



دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

## گزارش کارآموزی

نام و نام خانوادگی کارآموز: رسول بوسعیدی

شماره دانشجویی: ۹۶۲۴۴۱۳

استاد کارآموزی: دکتر سمانه حسینی

سرپرست کارآموزی: جناب آقای سید ظهیر قاضوی

محل کارآموزی: آزمایشگاه پژوهشی تیم‌های خود-هماهنگ-شونده هوشمند

آدرس: اصفهان، دانشگاه صنعتی اصفهان، ساختمان کارآفرینی، طبقه دوم

تلفن: ۰۳۱۳۳۹۱۲۲۱۰

تاریخ پایان: ۱۴۰۰/۰۶/۳۰

تاریخ شروع: ۱۴۰۰/۰۴/۰۱

## فرم شماره ۱

### گزارش خلاصه‌ای از فعالیت‌های هفتگی

شماره دانشجویی: ۹۶۲۴۴۱۳

دانشکده: برق و کامپیوتر

نام و نام خانوادگی کارآموز: رسول بوسعیدی

رشته تحصیلی: مهندسی کامپیوتر

|   |   |
|---|---|
| آشنایی با قسمت‌های مختلف ربات‌های دیفرانسیلی و ربات‌های پروازی  | <b>هفته اول</b><br>از تاریخ: ۱۴۰۰/۰۴/۰۱<br>لغایت<br>تاریخ: ۱۴۰۰/۰۴/۰۸   |
| بررسی فلایت‌کنترل‌های مرسوم در زمینه ربات‌های پروازی  | <b>هفته دوم</b><br>از تاریخ: ۱۴۰۰/۰۴/۰۹<br>لغایت<br>تاریخ: ۱۴۰۰/۰۴/۱۶   |
| کدنویسی قسمت‌های مربوط به دریافت اطلاعات حسگر‌های موجود در ربات پروازی اعم از GPS، IMU، دماسنجه و فشارسنج، قطب‌نما کنترل‌های PID برای کنترل دور موتورهای ربات پروازی و دیگر قسمت‌های مربوط به فلایت‌کنترل | <b>هفته سوم</b><br>از تاریخ: ۱۴۰۰/۰۴/۱۷<br>لغایت<br>تاریخ: ۱۴۰۰/۰۴/۲۴   |
| کدنویسی فلایت‌کنترل برای بورد STM32F103 بعنوان بورد کنترل کننده سنسورها و بورد اجرا کننده فلایت‌کنترل   | <b>هفته چهارم</b><br>از تاریخ: ۱۴۰۰/۰۴/۲۵<br>لغایت<br>تاریخ: ۱۴۰۰/۰۵/۰۱ |

نام و نام خانوادگی سرپرست کارآموزی  
امضاء سرپرست

سید جواد

## فرم شماره ۱ (ادامه)

### گزارش خلاصه‌ای از فعالیت‌های هفتگی

شماره دانشجویی: ۹۶۲۴۴۱۳

دانشکده: برق و کامپیوتر

نام و نام خانوادگی کارآموز: رسول بوسعیدی

رشته تحصیلی: مهندسی کامپیوتر

|   |  |
|---|--|
| <p>ساخت نمونه اولیه ربات پروازی برای تست‌های لازم و نهایی کردن کوادکوپتر برای ساخت ۵ نمونه دیگر از روی آن برای اجرای الگوریتم‌های هوش جمعی</p>  | <b>هفته پنجم</b><br>از تاریخ: ۱۴۰۰/۰۵/۰۲<br>لغایت<br>تاریخ: ۱۴۰۰/۰۵/۰۹ |
| <p>آشنایی با الگوریتم‌های مطرح هوش جمعی و خواندن مقاله‌های مرتبط با هر الگوریتم</p>   | <b>هفته ششم</b><br>از تاریخ: ۱۴۰۰/۰۵/۱۰<br>لغایت<br>تاریخ: ۱۴۰۰/۰۵/۱۷  |
| <p>آشنایی با الگوریتم‌های مطرح هوش جمعی و خواندن مقاله‌های مرتبط با هر الگوریتم و تست کد چندتا از الگوریتم‌های معروف برروی پلتفرم رزبری پای</p> | <b>هفته هفتم</b><br>از تاریخ: ۱۴۰۰/۰۵/۱۸<br>لغایت<br>تاریخ: ۱۴۰۰/۰۵/۲۵ |
| <p>آشنایی با الگوریتم‌های مربوط به پردازش تصویر و دریافت عمق تصاویر براساس یک دوربین (Monodepth and monocular image processing)</p>             | <b>هفته هشتم</b><br>از تاریخ: ۱۴۰۰/۰۵/۲۶<br>لغایت<br>تاریخ: ۱۴۰۰/۰۶/۰۲ |

نام و نام خانوادگی سرپرست کارآموزی  
امضاء سرپرست

سهراب

## فرم شماره ۲

### گزارش سرپرست کارآموزی

نام و نام خانوادگی سرپرست کارآموزی: سید ظهیر قاضوی  
 سمت: مسئول آزمایشگاه  
 نام واحد صنعتی: آزمایشگاه هوش جمعی دانشگاه صنعتی اصفهان  
 شماره دانشجویی: ۹۶۲۴۴۱۳  
 نام و نام خانوادگی کارآموز: رسول بوسعیدی  
 دانشکده: برق و کامپیوتر  
 گرایش تحصیلی: مهندسی کامپیوتر

| ردیف | اظهار نظر سرپرست کارآموز  | ضعیف | متوسط | خوب | عالی |
|------|---|------|-------|-----|------|
| 1    | رعایت نظم و ترتیب و انضباط در محیط کار                            |      |       |     | X    |
| 2    | میزان علاقه و همکاری با دیگران                                    |      |       |     | X    |
| 3    | علاقه به فرآیندی مطالب علمی و فنی                                 |      |       |     | X    |
| 4    | پیگیری وظایف و میزان پشتکار                                       |      |       |     | X    |
| 5    | ارزش پیشنهادات کارآموز در جهت بهبود کار                           |      |       |     | X    |
| 6    | کیفیت گزارش‌های کارآموزی (حداقل فرمهای شماره (1)                  |      |       |     | X    |
| 7    | میزان بهره گیری از امکانات موجود جهت ارتقاء<br>توانایی علمی و فنی |      |       |     | X    |

تعداد روزهای غیبت: ۷ تعداد روزهای مرخصی:

پیشنهادات سرپرست کارآموزی جهت بهبود دوره کارآموزی:

امضاء سرپرست کارآموزی



**فرم شماره ۳**  
**نظرات و پیشنهادات**  
**(در پایان دوره تکمیل شود)**

شماره دانشجویی: ۹۶۲۴۴۱۳

نام و نام خانوادگی کارآموز: رسول بوسعیدی

دانشکده برق و کامپیوتر

رشته تحصیلی: مهندسی کامپیوتر

شرح نظرات و پیشنهادات:

**۱- در مورد دوره کارآموزی و مراحل مختلف آن:**  
 با توجه به چالش‌هایی که در جریان کارآموزی وجود داشت و همچنین مرتبط بودن مطالب آموخته شده در راستای رشته تحصیلی و همچنین گرایش انتخاب شده، دوره بسیار پرباری بود و با خاطر ماهیت فنی/آکادمیکی که وجود داشت، با چالش‌هایی که در فضای صنعتی و همچنین فضای آکادمیک در حوزه مربوطه مطرح هست به صورت جامع آشنا شدم و تجربه‌های زیادی کسب کردم.

**۲- در مورد امور پژوهشی، فنی و تولیدی محل کارآموزی:**  
 همانطور که مطرح شد، به دلیل ماهیت کارآموزی و موضوعی که انتخاب شده بود، با چالش‌های فنی و تولیدی و همچنین چالش‌های پژوهشی و آکادمیکی به صورت موازی آشنا شدم و همچنین به دلیل اینکه موضوع انتخاب شده کاملاً در لبهٔ تکنولوژی قرار داشت و شرکت و آزمایشگاه‌های دیگری در ایران وجود ندارند که انحصاراً در این موضوع و در این فیلد از رباتیک کار کنند، دلیل شد که تجربه بدست‌آمده بسیار گران‌بها و بدیع باشد.

توجه: علاوه بر ارائه فرم نظرات و پیشنهادات در پیوست گزارش تفصیلی، در صورت تمایل یک کپی از این فرم را به دفتر ارتباط با صنعت دانشکده تحویل نمایید.

امضاء کارآموز

## فهرست مطالب

|     |   |
|-----|---|
| ۷.  | چکیده   |
| ۸.  | فصل اول: معرفی محل کارآموزی                                       |
| ۸.  | ۸-۱- معرفی آزمایشگاه پژوهشی تیم‌های خود-همانگ-شونده هوشمند (ICST) |
| ۹.  | فصل دوم: کارهای انجام شده در دوره کارآموزی                        |
| ۹.  | ۹-۱- مقدمه  |
| ۱۰. | ۱۰- برنامه‌نویسی  |
| ۱۰. | ۱۰-۱- مقدمه   |
| ۱۳. | ۱۳- برنامه‌نویسی الگوریتم‌های هوش جمعی                            |
| ۱۴. | ۱۴- مکانیک  |
| ۱۴. | ۱۴-۱- انواع موتورها   |
| ۱۸. | ۱۸- برق و الکترونیک   |
| ۱۸. | ۱۸-۲- پروتکل‌های ارتباطی:   |
| ۱۹. | ۱۹- ۲-۴- واحد اندازه‌گیری اینرسیایی (IMU):                        |
| ۲۰. | ۲۰- ۳-۴- ۲- کنترل کننده PID                                       |
| ۲۲. | ۲۲- ۲-۵- الگوریتم‌ها و هوش مصنوعی                                 |
| ۲۲. | ۲۲-۱-۲-۵- تعریف هوش جمعی (Swarm Intelligence)                     |
| ۲۳. | ۲۳- ۲-۵-۲- مقدمه  |
| ۳۱. | ۳۱- مراجع   |

## چکیده

در آزمایشگاه پژوهشی تیم‌های خود-هماهنگ-شونده هوشمند (ICST)، هدف اصلی ساخت یک دسته از ربات‌های ساخته از ربات‌های برای رسیدن به یک هدف مشترک همکاری می‌کند. برای رسیدن به این هدف قسمت‌های مختلفی مورد نیاز است. ابتدا برای ساده‌تر کردن فرایند، یک نمونه تست ربات ساخته می‌شود و تست‌های مختلف روی آن گرفته شده، سپس با توجه به خروجی تست‌ها، تصمیم به ساخت بقیه ربات‌ها گرفته می‌شود.

برای اینکه بتوانیم فعالیتی مشترک بین ربات‌ها داشته باشیم، هر ربات نیاز دارد:

1. نیاز داریم محیط اطراف خود را حس کند
2. با همسایگان خود تعامل داشته باشد که نیاز مند یک شبکه ویژه است
3. الگوریتمی داشته باشد که بوسیله داده‌های حس شده، بدون برخورد با همسایگانش، به هدف مشخص شده دستیابد

برای رسیدن به این اهداف، چالش‌هایی در قسمت‌های برنامه‌نویسی، انتخاب قطعات الکتریکی، ایرودینامیک و شاسی ربات و همچنین چالش‌های تئوری زیادی هستیم که در ادامه مفصل‌تر این چالش‌ها را توضیح خواهیم داد.

## فصل اول

### معرفی محل کارآموزی

#### ۱-۱- معرفی آزمایشگاه پژوهشی تیم‌های خود-همانگ-شونده هوشمند (ICST)

**زمینه‌های کاری:** سیستم‌های چند عاملی، هوش جمعی، یادگیری تقویتی، شبکه‌های سنسوری

آزمایشگاه پژوهشی تیم‌های خود-همانگ-شونده هوشمند در سال ۱۳۹۹ تحت سرپرستی دکتر سمانه حسینی و مسئولیت آزمایشگاه توسط آقای سید ظهیر قاضوی، در ساختمان مرکز فناوری، شروع به فعالیت کرده است. در حال حاضر ۱۰ نفر از دانشجویان دانشگاه صنعتی از ورودی‌ها و گرایش‌های مختلف از رشته‌های برق و کامپیوتر در حال فعالیت در آزمایشگاه می‌باشند.

تحقیقات آزمایشگاه ICST حول محور سیستم‌های چند عاملی خود همانگشونده است. در این آزمایشگاه به دنبال پاسخ‌دهی به سوالاتی نظری اینکه چطور عامل‌ها می‌توانند با همکاری و همانگی یکدیگر رفتار‌های جمعی هوشمندی از خود نشان دهند هستیم. در این تحقیقات، عامل‌ها می‌توانند از پرندگاه‌های بدون سرنشین (UAV) تا ربات‌های متحرک انسان‌نمای ساده، متغیر باشند و الگوریتم‌های مورد استفاده برای حل مسائل می‌توانند گسترده‌ای از الگوریتم‌های هوش جمعی تا الگوریتم‌های یادگیری تقویتی عمیق داشته باشند.

آزمایشگاه پژوهشی تیم‌های خود-همانگ-شونده هوشمند واقع در مرکز فناوری دانشگاه صنعتی اصفهان در حوزه رباتیک، فعالیتی ترکیبی و بین رشته‌ای و تلفیقی از رشته‌های هوش مصنوعی، برق، کامپیوتر و مکانیک دارد. فضای آکادمیک در این آزمایشگاه بر روی مسائل به روز نظری Multiagent Systems، Motion Planning Algorithms، Disturbuted Algorithms و الگوریتم‌های هوش جمعی تمرکز دارد.

یکی از اهداف اصلی آزمایشگاه که در برگیرنده بقیه شاخه‌های ذکر شده نیز می‌باشد، پیاده‌سازی الگوریتم‌های Motion Planning بر روی ربات‌های پروازی برای ایجاد اشکال نمایشی در فضای <sup>۳</sup> بعدی می‌باشد که کارهای پژوهشی دیگری در این بین تعریف می‌گردد.

## فصل دوم

### کارهای انجام شده در دوره کارآموزی

#### ۱-۲ - مقدمه

برنامه آزمایشگاه در دوره کارآموزی به صلاح دید مسئول‌های آن، آشنایی با ربات‌های دیفرانسیلی و پروازی و همچنین آشنایی با الگوریتم‌های مطرح در زمینه هوش جمعی بوده است.

در مرحله بعدی، ساخت ۶ ربات پروازی برای تست یکسری از الگوریتم‌های مطالعه شده و همچنین پیاده‌سازی الگوریتم‌های جدید برای گزارشات تجربی هدف قرار گرفت. در این میان با پلتفرم‌ها و روش‌های گوناگون آشنا شده و تلاش شد در اکثر قسمت‌ها دانشجویان قسمت‌های مختلف را فرا گرفته و خودشان پیاده‌سازی کنند و در مرحله آخر برای تست‌های آخر و محصول نهایی، از وسایل حرفه‌ای‌تر استفاده شده که این روند باعث شد دانش‌های گوناگون به صورت خیلی عمیق فراگرفته شود.

برای محقق آمدن این اهداف، هر هفته به صورت کاملاً منظم، حداقل یک جلسه (از اواسط دوره کارآموزی بدليل حجم بالای کارها جلسات به صورت هر هفته دو جلسه تغییر پیدا کرد) برگزار شد و افراد در مورد قسمت‌های مختلف گزارشی آماده کرده و ارائه می‌دادند و در جلسات حضوری تلاش می‌شد با توجه به تجهیزات فراهم آورده شده، ربات پروازی اولیه (به صورت MVP) ساخته شود تا تست‌های مورد نیاز روی آن انجام بگیرد.

در ادامه مطلب بالا به گزارش فعالیت‌های انجام‌شده و بخش‌بندی قسمت‌های مختلف می‌پردازیم:

## ۲-۲ - برنامه نویسی

۱-۲-۲ - مقدمه

بدلیل تسكیهای فراوانی که وجود داشت، با توجه به تقسیم کارهای صورت گرفته در قسمت‌های مختلف، هرکسی ممکن بود با چالش‌های متفاوتی در قسمت‌های کدنویسی مواجه شود که در ادامه، چندی از چالش‌هایی که شخص من با آن‌ها درگیر بودم را گزارش می‌کنم:

### ۱-۲-۲ : خواندن اطلاعات از سنسورها

از ابتدای کار و با توجه به گزارش هفتگی آورده شده در فرم‌های مذکور، نیاز بود تا هرکدام از سنسورهای مورد نیاز، راهاندازی شود. یک قسمت از این راهاندازی، خواندن اطلاعات سنسورها و همچنین ایجاد خروجی‌های مناسب با توجه به این اطلاعات است که با کمی بهینه‌سازی در مجموع این اعمال شرایطی رو فراهم می‌آورد که ربات حسی نسبت به شرایط اطراف خود داشته باشد و بتواند موقعیت خود را در هوا حفظ کند و یا به سمت هدف تعریف شده حرکت کند.

مجموعه این کدنویسی‌ها و بهینه‌سازی‌ها، مفهومی تحت عنوان **فلایت‌کنترل** ایجاد می‌کند که در واقع هسته مرکزی ربات برای حفظ تعادل‌هایش می‌باشد. در بحث ربات‌های پروازی، یکی از مهمترین قسمت‌ها بخش **فلایت‌کنترل** آن می‌باشد. این قسمت کنترل‌کننده اصلی تعادل ربات در آسمان و عملیات محاسبه نیرو و مناسب برای موتورها با توجه به اطلاعات دریافتی از سنسورها و همچنین شرایط ربات را انجام می‌دهد.

از آنجایی که این قطعه و الگوریتم خیلی مهم و حساس است، نیاز بود تحقیق مفصلی شکل گیرد و سپس بهترین روش‌ها و قطعات فراهم آید که در ادامه به معرفی چند قطعه و روش معروف در این زمینه می‌پردازیم.

سنسورهای استفاده شده عبارت‌اند از:

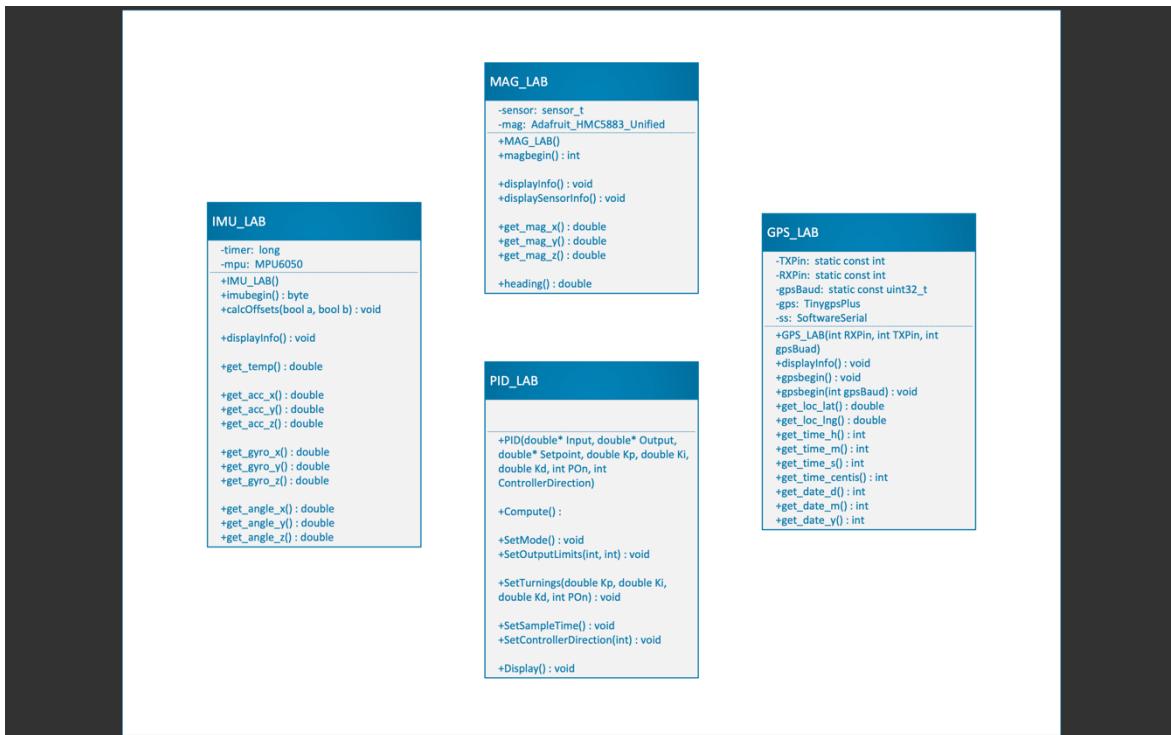
- GPS •
- IMU •
- ESC •
- مغناطیس •
- فشارسنج •
- دوربین •
- فاصله‌سنج •

توضیحات مربوط به هر یک از سنسورهای فوق در قسمت‌های بعدی (۴-۲) توضیح داده خواهد شد.

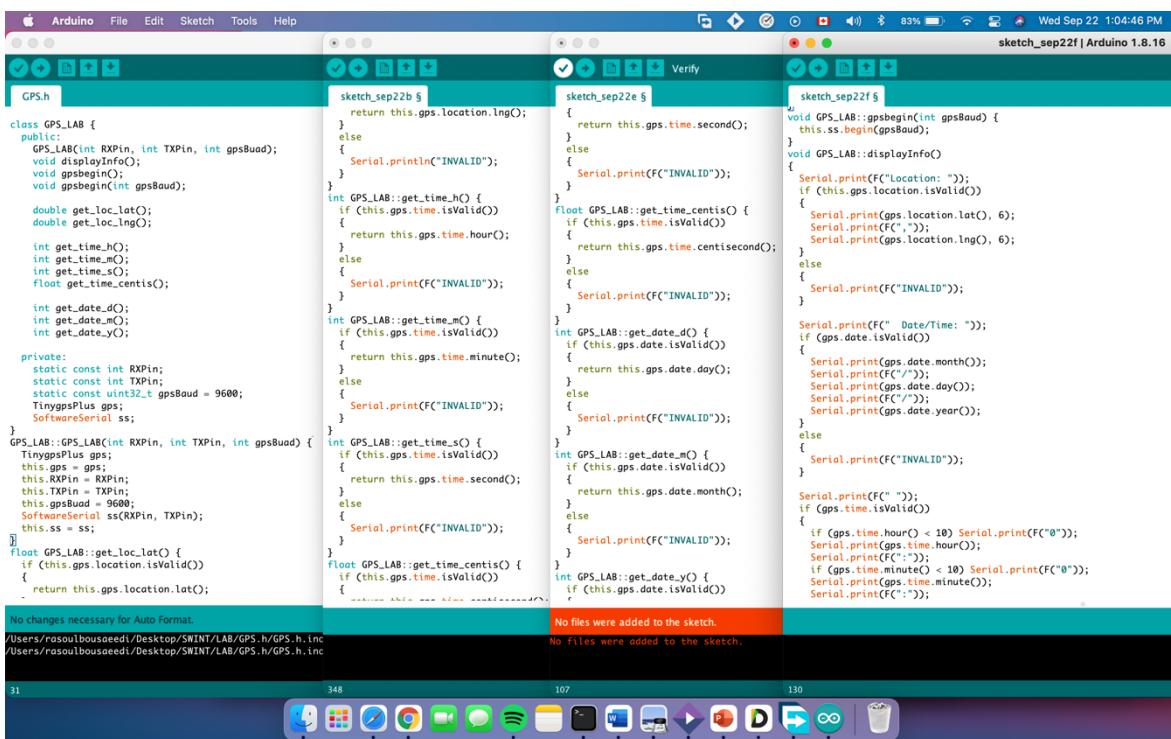
در مرحله اول برای هرکدام از این سنسورها بدلیل اینکه توابع آماده و حرفه‌ای زیادی موجود می‌باشد، از IDE معروف آردوینو برای کدنویسی استفاده شد. اگرچه در ادامه نیز کدنویسی داخل Arduino IDE ادامه پیدا کرد اما بوردهای خانواده ARM تغییر پیدا کردند که کمی کدنویسی‌ها و توابع متفاوت شد. همچنین تلاش شد تمامی کدها به صورت شیگرایی نوشته شود چون در ادامه کار و با توجه به جمعی بودن کار، توسعه‌پذیری بهبود پیدا می‌کرد.

کدهای مربوط به هر سنسور در عکس‌های زیر قابل مشاهده است:

همچنین لازم به ذکر است تمامی کدها در گیت آزمایشگاه (ساخته شده در گیت دانشگاه) قرار داده شده است.



شکل (۱-۲) دیاگرام مربوط به تمامی کلاس‌های نوشته شده



GPS\_LAB(۲-۲)

The screenshot shows the Arduino IDE interface with two open sketches:

- IMU\_LAB.h**: Header file for the IMU library. It includes the `MPU6050` library and defines the `IMU_LAB` class. The class contains methods for initializing the MPU6050, calculating offsets, displaying info, getting temperature, and reading various sensor values.
- sketch\_sep22g**: The main sketch. It includes the `IMU_LAB` header and initializes the IMU. It then enters a loop where it prints the temperature and various sensor readings (gyro and acc X, Y, Z) to the serial monitor.

شکل (۳-۲) IMU\_LAB (۳-۲)

The screenshot shows the Arduino IDE interface with one open sketch:

- MAG\_LAB.h**: Header file for the MAG library. It includes the `Adafruit_Sensor.h`, `Adafruit_HMC5883_U.h`, `SoftwareSerial.h`, and `Wire.h`. The `MAG_LAB` class is defined, which interacts with an `HMC5883` magnetometer.
- sketch\_sep22i**: The main sketch. It initializes the magnetometer, handles sensor events, calculates heading based on magnetic field components, and displays the heading and sensor info to the serial monitor.

شکل (۴-۲) MAG\_LAB (۴-۲)

## ۲-۲-۲- برنامه‌نویسی الگوریتم‌های هوش‌جمعی

در هنگام تست کردن الگوریتم‌های مطرح هوش‌جمعی (عنوانین الگوریتم‌ها در قسمت‌های بعدی توضیح داده خواهد شد) نیاز بود تا خیلی از الگوریتم‌ها متناسب با پلتفرم ما ساخته شوند که به همین دلیل، یکی از تسک‌های من پیاده‌سازی و تست این الگوریتم‌ها در بستر سخت‌افزارهای مورد استفاده ما (عمدها رزبری پای) بود که در بسیاری از مواقع کار طاقت‌فرسایی شد اما خروجی مناسبی گرفته شد و کدهای مربوطه در گیت‌هاب آزمایشگاه منتشر شده است.

شبکه‌کی که در شکل (۵-۲) مشاهده می‌کنید، در زبان پایتون پیاده‌سازی کردم که در این [لينک ۱](#) می‌توانید آن را در گوگل کولب مشاهده کنید.

همچنین کد مربوط به الگوریتم CADRL [3] را آموزش داده و به صورت آموزش‌داده شده و آمده‌ی استفاده و برای انتقال به رزبری، داخل گیتلاب و گیت‌هاب گذاشتم که می‌توانید کدهای مربوط به آن را در [لينک ۱](#) و [لينک ۲](#) مشاهده کنید.

همینطور کد مربوط به الگوریتم CADRL در رزبری‌پایی تست شد که پرسه بسیار زمان‌بری بود. دلیل سختی این فرایند، تفاوت پکیج‌های موجود و عدم تطابق با مدل آموزش دیده شده و همچنین سیستم‌عامل مخصوص آن که با توجه به معماری متفاوت CPU در رزبری، پکیج‌های موجود تفاوت‌های فراوانی داشتند.

متاسفانه این تست به دلیل بسیار سنگین بودن مدل تقویتی که در آن استفاده شده بود، ناموفق ماند و ترجیح داده شد از روش‌های کلاسیک مانند ORCA [4] و FMP [2] استفاده شود.

---

### Algorithm 1 FMP Algorithm

---

**Input:**

- $n$  = number of agents (= number of targets)
- $\mathcal{I}$  = Initial position of agents  $\mathcal{F}$  = final position of agents
- $d^*$  = required minimum separation distance between agents
- $v_{\max}$

**Output:**

- $\forall_{i=0}^n p_i(t), p_i(t) : (p_{ix}(t), p_{iy}(t), p_{iz}(t))$  = position of agent  $i$  at time  $t$
- $\forall_{i=0}^n v_i(t), v_i(t) : (v_{ix}(t), v_{iy}(t), v_{iz}(t))$  = velocity of agent  $i$  at time  $t$   
( $0 \leq t$  increases in  $\Delta t$  time intervals)

**begin**

```

 $v_i(0) = (0, 0, 0)$  // initialize agent velocity
 $\mathcal{A}$  = agent-target assignment // e.g., Hungarian
 $\mathcal{T}_i = \mathcal{F}_{\mathcal{A}_i}$  //initialize target position
Calculate  $d$  and  $r$  using (36) and (15)
for ( $t = 0; t < \text{MaxSimTime}; t+ = \Delta t$ )
  for  $0 \leq i < n$  do in parallel
    move_this_agent_to_new_position (i)
  end for
  MaxDis = agents_maximum_distanse_from_  $\mathcal{F}$ ()
  if ( $MaxDis < EndMaxDis$ ) do
    break
  end if
end for
end

```

---

**end**
**move\_this\_agent\_to\_new\_position (i) {**

```

 $f_i^R = \text{repulsive\_force} (i), f_i^Y = \text{navigational\_feedback} (i)$ 
 $\bar{u}_i = f_i^R + f_i^Y$ 
 $v_i(t) = v_i(t - \Delta t) + \bar{u}_i \Delta t$ 
cap_velocity (i) //A hard constraint on velocity
 $p_i(t) = p_i(t - \Delta t) + v_i(t) \Delta t$ 

```

---

**repulsive\_force (i) {**

```

 $f = 0$ 
for  $0 \leq j < n$  do
  dist =  $\|p_j - p_i\|$ 
  if ( $dist < r$  and  $j \neq i$ ) then
    ForceComponent =  $-\rho \times (dist - r)^2$ 
     $f += \text{ForceComponent} \times (p_j - p_i) / \|p_j - p_i\|$ 
  end if
end for
return f

```

---

**navigational\_feedback (i) {**

```

 $f = -c_1(p_i - \mathcal{T}_i) - c_2(v_i)$ 
return f

```

---

```

cap_velocity(i) {
   $v_i = \min(v_i, \|v_i\| \cdot r / \|v_i\|) \wedge v_{\max}$ 
}

```

---

شكل (۵-۲) شبکه‌کی الگوریتم FMP

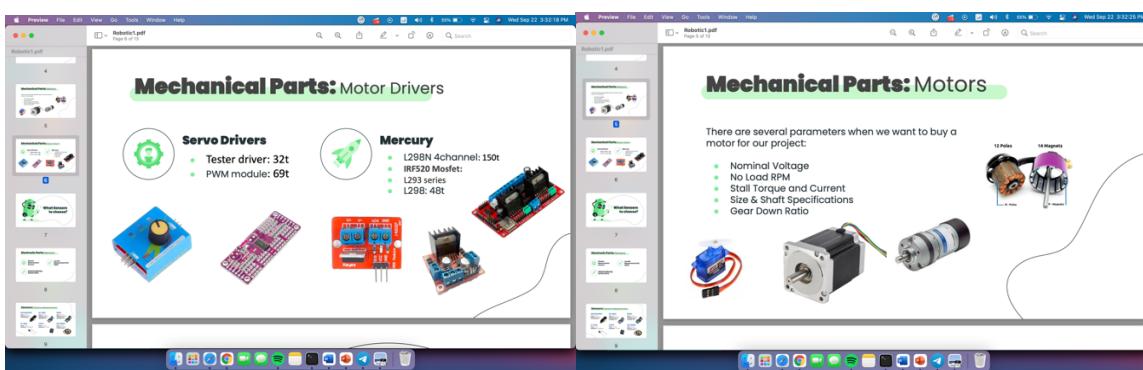
## ۱-۲-۳ - انواع موتورها

- **استپر موتور:** استپر موتور یک موتور DC است که در گام‌هایی گستره حرکت می‌کند. این موتورها دارای چندین سیمپیج هستند که در گروه‌هایی با عنوان "فاز" دسته‌بندی شده‌اند. با اعمال ولتاژ در یک توالی برای فازها، موتور در هر تغییر، یک گام به حرکت در خواهد آمد. با کنترل گام‌ها می‌توان به موقعیت بسیار دقیق و یا کنترل سرعت دست یافت، به همین دلیل استپر موتور برای کاربردهای دقیق کنترلی استفاده می‌شود.

- **سرورو موتور:** سروو موتور نوعی موتور است که می‌تواند با دقت زیادی دوران کند. به طور معمول این نوع موتور از یک مدار کنترل تشکیل شده است که بازخوردی را در مورد موقعیت فعلی شافت موتور ارائه می‌دهد، این بازخورد به موتورهای سروو اجازه می‌دهد تا با دقت زیادی بچرخد. اگر بخواهیم یک جسم را در چند زاویه یا فاصله خاص بچرخانیم، از سروو موتور استفاده می‌کنیم.

- **موتور براشلس DC:** موتورهای براشلس DC یا BLDC که به عنوان موتور الکترونیکی کموتاتور، موتورهای سنکرون AC یا سرووموتور DC نیز شناخته می‌شوند، به دلیل "کاربرد مؤثر، عمر طولانی، تحويل گشتاور صاف، و بهره‌برداری با سرعت بالایی که دارند به طور فزاینده‌ای جایگزین موتور DC براش شده‌اند.

- **موتور براش DC:** طیف گسترده‌ای از موتورها در این دسته قرار دارند. مدل‌های چینی و غیرچینی آن‌ها در بازار فراوان است و عمدهاً قیمت بسیار ارزان و به نسبت قیمت، کارایی خوبی دارند و بسیار در دسترس هستند.



شکل (۶-۲) عکس‌ها مریبوط به یکی از ارائه‌های آماده شده در مدت کارآموزی است. در این عکس‌ها می‌توان اطلاعات جامعی درمورد نوع موتور و برایور موتورهای مناسب و همچنین معیارهای مناسب در انتخاب این قطعات را مشاهده کرد

در عکس‌های فراهم آورده شده زیر، الگوریتم‌های مورد استفاده برای حرکت ربات قابل مشاهده است:

Motor Mixing Algorithm

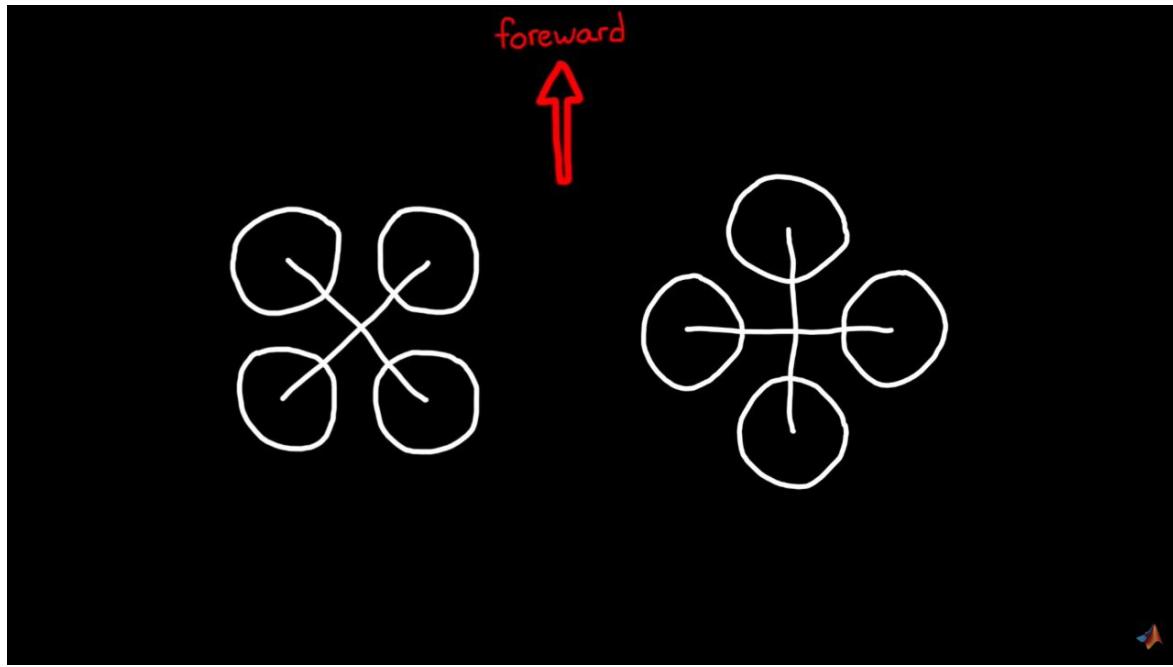
$$\text{Motor}_{\text{front right}} = \text{Thrust}_{\text{cmd}} + \text{Yaw}_{\text{cmd}} + \text{Pitch}_{\text{cmd}} + \text{Roll}_{\text{cmd}}$$

$$\text{Motor}_{\text{front left}} = \text{Thrust}_{\text{cmd}} - \text{Yaw}_{\text{cmd}} + \text{Pitch}_{\text{cmd}} - \text{Roll}_{\text{cmd}}$$

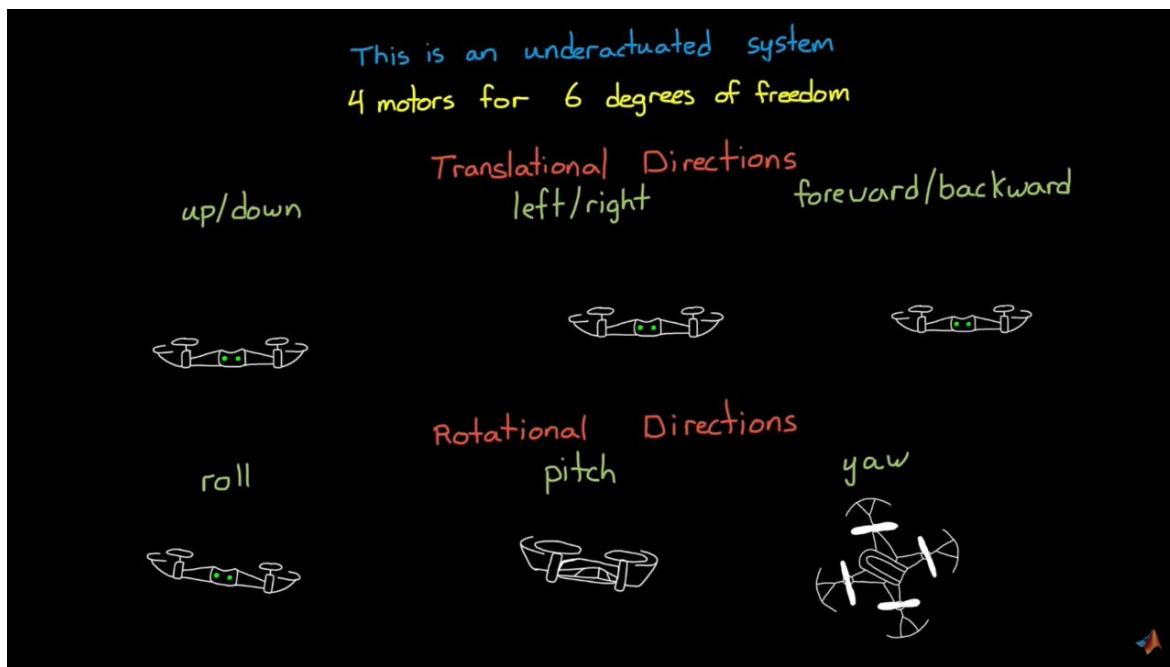
$$\text{Motor}_{\text{back right}} = \text{Thrust}_{\text{cmd}} - \text{Yaw}_{\text{cmd}} - \text{Pitch}_{\text{cmd}} + \text{Roll}_{\text{cmd}}$$

$$\text{Motor}_{\text{back left}} = \text{Thrust}_{\text{cmd}} + \text{Yaw}_{\text{cmd}} - \text{Pitch}_{\text{cmd}} - \text{Roll}_{\text{cmd}}$$

شکل (۱-۲) فرمول‌های مورد نیاز حرکت ربات پروازی در محورهای مختلف

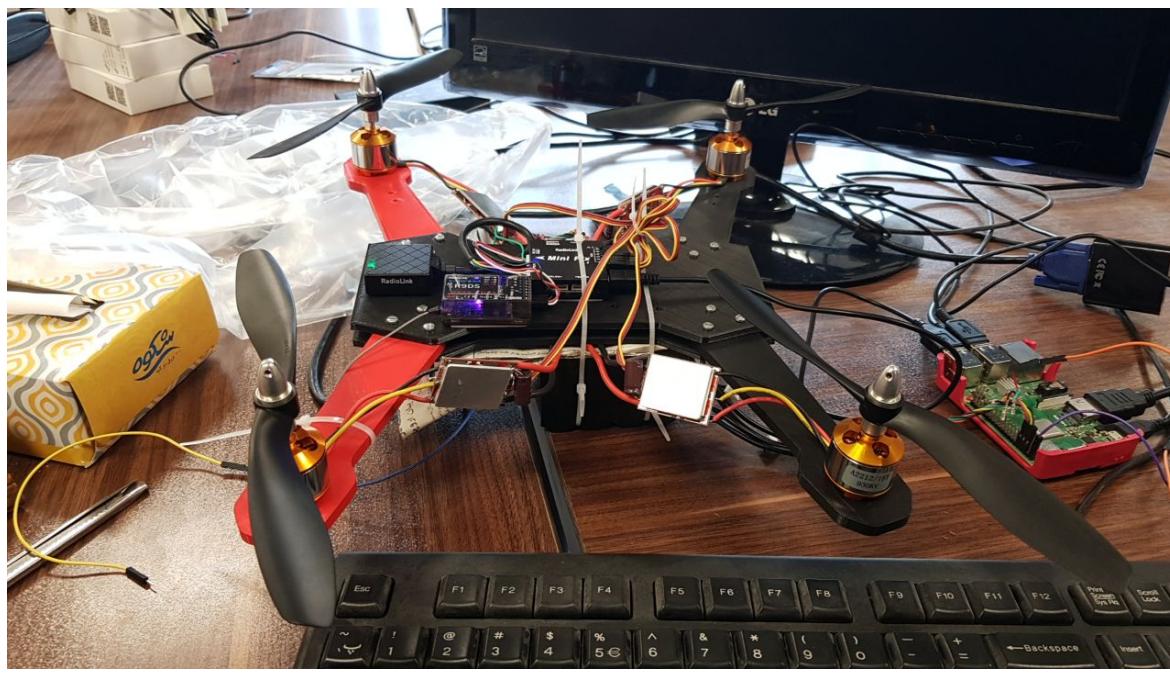


شکل (۱-۲) دو تا از الگوریتم‌های مرسوم برای حرکت رو به جلوی ربات‌های پروازی

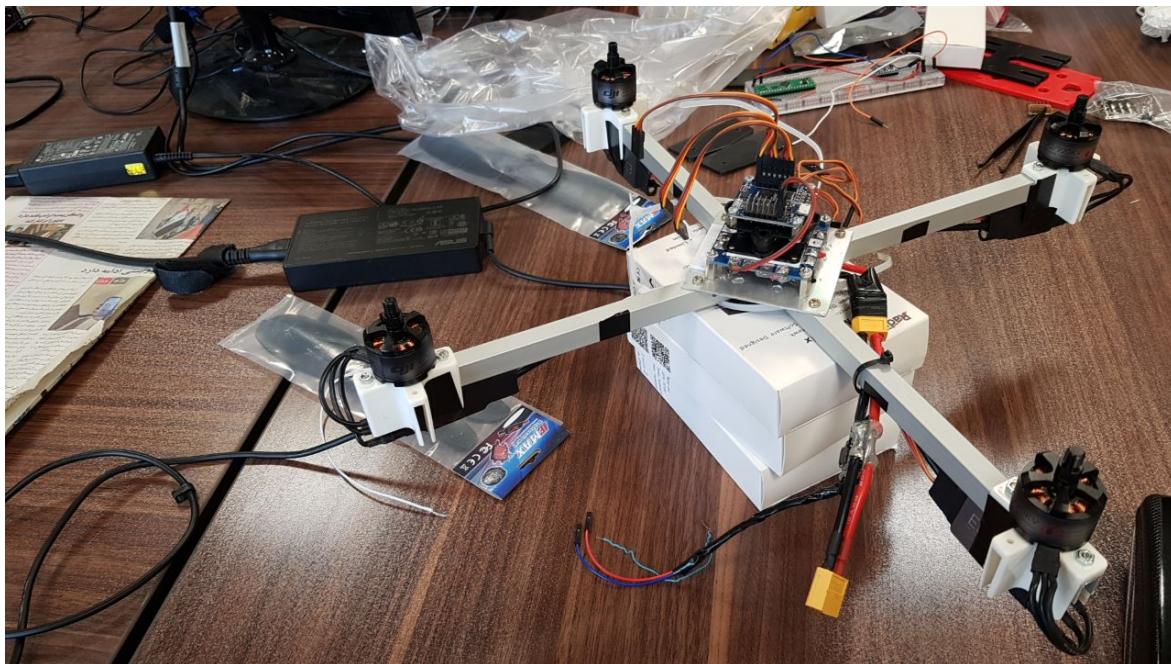


شکل (۹-۲) انواع حرکات مختلف ربات در محورهای متفاوت

در ادامه تعدادی عکس، از ربات‌های ساخته شده و شاسی ربات‌های پروازی، فضای کار آزمایشگاه و همچنین قطعات استفاده شده قابل مشاهده است:



شکل (۱۰-۲) نمونه اولیه ربات پروازی ساخته شده که تست‌هایی روی آن گرفته شده است



شکل (۱۱-۲) یکی از شاسی‌های قدیمی یک ربات پروازی



شکل (۱۲-۲) زمین داخل محل کارآموزی که تست‌های لازم برای الگوریتم‌های Indoor توسط ربات‌های پروازی بسیار کوچکی (CrazyFlie) که چهارتا از آن‌ها در تصویر به سختی قابل مشاهده است، در این جایگاه انجام می‌شود

## ۲-۴- برق و الکترونیک

در هفته‌های اول تلاش شد تا دانش‌های موردنیاز اولیه فرآگرفته شود. در این زمان، اطلاعات جامعی نسبت به روش کار حسگر‌های مورد استفاده در ربات‌های پروازی و دیفرانسیلی فراهم آورده شد و همچنین در کنار آن، نوع قطعات مطرح و در دسترس بازار تحقیق شده و با توجه به قیمت آن‌ها گزارش شد تا بازار تجهیزات نیز شناخته شود. برای این امر به صورت حضوری در خیابان طلاقانی اصفهان و همچنین سایت‌های معروف توزیع این تجهیزات گزارشی تهیه و فراهم شد که در ادامه تصاویر و مطالبی از این تحقیقات ارائه شده است:



شکل (۱۳-۲) قسمتی از ارائه آماده شده در زمان کارآموزی در مورد نوع سنسورهای مختلف. مشاهده می‌شود که قیمت و نوع سنسورها و کمی از اطلاعات فنی سنسورها قابل مشاهده است

### ۱-۴-۲ پروتکل‌های ارتباطی:

در زیر جدولی فراهم آورده شده است که تفاوت ۳ پروتکل ارتباطی معروف را می‌توانید مشاهده کنید:

جدول (۱-۲)

| UART         | I2C          | SPI           |
|--------------|--------------|---------------|
| 1 to 1       | Simplex      | Duplex        |
| Asynchronous | Synchronous  | Synchronous   |
| 2 wires      | 2 wires      | 4 wires       |
| 20Kbps       | 1Mbps        | 25Mbps        |
| Medium power | Medium power | Lower power   |
| 15m distance | 1m distance  | 20cm distance |

## ۲-۴-۲- واحد اندازه‌گیری اینرسیایی (IMU):

یک سیستم ناوبری اینرسیایی (INS) خروجی را از واحد اندازه‌گیری اینرسیایی (IMU) گرفته و با داده‌های شتاب و چرخش، با اطلاع از مقادیر اولیه مکان، سرعت و وضعیت ترکیب می‌کند. سپس خروجی ناوبری را با همه اندازه‌گیری‌های جدید (Mechanization) تحويل می‌دهد.

- **IMU:** واحد اندازه‌گیری اینرسیایی
- **INS:** سیستم ناوبری اینرسیایی
- **Mechanization:** الگوریتم تولید مکان، سرعت و وضعیت از شتاب و نرخ وضعیت در طول زمان.

یک IMU بطور معمول از اجزاء زیر تشکیل شده است:

- سه شتابسنج
- سه جایروسکوپ
- سخت افزار/نرم افزار پردازش سیگنال دیجیتال
- نرم افزار/سخت افزار ارتباطی
- محفظه

سه شتابسنج عمود بر هم قرار گرفته‌اند تا بتوانند شتاب را به طور مستقل در سه محور X، Y و Z اندازه بگیرند. سه جایروسکوپ نیز عمود بر هم قرار گرفته‌اند تا سرعت زاویه‌ای هر سه محور را اندازه بگیرند. سیگنال‌های خروجی از شتابسنج‌ها و جایروسکوپ‌ها در بخش پردازش سیگنال با سرعتی بسیار زیاد پردازش می‌شوند. سپس اندازه‌گیری‌ها جمع شده و یک شتاب و چرخش کلی برای دوره نمونه IMU داده می‌شود. مثلاً در یک IMU ۲۰۰ هرتز، دوره نمونه نمایانگر حرکت کلی IMU در یک بازه ۵ میلی‌ثانیه است.

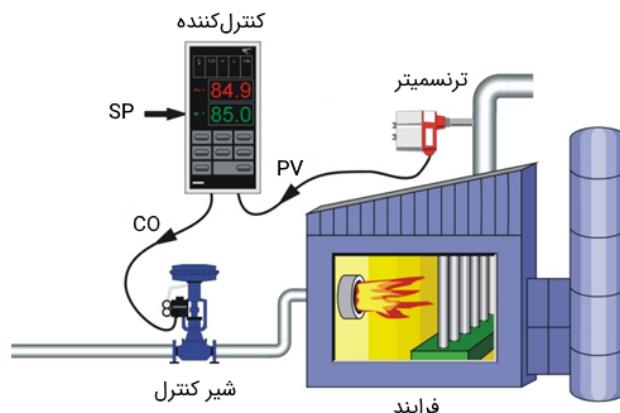
داده‌های حرکت IMU به INS ارسال می‌شود. این اندازه‌گیری‌ها به عنوان ورودی فیلتر INS، استفاده می‌شود. نتیجه نهایی ترکیب داده‌های IMU با داده‌های GNSS عبارت است از مکان، سرعت و وضعیت. اندازه‌گیری‌های IMU بعنوان بخشی از فیلتر INS نسبت به زمان، انتگرال‌گیری شده و در نتیجه هر خطای در اندازه‌گیری در گذر زمان بزرگ می‌شود. برای کاهش اثر خطای اندازه‌گیری، می‌بایست این خطای درک شده، تخمین زده شده و سپس تصحیح شوند.

### ۳-۴-۲- کنترل کننده PID

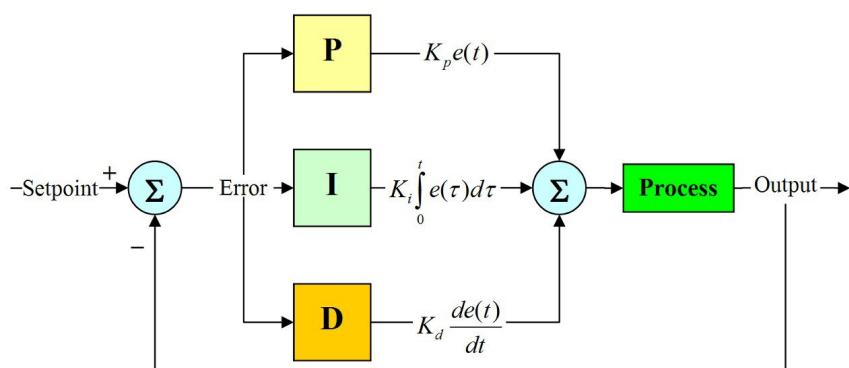
کنترل کننده PID یا تناوبی-انتگرالی-مشتقی، یک الگوریتم و روش کنترل حلقه بسته با بهره‌گیری از مفهوم فیدبک است که در بسیاری از فرایندهای صنعتی برای کنترل سرعت موتورهای DC، کنترل فشار، کنترل دما و ... به کار می‌رود.

کنترل کننده تناوبی-انتگرالی-مشتقی (Proportional–Integral–Derivative Controller) یک نقطه تنظیم (Set Point) یا SP دارد که اپراتور می‌تواند آن را روی به طور مثال، دمای مطلوب تنظیم کند. خروجی کنترل کننده (Controller's Output) یا CO موقعیت شیر کنترل را تنظیم می‌کند. و در نهایت، دستگاه اندازه‌گیری دما، که متغیر فرایند (Process Variable) یا PV نامیده می‌شود، فیدبک لازم را به کنترل کننده می‌دهد. متغیر فرایند و خروجی کنترل کننده معمولاً با سیگنال‌های ۴ تا ۲۰ میلی‌آمپری یا فرمان‌های دیجیتال روی یک فیلدباس منتقل می‌شوند.

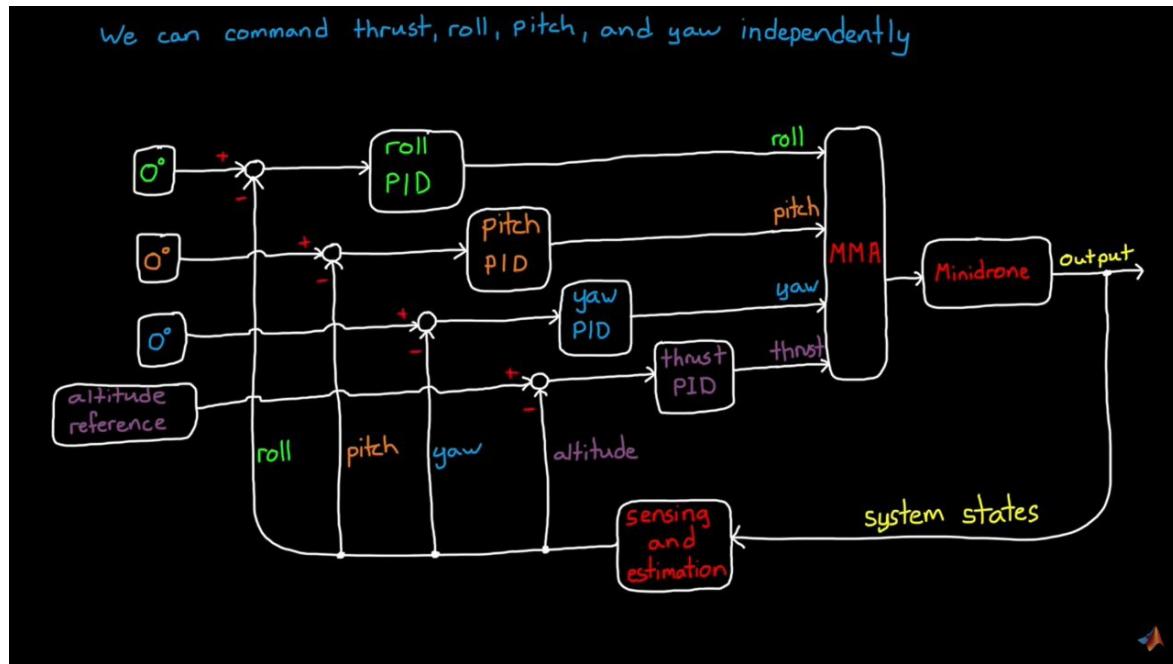
وقتی همه اجزا در جای خود قرار گیرند، کنترل کننده PID متغیر فرایند را با نقطه تنظیم مقایسه کرده و اختلاف بین دو سیگنال را محاسبه می‌کند که خطا (Error) یا E نامیده می‌شود. سپس، بر اساس خطا و ثابت‌های تنظیم کنترل کننده PID، کنترل کننده یک خروجی مناسب را محاسبه کرده و برای نگه داشتن دما در نقطه تنظیم، شیر را می‌چرخاند. اگر لازم باشد دما به بالاتر از مقدار نقطه تنظیم تغییر کند، کنترل کننده موقعیت شیر را در جهت عکس تغییر می‌دهد و بالعکس.



شکل (۱۴-۲) یک کنترل کننده PID که کنترل خودکار را انجام می‌هد



شکل (۱۵-۲) در این تصویر معنی PID بسیار روشن نشان داده شده است و قسمت‌های ریاضی آن قابل مشاهده است



شکل (۲-۱) این تصویر کل فرایندی که ما پیش رو داریم را به طور کامل نمایان می کند.

```

Arduino File Edit Sketch Tools Help
Control sketch_sep22j § sketch_sep22j | Arduino 1.
Wed Sep 22 1:09:17 PM
#include <Adafruit_Sensor.h>
#include <Adafruit_HMC5883_U.h>
#include "Wire.h"
#include <MPU6050.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

MPU6050 mpu(Wire);
long timer = 0;

int RX = 11;
int TX = 10;
TinyGPSPlus gps;
SoftwareSerial gpsPort(RX, TX);
double gps_x0 = 0;
double gps_y0 = 0;
double gps_x = 0;
double gps_y = 0;
double gps_thrust0 = 0;
double gps_thrust = 0;

Adafruit_HMC5883_Unified mag = Adafruit_HMC5883_Unified(12345);
double heading = 0;
double declinationAngle = 0.22;
double headingDegrees = 0;

// motor pins definition
#define M1 0 //front left
#define M2 1 //front right
#define M3 2 //back left
#define M4 3

// pid variables for roll pitch yaw and thrust
// roll variables
double roll_setpoint, roll_input, roll_output;
double roll_Kp = 2, roll_Ki = 5, roll_Kd = 1;
PID roll_pid(&roll_input, &roll_output, &roll_setpoint, roll_Kp, roll_Ki, roll_Kd, DIRECT);

// pitch variables
double pitch_setpoint, pitch_input, pitch_output;
double pitch_Kp = 2, pitch_Ki = 5, pitch_Kd = 1;
PID pitch_pid(&pitch_input, &pitch_output, &pitch_setpoint, pitch_Kp, pitch_Ki, pitch_Kd, DIRECT);

// yaw variables
double yaw_setpoint, yaw_input, yaw_output;
double yaw_Kp = 2, yaw_Ki = 5, yaw_Kd = 1;
PID yaw_pid(&yaw_input, &yaw_output, &yaw_setpoint, yaw_Kp, yaw_Ki, yaw_Kd, DIRECT);

// thrust variables
double thrust_setpoint, thrust_input, thrust_output;
double thrust_Kp = 2, thrust_Ki = 5, thrust_Kd = 1;
PID thrust_pid(&thrust_input, &thrust_output, &thrust_setpoint, thrust_Kp, thrust_Ki, thrust_Kd, DIRECT);

// motor pins
double m1 = 0;
double m2 = 0;
double m3 = 0;
double m4 = 0;

void setup() {
    // put your setup code here, to run once:
    // set up pid variables here setpoint, input
    ///////////////////////////////////////////////////////////////////
    Serial.begin(9600);
    gpsPort.begin(9600);
    Wire.begin();
    byte status = mpu.begin();
    Serial.print("MPU6050 status: ");
    Serial.println(status);
    while (status != 0) { // stop everything if could not connect to MPU6050
        Serial.println("Calculating offsets, do not move MPU6050");
        delay(1000);
        mpu.calcOffsets(true, true); // gyro and accelero
        Serial.println("Done!\n");
        // if connection successful break
    }
}

Name for new file:

```

شکل (۲-۲) قسمتی از کد مربوط به PID و ترکیب آن با خواندن اطلاعات از سنسورها و همچنین خروجی های موتورها

## ۲-۵-۱- الگوریتم ها و هوش مصنوعی

### ۲-۵-۲- تعریف هوش جمیعی (Swarm Intelligence)

رباتیک یک شاخه نسبتاً جدید و رو به رشد از علم است که بخصوص در صنعت و برای اتوماتیک کردن کارها استفاده های فراوانی دارد.

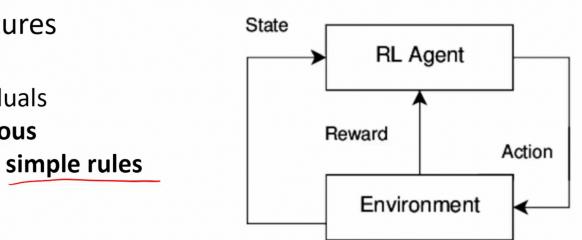
از سال های بسیار دور، ارتباط بین ربات ها و ایجاد هوشمندی در آن ها چالش هایی برای محققان ایجاد کرد که با مطالعه این شاخه از علم و این چالش ها، باعث شد پیشرفت هایی در زمینه این هوشمندی ها ایجاد شود و پس از گذشت سال های زیاد از این تحقیق ها، تفکیک پذیری های متفاوتی برای این هوشمندی ایجاد شد.

یکی از شاخه های این هوشمندی، به بررسی ارتباط ربات ها با هم می پردازد و تلاش می شود بین ربات های ارتباط هایی شکل بگیرد که به یک همکاری مشترک بین ربات ها بیانجامد که در آینده هی پیشرو کمک می شود این ارتباط و همکاری مشترک، به یک هوشمندی مشابه انسان ها تبدیل شود.

در این الگوریتم ها نقاط مشترکی وجود دارند. یکی از این اشتارکات، آن است که در این همکاری جمعی، معمولاً شخص یا موجود برتری خاصی ندارد و هیچ موجودی بقیه را کنترل نمی کند بلکه هر عضو با توجه به داده های خود و همچنین داده های همسایگانش، سعی به رسیدن به هدف مشترک کل اجتماع می کند. این کار نیاز به الگوریتم های **Disturbuted** دارد و این سیستم های چند عاملی، بدون نیاز به کنترل کننده مرکزی هدف خود را محقق می سازند. در این میان می توان به انعطاف پذیری شبکه، هزینه و پردازش کمتر در مقیاس های بزرگ و همچنین محدودیت کمتر در ایجاد شبکه و سیغت بعنوان مزیت های این شبکه ها اشاره کرد. در مقابل آن شبکه های **Centralized** وجود دارند که کاربردهای متفاوتی دارند و در ادامه مفصل توضیح داده خواهد شد.

## Swarm Intelligence

- AI technique based on the collective behavior in decentralized, self-organized **multi-agent** Systems
- Generally made up of agents who interact with each other and the environment
- No centralized** control structures
- Characteristics
  - Composed of many individuals
  - Individuals are **homogeneous**
  - Local interaction based on **simple rules**
  - Self-organization**



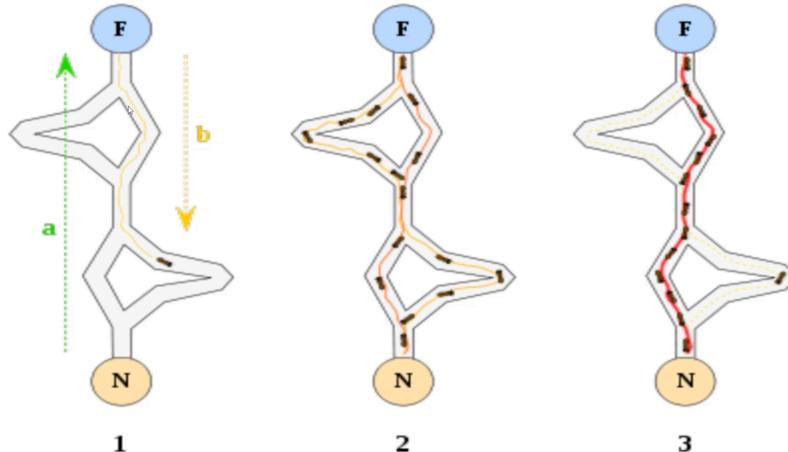
شکل (۲-۲) اطلاعات بالا به صورت خلاصه در این اسلاید فراهم آورده شده است. مفهوم **homogeneous** به معنی این است که همه عامل ها شبیه بهم هستند و هیچ عاملی برتری ای ندارد.

لازم به ذکر است اکثر الگوریتم‌هایی که در این زمینه از سال‌های گذشته مطرح شده، الهام گرفته شده از طبیعت است و رفتارهای موجودات زنده در یافتن الگوریتم‌های جدید بسیار کمک‌کننده است. در این خصوص می‌توان به الگوریتم حرکت مورچه‌ها، کلونی زنبورها و همینطور حرکت دسته‌جمعی پرندگان اشاره کرد.

#### ۲-۵-۲- مقدمه

برای اینکه الگوریتم‌های هوش جمعی کمی ملموس‌تر شوند، به حرکت دسته‌جمعی مورچه‌ها نگاه کنید:

## Intelligence



Swarm Intelligence, Dr. Samaneh Hosseini

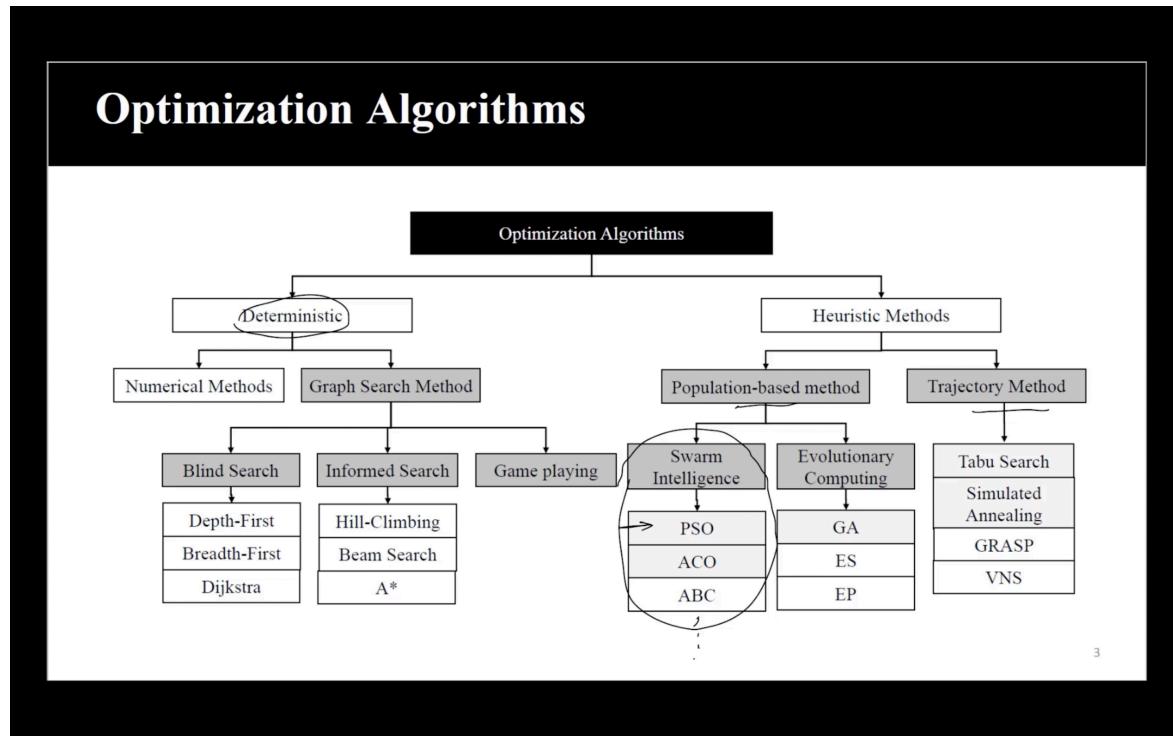
13

شکل (۱۹-۲) حرکت دسته‌جمعی مورچه‌ها

در تصویر (۱۹-۲) مشاهده می‌کنید که بعد از چند بار رفت و برگشت مورچه‌ها، مورچه جدید مسیر کوتاه‌تر را انتخاب می‌کند. این امر به این دلیل شکل می‌گیرد که هر مورچه در حرکت خود ماده‌ای به نام فرومون ترشح می‌کند که با بو کشیدن آن، می‌تواند مسیر برگشت خود به لانه را پیدا کند. همچنین اگر از لانه بخواهد به سمت مقصدی حرکت کند، از مسیری حرکت می‌کند که این بو بیشتر باشد تا بتوانند حرکت دسته‌جمعی داشته باشند. به همین خاطر پس از گذشت چند مرحله، مورچه‌هایی که از مسیر کوتاه‌تر رفته بودند، زودتر به لانه بر می‌گردند و این باعث می‌شود در حرکات بعدی، بوی مسیر کوتاه‌تر بیشتر باشد.

در اینجا هر مورچه با دسته‌ای از اطلاعات سر و کار دارد که مقداری از آن وابسته به همسایگی خود و دسته‌ای وابسته به اطلاعات خود مورچه است که باعث می‌شود کل مورچه‌ها حرکتی بی‌نقص داشته باشند.

الگوریتم‌های هوش جمعی نیز به همین صورت عمل می‌کنند. معمولاً هر عضو، صرفاً با دانستن دسته‌ای از اطلاعات شخصی و در اختیار داشتن دسته‌ای از اطلاعات از همسایگانش، مسیرش را انتخاب می‌کند و باعث می‌شود کل شبکه حرکتی دسته‌جمعی و زیبا نمایش دهد که در بعضی از مقالات، علت این امر به صورت ریاضی اثبات شده است.



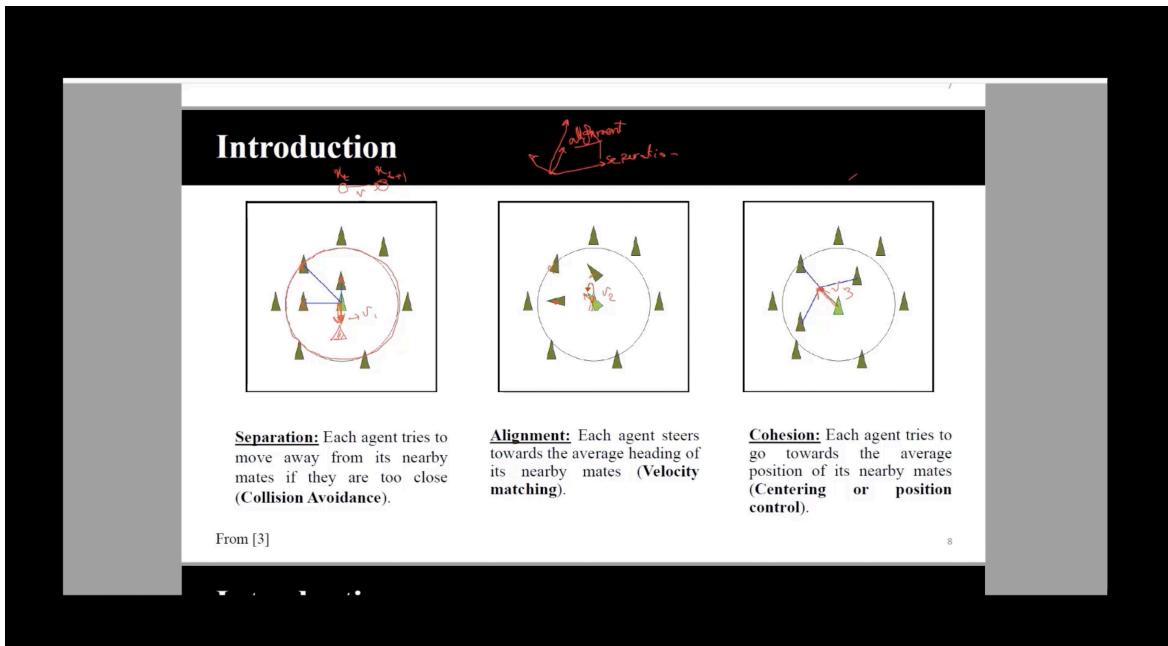
شکل (۲۰-۲) جایگاه الگوریتم‌های هوش جمعی در مقایسه با دیگر الگوریتم‌های هوش مصنوعی

در ادامه مقدمه‌ای بر شکل‌گیری چندی از این الگوریتم‌ها و همچنین کارهای مرتبط انجام شده در آزمایشگاه پرداخته خواهد شد:

در سال ۱۹۸۷ فردی به نام Craig Reynolds برای ساخت فیلمی، نیاز به شبیه‌سازی گرافیکی پرواز پرنده‌ها داشت. از این رو به مطالعه رفتار آن‌ها پرداخت و قوانینی تعیین کرد که پرنده‌گان مهاجر برای اینکه مهاجرتی موفق داشته باشند، از آن‌ها استفاده می‌کنند. این ۳ قانون ساده اما کاربردی به شرح زیر است.

هر پرنده تلاش می‌کند:

- با پرنده‌گان همسایه برخورد نکند
- همسو با هدفی که همگان در حال حرکت به سمت آن هستند حرکت کند
- با سرعتی متناسب با سرعت همسایگانش حرکت کند



شکل (۲۱-۲) ۳ قانون ذکر شده توسط رینولدز، در این اسلاید قابل مشاهده است

از آن پس الگوریتم‌های زیادی براساس این ۳ قانون مطرح شده‌اند و این ۳ قانون پایه اکثر الگوریتم‌های کلاسیک ارائه شده قرار گرفتند.

تا مدت‌ها براساس این قوانین الگوریتم‌های زیادی مطرح و به صورت تجربی استفاده می‌شد اما در سال ۱۹۹۵، مقاله‌ای تحت عنوان Flocking [1] مطرح شد که این قوانین را به صورت ریاضی اثبات کرد.

بعد از مدت‌ها الگوریتمی تحت عنوان Semi Flocking مطرح شد که در آن صرفاً یک هدف واحد برای کل مجموعه در نظر گرفته نمی‌شد و این امکان وجود داشت حداقل به اندازه‌ی هر عضو، هدف متفاوت داشت که برای اینکار، چندتا از term‌های الگوریتم آپدیت شد.

بعد از آن در سال ۲۰۲۰ توسط دکتر حسینی و چندی از همکارانشان، مقاله‌ای تحت عنوان Force-Based Motion Planning Algorithm یا مختصرًا FMP ارائه شد که روش جدیدی برای این رفتار ارائه کردند. در این روش هر ربات صرفاً با دانستن موقعیت همسایگانش می‌توانست به نتیجه قبلی دست پیدا کند. در الگوریتم قبلی، علاوه بر موقعیت همسایگان، نیاز به دانستن سرعت آن‌ها نیز بود که الگوریتم ارائه شده باعث شد با حذف یکی از پارامترهای اصلی، از نظر کارایی و سرعت محاسبات، بهبود فراوانی ایجاد کند.

در این الگوریتم به تعداد هر عضو هدف وجود دارد و الگوریتم کلی دستخوش تغییراتی شد. همانطور که از عنوان این الگوریتم پیداست، نیروهای بین ربات‌ها مورد بحث قرار می‌گیرد و هر ربات با نزدیک شدن به ربات کناری، به نسبت این فاصله‌ی ایجاد شده، با شدت متناسبی ربات کناری را دفع می‌کند که این الگوریتم کاملاً براساس رفتار ذرات و منطبق بر نیروهای علم فیزیک مطرح شده است. پیروی از این الگوریتم و این نیروهای دافعه باعث می‌شود از برخورد ربات‌ها با هم جلوگیری شود.

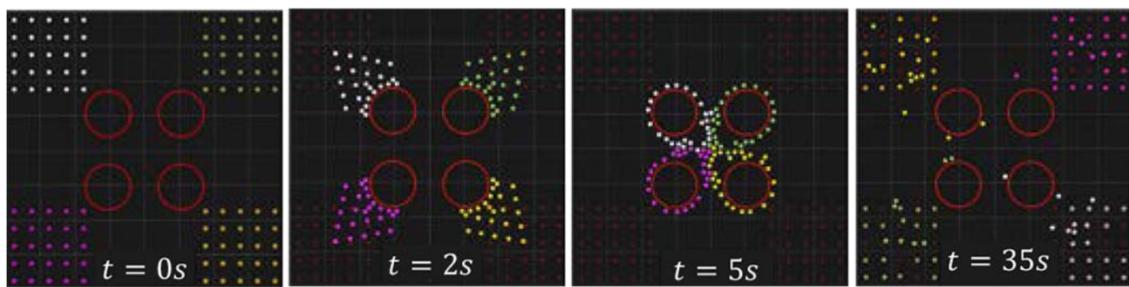


Fig. 7. Simulation of four groups of 25 agents that moves with opposite directions through a narrow passage formed by the presence of four static obstacles. Red dots represent initial and goal positions—designated times indicating transition time. The full video is available as Extension.

شکل (۲۲-۲) نتایج الگوریتم FMP. در این شبیه‌سازی قرار است جایگاه هر دسته از عامل‌ها بدون هیچ‌گونه برخورد، با دسته‌ی عامل‌های روپرتویی جابجا شود

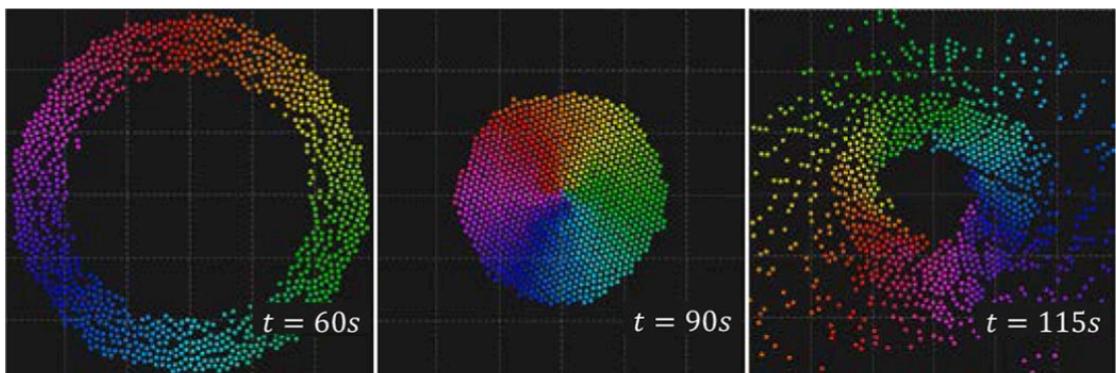


Fig. 6. Simulation of 1000 agents trying to move to antipodal positions at transition times: 60, 90, and 115 s. Agents smoothly move through the congestion that forms in the center. The full video is available as Extension.

شکل (۲۳-۲) در این آزمایش نیز قابل مشاهده است که عامل‌ها بدون برخورد متراکم شده و پس از چرخش ۱۸۰ درجه، دوباره پراکنده می‌شوند

لازم به ذکر است توضیحات ارائه شده بسیار کلی بوده و خلاصه‌ای از روند تکامل الگوریتم‌ها بوده است و اصل کار بسیار مفصل‌تر است که باید مقاله‌های ارائه شده مطالعه گردد.

در ادامه روش‌های دیگری برای رسیدن به هدف‌های بحث شده، مورد استفاده قرار گرفت. یکی از این روش‌ها که بسیار معروف است، تحت عنوان CADRL در سال ۲۰۱۷ یک شد که در این روش، از یک شبکه عمیق تقویتی از پیش آموزش دیده شده استفاده می‌شود تا در هنگام شرایط برخورد، ربات‌ها از این شبکه استفاده کنند.

---

**Algorithm 1: CADRL (Coll. Avoidance with Deep RL)**

---

```

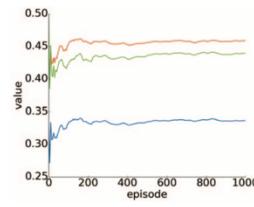
1 Input: value network  $V(\cdot; \theta)$ 
2 Output: trajectory  $s_{0:t_f}$ 
3 while not reached goal do
4   update  $t$ , receive new measurements  $s_t$ ,  $\tilde{s}_t^o$ 
5    $\hat{\tilde{V}}_t \leftarrow \text{filter}(\tilde{V}_{0:t})$ 
6    $\hat{s}_{t+1}^o \leftarrow \text{propg}(\tilde{s}_t^o, \Delta t \cdot \hat{\tilde{V}}_t)$ 
7    $U \leftarrow \text{sampleActions}()$ 
8    $u_t \leftarrow \arg\max_{u_t \in U} R(s_t^{jn}, u_t) + \bar{\gamma} V(\hat{s}_{t+1}, \hat{s}_{t+1}^o)$ 
     where  $\bar{\gamma} \leftarrow \gamma^{t-t_{pref}}$ ,  $\hat{s}_{t+1} \leftarrow \text{propg}(s_t, \Delta t \cdot u_t)$ 
9 return  $s_{0:t_f}$ 

```

---



(a) Value network



(b) Convergence

---

**Algorithm 2: Deep V-learning**

---

```

1 Input: trajectory training set  $D$ 
2 Output: value network  $V(\cdot; \theta)$ 
3  $V(\cdot; \theta) \leftarrow \text{train\_nn}(D)$  //step 1: initialization
4 duplicate value net  $V' \leftarrow V$  //step 2: RL
5 initialize experience set  $E \leftarrow D$ 
6 for episode=1, ...,  $N_{eps}$  do
7   for m times do
8      $s_0, \tilde{s}_0 \leftarrow \text{randomTestcase}()$ 
9      $s_{0:t_f} \leftarrow \text{CADRL}(V), \tilde{s}_{0:t_f} \leftarrow \text{CADRL}(V)$ 
10     $y_{0:T}, \tilde{y}_{0:t_f} \leftarrow \text{findValues}(V', s_{0:t_f}, \tilde{s}_{0:t_f})$ 
11     $E \leftarrow \text{assimilate}\left(E, (y, s^{jn})_{0:t_f}, (\tilde{y}, \tilde{s}^{jn})_{0:t_f}\right)$ 
12     $e \leftarrow \text{randSubset}(E)$ 
13     $\theta \leftarrow \text{RMSprop}(e)$ 
14    for every  $C$  episodes do
15      Evaluate( $V$ ),  $V' \leftarrow V$ 
16 return  $V$ 

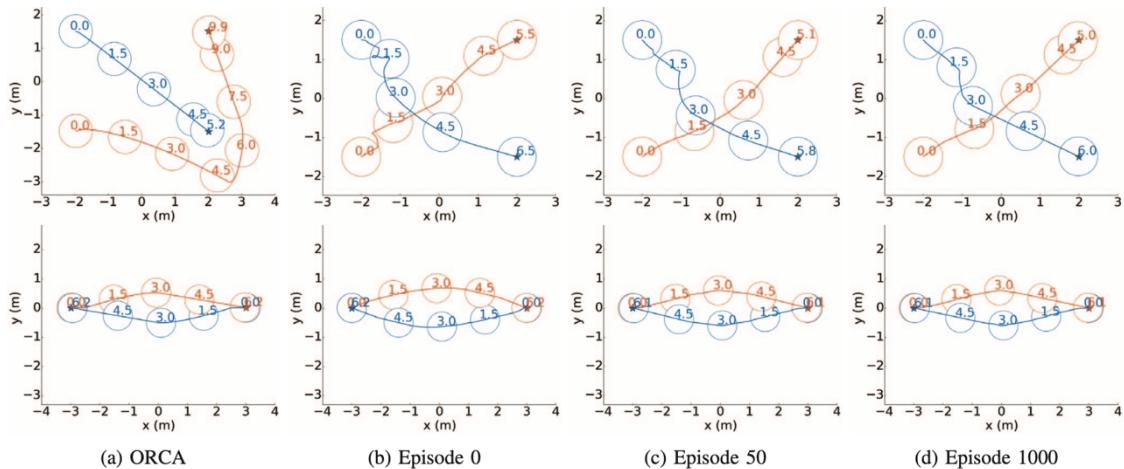
```

---

$$\operatorname{argmin}_{\theta} \sum_{k=1}^N \left( y_k - V(s_k^{jn}; \theta) \right)^2.$$

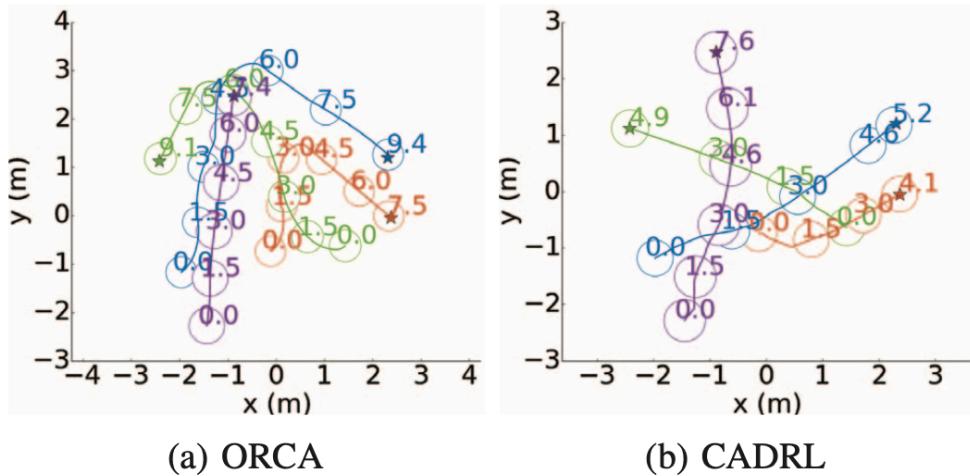
This work uses optimal

شکل (۲۴-۲) در این شکل شبکه‌کد الگوریتم CADRL مشاهده می‌شود. در سمت راست مدل آموزش می‌بیند و در سمت چپ شبکه‌کدی است که هر عامل در هنگام مواجه با شرایط ویژه، اقدام به تصمیم‌گیری بر اساس مدل آموزش دیده شده می‌کند.



شکل (۲۵-۲) در شکل فوق، نتایج الگوریتم ORCA که یک الگوریتم معروف کلاسیک است را در مقایسه با الگوریتم CADRL در دو سناریو مختلف مشاهده می‌کنید. قابل مشاهده است که در سناریوهای ساده تفاوت چندانی مشاهده نمی‌شود اما در سناریوهای پیچیده، نتایج بست آمده توسط الگوریتم CADRL مسیر بهینه‌تری را تضمین می‌کند.

در ادامه چندین عکس برای فهم بهتر این مطالب ارائه شده است که می‌توان با توجه به پانویس‌ها، از محتواهای آن‌ها استفاده برد.



شکل (۲۶-۲) مقایسه عملکرد دو الگوریتم *ORCA* و *CADRL* در یک سناریو پیچیده نیگر

#### IEEE TRANSACTIONS ON CYBERNETICS

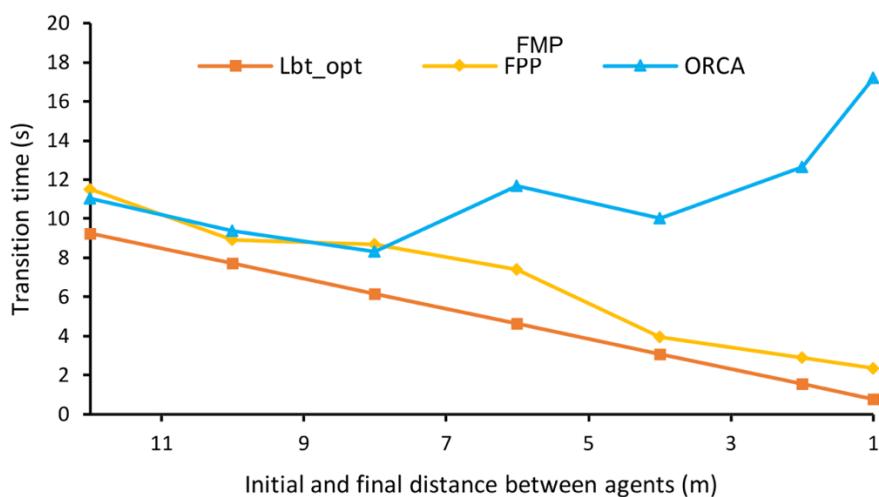
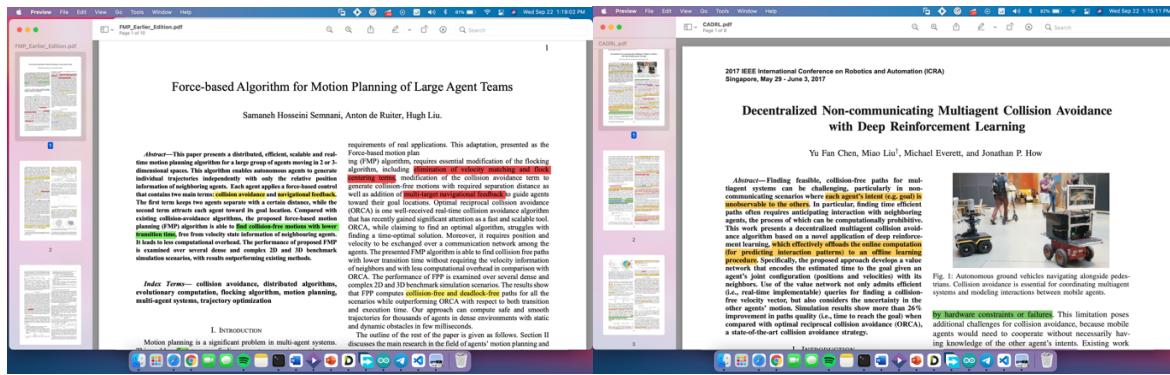
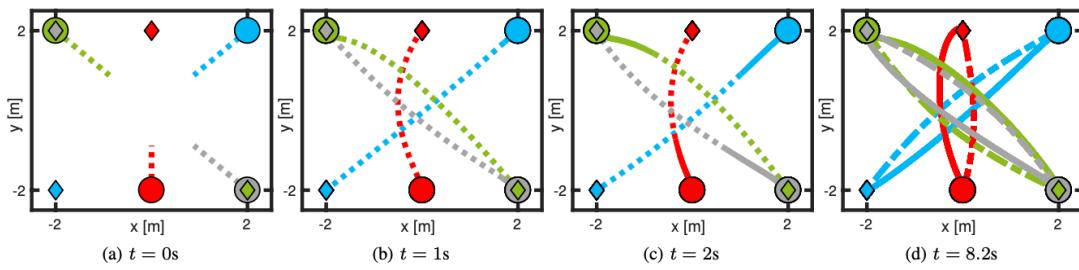


Fig. 9. Comparison of FMP and ORCA produced motions transition times with theoretical lower bound of optimal transition time ( $Lbt_{opt}$ ).

شکل (۲۷-۲) مقایسه عملکرد الگوریتم‌های *ORCA*، *FMP* و *CADRL*



شکل (۲۸-۲) دو از مقاله‌های خوانده شده در مدت کارآموزی، مقاله‌ها مربوط به الگوریتم‌های FMP و CADRL می‌باشند.



شکل (۲۹-۲) یکی دیگر از تست‌هایی که بر روی الگوریتم CADRL استفاده شده و نتایج خوبی داشته (به بازه‌های زمانی نقط شود)

|                                     | FMP   | ORCA    |
|-------------------------------------|-------|---------|
| Number of deadlocks                 | 0     | 9       |
| Overall min separation distance (m) | 5.07  | 4.98    |
| Average Transition time (s)         | 47.30 | 422.51  |
| Average Execution time (ms)         | 99.24 | 1527.45 |

شکل (۳۰-۲) نتایج سرعت اجرا و موقعیت‌های deadlock در دو الگوریتم FMP و ORCA

فیلم‌هایی از تست ربات و همچنین شبیه‌سازی الگوریتم ORCA و CADRL در این [لينك](#) قابل مشاهده است.

## مراجع

- [1] Flocking: <https://arxiv.org/pdf/1310.3601.pdf>
- [2] FMP: <https://arxiv.org/abs/1909.05415>
- [3] CADRL: <https://arxiv.org/abs/1609.07845>
- [4] ORCA:  
[https://www.researchgate.net/publication/347505481\\_ORCA\\_Optimization\\_Algorithm\\_A\\_New\\_Meta-Heuristic\\_Tool\\_for\\_Complex\\_Optimization\\_Problems](https://www.researchgate.net/publication/347505481_ORCA_Optimization_Algorithm_A_New_Meta-Heuristic_Tool_for_Complex_Optimization_Problems)