با تلمتری

ماژول تله‌متري زمين به لپ‌تاپ وصل مي‌شود و بعد از مدت کوتاهی چراغ‌های فرستنده و گيرنده تلمتری ثابت مي‌شوند و چشمک نمی‌زنند. (اين نشان دهنده برقراری ارتباط بین آنهاست). سپس دكمه اتصال در Mission Planner زده مي‌شود اما در اينجا باید توجه شود که نرخ باد (Baud rate) باید 57600 باشد نه 115200. مي‌شود اين تنظيمات در قسمت تنظيمات پaramترها تغيير داد [41] [42]. (شکل 7-3 و شکل 8-3)



شکل 7-3 دكمه connect در نرم‌افزار Mission Planner



شکل 8-3 در حال اتصال با کوادروتور توسط تله‌متري Mission Planner

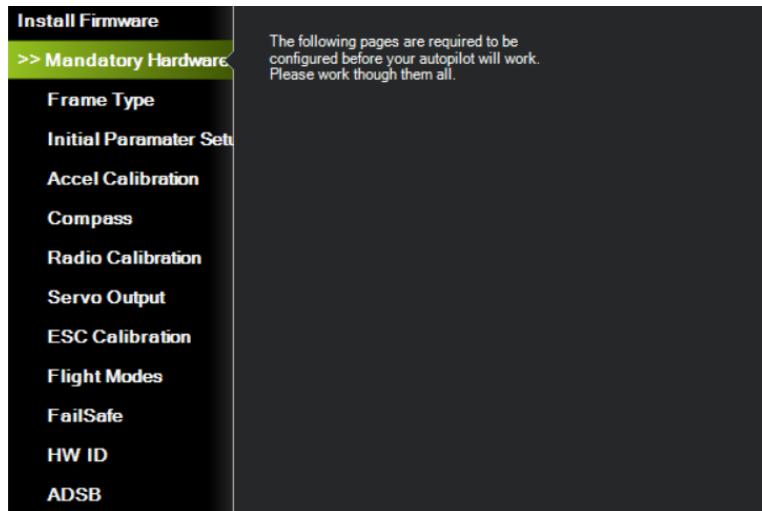
4-2-3 کالibrاسيون

با استفاده از سریال usb یا تله‌متري، به Mission Planner متصل شويد و سپس به گزینه setup برويد، سپس Mandatory Hardware را انتخاب کنيد و همانطور که در Mission Planner توضیح داده شده است، کالibrاسيون را شروع مي‌شود (شکل 9-3).

ابتدا نوع فريم را انتخاب مي‌شود، سپس شتاب‌سنج و قطب‌نما و سپس ESC و Radio ها کالibrه مي‌شوند. اينها اصلی‌ترین قسمت‌هایی هستند که باید کالibrه شوند. توجه داشته باشيد که تمام کالibrاسيون‌ها باید در خارج از ساختمان انجام شود

و در هنگام کالیبراسیون ESC‌ها، باید، برای ایمنی، ملخ‌ها یا اتصالات USB جدا کرد. برای کالیبره کردن رادیو به یک فرستنده RC نیاز هست و برای کالیبره کردن ESC باید دیتاشیت^{۳۱} آن با دقت خوانده شود و دقیقاً باید همان روشی که سازنده می‌گوید انجام شود.

کالیبره کردن قطعات آسان و مستقیم است اما اطلاعات بیشتر و دقیق‌تر برای کالیبراسیون قطب نما [43] و شتاب‌سنج [44] و رادیو [45] و اسپید کنترلر [46] موجود است.



شکل 9-3 صفحه که شامل کالیبراسیون قطعات متنوع در Mission Planner است

3-3 روشن کردن موتورها با استفاده از RC و Mission Planner

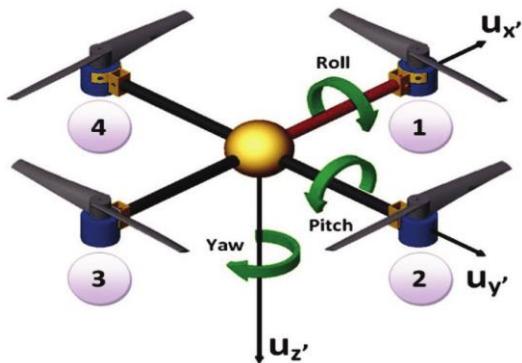
برای مسلح کردن موتورها (Motor Arming) ابتدا از فرستنده RC استفاده شده است. برای اهداف آزمایشی، موتورها در داخل آزمایشگاه روشن شده‌اند. برای انجام این کار، باید پارامتر بررسی ایمنی (Safety Check Parameter) را در بخش پارامترها در Mission Planner غیرفعال شود. از آنجایی که پهپاد طوری برنامه‌ریزی شده است که پرواز نکند مگر اینکه همه حسگرهای 100٪ خوب کار کنند و برای اینکه آسیب پیدا نشود، پهپاد باید بیرون باشد.

1-3-3 حالت‌های پرواز (Flight Modes)

از آنجایی که حالت‌های پرواز با روشن شدن موتورها مرتبط است، ابتدا چند حالت اصلی پرواز را توضیح می‌دهیم. فیرمور گزینه‌هایی به نام حالت‌های پرواز را ارائه می‌دهد تا به کاربران کمک کند بهترین گزینه را برای ماموریت خود انتخاب کنند. برای آزمایش و شروع، پنج حالت پرواز توصیه شده برای استفاده از آنها وجود دارد. این پنج حالت را توضیح خواهیم داد.

³¹ Datasheet

حالت ثبیت کردن (Stabilize): محور roll و pitch را خودتراز می‌کند. (شکل 10-3)



شكل 10-3 سه محور اصلی کوادراتور: roll، yaw و pitch

حالت نگه داشتن ارتفاع (Alt Hold): ارتفاع را نگه می‌دارد و roll و pitch را خودتراز می‌کند.

حالت سرگردان (Loiter): ارتفاع و موقعیت را حفظ می‌کند، از GPS پرای حرکات استفاده می‌کند.

حالت برگشت به مبدأ (Return to Launch) یا RTL: بازگشت بالای محل برخاستن، ممکن است شامل فرود نیز باشد.

حالت خودکار (Auto): مامو بیتی که از پیش تعریف شده را اجرا می‌کند.

⁴⁸ جالت‌های بواز مختلف در ابن ماجه [48] به تفصیل موحود است.

RC، وشن: موقع‌ها با استفاده از 2-3-3

به طور خیلی مختصر: پس از کالیبره کردن رادیو و ESCها و پس از اتصال پهپاد به باتری، فقط باید RC روشن شود و **Rudder**، **Throttle**، **Aileron** و **Elevators** به سمت راست بروند.

کل طویله

فرستنده خود را روشن کنید. باتری LiPO را به برد وصل کنید. چراغ‌های قرمز و آبی باید برای چند ثانیه چشمک بزنند زیرا ژیروسکوپ کالیبره شده است (کوادروتور را حرکت ندهید). بررسی‌های قبل از مسلح شدن (Pre-arming Checks) به طور خودکار اجرا می‌شود و در صورت مشاهده هرگونه مشکل، LED زرد چشمک می‌زنند و خرابی در ایستگاه زمینی نمایش داده می‌شود. بررسی کنید که سوئیچ حالت پرواز روی PosHold، AltHold، ACRO، Stabilize، Loiter یا تنظیم شده باشد (در مورد حالت‌های پرواز در قسمت قبلی صحبت شده است). اگر از خلبان خودکار با سوئیچ اینمنی استفاده می‌کنید، آن را فشار دهید تا چراغ ثابت شود. اگر قصد استفاده از یک حالت خودکار (مانند Auto، RTL، Loiter، وغیره) را دارید، باید کوادروتور به PosHold یا Loiter تغییر شود و منتظر بماند تا LED‌ها سبز شوند که نشان دهنده GPS خوب است. با یافتن نگه داشتن Throttle، موتورها را مسلح کنید و به مدت 5 ثانیه Rudder به سمت راست

حرکت کنید (شکل 113). معنای Throttle و Rudder را برای مدت طولانی در شکل 12-3 واضح است. AutoTrim را شروع خواهید کرد. پس از مسلح شدن، (بیش از 15 ثانیه) درست نگه ندارید، در غیر این صورت ویژگی AutoTrim را شروع خواهید کرد. پس از LED ها جامد می شوند و پروانهها شروع به چرخش می کنند. Throttle را برای پرواز بالا ببرید [49].



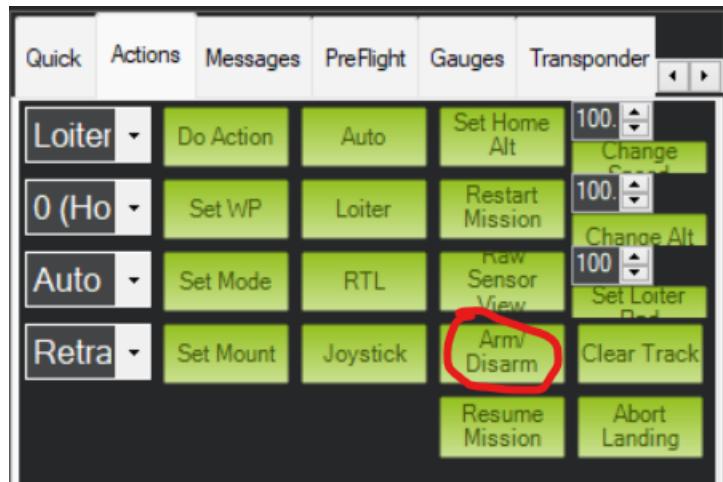
شکل 11-3- حالت مسلح کردن موتورها در RC



شکل 12-3 تقسیم RC در یک Throttle ،Aileron ،Rudder و Elevator [50]

3-3-3 روشن موتورها با استفاده از Mission Planner

همانطور که در قسمت های قبل گفته شد پهپاد را به Mission Planner متصل شده و سپس فقط دکمه Arm را فشار شود و موتورها روشن می شوند (شکل 13-3). حتما باید باتری وصل شود.



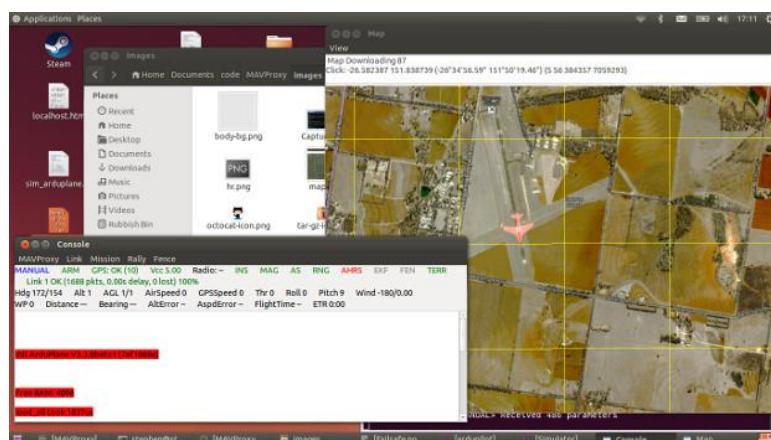
شکل 3-13-3 قسمت Mission Planner در برای روشن و خاموش کردن موتورها

4-3 شبیه‌سازی با MAVProxy و GAZEBO

چیست؟ 1-4-3 MAVProxy

بسته نرمافزاری ایستگاه زمینی پهپاد برای سیستم‌های مبتنی بر GCS یک MAVProxy با عملکرد کامل برای پهپادها است که به عنوان یک ایستگاه زمینی قابل حمل و قابل گسترش برای هر سیستم مستقلی که از پروتکل MAVLink پشتیبانی می‌کند (مانند سیستمی که از ArduPilot استفاده می‌کند) طراحی شده است. می‌توان آن را از طریق مژوی‌های افزودنی گسترش داد یا با ایستگاه زمینی دیگری مانند APM Planner 2، Mission Planner 2، QGroundControl وغیره تکمیل کرد تا یک رابط کاربری گرافیکی ارائه دهد [31]. (شکل 143)

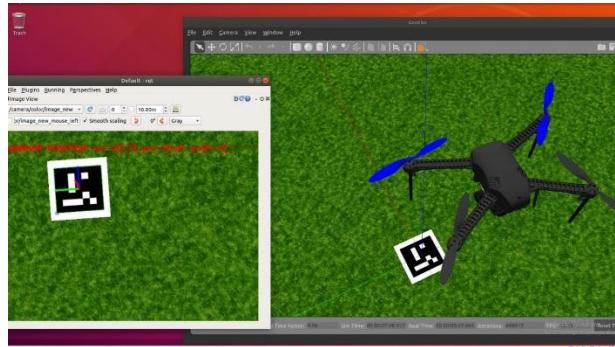
MAVProxy معمولاً توسط توسعه دهندگان (مخصوصاً با SITL) برای آزمایش بیلدہای (Builds) جدید استفاده می‌شود. بنابراین برای جمع‌بندی، این یک ایستگاه زمینی است که با شبیه‌سازی SITL استفاده می‌شود تا کاربران بتوانند برنامه‌های خود را آزمایش کنند. در این مرجع [31] ویژگی‌های این ایستگاه زمینی موجود است.



شکل 3-14-3- ایستگاه زمینی MAVProxy در سیستم عامل لینوکس [31]

3-4-2 شبیه‌ساز GAZEBO چیست؟

یک شبیه‌ساز پویا سه بعدی با قابلیت شبیه‌سازی دقیق و کارآمد جمعیت روبات‌ها در محیط‌های پیچیده داخلی و خارجی است (شکل 153). Gazebo پیشرو در شبیه‌سازی ربات است [51]. این ابزاری است که صدها هزار کاربر و توسعه‌دهنده در سراسر جهان به آن اعتماد دارند [52].



شکل 15-3- شبیه‌سازی کوادرورتور در شبیه‌ساز GAZEBO

3-4-3 تنظیم محیط SITL و نصب GAZEBO

برای شروع شبیه‌سازی، ابتدا باید MAVProxy و GAZEBO نصب شوند. هم روی ویندوز و هم روی لینوکس اجرا می‌شود اما GAZEBO فقط روی لینوکس اجرا می‌شود (راهی هست که روی ویندوز اجرا شود اما بهتر است روی لینکس نصب شود). برای آن از لینوکس استفاده شده است. برای تنظیم محیط SITL و نصب GAZEBO، رجوع شود به پیوست 1 : تنظیم محیط برای SITL و اجرای آن و پیوست 2 : نصب GAZEBO و اجرای آن .

5-3 خلاصه و جمع‌بندی

برای جمع‌بندی، در این فصل در مورد پروتکل MAVLink، اتصال کوادرورتور با ایستگاه زمینی، کالیبراسیون سنسورهای آن، اتصال آن به RC و شبیه‌سازی کوادرورتور در نرم‌افزارهای مختلف صحبت شده است. اکنون همه چیز در پهپاد آماده به جز اتصال ROS است. با تمام کارهای انجام شده در بالا، پهپاد اکنون آماده اتصال به سیستم عامل ربات است. در فصل بعد، این موضوع را مورد بحث قرار خواهیم داد.

فصل چهار: ارتباط با سیستم عامل ربات (ROS)

1-4 سیستم عامل ربات (ROS)

1-1-4 ROS چیست؟

ربات‌ها عموماً دارای واحد ادراکی (Sensor) و واحد عملگر (Actuator) هستند. در این حالت برای اینکه ربات به درستی بتواند ماموریت خود را انجام دهد، نیاز به این است که برای هر کدام از واحدهای فوق کدهایی نوشته شود و ارتباط مؤثری بین این کدها برقرار گردد. سیستم عامل ربات‌ها (ROS)، بستری قدرتمند و سریع را برای یکپارچه‌سازی کدهای مربوط به هر یک از واحدهای ربات فراهم می‌سازد. به کمک ROS می‌شود تحت لینوکس کد مربوط به هر یک از واحدهای فوق را در یک گره (Node) نوشت و ارتباط موثر بین این گره‌ها را از طریق پیام‌هایی به نام (Topic) ایجاد شود [53].

پیرامون سال ۲۰۰۷ اولین نسخه سیستم عامل ربات در دانشگاه استنفورد^{۳۲} ایجاد شد. از آن زمان تاکنون ROS گسترش زیادی پیدا کرده است. جامعه آماری بزرگی از برنامه‌نویسان در حوزه توسعه ROS فعالیت دارند. ROS در حقیقت یک سیستم عامل نیست، بلکه چارچوبی (Framework) برای برنامه‌نویسی ربات فراهم می‌کند که با داشتن استانداردهای تبیین شده، اعضای یک تیم بتوانند هم زمان برای برنامه‌نویسی یک ربات فعالیت کنند. با هر یک از زبان‌های پایتون^{۳۳} یا C++ می‌توان برای ROS کدنویسی کرد [53].

2-1-4 چرا از سیستم عامل ربات استفاده می‌کنیم؟

دلایل زیادی وجود دارد که باعث می‌شود مهندسان و توسعه دهندهای استفاده از ROS را برای پروژه‌های رباتیک خود ترجیح دهند که در زیر به برخی از این دلایل اشاره می‌شود:

³² Stanford University

³³ Python

الف- همان کد را می‌توان برای انواع مختلفی از ربات‌ها اعمال کرد: بازوهای رباتیک، هواپیماهای بدون سرنشین، پایگاه‌های متحرک، ... هنگامی که در مورد نحوه برقراری ارتباط بین تمام گره‌های برنامه یاد گرفته شد، می‌توان به راحتی قطعات جدید را راهاندازی کرد. اگر کار بین ربات‌های متفاوت باشد، به راحتی می‌توان بین این ربات‌ها رفت [54].

ب- با استفاده از ROS می‌توان برنامه‌ها را با استفاده از پایتون و C++ نوشت و بین آنها ارتباط برقرار کرد.

ج- ROS می‌تواند با چندین ROS Master کار کند. این بدان معناست که می‌توان ربات‌های مستقل زیادی داشت که هر کدام سیستم ROS خاص خود را دارند و همه ربات‌ها می‌توانند بین یکدیگر ارتباط برقرار کنند [54].

د- پایه اصلی ROS فضا و منابع زیادی را نمی‌گیرد. می‌توان به سرعت بسته‌های اصلی را نصب کرد و در عرض چند دقیقه شروع به کار کرد. به علاوه، می‌توان از ROS در رایانه‌های جاسازی شده (embedded) مانند بردهای Raspberry Pi نیز استفاده کرد (در این پروژه از این خاصیت استفاده شده است). بنابراین به راحتی می‌توان یک پروژه جدید را بدون دردسر زیاد شروع کرد [54].

ه- ROS به توسعه دهنگان این امکان را می‌دهد تا قبل از اعمال کردن هر کدی یا برنامه در دنیای واقعی، ربات خود را به راحتی در هر محیطی شبیه‌سازی کنند. ابزارهایی مانند Gazebo حتی امکان می‌دهند با روبات‌هایی که در اختیار نیست شبیه‌سازی کنند.

3-1-4 معماری ROS

نمی‌شود به جزئیات سیستم عامل ربات وارد کرد زیرا خود آن یک عالم است، اما برای گفته‌های بعدی شاید به این واژه‌شناسی نیاز باشد، لذا اگر خواننده اطلاعات قبلی در مورد ROS ندارد، بهتر است، برای بخش‌های آینده، معماری ROS را بداند.

ابتدا خوب است بگوییم که ROS سه نقش اساسی در کنترل دارد. حس می‌کند، فکر می‌کند و عمل می‌کند. بخش حس کردن می‌تواند دوربین، حسگر لیزری، حسگر اولتراسونیک یا GPS³⁴ باشد. بخش فکر کردن ممکن است هوش مصنوعی، یادگیری ماشینی یا پردازش سیگنال باشد. قسمت عمل می‌تواند انواع محرک‌ها باشد. برای جمع‌بندی، یک مثال بسیار شناخته شده می‌گوییم، ماشین خودران است. تمام اطراف خود را حس می‌کند، فکر می‌کند که در مرحله بعد چه کاری انجام دهد و با استفاده از محرک‌ها اقداماتی را انجام می‌دهد. اتومبیل‌های خودران با استفاده از ROS حرکت می‌کنند.

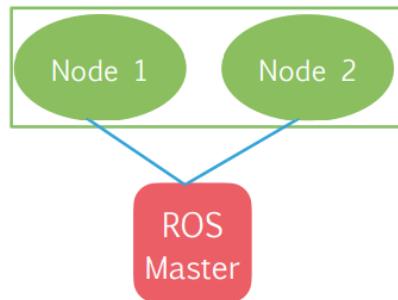
اسکریپتها یا کدهای ROS گره‌ها³⁵ نامیده می‌شوند. ارتباط بین این گره‌ها از طریق گره‌ای به نام ROS master انجام می‌شود (شکل 14). به آن ROS core نیز می‌گویند. اگر گره اصلی³⁶ خراب یا متوقف باشد، تمام سیستم از کار می‌افتد.

³⁴ Global Positioning System

³⁵ Node

³⁶ Ros master

این یک نقطه ضعف در ROS است که تمام سیستم به روشن یا خاموش یک گره وابسته است. این مشکل در ROS 2.0 رفع شد.



شکل 4-1- ارتباط دو گره با گره اصلی [55]

هر گره می‌تواند ناشر^{۳۷} یا مشترک^{۳۸} باشد. گره ناشر داده یا اطلاعات را منتشر یا ارسال می‌کند. گره مشترک این داده‌ها را دریافت می‌کند. ارسال داده‌ها از طریق چیزی به نام تاپیک‌ها^{۳۹} انجام می‌شود. رابطه ناشر-مشترک یک راه است، به این معنی که فقط ناشر می‌تواند اطلاعات ارسال کند. گره ناشر می‌تواند داده‌ها یا اطلاعات را برای بیش از یک گره مشترک منتشر کند. برای هر تاپیک باید نوع داده که می‌خواهد ارسال کند، مشخص کرد. اطلاعات یا داده‌هایی که توسط ناشر ارسال می‌شود به اسم پیام^{۴۰} معروف است. نوع پیام باید قبل از ارسال داده مشخص شود. مثلاً نوع پیامی که حسگر لیزر ارسال می‌کند float32 هست چون دارد فاصله اشیای محیط را محاسبه می‌کند. (شکل 4-2)

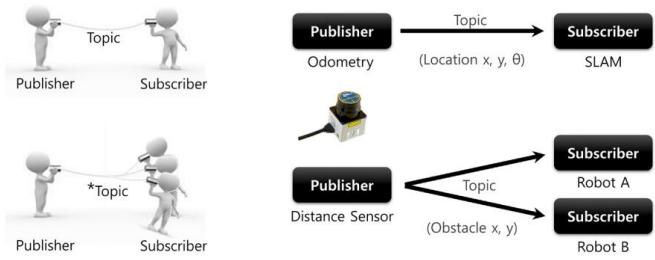
تاپیک‌ها سیم‌های مجازی هستند که داده‌ها را با نوع پیام خاصی ارسال می‌کنند. برای اجازه دادن به این ارتباط بین گره‌ها، نام تاپیک‌ها باید در هر دو گره یکسان باشد. (شکل 34)

³⁷ Publisher

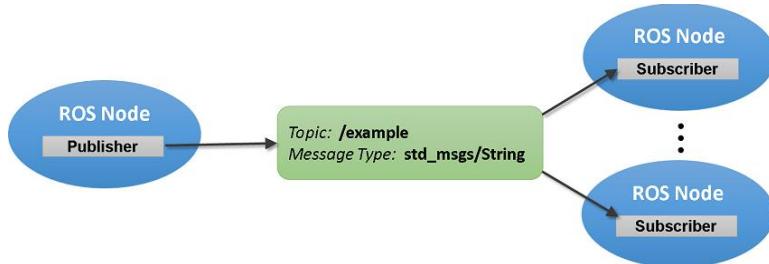
³⁸ Subscriber

³⁹ Topics

⁴⁰ Message



[55] ROS ارتباط ناشر و مشترک در



[56] شکل 4-3- ارتباط ناشر و مشترک توسط تاپیک و تعیین اسم تاپیک و نوع پیام

گره‌ها همچنین می‌توانند سرور^{۴۱} یا کلاینت^{۴۲} باشند. همانطور که ناشر داده‌ها را برای یک مشترک ارسال می‌کند، سروری نیز چیزی را برای مشتری(کلاینت) ارسال می‌کند.

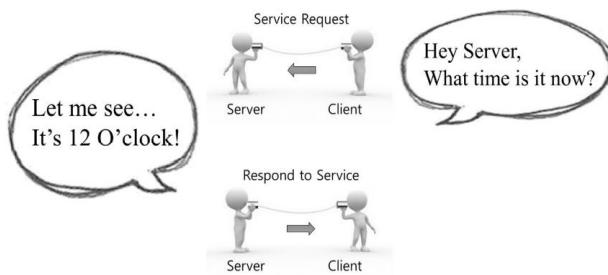
سیم مجازی این بار سرویس^{۴۳} نام دارد و به اختصار **Service** نامیده می‌شود. همانند یک تاپیک، یک پیام سرویس باید مشخص شود. گره مشتری(یا همان کلاینت) یک سرویس از گره سرور درخواست می‌کند و منتظر پاسخ می‌ماند. مشتری منتظر خواهد ماند تا زمانی که سرور پاسخ خود را ارسال کند. (شکل 4-4)

به عنوان مثال، یک سرویس "goToPoint"(یک سرویس که ربات را به یک نقطه معین ارسال می‌کند) ایجاد می‌شود: اگر ربات به نقطه مورد نظر رسیده باشد، سرویس در پایان به ما پاسخ می‌دهد. (برعکس اکشن‌ها، که در قسمت بعدی درموردان صحبت خواهیم کرد).

⁴¹ Server

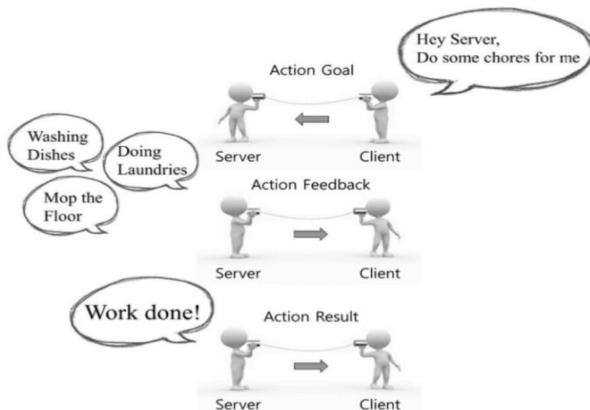
⁴² Client

⁴³ Service



شکل 4-4 ارتباط سرور و کلاینت (با سرویس‌ها) [55]

سرورها و مشتریان (کلاینت‌ها) همچنین ممکن است از طریق چیزی به نام اقدامات^{۴۴}(اکشن‌ها) با هم صحبت کنند. مانند خدمات(سرویس‌ها) است اما با یک تفاوت و آن اینکه اقدامات(اکشن‌ها) ناهمزمان هستند. همچنین نوع پیام‌های مربوط به اقدامات(اکشن‌ها) باید مشخص شود. به عنوان مثال، اگر یک اکشن "goToPoint" انجام دهیم، این اکشن در هر X میلی ثانیه برای شما ارسال می‌کند که چند متر باقی مانده است تا به نقطه مورد نظر برسید. سرویس‌ها چنین خاصیتی را ندارند. (شکل 5-4)

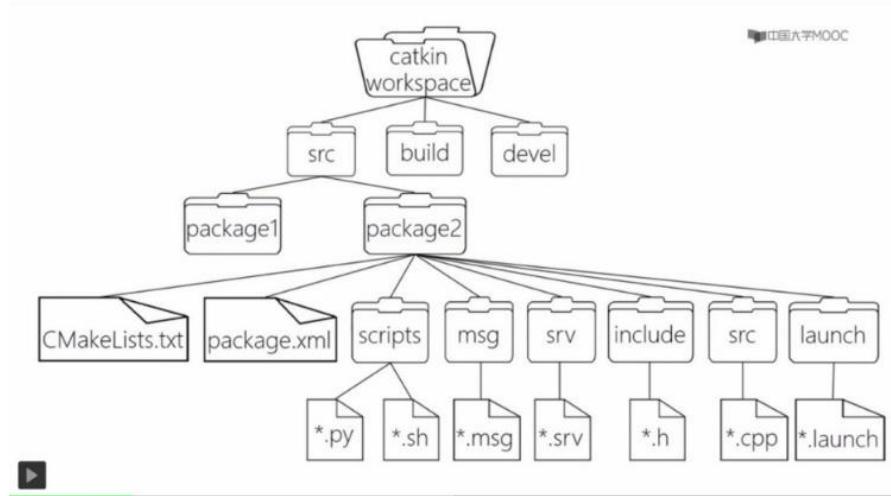


شکل 4-5 ارتباط بین سرور و کلاینت (با اکشن‌ها) [55]

4-1-4 فضای کاری و بسته ROS چی هستند؟

فضای کاری یا workspace به سادگی پوشه‌ای است که شامل تمام بسته‌های ROS است و یک بسته پوشه‌ای است که شامل برنامه‌ها و اسکریپت‌ها است. پس از ساخت فضای کاری برای کار، باید یک پوشه src ایجاد شود تا همه بسته‌ها در آن قرار بگیرند. سپس در پوشه‌ی بسته، اسکریپت‌های C++ پروژه در یک پوشه scripts قرار داده می‌شوند. فایل‌های دیگر مثل launch یا msg و اسکریپت‌های پایتون در پوشه src موجود است. (شکل 6-4)

⁴⁴ Actions



[58] ساختار فضای کاری ROS

Raspberry Pi 2-4 چیست؟

در این پژوهه و همانطور که قبلاً ذکر شد از برد Raspberry Pi 3B ROS (شکل 7-4) برای دانلود ROS و MAVROS استفاده شده است. این برد رزبری پای کامپیوتر پژوهه بود. رزبری پای رایانه کوچکی است که از سال 2006 در حال توسعه است و قطعات آن روی یک مادربرد به اندازه کارت بانکی سوار شده و Raspbian را اجرا می‌کند که یک نسخه اختصاصی از سیستم عامل لینوکس است که اختصاصاً برای این رایانه طراحی شده است. از Ubuntu mate 16.04 به عنوان سیستم عامل این کامپیوتر استفاده شده است زیرا نصب ROS روی راسپیان تقریباً یک روز کامل طول می‌کشد و برای build کردن (یا کمپایل کردن) یک فضای کاری باید هر بار بین دو تا هشت ساعت منتظر ماند و شاید در این وقت مشکل پیش بیاید که باعث خراب کل کار می‌شود. رزبری پای کاربردهای محاسباتی ابتدایی اداری، بازی‌های سطح پایین، دسترسی به اینترنت و ایمیل، بازپخش ویدئو و بسیاری قابلیت‌های دیگر دارد که به طور معمول از یک رایانه در قرن بیست و یکم انتظار می‌شود. رزبری پای همه این امکانات را با تعداد بسیار کمی از قطعات از جمله یک پردازنده ARM و قیمت بسیار پایین عرضه می‌کند.



[59] raspberry pi 3B شکل 4-7

برای نصب کردن سیستم عامل روی رزپری پای، به یک حافظه sd card نیاز است. کل مراحل نصب و تهیه به تفصیل در این قسمت [60] از مراجع موجود است. و نصب سیستم عامل ربات، همان روی لپتاپ، به دنبال این مرجع [61] می‌شود. باید ذکر شود که ویرایش سیستم عامل ربات که نصب شد kinetic می‌باشد و این ویرایش EOL⁴⁵ یا پایان زندگی می‌باشد یعنی دیگر ساپرت نمی‌شود. اما تا با ویرایش قدیمی MAVROS کار کند باید ویرایش kinetic نصب شود. و کل این کار برای سازگار بودن نرم‌افزار با کنترل کننده پرواز است.

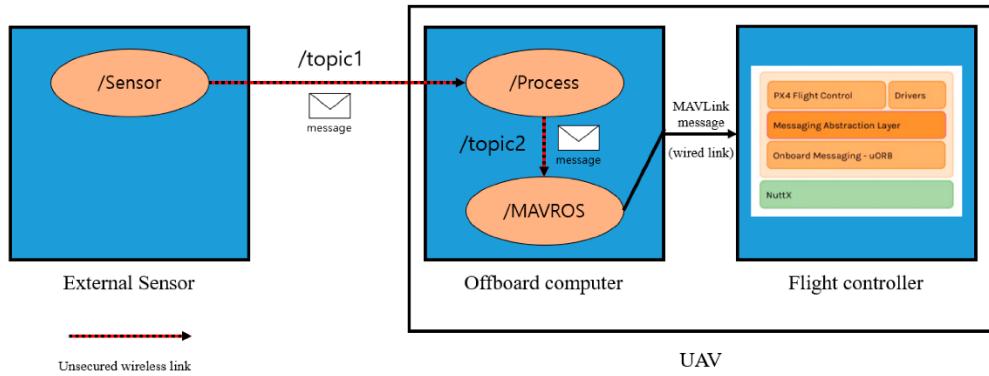
3-4 بسته MAVROS

1-3-4 تعریف بسته MAVROS

این بسته درایور ارتباطی برای خلبان‌های خودکار مختلف با پروتکل ارتباطی MAVLink فراهم می‌کند. علاوه بر این، MAVROS پل ارتباطی بین ROS و پروتکل MAVLink محسوب می‌شود. (از اسمش مشخص است که MAVLink+ROS هست). این بسته توسط مهندس روسی ولادیمیر ارمакوف که بعد از اتمام تحصیلات خود از دانشگاه صنعتی روسیه (MIREA)⁴⁶، در حوزه رباتیک و توسعه نرم‌افزار مشغول به کار شده است. بسته MAVROS را در سال 2014 نوشته است. هشتاد و پنج درصد از این بسته به زبان C++ و دوازده درصد به زبان پایتون نوشته شده است [1]. بقیه سه درصد فایل‌های متفرقه مانند xml یا txt هستند. در شکل 4-84- یک توضیحی مختصر در مورد ارتباط پهپاد با ROS با استفاده از بسته MAVROS توسط تاپیک‌ها نشان می‌شود.

⁴⁵ End Of Life

⁴⁶ Moscow Institute of Radio, Electronics and Automation



شکل 4-8- نقشه مختصر روند ارتباط کنترلر پرواز با ROS و همچنین با حسگر خارجی [62]

2-3-4 ساختار بسته MAVROS

پکیج MAVROS مانند بسته‌های دیگر از گره‌های مختلف و تاپیک‌ها تشکیل شده است که امکان ارتباط بین این گره‌ها را فراهم می‌کند. گره اصلی که تقریباً تمام کارها را انجام می‌دهد `mavros_node` است. اول به شرح ساختار `mavros_node` خواهیم پرداخت.

گره ارتباط اصلی محسوب می‌شود که خودش به تاپیک به نام `mavlink/to` مشترک است. این تاپیک داده‌های MAVLink که از ArduPilot می‌آید منتقل می‌کند.

نasher دو تاپیک به نام `Mavros_node` می‌باشد. `Mavlink/from` و `diagnostics` می‌باشد. `Mavros_node` داده‌های MAVLink به کنترل کننده پرواز و `diagnostics` مسؤول تشخیص وضعیت اتصال وغیره می‌باشد.

همانطور که قبلا در مورد MAVLink صحبت شد که بسته MAVLink دارای پارامترها مانند `system id` و `component id`، اینجا هم، در گره `mavros_node`، پارامترها شبیه آن وجود دارد. اینها به ترتیب عبارتند از:

:`system_id` - در آن عدد صحیح نوشته می‌شود که شناسه سیستم گره MAVLink است.

:`component_id` - این هم عدد صحیح می‌گیرد که شناسه مؤلفه گره MAVLink است.

:`target_system_id` - شناسه سیستم کنترل کننده پرواز (FCU).⁴⁷

:`component_system_id` - شناسه مؤلفه کنترل کننده پرواز.

:`fcu_url` - رشته را می‌گیرد. نشانه‌ی آدرس کنترل کننده پرواز. مثلا اگر کنترل کننده پرواز توسط یک تله‌متري

به کامپیوتر وصل شده باشد، آدرس می‌شود: `/dev/ttyUSB0:57600`.

در علم شبکه، و در سیستم عامل لینوکس، در پوشه‌ی `dev` می‌توان کل پورت‌ها را دید که در این حالت پورتی که تله‌متري وصل شده است به نام `ttyUSB0`

مشخص شد و نرخ باد 57600 می‌باشد.

⁴⁷ Flight Control Unit

- **fcu_protocol**: ویرایش MAVLink را مشخص می‌کند. (یا 1.0 یا 2.0). پیش‌فرضش 2.0 است. اما در این پروژه از ویرایش v1.0 به خاطر مشکل ناسازگار بودن کنترل کننده پرواز با ویرایش v2.0 استفاده شده است.

- **gcs_url**: اگر به ایستگاه زمینی قرار شود وصل کنیم، باید آدرس این GCS را تعیین کرد. که اتصال **udp** می‌شود. مثلاً آدرس به این شکل می‌شود: `gcs_url:=udp://:14855@14855` این یعنی ایستگاه زمینی روی پورت 14855 با پروتوكول **udp** وصل می‌شود.

پیشنهاد می‌شود که برای فهم عمیق و بهتر ساختار این بسته بزرگ، به مرجع اصلی این بسته مراجعه شود. [63] با ذکر مثال‌ها در قسمت‌های بعدی، ایده بیشتر روش می‌شود.

3-3-4 نصب بسته‌ی MAVROS

MAVROS با تمام نسخه‌های اخیر ROS از جمله Kinetic، Melodic و Noetic سازگار است. بسته‌های ROS گاهی به کتابخانه‌ها و ابزارهای خارجی نیاز دارند که باید توسط سیستم عامل ارائه شوند. این کتابخانه‌ها و ابزارهای مورد نیاز معمولاً به عنوان وابستگی‌های سیستمی (System Dependencies) شناخته می‌شوند. برای نصب MAVROS سیستم نیاز به بعضی از این وابستگی‌ها است. اصلی‌ترین dependency برای MAVROS **GeographicLib** می‌باشد. (همان که اگر از پایتون در سیستم عامل ربات استفاده شود، باید dependency به نام **rospy** نصب باشد). این وابستگی مسؤول واحدهای مختصات زمینی می‌باشد. تفصیل این وابستگی خارج از موضوع اصلی است. اطلاعات بیشتر در این مرجع [64] وجود دارد. به هر حال، می‌شود قبل از نصب وابستگی‌ها، بسته را نصب کرد و بعداً وابستگی‌ها را نصب می‌شوند.

برای مراحل نصب MAVROS روی لپتاپ و روی raspberry pi، به پیوست 3 : نصب بسته MAVROS روی لپتاپ؛ و پیوست 4 : نصب بسته raspberry pi روی MAVROS مراجعه کنید.

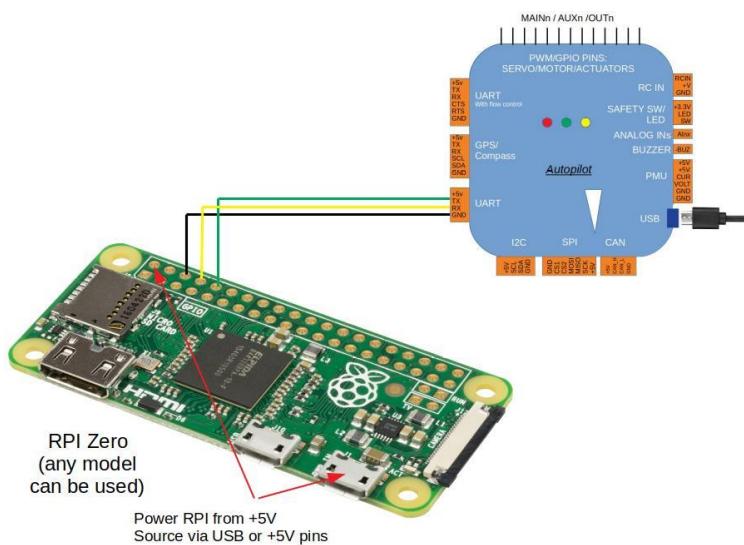
4-4 اتصال به ROS در SITL روی لپتاپ

SITL همانطور که قبلاً گفته شد، همیشه بهتر است قبل از رفتن به دنیای واقعی در یک شبیه‌سازی تست انجام شود. ساده‌ترین راه برای آزمایش است زیرا سخت‌افزاری وجود ندارد. به این ترتیب برنامه‌های ROS در صورت داشتن هرگونه مشکل به راحتی قابل بررسی هستند. مراحل اتصال به ROS در پیوست 5 : اتصال به SITL در ROS روی لپتاپ موجود است.

4-5 ارتباط واقعی با ROS

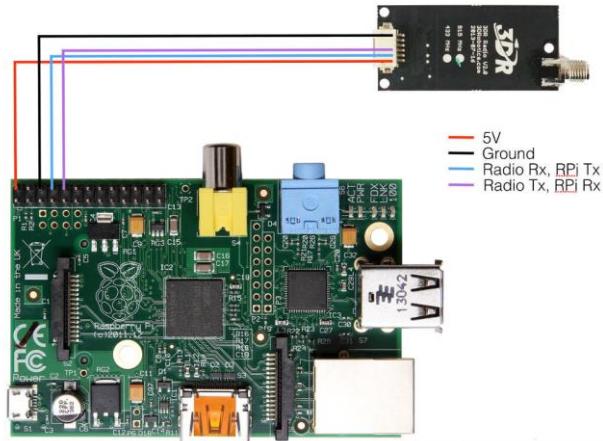
تا اینجا، MAVROS و ROS باید روی raspberry pi که اوبونتو 16.04 را اجرا می‌کند، نصب شده باشند. اکنون چندین راه وجود دارد که می‌توانیم کنترل کننده پرواز را به رزبری پای متصل کنیم که سه تا از آنها را می‌گوییم. استفاده از کابل USB یا تله‌متری (همانطور که قبل از اتصال کنترلر پرواز با MissionPlanner در قسمت ارتباط با کابل USB و ارتباط با تلمتری گفته شد) و از طریق اتصال پرت سریال (Serial Port). اتصال به رزبری پای ممکن است فقط برای استفاده از ROS نباشد، بسیاری از کارها را می‌توان با استفاده از رزبری پای غیر از ROS انجام داد.

1-5-4 ارتباط Serial Port با



[65] روند اتصال کنترل کننده پرواز با raspberry pi با استفاده از serial port

همانطور که در شکل 4-9 مشخص است، باید قسمت TELEMETRY یا همان جایی وصل در کنترل کننده پرواز به raspberry pi وصل کرد. Ground مال UART به رازبری پای وصل شده، RX به TX و TX به RX. رازبری پای را می‌توان با اتصال منبع 5 ولت به پین 5+ ولت یا از طریق USB تغذیه کرد. (شکل 104-)

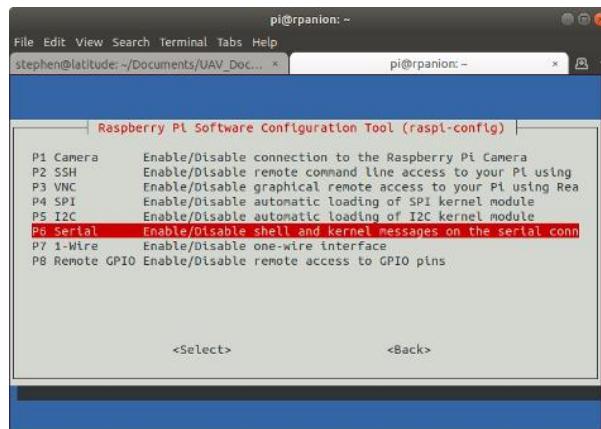


شکل 4-10- اتصال raspberry pi 3b با پورت‌های TELEMETRY کنترل کننده پرواز [66]

بعد فقط در ترمینال raspberry pi این دستور را اجرا می‌شود:

```
sudo raspi-config
```

ویک صفحه ظاهر می‌شود که روی گزینه Interfacing Options زده می‌شود و بعد Serial فعال می‌شود. (شکل 11-4)



شکل 4-11- گزینه Serial غیر فعال بودن

با این اتصال، الان ارتباط بین کنترل کننده پرواز و رزپری پای وجود دارد و می‌توان به ROS وصل کرد.

2-5-4 ارتباط با کابل usb یا تله‌متری

برای ارتباط با کابل usb کافی است که طرف micro USB کابل به کنترل کننده پرواز و طرف USB به رزپری پای متصل شوند. این باعث می‌شود که پرت ttyACM0 باز شود.

حال با استفاده از این دستور، به ROS متصل می‌شود:

```
roslaunch mavros apm.launch fcu_url:=/dev/ttyACM0: 115200 fcu_protocol: -v1.0
```

برای شروع کد `mavros` که در بسته `apm.launch` موجود است، می‌باشد.
`roslaunch mavros apm.launch`
`Fcu_url` برای تعیین آدرس کنترل پرواز که با کابل متصل شده است.
`115200` برای تعیین نرخ باد

`2.0` برای تعیین ویرایش `MAVLink` استفاده شده. از ویرایش `1.0` نه `2.0` استفاده شده چون `2.0` روی سخت‌افزار ما کار نمی‌کند.

الآن می‌توان کدهای مختلف را که ما نوشتمیم یا دستورهای آمده اجرا کرد و باید کوادروتور را جواب دهد.

برای ارتباط با تله‌متري، باید ماژول زمیني تله‌متري را به رزپری پای وصل کرد. با این اتصال باید ارتباط بین دو ماژول برقرار شود. الآن پورت `ttyUSB0` رزپری پای فعال می‌شود.

حال با استفاده از این دستور، به ROS متصل می‌شود:

```
roslaunch mavros apm.launch fcu_url:=/dev/ttyUSB0: 57600 fcu_protocol: -v1.0
```

فقط قسمت پورت و نرخ باد تغییر می‌شود.

6-4 ROS برنامه اجرا

اکنون که ارتباط با ROS رخ داده است، هر برنامه‌ای را می‌توان با استفاده از پایتون یا سی‌پلاس‌پلاس نوشت. نوشتن و اجرای یک برنامه ROS برای کنترل کوادروتور در پیوست 6 : نوشتن و اجرا کردن برنامه ROS نوشته شده است.

با اجرای دستور `rosrun apm_control apm_control.py` (البته بعد از برقرار ارتباط با ROS) : (شکل 124- و شکل 134- و شکل 144-)

```

PLEASE CHOOSE A COMMAND -- Press CTRL+C to EXIT

1: to set mode to GUIDED
2: to set mode to STABILIZE
3: to ARM the drone
4: to DISARM the drone
5: to TAKEOFF (10 meters high)
6: to LAND
7: print coordinates

Enter your input: █

```

شکل 4-12- گزینه‌ها داده شده با اجرای برنامه `apm_control.py`

```

Enter your input: 7

LONGITUDE: 149.1652374
LATITUDE: -35.3632621

```

شکل 4-13- رفتار برنامه `apm_control.py` با زدن عدد 7

```

Enter your input: ^C
Exit
Written by NABIL

```

شکل 4-14- رفتار برنامه `apm_control.py` با نوشتن `^C`

7-4 موارد اضافی: ارتباط لپ‌تاپ به رزپری پای توسط SSH

از پروتکل (Secure Shell) SSH برای ایجاد ارتباطی امن بین کاربر و سرور استفاده می‌شود. در این پروتکل با استفاده از کلید عمومی و رمزنگاری متقاضی، تمام محتوای ارسال شده بین کاربر و سرور، رمزنگاری می‌شود و تنها دو طرفی که کلید توافق شده‌ی مشترک دارند می‌توانند به محتوای اصلی دسترسی داشته باشند.

اگر لپ‌تاپ و رزپری پای روی یک شبکه انترنت باشند، می‌توان این ارتباط را برقرار کرد. ابتدا باید `ssh` روی هردو ماشین نصب شود:

```
sudo apt-get install ssh
```

بعد خوب است `ssh restart` اجرا شود.

در ترمینال لپتاپ `ssh raspberrypi_username@ipaddress` اجرا می‌شود. البته اسم حقیقی به جای `raspberrypi_username` نوشته می‌شود و `ipaddress` رزپری پای همینطور. آن کل دستورات را که کاربر می‌خواهد اعمال کند از لپتاپ اعمال می‌شوند.

فصل پنجم: نتائج

هوایپیماهای بدون سرنشین ابزارهای شگفت انگیزی هستند که مهندسان را همیشه مشتاق دیدن آینده این صنعت می‌کند. البته پیشرفت سریع و شگفت انگیز فناوری هر روز ما را با دستگاه‌های تکنولوژیکی جدید شگفت زده می‌کند که می‌تواند فکر کنند و یاد بگیرند. البته در مورد دنیای هوش مصنوعی صحبت می‌کنم. برای این پروژه، اولین قدم به این دنیا در زمینه پهپادها انجام شده است اما مراحل بسیار دیگری نیز وجود دارد. در اینجا توانتیم یک پهپاد را با استفاده از سیستم عامل ربات کنترل کیم که امروزه بهترین و ساده‌ترین پلتفرم برای برنامه‌نویسی ربات محسوب می‌شود. در این پروژه با Raspberry Pi کار کردیم که با آن می‌توان بسیاری از ارتقاء‌های ویژه را به پهپاد اضافه کرد. کد نوشته شده که به ما توانایی کنترل پهپاد را از ROS می‌دهد ساده است. بسیاری از گزینه‌های دیگر را می‌توان اضافه کرد و پهپاد ممکن است اقدامات بسیار پیچیده‌تری انجام دهد. اما از آنجایی که هدف پروژه انجام شد و مفهومی که به ما امکان انجام کارهای بیشتری را می‌دهد درک شد، این بدان معناست که وقت آن رسیده است که به راههای دیگری برای ارتقاء پهپاد در زمینه هوش مصنوعی فکر کنیم.

ممکن است فکر کنیم، بعد چه می‌شود؟ ایده‌های زیادی وجود دارد که می‌توان آنها را مطالعه و انجام داد. برای مثال، پیاده‌سازی الگوریتم‌های SLAM⁴⁸ روی وسایل نقلیه پرنده ممکن است یک چالش باشد، اما این تنها یک جهان است. ایده دیگر می‌تواند کنترل و پرواز پهپاد در داخل ساختمان‌ها باشد. برخی از ایده‌ها ممکن است در زمینه کشاورزی، برخی نظامی و برخی در زمینه آتش نشانی و نجات باشد. خوبیش این است که در این زمینه وقت استراحت ندارد.

⁴⁸ Simultaneous Localization And Mapping

پیوست

پیوست 1: تنظیم محیط برای SITL و اجرای آن

هم روی ویندوز و هم روی لینوکس اجرا می‌شود اما GAZEBO فقط روی لینوکس اجرا می‌شود (راهی هست که روی ویندوز اجرا شود اما بهتر است روی لینوکس نصب شود). برای آن از لینوکس استفاده شده است.

برای راه اندازی محیط لینوکس مراحل زیر دنبال شده است: [67]

دانلود گیت (git):

- `sudo apt-get update`
- `sudo apt-get install git`
 - `sudo apt-get install gitk git-gui`

سپس مخزن ArduPilot کلون می‌شود:

```
git clone https://github.com/ArduPilot/ardupilot.git
cd ardupilot
git submodule update --init --recursive
```

و اکنون سیستم آماده اجرای شبیه‌سازی MAVProxy در SITL شده است.

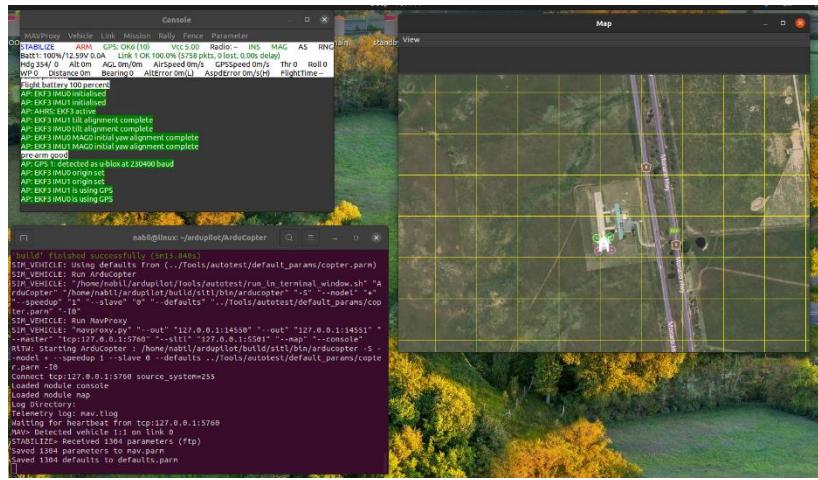
برای راه اندازی SITL به پوشه ArduCopter می‌رویم:

```
cd ardupilot/ArduCopter
```

سپس این دستور را برای اجرای فایل مسؤول شبیه‌سازی SITL اجرا می‌کنیم:

```
sim_vehicle.py -w
```

چنین چیزی شبیه شکل 3-16 می‌آید



شکل 0-1- سمت چپ بالا: console و سمت چپ پایین: ترمینال و سمت راست: نقشه

برای تست، بعضی از دستورات به شبیه‌سازی داده می‌شود، مثلا:

stabilize: برای تعیین حالت mode stabilize

arm throttle: برای روشن کردن موتورها

takeoff 10: برای پرواز کردن 10 متر

disarm throttle: برای خاموش کردن موتورها

mode circle: برای تبدیل حالت به حالت دایره (کواد به شکل دایره پرواز می‌کند)

mode land: برای تبدیل به حالت فرود.

لیست کامل دستورات در اینجا [68] موجود است.

پیوست 2 : نصب GAZEBO و اجرای آن

برای نصب gazebo در لینوکس، این مراحل باید دنبال شود: [69]

این دستورها در Terminal نوشته می‌شوند. دستور اول به این معنی است که رشته بعد از "echo" در "stable.list" نوشته شده است.

```
sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable `lsb_release -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'
```

دستور **w-get** برای نصب و باز کردن کلید بسته می‌باشد.

```
wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
```

```
sudo apt update
```

نصب gazebo ویرایش نهم.

```
sudo apt install gazebo9 libgazebo9-dev
```

کپی پوشه **ardupilot** مسئول شبیه‌سازی در gazebo

```
git clone https://github.com/khancyr/ardupilot_gazebo
```

ورود به پوشه **ardupilot_gazebo**

```
cd ardupilot_gazebo
```

ساختن یک پوشه به نام **build** و وارد آن برای **build** کردن بسته

```
mkdir build  
cd build
```

دستور **cmake** برای ساختن فایل‌های **build** در پوشه‌ی **build**

```
cmake ..
```

دستور **make** برای **build** کردن

```
make -j4
```

فقط فایل‌های کامپایل شده (در مرحله قبل) را در مکان‌های مناسب کپی می‌کند.

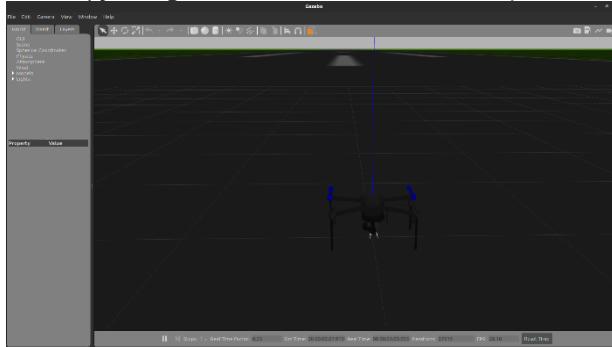
```
sudo make install
```

برای شروع gazebo با شبیه‌ساز SITL

```
gazebo --verbose worlds/iris_arducopter_runway.world
```

```
cd ~/ardupilot/ArduCopter
```

```
../Tools/autotest/sim_vehicle.py -f gazebo-iris --console --map
```



شکل 2-0 شبیه‌ساز GAZEBO با ArduCopter مجازی در آن

دستورات ذکر شده در مرجع [68] برای GAZEBO هم استفاده می‌شود.

پیوست 3 : نصب بسته MAVROS روی لپتاپ:

اول روی لپتاپ نصب میشود. بعد از نصب سیستم عامل ربات روی کامپیوتر خود، باید این مراحل دنبال شوند:

```
sudo apt-get install ros-kinetic-mavros ros-kinetic-mavros-extras
```

این دستور در ترمینال اجرا میشود. اگر ویرایش ROS که نصب شده بود غیر از kinetic باشد، باید به جای `kinetic` ویرایش خود را نوشت. این دستور بسته‌ی mavros و بسته‌ی mavros-extras [70] را نصب میکند.

```
Wget
https://raw.githubusercontent.com/mavlink/mavros/master/mavros/scripts/install_geographiclib_datasets.sh
```

```
./install_geographiclib_datasets.sh
```

این دو دستور، dependency مطلوب را نصب میکنند.

مراحل فوق به راحتی MAVROS را نصب میکنند. روش نصب دنبال شده به نام Binary Installation یا نصب باینتری است. حتما برای اطلاعات دقیق‌تر و واضح‌تر این قسمت در مراجع [71] را دنبال کنید. این روش نصب برای لپتاپ خوب است زیرا جدیدترین ویرایش MAVROS را نصب میکند و هیچ مشکلی با ناسازگار بودن وجود ندارد. اما وقتی که روی raspberry pi قرار شد نصب کنیم، باید از یک روشی که به ما اجازه می‌دهد که ویرایشی دلخواه انتخاب کنیم را استفاده کنیم. این روش به نام Source Installation یا نصب از منبع می‌باشد.

پیوست 4 : نصب بسته MAVROS روی raspberry pi

(فرض شده است که ویرایش سیستم عامل لینوکس UBUNTU 16.04 و سیستم عامل ربات kinetic می‌باشد) مرحله اول نصب `python-rosinstall-generator` و `python-catkin-tools` است. این دو به ترتیب مسؤول کردن فضای کاری توسط `rosinstall build` و دوم فایل‌های `catkin tools` حاوی اطلاعات مخازن با بسته‌های ROS تولید می‌کند.

```
sudo apt-get install python-catkin-tools python-rosinstall-generator -y
```

فضای کاری برای نصب بسته‌ها ساخته می‌شود:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
catkin init
wstool init src
```

در قسمت نصب روی لپ تاپ، بسته MAVLink توسط دستور نصب نشده بود اما خود به خود با MAVROS نصب می شود. اینجا باید MAVLink را نصب کرد. این فقط نسخهی ROS از پروتوكل MAVLink است:

```
rosinstall_generator --rosdistro kinetic mavlink | tee /tmp/mavros.rosinstall
```

: MAVROS نصب

```
rosinstall_generator --upstream mavros --deps | tee -a /tmp/mavros.rosinstall
```

واکشی بسته MAVROS در فضای کاری و افزودن وابستگی ها:

```
wstool merge -t src /tmp/mavros.rosinstall
wstool update -t src -j4
rosdep install --from-paths src --ignore-src -y
```

:geographiclib نصب وابستگی

```
./src/mavros/mavros/scripts/install_geographiclib_datasets.sh
```

Build کردن فضای کاری:

```
catkin build
```

البته باید فایل setup.bash که در پوشهی devel در فضای کاری موجود است، source شود. این تا دستور MAVROS بتواند بسته rosrun را پیدا کند.

```
source devel/setup.bash
```

اکنون که می خواهیم کنترل کننده پرواز (APM2.8) را با MAVROS وصل کنیم، هیچ ارتباطی برقرار نمی شود. پس از چندین ساعت و روز عیب یابی، آقای ارمکوف راهی برای تغییر نسخه های MAVROS و MAVLink به منظور سازگاری APM2.8 باز ارائه کرد.

به پوشهی mavros که در فضای کاری موجود است وارد می شویم و این دستور را برای تغییر ویرایش mavros به 0.30.0 اجرا می کنیم:

```
git checkout v0.30.0 -b release/0.30.0
```

: و همچنین برای mavlink وارد پوشهی آن می شویم و این دستور را اجرا می کنیم:

```
git checkout upstream/2019.7.7 -b release/2019.7.7
```

پیوست 5 : اتصال به SITL در ROS روی لپ تاپ

اول، یک عادت خوب ROS این است که همیشه در یک پوششی فضای کاری کار انجام شود، بنابراین باید یکی را ایجاد شود: در یک ترمینال، وهمان مراحل ایجاد فضای کاری، این دستورها را اجرا می‌شوند:

```
mkdir -p ardupilot_ws/src
cd ardupilot_ws
catkin init
cd src
```

آن ROS آماده کار است. الان باید یک شبیه‌ساز SITL را اجرا کرد (SITL همان طور که قبلا در قسمت شبیه‌سازی با **GAZEBO** و **MAVProxy** گفته شده بود، اجرا می‌شود) :

```
cd ..
cd ardupilot/ArduCopter
sim_vehicle.py -v ArduCopter --console --map
```

اکنون یک نمونه SITL در ترمینال خواهد بود که با دسترسی TCP و UDP راه اندازی شده است، باید خطی مانند زیر داشته باشد:

```
"mavproxy.py" "--master" "tcp:127.0.0.1:5760" "--sitl" "127.0.0.1:5501" "--out" "127.0.0.1:14550" "--out" "127.0.0.1:14551" "--map" "--console"
```

هر دو «out» به اتصال UDP ایجاد شده توسط MAVProxy اشاره دارند. از دسترسی UDP با mavros استفاده خواهیم کرد. به ترمینال ROS خود برگردیم. یک پوششی جدید برای فایل launch ایجاد کنیم. (کار در پوشش ~/ardupilot_ws/src می‌باشد):

```
mkdir launch
cd launch
```

فایل launch پیش‌فرض MAVROS را برای ArduPilot کپی می‌کنیم:

```
roscp mavros apm.launch apm.launch
```

باید یک فایل جدید به نام "apm.launch" در پوشش خود موجود باشد. دستور "roscp" فقط کمکی برای فرآخوانی فرمان سیستم در بسته ROS است. آن را با gedit یا هر ویرایشگر دیگر باز شود و در آن به جای خط اول که این است:

```
<arg name="fcu_url" default="/dev/ttyACM0:57600" /> <!-- Port et baudrate of the connexion with Pixhawk -->
```

این خط گذاشته می‌شود:

```
<arg name="fcu_url" default="udp://127.0.0.1:14551@14555" />
```

بعد این دستور را اجرا می‌شود:

```
roslaunch apm.launch
```

اگر چنین چیزی ظاهر شود یعنی کل کار درست شد:

```
[ INFO] [1496336768.500953284]: CON: Got HEARTBEAT, connected. FCU: ArduPilotMega /  
ArduCopter  
[ INFO] [1496336768.536761724]: RC_CHANNELS message detected!  
[ INFO] [1496336769.533950451]: VER: 1.1: Capabilities 0x0000000000001bcf  
[ INFO] [1496336769.534021653]: VER: 1.1: Flight software: 03060000 (8a4a2722)  
[ INFO] [1496336769.534146986]: VER: 1.1: Middleware software: 00000000 ( )  
[ INFO] [1496336769.534195446]: VER: 1.1: OS software: 00000000 ( )  
[ INFO] [1496336769.534280663]: VER: 1.1: Board hardware: 00000000  
[ INFO] [1496336769.534309086]: VER: 1.1: VID/PID: 0000:0000  
[ INFO] [1496336769.534331512]: VER: 1.1: UID: 00000000000000000000  
[ WARN] [1496336769.534370049]: CMD: Unexpected command 520, result 0  
[ INFO] [1496336778.533962739]: FCU: APM:Copter V3.6-dev (8a4a2722)  
[ INFO] [1496336778.534247677]: FCU: Frame: QUAD  
[ INFO] [1496336779.021134163]: PR: parameters list received  
[ INFO] [1496336783.535151119]: WP: mission received
```

سطر اول مهمترین سطر می باشد. این یعنی یک تپش قلب از کنترل کننده پرواز رسید یعنی متصل شد. می توان دید که توانست اطلاعات کنترل کننده پرواز بخواند و الان فقط منتظر دستورات است.

اطلاعات در مورد اجرای ROS با SITL در این قسمت [72] موجود است.

برای تست، می توان یک کد پایتون یا C++ را نوشت و اجرا کرد، یا می توان مستقیماً با استفاده از بعضی از دستورات که در بسته MAVROS موجود هستند، استفاده کرد. قسمت نوشتند که با پایتون برای قسمت آخر می گذاریم و در این قسمت با دستورات آماده کار می کنیم. هر دو روش (نوشتند کد یا استفاده از دستورات آماده) هم برای شبیه سازی و هم برای ارتباط در واقع قابل اعمال هستند. کل این دستورات اینجا [1] در قسمت Utility commands موجود هستند.

مثلا برای تغییر حالت پرواز:

```
rosrun mavros mavsys mode -c 0
```

این دستور حالت پرواز به 0 تغییر می کند (یعنی STABILIZE). لیست کامل اعداد حالت های پرواز در Error! Reference source not found. می باشد.

```

STABILIZE = 0, // manual airframe angle with manual throttle
ACRO = 1, // manual body-frame angular rate with manual throttle
ALT_HOLD = 2, // manual airframe angle with automatic throttle
AUTO = 3, // fully automatic waypoint control using mission commands
GUIDED = 4, // fully automatic fly to coordinate or fly at velocity/direction using GCS immediate commands
LOITER = 5, // automatic horizontal acceleration with automatic throttle
RTL = 6, // automatic return to launching point
CIRCLE = 7, // automatic circular flight with automatic throttle
LAND = 9, // automatic landing with horizontal position control
DRIFT = 11, // semi-autonomous position, yaw and throttle control
SPORT = 13, // manual earth-frame angular rate control with manual throttle
FLIP = 14, // automatically flip the vehicle on the roll axis
AUTOTUNE = 15, // automatically tune the vehicle's roll and pitch gains
POSHOLD = 16, // automatic position hold with manual override, with automatic throttle
BRAKE = 17, // full-brake using inertial/GPS system, no pilot input
THROW = 18, // throw to launch mode using inertial/GPS system, no pilot input
AVOID_ADSB = 19, // automatic avoidance of obstacles in the macro scale - e.g. full-sized aircraft
GUIDED_NOGPS = 20, // guided mode but only accepts attitude and altitude
SMART_RTL = 21, // SMART_RTL returns to home by retracing its steps
FLOWHOLD = 22, // FLOWHOLD holds position with optical flow without rangefinder
FOLLOW = 23, // follow attempts to follow another vehicle or ground station
ZIGZAG = 24, // ZIGZAG mode is able to fly in a zigzag manner with predefined point A and point B
SYSTEMID = 25, // System ID mode produces automated system identification signals in the controllers
AUTORotate = 26, // Autonomous autorotation

```

شکل ۰-۳- لیست کامل اعداد حالت‌های پرواز [73]

برای مسلح کردن موتورها (روشن کردن موتورها):

```
rosrun mavros mavsafe arm
```

و برای خاموش کردن موتورها به جای disarm arm نوشته می‌شود.

وهمینطور برای دستورات مختلف و متنوع می‌توان GAZEBO را شروع کرد و شبیه‌سازی را به صورت سه بعدی دید.
 (اجرا کردن GAZEBO در قسمت Error! Reference source not found. توضیح شده بود)

پیوست ۶ : نوشتن و اجرا کردن برنامه ROS

اول یک فضای کار برای این پروژه ساخته می‌شود. بعد در این فضای کار یک بسته به نام دلخواه (مثال: (apm_control بوجود آورده می‌شود:

```

mkdir -p ~/proj/src
cd ~/proj/
catkin_make

cd ~/proj/src
catkin_create_package apm_control std_msgs rospy

cd ~/proj/
catkin build

```

الآن یک پوشه را در `src` بسته به نام `scripts` ساخته می‌شود:

```
cd ~/proj/src
mkdir scripts
```

الآن فایلی که در آن می‌خواهیم کد را بنویسیم، بوجود آورده می‌شود:

```
touch apm_control.py
```

و کد در آن نوشته می‌شود و محیط کار دوباره `build` می‌شود. باید فراموش نوشد که همیشه باید فایل `(source setup.bash)` کرد. `source setup.bash` را `source setup.bash`

با اجرای این دستور، کد اجرا می‌شود:

```
rosrun apm_control apm_control.py
```

کد `:apm_control.py`

```
1. #!/usr/bin/env python
2. #وارد کردن کنابخانه‌ها و انواع پیام‌ها مورد نیاز این برنامه
3. import rospy
4. from sensor_msgs.msg import NavSatFix #پیام مسؤول انتقال مختصات GPS می باشد
5. from mavros_msgs.srv import *
6. #تعریف دو متغیر برای مختصات
7. latitude = 0.0
8. longitude= 0.0
9.
10. def setToGuided(): #تعریف تابع برای تبدیل حالت پرواز
11.     rospy.wait_for_service('/mavros/set_mode') #مدا کردن سرویس
12.     try:
13.         flightModeService =
14.             rospy.ServiceProxy('/mavros/set_mode', mavros_msgs.srv.SetMode)
15.             isModeChanged =
16.                 flightModeService(custom_mode='GUIDED') #تبدیل حالت پرواز
17.             except rospy.ServiceException as e:
18.                 print ("service set_mode call failed: %s. GUIDED
19.                     Mode could not be set. Check that GPS is enabled"%e) #اگر امکان تبدیل حالت نشد یک پیام به کاربر ارسال می‌شود که شاید مشکل از تشخیص موقعیت باشد
20.             def setToStabilize(): #تعریف تابع برای تبدیل حالت پرواز
21.                 rospy.wait_for_service('/mavros/set_mode')
22.                 try:
```

```

21.         flightModeService =
22.             rospy.ServiceProxy('/mavros/set_mode', mavros_msgs.srv.SetMode)
23.             isModeChanged =
24.                 flightModeService(custom_mode='STABILIZE') #return true or false
25.             except rospy.ServiceException as e:
26.                 print ("service set_mode call failed: %s. GUIDED
27. Mode could not be set. Check that GPS is enabled"%e)
28.             تعريف تابع برای فرود کوادرورتور
29.             def setToLand():
30.                 rospy.wait_for_service('/mavros/cmd/land')
31.                 try:
32.                     landService =
33.                         rospy.ServiceProxy('/mavros/cmd/land',
34. mavros_msgs.srv.CommandTOL)
35.                         isLanding = landService(altitude = 0, latitude = 0,
36. longitude = 0, min_pitch = 0, yaw = 0)
37.                         except rospy.ServiceException as e:
38.                             print ("service land call failed: %s. The vehicle
39. cannot land "%e)
40.             تعريف تابع برای روشن کردن موتورها
41.             def setToArm():
42.                 rospy.wait_for_service('/mavros/cmd/armming')
43.                 try:
44.                     armService =
45.                         rospy.ServiceProxy('/mavros/cmd/armming',
46. mavros_msgs.srv.CommandBool)
47.                         armService(True)
48.                         except rospy.ServiceException as e:
49.                             print ("Service arm call failed: %s"%e)
50.             تعريف تابع برای خاموش کردن موتورها
51.             def setToDisarm():
52.                 rospy.wait_for_service('/mavros/cmd/armming')
53.                 try:
54.                     armService =
55.                         rospy.ServiceProxy('/mavros/cmd/armming',
56. mavros_msgs.srv.CommandBool)
57.                         armService(False)
58.                         except rospy.ServiceException as e:
59.                             print ("Service arm call failed: %s"%e)
60.             تعريف تابع برای پرواز به ارتفاع ده متر
61.             def setToTakeoff():
62.                 rospy.wait_for_service('/mavros/cmd/takeoff')
63.                 try:

```

```

54.             takeoffService =
55.                 rospy.ServiceProxy('/mavros/cmd/takeoff',
56.                         mavros_msgs.srv.CommandTOL)
57.             takeoffService(altitude = 10, latitude = 0,
58.                           longitude = 0, min_pitch = 0, yaw = 0)
59.         except rospy.ServiceException as e:
60.             print ("Service takeoff call failed: %s"%e)
61.
62.         def globalPositionCallback(globalPositionCallback):
63.             تعریفتابع مسؤول تعیین مختصات موقعیت کوادرورتور#
64.             global latitude
65.             global longitude
66.             latitude = globalPositionCallback.latitude
67.             longitude = globalPositionCallback.longitude
68.             #print ("longitude: %.7f" %longitude)
69.             #print ("latitude: %.7f" %latitude)
70.         تعریفتابع برای نوشتن گزینه‌ها مختلف در ابتدای # برنامه
71.         #print ("\nWritten by Nabil\n")
72.         print ("\nPLEASE CHOOSE A COMMAND\n")
73.         print ("1: to set mode to GUIDED")
74.         print ("2: to set mode to STABILIZE")
75.         print ("3: to ARM the drone")
76.         print ("4: to DISARM the drone")
77.         print ("5: to TAKEOFF (10 meters high)")
78.         print ("6: to LAND")
79.         print ("7: print coordinates\n")
80.
81.     تعریفتابع که براساس انتخاب کاربر توابع # مختلف را اجرا می‌کند
82.     x='1'
83.     while ((not rospy.is_shutdown())):
84.         menu()
85.         x = input("Enter your input: ");
86.         if (x=='1' or x=="GUIDED" or x=="guided"):
87.             setToGuided()
88.         elif(x=='2' or x=="STABILIZE" or x=="stabilize"):
89.             setToStabilize()
90.         elif(x=='3' or x=="ARM" or x=="arm"):
91.             setToArm()
92.         elif(x=='4' or x=="DISARM" or x=="disarm"):
```

```

93.             setToDisarm()
94.         elif(x=='5' or x=="TAKEOFF" or x=="takeoff") :
95.             setToTakeoff()
96.         elif(x=='6' or x=="LAND" or x=="land") :
97.             setToLand()
98.         elif(x=='7' or x=="PRINT" or x=="print") :
99.             global latitude
100.            global longitude
101.            print ("\nLONGITUDE: %.7f" %longitude)
102.            print ("LATITUDE: %.7f\n" %latitude)
103.        else:
104.            print ("Exit")
105.            print ("Written by NABIL")
106.
107.
108.
109.
110.    if __name__ == '__main__':
111.        rospy.init_node('apm_control', anonymous=True)
112.        rospy.Subscriber("/mavros/global_position/raw/fix",
113.                         NavSatFix, globalPositionCallback)
114.        choices()
115.        rospy.spin()
116.

```

تعریف تابع اصلی که در آن تابع # : گزینه‌ها اجرا می‌شود و برنامه خاموش نمی‌شود مگر کابر این را بخواهد.
اینجا هم اسم گره را مشخص می‌شود

مراجع

- [1] V. Ermakov, "mavros," [Online]. Available: <https://github.com/mavlink/mavros>.
- [2] dji, "flame wheel arf," dji, [Online]. Available: <https://www.dji.com/cz/flame-wheel-arf/feature>.
- [3] ROBO.IN, "1045(10x4.5) SF Propellers Black 1CW+1CCW-1pair-Normal Quality," Generic (China), [Online]. Available: <https://robu.in/product/10x4-5-sf-props-black-cw-2pc/>.
- [4] EMAX, "EMAX Cooling New MT2216 II 810KV Brushless Motor CW CCW with 1045 Propeller for RC Multicopter," EMAX, [Online]. Available: <https://emaxmodel.com/products/emax-cooling-new-mt2216-ii-810kv-brushless-motor-cw-ccw-with-1045-propeller-for-rc-multicopter>.
- [5] ROBO.IN, "emax-mt2216-810kv-bldc-motor-cw-prop1045-combo-original," EMAX, [Online]. Available: <https://robu.in/product/emax-mt2216-810kv-bldc-motor-cw-prop1045-combo-original/>.
- [6] EMAX, "EMAX Simon Series 20A For Muti-Copter," EMAX, [Online]. Available: <https://emaxmodel.com/products/emax-simon-series-20a-for-muti-copter>.
- [7] Amazon, "Dilwe RC Drone ESC Emax Simon Series BLHeli 12A 20A 30A ESC 2-3S Battery for 130-210mm Racing Drone RC Part(20A)," Dilwe, [Online]. Available: <https://www.amazon.com/Dilwe-BLHeli-Battery-130-210mm-Racing/dp/B07MFQJBGS>.
- [8] T. Gudde, "Flight Controllers explained for everyone.," Fusion Engineering, [Online]. Available: <https://fusion.engineering/flight-controllers-explained-for-everyone/>.
- [9] ArduPilot, "Archived:APM 2.5 and 2.6 Overview," ArduPilot, 2021. [Online]. Available: <https://ardupilot.org/copter/docs/common-apm25-and-26-overview.html>.
- [10] pixhawk, "Products," [Online]. Available: <https://pixhawk.org/products/>.
- [11] ArduPilot Dev Team, "History of ArduPilot," [Online]. Available: <https://ardupilot.org/planner2/docs/common-history-of-ardupilot.html>.
- [12] H. JIN, "Best Drone Controller 2022: Top Review For You," LUCIDCAM, 05 09 2022. [Online]. Available: <https://lucidcam.com/best-drone-controller/>.
- [13] Banggood, "Ardupilot APM 2.8 Flight Controller Board Bend Pin with Protective Case for RC Multi Rotor Drone - With Compass," Topacc, [Online]. Available:

- https://usa.banggood.com/Ardupilot-APM-2_8-Flight-Controller-Board-Bend-Pin-with-Protective-Case-for-RC-Multi-Rotor-Drone-p-1123217.html.
- [14] HobbyKing, JR, [Online]. Available: https://hobbyking.com/en_us/jr-xg6-6-channel-2-4ghz-dmss-transmitter-w-telemetry-rg612bx-receiver-mode-2.html.
- [15] AKCELL, "JR RG831B 8 CHANNEL DMSS RECEIVER," JR, [Online]. Available: <https://akcelltrader.com/product/jr-rg831b-8-channel-dmss-receiver/>.
- [16] Wikipedia, "MAVLink," [Online]. Available: <https://en.wikipedia.org/wiki/MAVLink>.
- [17] ArduPilot Dev Team, "SiK Telemetry Radio," ArduPilot, [Online]. Available: <https://ardupilot.org/copter/docs/common-sik-telemetry-radio.html>.
- [18] Iranian Robotic, "3 فلایت کنترلر و مژویل ها DR," [Online]. Available: <http://robosigma.com/single-ttl-3drobotics-3dr-radio-telemetry-kit-433mhz-500mw-for-apm-2-6-px4-flight-controller>.
- [19] Alitools, "NEO-M8N Flight Controller GPS Module with On-board Compass M8 Engine PX4 Pixhawk TR For OCDAY Drone GPS," OXA, [Online]. Available: <https://alitools.io/en/showcase/neo-m8n-flight-controller-gps-module-with-on-board-compass-m8-engine-px4-pixhawk-tr-for-octday-drone-gps-4001339980107>.
- [20] Techplayon, "What is GPS, how GPS module is used for base station applications, how to choose GPS module ?," 04 10 2017. [Online]. Available: <https://www.techplayon.com/gps-gps-module-used-base-station-applications-choose-gps-module/>.
- [21] "TATTU 4200mAh 3s 35c Lipo Battery," TATTU, [Online]. Available: <https://www.getfpv.com/tattu-4200mah-3s-35c-lipo-battery.html>.
- [22] RC Corner, "GENS ACE 11.1V 50C 3S 4000MAH LIPO BATTERY PACK WITH XT60 PLUG," Gens Ace, [Online]. Available: <https://www.rccorner.ae/gens-ace-11-1v-50c-3s-4000mah-lipo-battery-pack-with-xt60-plug>.
- [23] Solidworks, Dassault Systemes, [Online]. Available: <https://www.solidworks.com/>.
- [24] Wikipedia, "Polylactic acid," [Online]. Available: https://en.wikipedia.org/wiki/Polylactic_acid.
- [25] ArduPilot Dev Team, "Archived:APM2.x Wiring QuickStart," ArduPilot, [Online]. Available: <https://ardupilot.org/copter/docs/connecting-the-apm2.html>.
- [26] ArduPilot, "What is ArduPilot?," ArduPilot, [Online]. Available: <https://ardupilot.org/>.
- [27] ArduPilot Dev Team, "Vehicle Types Supported by ArduPilot," ArduPilot, [Online]. Available: <https://ardupilot.org/copter/docs/common-all-vehicle-types.html>.
- [28] ArduPilot autotest system, "ArduPilot Firmware builds," ArduPilot, [Online]. Available: <https://firmware.ardupilot.org/>.

- [29] M. Oborne, "Mission Planner Home," [Online]. Available: <https://ardupilot.org/planner/>.
- [30] ArduPilot Dev Team, "APM Planner 2 Home," ArduPilot, [Online]. Available: <https://ardupilot.org/planner2/>.
- [31] ArduPilot Dev Team, "MAVProxy," 2019. [Online]. Available: <https://ardupilot.org/mavproxy/#:~:text=MAVProxy%20is%20a%20fully%2Dfunctioning,%E2%80%9Cdeveloper%E2%80%9D%20ground%20station%20software..>
- [32] ArduPilot Dev Team, "Choosing a Ground Station," ArduPilot, [Online]. Available: <https://ardupilot.org/copter/docs/common-choosing-a-ground-station.html>.
- [33] [و. خاکپور, "نحوه نصب لینوکس در ویندوز با ماشین مجازی," 2021 01 05] [Online]. Available: <https://kaliboyz.com/install-linux-on-virtualbox/>.
- [34] S. Loretz, "Ubuntu install of ROS Noetic," Robot Operating System, [Online]. Available: <http://wiki.ros.org/Installation/Ubuntu>.
- [35] ArduPilot Dev Team, "SITL Simulator (Software in the Loop)," ArduPilot, [Online]. Available: <https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html>.
- [36] 3D Robotics, "Setting up a Simulated Vehicle (SITL)," 3D Robotics, 2015-2016. [Online]. Available: https://dronekit-python.readthedocs.io/en/latest/develop/sitl_setup.html.
- [37] S. Balasubramanian, "MavLink Tutorial for Absolute Dummies (Part –I)".
- [38] MAVLink, "MAVLink Messages," [Online]. Available: https://mavlink.io/en/messages/common.html#COMMAND_LONG.
- [39] ArduPilot Dev Team, "MAVLink Basics," 2021. [Online]. Available: <https://ardupilot.org/dev/docs/mavlink-basics.html>.
- [40] MAVLink, "MAVLink Commands (MAV_CMD)," [Online]. Available: https://mavlink.io/en/messages/common.html#MAV_CMD.
- [41] ArduPilot Dev Team, "Configuring a Telemetry Radio using Mission Planner," 2021. [Online]. Available: <https://ardupilot.org/copter/docs/common-configuring-a-telemetry-radio-using-mission-planner.html>.
- [42] ArduPilot Dev Team, "Complete Parameter List," [Online]. Available: <https://ardupilot.org/copter/docs/parameters.html>.
- [43] ArduPilot Dev Team, "Compass Calibration," [Online]. Available: <https://ardupilot.org/copter/docs/common-compass-calibration-in-mission-planner.html>.
- [44] ArduPilot Dev Team, "Accelerometer Calibration," [Online]. Available: <https://ardupilot.org/copter/docs/common-accelerometer-calibration.html#:~:text=Calibration%20steps,->

Warning&text=Under%20Setup%20%7C%20Mandatory%20Hardware%2C%20select,from%20the%20left%2Dside%20menu.&text=Click%20Calibrate%20Accel%20to%20start,each%20ax.

- [45] ArduPilot Dev Team, "Radio Control Calibration," [Online]. Available: <https://ardupilot.org/copter/docs/common-radio-control-calibration.html>.
- [46] ArduPilot Dev Team, "Electronic Speed Controller (ESC) Calibration," [Online]. Available: <https://ardupilot.org/copter/docs/esc-calibration.html>.
- [47] V. Prithiviraj, "Yaw, pitch and roll rotations of a Quadcopter.," 04 2015. [Online]. Available: https://www.researchgate.net/figure/Yaw-pitch-and-roll-rotations-of-a-Quadcopter_fig1_280573614.
- [48] ArduPilot Dev Team, "Flight Modes," [Online]. Available: <https://ardupilot.org/copter/docs/flight-modes.html>.
- [49] ArduPilot Dev Team, "Arming the motors," [Online]. Available: https://ardupilot.org/copter/docs/arming_the_motors.html.
- [50] aLDEID, "Category:Drones/Radio," 18 03 2018. [Online]. Available: <https://www.aldeid.com/wiki/Category:Drones/Radio>.
- [51] GAZEBO, "Beginner: Overview," 2014. [Online]. Available: https://classic.gazebosim.org/tutorials?cat=guided_b&tut=guided_b1.
- [52] B. Aderinola, "What is Gazebo simulation?," 18 09 2019. [Online]. Available: <https://www.theconstructsim.com/ros-5-mins-028-gazebo-simulation/>.
- [53] ن. ضرایبی, "آموزش آشنایی با سیستم عامل ربات ها ROS," [Online]. Available: <https://faradars.org/courses/fvmec9809-introduction-to-robot-operating-system>.
- [54] NIRYO, "8 reasons why you should use ROS for robotics projects," [Online]. Available: <https://service.niryo.com/en/blog/8-reasons-use-ros-robotics-projects>.
- [55] Y. Pyo, H. Cho, L. Jung and D. Lim, ROS Robot Programming, Seoul: ROBOTIS Co.,Ltd, 2017.
- [56] A. Anwar, "Part 3: Create Your First ROS Publisher and Subscriber Nodes," 11 02 2021. [Online]. Available: <https://medium.com/swlh/part-3-create-your-first-ros-publisher-and-subscriber-nodes-2e833dea7598#:~:text=A%20ROS%20Node%20can%20be,is%20published%20to%20the%20topic..>
- [57] ROS, "ROS Tutorials," [Online]. Available: <http://wiki.ros.org/ROS/Tutorials>.
- [58] ProgrammerSought, "ROS file structure," [Online]. Available: <https://www.programmersought.com/article/86944167082/>.

- [59] Raspberry Pi, "Raspberry Pi 3 Model B," [Online]. Available: <https://www.raspberrypi.com/products/raspberry-pi-3-model-b/>.
- [60] Chinmay, "Installing Ubuntu MATE on a Raspberry Pi," 04 April 2020. [Online]. Available: <https://itsfoss.com/ubuntu-mate-raspberry-pi/>.
- [61] ROS, "Ubuntu install of ROS Kinetic," [Online]. Available: <http://wiki.ros.org/Installation/Ubuntu>.
- [62] H. Lee, J. Yoon, M.-S. Jang and K.-J. Park , "A Robot Operating System Framework for Secure UAV Communications," *sensors*, p. 19, 2021.
- [63] V. Ermakov, "mavros," 2018. [Online]. Available: <http://wiki.ros.org/mavros>.
- [64] C. Karney, "geographiclib," [Online]. Available: <https://github.com/sotex/geographiclib>.
- [65] ArduPilot Dev Team, "Communicating with Raspberry Pi via MAVLink," [Online]. Available: <https://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html>.
- [66] ArduPilot Dev Team, "Making a MAVLink WiFi bridge using the Raspberry Pi," [Online]. Available: <https://ardupilot.org/dev/docs/making-a-mavlink-wifi-bridge-using-the-raspberry-pi.html>.
- [67] ArduPilot Dev Team, "Setting up SITL on Linux," [Online]. Available: <https://ardupilot.org/dev/docs/setting-up-sitl-on-linux.html>.
- [68] ArduPilot Dev Team, "Using SITL," [Online]. Available: <https://ardupilot.org/dev/docs/using-sitl-for-ardupilot-testing.html>.
- [69] ArduPilot Dev Team, "Using Gazebo Simulator with SITL," [Online]. Available: <https://ardupilot.org/dev/docs/using-gazebo-simulator-with-sitl.html>.
- [70] V. Ermakov, "mavros_extras," [Online]. Available: http://wiki.ros.org/mavros_extras.
- [71] V. Ermakov, "mavros-installation," [Online]. Available: <https://github.com/mavlink/mavros/tree/master/mavros#installation>.
- [72] ArduPilot Dev Team, "ROS with SITL," [Online]. Available: <https://ardupilot.org/dev/docs/ros-sitl.html>.
- [73] A. Khanal, "Developing in Ardupilot(1)," [Online]. Available: <https://www.prokurainnovations.com/developing-in-ardupilot1/>.
- [74] ROS, "Home," 2022. [Online]. Available: <https://www.ros.org/>.
- [75] L. Meier, "MAVLink Developer Guide," 2009. [Online]. Available: <https://mavlink.io/en/>.

- [76] Y.-M. Kwon, J. Yu and K.-J. Park, "Empirical Analysis of MAVLink Protocol Vulnerability for Attacking Unmanned Aerial Vehicles," *Semantic Scholar*, vol. 6, p. 3, 2018.