



جامعة تشرين

(أطروحة بكالوريوس)

تصميم وتنفيذ خوارزمية تخطيط مسار حسابية لسرب روبوتات اعتقاداً على الحل الرقي للمعادلات التفاضلية الجزئية الخاصة
بجزء الموضع

كلية الهندسة الميكانيكية والكهربائية
برنامج هندسة الميكترونيكس للمتميزين

أسماء الطلاب المتفذدين

عثمان سعود

ندى سليمان

صبا عيسى

بإشراف

د. عيسى الغنام

د. مثنى القبيلي

2021/06



Tishreen University

(Bachelor's Thesis)

**Design and Implementation of a computational motion planning algorithm
based on fluid dynamics PDE for Robotic Swarms**

**Faculty of Mech. and Elec. Engineering
Program of Mechatronics For the Distinguished**

Prepared by:

Siba Issa

Nada Salman

Adnan Saood

Supervised by

**Mothanna Alkubeily,
Ph.D.**

Issa AlGhannam, Ph.D.

2021/06



تصميم وتنفيذ خوارزمية تطبيط مسار حسابية لصربة روبوتاتك المتميزة على العمل الرقمي
المعادلات التفاضلية الجزئية الخاصة بجريان المواقع

أعده الطالب:

صبا عيسى، ندى سليمان، وعدنان سعود

ياشراف:

الدكتور مثنى القبيلي، الدكتور عيسى الغنام

جامعة تشرين، اللاذقية، سوريا

06.2021



Design and Implementation of a computational motion planning algorithm based on fluid dynamics PDE for Robotic Swarms

Prepared by

Siba Issa, Nada Salman, Adnan Saood

Supervisor by

Prof. Mothanna Alkubeily, Prof. Issa AlGhannam

at
Tishreen University, Latakia, Syria

02.2021

تصادق لجنة الحكم بعد قراءتها ومناقشتها لمشروع التخرج
على أنه ملائم من حيث النوعية والأهمية ليكون بحثاً
لمشروع التخرج.

أعضاء لجنة الحكم:

We, the Exam committee, certify that we have
read this project and that in our opinion it is
fully adequate, in scope and quality, as a
thesis for final project.

Exam committee:

Abstract

Formation of non-holonomic swarm robots has received a lot of considerations in both academia and industry over last two decades. Many path planning methods produce collision-free paths for a group of multiple robots, but many of them are computationally expensive and complex to implement and develop. In addition, these methods do not find a solution in all environments, or they find a solution that is not optimal. In this project, we present a computational motion planning algorithm based on fluid dynamics PDE for Robotic Swarms. This proposed method features a computationally cheap algorithm, and it is easy to develop using updated tools and hardware. But, most importantly, it always ensures finding a path, because partial differential equations of fluid dynamics always finds a solution that describes the movement of the fluid's particles towards a goal. The aforementioned feature gives this algorithm a clear advantage because it is reliable, and it produces a smooth path. The validity of this algorithm will be tested on a robotic swarm platform that we designed and implemented to ensure high quality and ease of use. The control of the swarm is preformed centrally, and a visual feedback is provided continually to the central computer from a camera that is fixed above the robots.

1.0 موجز البحث

تعد مضمونات هذا البحث أحد القضايا التي تطرق لها الباحثون بكثرة واهتمام في الفترة الأخيرة: التشكيل السريري للروبوتات غير الهولنومية. هناك العديد من الطرق التي قد توصل مجموعة من الروبوتات من منطقة إلى أخرى في فضاء العمل بسلام (أحياناً)، لكن عددها إما تحوي تعقيدات برمجية وبنية خوارزمية ومخطط حالة غاية في التعقيد، أو أنها لا توفر حل في أغلب الأحوال. نحاول في هذا البحث تقديم خوارزمية تخطيط مسار لسرب من الروبوتات الغير هولنومية في بيئه مكتظة باعتبارها تابعة بالحالة للحل الرقمي لمعادلة جريان المائع من منطقة إلى الأخرى بشكله المستقر. من مزايا الحل المقدم أنه رخيص حسابياً ويملك العديد من العناصر القابلة للترشيع باستخدام أدوات حسابية وعتاد حاسوبي جديد، لكن محور البحث يمكن في مكان آخر. نحاول في هذا البحث تقديم طريقة حتمية لإيجاد مسارات حركة هذه الآليات من منطقة لأخرى في فضاء العمل، فالمعادلات الرياضية لجريان السوائل حتماً ستجد حلّاً يقوم بإ يصل جزيئات المائع من المنطق للهدف. هذه الحتمية ستعطي أفضلية واضحة للطريقة المقترحة لسببين: موثوقية الطريقة (إذا كان الوصول ممكناً فهو حتماً ضمن مجموعة الحل)، ونعومة المسارات المنتجة (ما يمكن الآليات الغير هولنومية من تقليل الفارق العتادي مع الهولنومية). سيتم اختبار طريقة الحل المقترحة على منصة روبوتات قمنا بتصميمها وتنفيذها بشكل كامل، مع مراعاة عوامل مثل جودة التصنيع والمرنة والسهولة في الاستخدام والتعديل والتكلفة المناسبة. تم تصميم المنصة بحيث أن يتم التحكم والتواصل مع الروبوتات بواسطة حاسوب مركزي ويتم مراقبتها بواسطة كاميرا مثبتة أعلى المنصة.

المحتويات

9	1.0	موجز البحث
13	1	مقدمة البحث
13	1.1	خلفية وأهداف البحث
13	2.1	الدراسات المرجعية
13	1.2.1	التصنيف والتصميم
14	2.2.1	تخطيط المسار
17	3.2.1	الاتصال
18	3.1	أهداف المشروع
18	4.1	مخطط العمل
18	5.1	مخطط الأطروحة
21	2	الدراسة الرياضية والنمذجة
21	1.2	تخطيط المسار لروبوت واحد باستخدام الطرق التقليدية
21	1.1.2	الخوارزميات المعتمدة على البيان
21	2.1.2	خوارزمية الحقل الكموني
24	2.2	تعميم الحل للأسراب
25	1.2.2	تعريف مسألة CBS
26	3.2	تخطيط المسار عن طريق المعادلات التفاضلية الجزئية لحركة المواقع
26	1.3.2	الخوارزمية المقترنة لروبوت واحد
30	2.3.2	الخوارزمية المقترنة للسرب
33	4.2	التغذية الراجعة لشعاع حالة الروبوتات
33	5.2	النموذج الرياضي العام لروبوت
34	6.2	دراسة العجلات
39	3	بنية النظام الصلبة والبرمجية
39	1.3	مقدمة

39	فضاء العمل	2.3
40	بنية النظام الصلبة	3.3
40	نظرة عامة	1.3.3
40	الأدوات	2.3.3
41	وحدة التحكم	3.3.3
42	التغذية	4.3.3
43	نظام الاتصال	5.3.3
43	الدارة الإلكترونية المطبوعة PCB	6.3.3
44	عملية تصنيع الدارات	7.3.3
44	اختبار المنتج	8.3.3
45	الجسم الخارجي والتحريك	9.3.3
49	نظام الاتصال	10.3.3
49	بروتوكول MQTT	11.3.3
51	بنية النظام البرمجية	4.3
51	نظرة عامة	1.4.3
51	البنية البرمجية للروبوت منخفضة المستوى	2.4.3
53		4 النتائج
63		5 الملحقات
63	Manual Reference Code C AVR	1.5

باب 1

مقدمة البحث

1.1 خلفية وأهداف البحث

إن مفهوم الأسراب مأخوذ من الطبيعة، حيث تمتمحاكاة أسلوب التعاون بين الأسراب البيولوجية (النمل، النحل، الأسماك..) ونقل هذا المفهوم إلى الروبوتات. يمكننا تصنيف الأسراب البيولوجية إلى أسراب أرضية وأخرى جوية، وكذلك الأمر بالنسبة للروبوتات. يختلف هذان التواعان فيما بينهما من ناحية التطبيق بشكل أساسي وقد اعتمد البحث [1] على تصنیف التطبيقات حسب نوع المهمة التي تتولاها الروبوتات (تغطية منطقة ما، مهامات خطيرة، مهامات يلعب الوقت فيها دور أساسي...). بعد تخطيط المسار الحسابي للروبوتات الأرضية مشكلة مطروقة جداً في الوقت الحالي لما لها من أهمية في توفير الوقت، الطاقة، والعتاد الحسابي. حسابياً، إن مشكلة تخطيط المسار الأفضل لأي آلية هي مسألة NP-Hard [2, 3]. تتعقد آليات الحساب وتبعاته عند التعامل مع بيئة ديناميكية، أو عند القيام بتخطيط مسار أكثر من كائن في ذات الوقت. يهدف المشروع إلى البحث في إمكانية تطوير آليات لتخطيط المسار الحسابي لمجموعة كائنات اعتماداً على معادلات الجريان الناتجة عن حل المعادلات التفاضلية الجزئية لحركة المولاع. لقد قدم كل من نافير وستوكس معادلات كافية لمنفذة حركة السائل في الظروف المختلفة، وسيتم الاعتماد عليها في إنشاء المسارات.

2.1 الدراسات المرجعية

يقدم هذا القسم مجمل ما تم مراجعته بهدف تصميم وتنفيذ خوارزمية البحث والمنصة الفعالة.

1.2.1 التصنیف والتصميم

إن الأنظمة متعددة الروبوتات تلاقي اهتماماً كبيراً في المراجع ولاسيما في الآونة الأخيرة؛ نظراً إلى أن مجالات التطبيق والمهام التي تواجهها تزداد تعقيداً. لقد تم اقتراح تصنیف جديد في عام 2004 اعتماداً على ثلث نقاط أساسية: (1) عقلانية التصميم، (2) الوظائف التقنية الأساسية (الكل من العتاد الصلب والبرمجيات)، (3) المهام التي يجب أن تقوم الروبوتات بتنفيذها. حيث تم التصنیف على عدة مستويات ضمن فئتين أساسيين: الأبعاد التنسيقية، وأبعاد النظام كما هو موضح في الجدول 1.1 [4].

وفي العام التالي طرحت الورقة البحثية [5] تبين أن أي فعل يقوم به أي روبوت يأخذ بالحسبان الأفعال التي تقوم بها بقية الروبوتات، وهكذا ينتج مجموعة من الروبوتات التي تعمل باتساق لتحقيق أفضل أداء ممكن. وقد أظهرت هذه الآلة من التنسيق أداءً عالٍ؛ حيث

Dimensions System	الأبعاد التنسيقية
Communication	التعاون
Composition team	المعرفة
Architecture System	التنسيق
Size Team	التنظيم

جدول 1.1: الأبعاد التنسيقية وأبعاد النظام

تمكنت الروبوتات من اكتشاف بيئتها في وقت أقل مقارنةً مع المنهجيات الأخرى التي لا تعتمد على التنسيق بين الروبوتات بشكل صريح. وبالرغم من النتائج الجيدة الذي حققها النهج المتبعة في البحث [5]، إلا أن هناك بعض النقاط التي تحتاج إلى تطوير، حيث يجب مناقشة الحالات التي يتم فيها تعطل أحد الروبوتات أو حدوث تغير ما في البيئة المحيطة بها.

ومن ناحية أخرى، نالت أعداد الروبوتات التي تشكل السرب اهتماماً كبيراً في مجال البحث العلمي. على الرغم من أن سلوك كل روبوت على حدا يُعد سلوكاً بسيطاً، إلا أنه عندما تعمل هذه الروبوتات معاً يصبح سلوكها أكثر تعقيداً. أشار [6] إلى أن فهم الروبوتات بشكل فردي هو أمرٌ سهل، أما بالنسبة لتوقع سلوكها معاً فهو أمرٌ صعب المنال. لقد تم عرض النهج المعتمد من قبل [6] من خلال التحليل العميق لخوارزمية السرب المتبعة والنماذج الاحتمالي المرتبط بها؛ حيث تم فحص خوارزمية التحكم باستخدام فضاء الحالة المطورة من قبل [7] على مجموعة من الروبوتات الباحثة عن طعام.

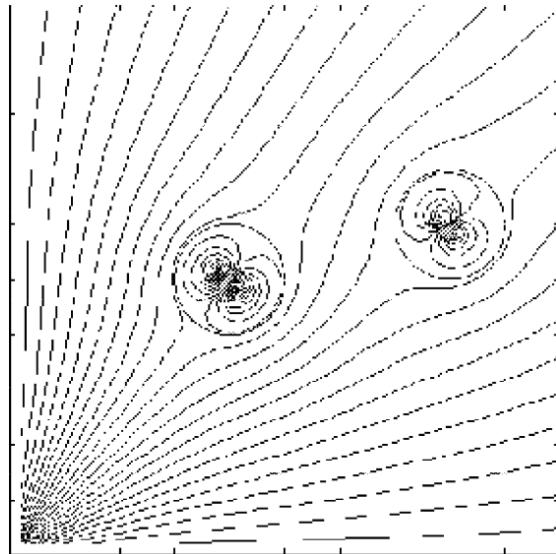
لقد قام [6] باستخدام خوارزمية الـ *الـ VG* في شبكة الاتصال اعتماداً على المنطق الرمزي (*Temporal logic*)، حيث تم التركيز على حالة الروبوتات بدلاً من الموقع وتفاصيل حركة كل روبوت. ولكن وبالرغم من النتائج المرضية وصل إليها البحث، إلا أنه لم يتمكن من إجراء عملية التحليل سوى على عدد قليل من الروبوتات بسبب مشكلة انفجار الحالة (*state explosion*) التي ظهرت أثناء عملية فحص النماذج الرياضي.

2.2.1 تخطيط المسار

تُعد مشكلة تخطيط المسار متعدد الروبوتات مشكلة معقدة حسابياً لأن صعوبة الحساب تنمو مع عدد الروبوتات. إن وجود قيود وعوائق في المسار يولّد مشكلة كبيرة، ومن هنا بدأت عملية البحث عن خوارزميات لحل هذه المشكلة. أحد الخوارزميات المُتبعة لحل مشكلة المسار المقيد هي الـ *VG*. تبدأ هذه الطريقة بتشكيل الرسم البياني عن طريق رسم العقد (*Nodes*) والحواف (*Edges*) باستخدام تقنية التصوير بالأشعة، ومن ثم تحديد البُيان الموجه. يتم بعد ذلك إيجاد المسار الأقصر باستخدام خوارزميات إيجاد الطريق الأقصر في البُيان. ذكر هنا أن المسار الناتج عادةً قريب جداً من العوائق.

وكان البحث [8] قد قدم عام 1985 طريقة فريدة لقادري العوائق اعتماداً على تقنية الـ *PotentialField*، حيث تم التحكم بالروبوتات على عدة مستويات وبالرغم من الحقيقي. وتتميز هذه الطريقة بإمكانية توسيع الحل ليشمل البُيانات ذات العوائق المُتحركة باستخدام حقل كموني متغير مع الزمن. وفي عام 2006 قدم العمل [9] طريقة حقل كموني لتخطيط مسار الروبوت بالاعتماد على ناتج جمع تابعين هما: تابع (*repulsive*) الذي يتَجنب المسارات القريبة من العوائق، وتتابع (*Attractive*) قيمته تتَعلق بالمسافة عن الهدف. إن طريقة الـ *Potentialfield* هي طريقة مباشرة لحساب الحقل المتجهات (*vector field*) اعتماداً على موقع الهدف والعوائق، ولكنها تتطلب روبوتات متحكم بها بشكل كامل (قادرة على التحرك بأي اتجاه بشكل آني). على خلاف الحقل الكموني، طرق تخطيط المسار المعتمدة على الدوال التتفافية (*Streamfunction*) لا تتطلب التحكم بذلك، كما أنها تضمن تحويل جميع نقاط التوازن إلى نقاط سرجية، مما يعني ضمان وصول الروبوت إلى هدفه. لقد أظهرت عملية تخطيط المسار المعتمدة على الدوال التتفافية المطروحة في [10] فعالية وسرعة عالية في التكيف مع التغيرات في الشروط. على عكس طرق تخطيط المسار الأخرى، الطريق الذي تعتمد على الدوال تتَميز بمظاهرها الحدسية بالنسبة لمراقب الخارجي. إن وجود أكثر من حاجز لا يضمن دائماً الحصول على نتائج صحيحة، لكن هذا المثال

يوضح أن السلوك لا يزال مقبولاً كما هو موضح في الشكل 1.1.



شكل 1.1: مثال على الدوال التدفقية

في عام 2015 تم طرح سلالات فورونوي VORONOI STRAINS كخوارزمية جديدة لتخطيط المسار والتي حققتفائدة مهمة على صعيد البيانات المعقّدة [9]. لقد حققت هذه الخوارزمية نتائج مثيرة للاهتمام في مساحات عمل فيها عدد كبير من العوائق، فقد أظهرت حلولاً أفضل بكثير من خوارزمية PSO البسيطة والمُطورة من أجل نفس عدد التكرارات. إضافة إلى ذلك، أثبتت أنها أكثر صلادة وأقل اعتماداً على الحالة الابتدائية (العشوانية). كما تتميز هذه الخوارزمية باحتمالية أكبر لتجنب النقاط الصغرى المحلية، ولكن قد أثبتت التجارب أن عملية الأمثلة هذه غير مجده مالم تكون البيئة معقّدة بشكل كافٍ.

أثناء تخطيط الحركة يمكننا تصنيف العمل إلى صنفين أساسين؛ فإنما أن يتم تخطيط المسار بشكل كامل قبل البدء بالحركة، (offline) أو أن تتم عملية التخطيط بشكل تدريجي أثناء الحركة. (online)

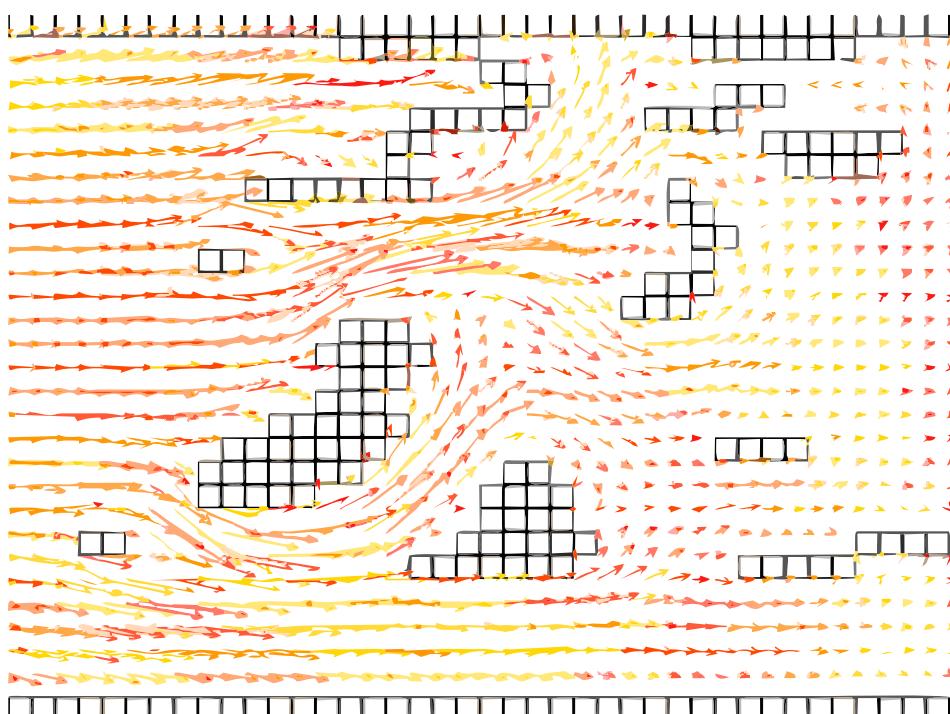
في [11] ، تم التركيز على تخطيط المسار بشكل كامل قبل بدء الحركة، بالإضافة إلى وجود حواجز ثابتة، وهي إحدى الأمور التي أعطت هذا البحث محدوديته. تتميز طريقة تخطيط المسار المقترنة في [11] بأنها تمكننا من الحصول على مسار ثلاثي الأبعاد مثالي يمكن الروبوتات من التحليق على ارتفاعات منخفضة في بيئه ذات تضاريس معقّدة.

وفي عام 2020 طرحت الورقة البحثية [12] بهدف توجيه روبوتات طائرة في مجال جوي مفتوح وتصاريص غير معلومة لتنتفو بذلك على البحث [11] وذلك عبر حل معادلة الاستمرارية للموائع. لقد حققت هذه الدراسة وقتاً أقل من أربع طرق أخرى (A^* , Dijkstra, DP, PE)

إضافة إلى الخوارزميات التقليدية، العديد من الدراسات أظهرت نتائج خوارزميات التعلم الآلي لحل هذه النوع من المشكلات. في [13] تم استخدام الشبكات العصبية الالتفافية لتحسين حلول الخوارزميات التقليدية كما نجد في [14] أنه تم استخدام خوارزمية تعلم Q-learning معتمدة على الخوارزمية المحسنة improved particle swarm optimization. تهدف الخوارزميات المستخدمة إلى تقليل طول المسار وزمن الوصول وزوايا دوران الروبوتات لتقليل استهلاك الطاقة. تمت المقارنة بين أربع خوارزميات وايجاد النتائج باختلاف عدد الروبوتات في السرب هي CQL و PSO و IPSO-DV و QIPSO-DV وأظهرت النتائج تفوق QIPSO-DV على باقي الخوارزميات كما يمكن ملاحظة تراجع أداء هذه الخوارزمية مع زيادة عدد الروبوتات ولاسيما من حيث زمن التنفيذ وعدد التكرارات

اللازمة للوصول الى الحل الأمثل (إن وجد) . في [15] تم اقتراح تقسيم الخريطة الكلية الى خرائط بيان جزئية sub-graphs مع قيود على الدخول والخروج من كل خريطة وذلك لضمان إيجاد المسار. بعد التجريب، تم التوصل الى حل فعال وسريع نسبيا لكنه لا يضمن ايجاد مسار أمثل وبحاجة إلى عمليات معالجة لاحقة لتقليل المسافات الضائعة.

بعد موضوع انشاء طريق باستخدام الحل الرقمي للمعادلات التفاضلية الجزئية موضوع حديث عالمياً ويحتاج الكثير من العمل، قدمت واحدة من الأبحاث القائمة على موضوع استغلال جريان السوائل في توليد المسار عام 2017، حيث تهدف [16] الى انشاء المسار الأفضل لحركة سفينة باستخدام طريقة latticeBoltzmann الرقمية. البحث لم يناقش آلية انشاء طريق من نقطة إلى أخرى في فضاء العمل، بل ناقش عبور منطقة تحتوي عائق بشكل عام. إضافة لذلك، فإن الحل الرقمي باستخدام latticeBoltzmann لم ينتج حقول متوجهات سرعة تحمل نفس السلوك الفيزيائي للسوائل، يمكن التتحقق من ذلك بالنظر في سلوك الحقل عند الحاجز في الشكل 2.1.



شكل 2.1: التدفق المولد باستخدام الخوارزمية latticeBoltzman . يمكن ملاحظة السلوك غير الفيزيائي للسائل بالنظر الى تداخل السائل مع الجدران.

وكان قد طُرِح في عام 2009 نظام قوي وسريع لخطيط المسار للملاحة الآلية [17]، وال فكرة الأساسية التي اعتمد عليها هذا البحث هي انشاء المسار عبر نشر موجتين متاليتين، كما في الشكل 3.1. تبدأ بموجة أولى تكون ذات سرعة قليلة نسبياً تُمكّن الروبوت من نقل طوبولوجيا الخريطة إلى الحاسوب، بينما تنتقل الموجة الثانية ذات السرعة العالية في الخريطة وذلك من أجل تكوين مسار من نقاط جهات الانتشار ذات التغير الأعلى. وفي النهاية يمكن التحكم في المسارات المثلث المخططة اعتماداً على معامل واحد: المسار الأقصر، الأكثر أماناً، أو الهجين.



شكل 3.1: الأمواج المتقدمة والمتاخرة في العمل [17]

3.2.1 الاتصال

كما أشرنا سابقاً، فإن فكرة الأسراب مأخوذة من الأسراب البيولوجية، حيث أن أساس التحكم في سلوك هذه الأسراب هو تبادل المعلومات. يتَّكَّنُ الأفراد في الأسراب البيولوجية من تبادل المعلومات بشكل مباشر عن طريق المجرسات، الحركة، أو الصوت. استوحي [18] تصنيف أسراب الروبوتات من الأسراب البيولوجية، حيث تم التصنيف إلى مجموعتين أساسيتين:

• **أولاً: التفاعل عن طريق التحسس:** هو الطريقة الأبسط للتواصل بين الروبوتات. تتطلب من الروبوت التمييز بين الأقران والبيئة المحيطة به. يطلق على هذا النوع من الاتصال بالتصريح.

• **ثانياً: التفاعل عن طريق البيئة:** حيث تستخدم الروبوتات البيئة كوسيلة للتواصل فيما بينها. يسمى هذا النوع بالتواصل الضمني. أسراب النمل مثل شهير على الاتصال الضمني؛ حيث يتواصل أفراد النمل عن طريق إفراز وتبث مادة الفيرومون [18].

هناك العديد من طرق الاتصال المستخدمة في أسراب الروبوتات، ومن التقنيات المستخدمة للاتصال بين الروبوتات في الترب الواحد (الأشعة تحت الحمراء، البلوتوث، ZigBee، Wi-Fi,...). تعد عملية اختيار التقنية المناسبة مهمة جداً لأنها مؤثر مباشر على أداء السرب، لذلك تتم بناء على عدة معايير (البيئة التي تعمل بها الروبوتات، مدى الاتصال، معدل نقل البيانات، ...) وقد قام العمل [19] بمقارنة أداء هذه الطريق.

وعندما نتوجه نحو الاتصال اللاسلكي فإن شرائح الـ ESP8266 توفر ما يجعلها الخيار الأمثل، فهي منخفضة الكلفة مقارنة بأدائها، وتدعم بروتوكولات TCP/IP بشكل كامل. تتكون من SoC يحوي معالج 32bit Tensilica Xtensa وشريحة Wi-Fi تدعم بروتوكول IEEE 802.11b/g/n. وقد قام العمل [20] بمقارنة أداء الاتصال اللاسلكي بين ESP8266 وـ PainlessMesh.

مجتمع الـ OpenSource الخاص بشريحة الـ ESP وفر مكتبة PainlessMesh المبنية خصيصاً لتشكيل شبكات تعتمد على أجهزة ESP8266؛ حيث أنها تبني شبكة خالية من الطبقات وتقوم بالتحديث كل 3 ثواني. إلا أنه ومع الأسف كان هناك انخفاض ملحوظ في أداء الشبكة مع ارتفاع العقد المُتصَّلة بها كما كان واضحاً في نتائج التجارب التي قام بها [20].

بصورة عامة، تكمن الغاية الأساسية من تصنيع وتركيب الروبوتات بشكل عام في الوظيفة التي ستقوم بها على أرض الواقع، ومن هنا يصبح للتنفيذ الفعلي أهمية كبيرة حيث أن محاكاة أداء الروبوتات ومهمها كان دقيقاً، فإنه يستخدم فقط لتقدير التصميم قبل التنفيذ الفعلي. عالمياً، نجد تنوع كبير في منصات أسراب الروبوت وفق الغاية منها، بعضها تكون لأهداف تعليمية والبعض الآخر لأهداف بحثية لختبار صحة الخوارزميات كما في بحثنا هذا. تقدم [21] مراجعة عامة لأهم وأشهر منصات أسراب الروبوت ومقارنة من حيث الخصائص العامة والحجم والسعر والحساسات المستخدمة. تم التوصل إلى أن معظم هذه المنصات مناسبة جداً للأغراض البحثية، كما أن بعض المنصات مثل MarXbot و sbot و Pheeno مناسبة للاستخدام على أرض الواقع وذلك لمرونة التصميم وإمكانية عبورها ضمن البيئات الصعبة. نجد في الجدول ملخصاً عن المقارنة بين المنصات.

الخصائص المميزة	العمر	النكلفة	الحجم
	البطارية (ساعة)	(دولار) (سم)	(دولار)
وحدة استشعار بالأشعة تحت الحمراء ذات مدى طويلاً	4	4	25 [22] Colias
حساسات ذات درجة حساسية عالية	2	5.6	65 [22] AmiR
دواريب ذات حركة بخط Omni	3	3.3	14 [23] Kilobot
قابلة للتغير بشكل ذاتي، آلية اقتران	6	10	220 [24] R-one
آلية اقتران، قابلية تجمع بشكل ذاتي، كاميرا متحركة بكل الاتجاهات	1	15	- [25] s-bot
يمكن محاكاتها باستخدام محاكي ENKI و WEMBOTS	2	75	1200 [26] e-puck
قابلة للتغير بشكل ذاتي، آلية اقتران، كاميرا متحركة بكل الاتجاهات، تصميم بنية معيارية	8	17	- [27] MarxBot
آلية قبض، كاميرا	-	7.12	270 [28] Pheeno

3.1 أهداف المشروع

- تصميم وتنفيذ روبوتات صغيرة الحجم وعالية الجودة وسهلة الاستخدام.
- تصميم منصة عمل مناسبة تمكننا من اختبار أي خوارزمية تخطيط مسار ضمن بيئه عمل معروفة سابقاً.
- تطوير وبرمجة خوارزمية تخطيط حسابي تعتمد على المعادلات التقاضية الجزئية لروبوت واحد ولسرير من الروبوتات.
- برمجة خوارزميات شهيرة أخرى في تخطيط مسار الروبوت لروبوت واحد ولسرير من الروبوتات.
- استخلاص نتائج تؤكد صحة الخوارزمية المقترحة وتقويها على الخوارزميات التقليدية وغير التقليدية من ناحية حتمية إيجاد المسار وعوامل أخرى.

4.1 مخطط العمل

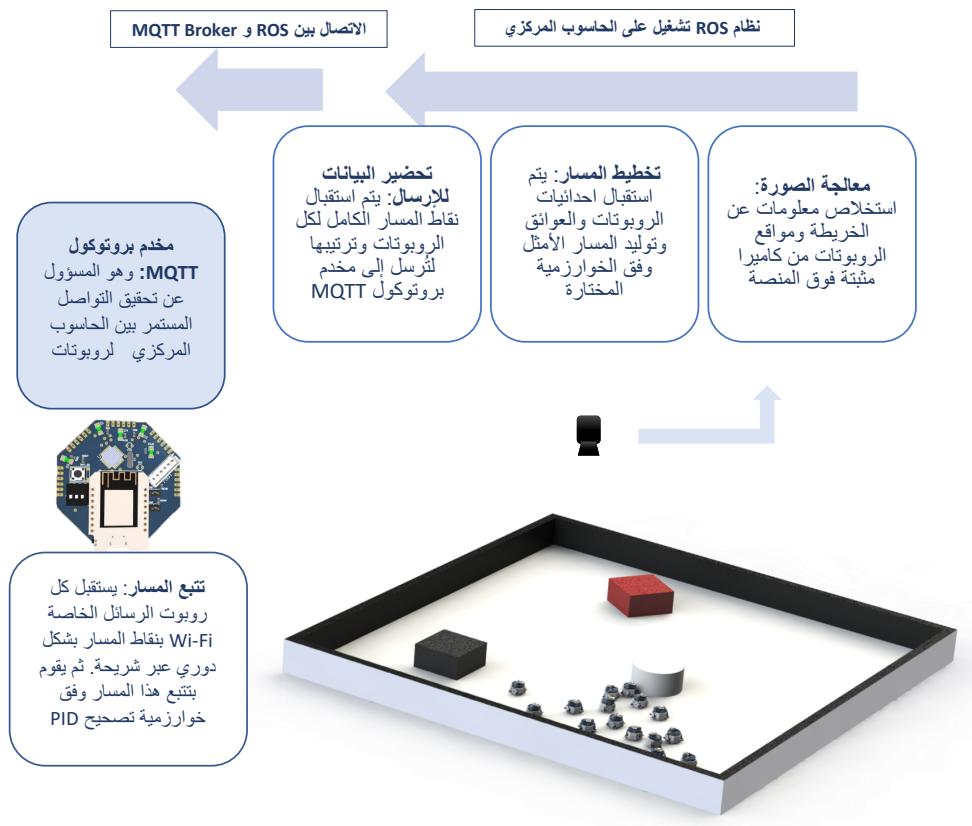
يوضح المخطط التالي آلية كاملة لعمل المشروع. نبدأ من عرض البيئة البرمجية الموجودة على الكمبيوتر المركزي والتي تتكون من نظام تشغيل روس مكون من ثلاثة عقد تتصل مع بعضها بتنقية Publish / Subscribe، ثم نوضح تواصل هذه النظم مع مخدم بروتوكول MQTT المسؤول عن تحقيق التواصل مع الروبوتات حيث تشتراك الروبوتات بـ topic التي تحوي الرسائل الحاملة لنقطات المسار. بالنسبة للنظام البرمجي الموجود على كل من الروبوتات، تستقبل شريحة الاتصال الرسالة من المخدم وتتلقاها إلى المعالج ليثيوم باترياتها وفق خوارزمية التصحيح. أنظر الشكل 4.1.

5.1 مخطط الأطروحة

بعد أن قدمنا معلومات عن خلفية البحث والدراسات المرجعية وعرضنا مخططاً يوضح آلية عمل المشروع بكامل أجزائه، ننتقل إلى الفصل الثاني لشرح الدراسات النظرية والمنفذة والمعادلات الرياضية المستخدمة وذلك بما يتضمن خوارزميات تخطيط المسار ونظام الاتصال بشكل أساسي. نقدم في الفصل الثالث المعلومات الكاملة عن عملية تصنيع الروبوتات بما في ذلك اختيار القطع الكهربائية والالكترونية وتصنيع الدارات وطباعة الأجسام كما نذكر شرحاً تفصيلياً عن نظام العمل البرمجي المطور لتشغيل الروبوتات والكمبيوتر المركزي. في الفصل الرابع نقدم شرحاً عن النتائج التي توصلنا إليها حتى الآن.

5.1. مخطط الأطروحة

19



شكل 4.1

باب 2

الدراسة الرياضية والنمذجة

1.2 تخطيط المسار لروبوت واحد باستخدام الطرق التقليدية

1.1.2 الخوارزميات المعتمدة على البيان

تعتمد هذه الخوارزمية على فضاء العمل لإنشاء بيان يمثل كل الأماكن الممتاحة في فضاء العمل متضمناً نقطتي البداية والنهاية. يضمن هذا التمثيل ترميز وجود كل الطرق الممتاحة وكذلك يضمن آلية لتقادري العوائق عن طريق حذف عقدتها من البيان. بعد الانتهاء من القطعية نعبر البيان من نقطة البداية إلى النهاية باستخدام أحدى خوارزميات عبور البيان الشهيرة (A^* , *Disjkstra*). يمثل الشكل 1.2 طريقة نقل فضاء العمل إلى البيان.

المسار باستخدام A^*

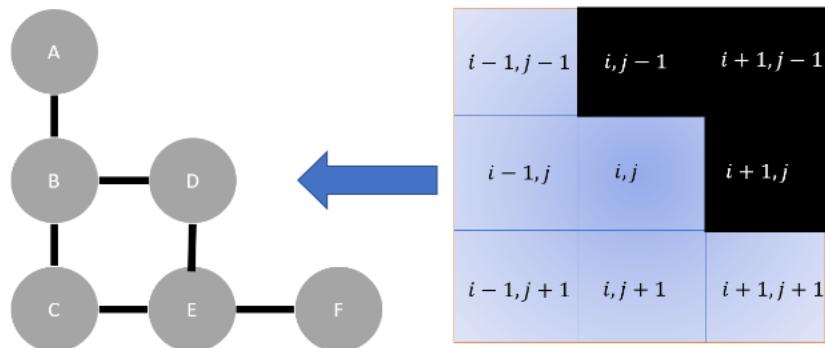
خوارزمية عبور بيان وإنشاء مسار تستخدم في كثير من مجالات علم الحاسوب نظرً لكمالها، مثاليتها، وكفاءتها. في الأنظمة التي من الممكن حساب الطريق بشكل استباقي، يمكن تجاوز أدائها من قبل الخوارزميات التي تستطيع حساب المسار قبل البدء بالتنفيذ [29]. واحدة من أكبر مشاكلها هي تعقيدها المكاني: $O(b^d)$.

المسار باستخدام *Dijkstra*

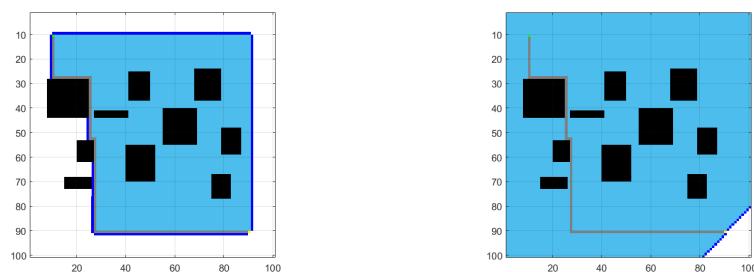
هي خوارزمية صممت لإيجاد المسار الأقصر بين عقدتين في بيان معين. تعتمد الخوارزمية على إيجاد الطريق من خلال عبور عقد البيان بالتدريج ثم حساب معاملات المسافة وإيقاف البحث في حال مواجهة عائق [30]

2.1.2 خوارزمية الحقل الكموني

تعتمد خوارزمية potential field على بناء خريطة افتراضية يتحرك الروبوت وفقها اعتماداً على موقع المبدأ والهدف والواحجز . يمكن أن يتم حساب المسار اعتماداً على خريطة حواجز معروفة مسبقاً أو بالزمن الحقيقي اعتماداً على حساسات مثبتة على الروبوت أو على



شكل 1.2: تحويل فضاء العمل المقطعي إلى بيان غير موجه



جدول 1.2: خرائط لحل كل من خوارزميتي البيان.

1.2. تخطيط المسار لروبوت واحد باستخدام الطرق التقليدية

الحلبة. تم عرض هذه الخوارزمية لأول مرة عام 1985 [8] حيث قدم البحث الحقول *potential field* كجزئين، جزء جاذب وجزء منفر. الجزء الجاذب يقوم بسحب الروبوت نحو الهدف، أما الجزء المنفر يقوم بإبعاد الروبوت عن العائق بمسافة أمان تحدد مسبقاً.

تستخدم هذه الخوارزمية بكثرة في تطبيقات تخطيط المسار وذلك لبساطتها وسهولة تطبيقها ولا سيما أنها تعطي نتائج جيدة في البيانات البسيطة التقليدية، لكن يوجد عدة سلبيات مرتبطة باستخدام هذه الخوارزمية. أحد السلبيات هي أن الروبوت يمكن أن يحتجز في قيمة صغيرة محلية مما يؤدي إلى توقيه عن الحركة نهائياً وعدم وصوله إلى الهدف. تم تطوير العديد من الطرق لحل هذه المشكلة مثل تتبع الحائط [31] وإضافة حواجز افتراضية [32]. من السلبيات الأخرى هي عدم قدرة الروبوت على المرور بين حواجز مقربة. عند عبور الروبوت بممر ضيق يقوم بحركة اهتزازية غير مستقرة تأثره بقوى الحقل المنفر [33].

توليد المسار وفق *:potential field*

شعاع الحقل الجاذب يحسب من موقع الروبوت إلى الهدف، أما شعاع الحقل المنفر يكون من موقع الحاجز إلى موقع الروبوت. بجمع الشعاعين نحصل على شعاع يحدد اتجاهه اتجاه حركة الروبوت، ومطالله سرعة الروبوت. ليكن (x_r, y_r) موقع الروبوت الابتدائي و (x_g, y_g) موقع الهدف و (x_o, y_o) موقع الحاجز في خريطة ثنائية الأبعاد.

يعطى شعاع الحقل الجاذب بالعلاقات التالية:

$$\vec{v}_{attr} = \begin{pmatrix} x_{attrr} \\ y_{attrr} \end{pmatrix} \quad (1)$$

$$\begin{pmatrix} x_{attrr} \\ y_{attrr} \end{pmatrix} = \begin{pmatrix} x_g - x_r \\ y_g - y_r \end{pmatrix} \quad (2)$$

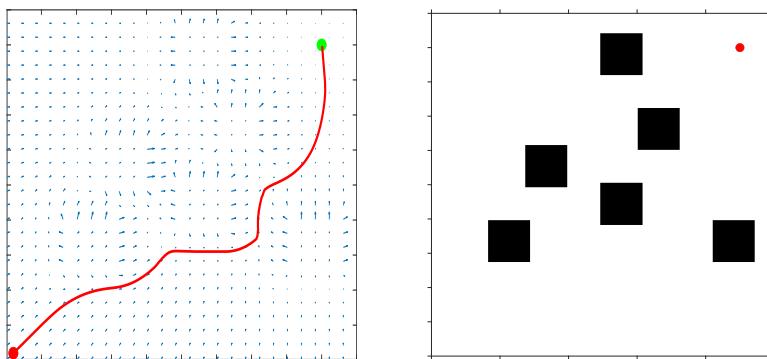
يعطى المسافة والزاوية بين الروبوت والهدف بالعلاقات التالية:

$$d_o = \sqrt{(x_r - x_o)^2 + (y_r - y_o)^2} \quad (3)$$

$$\theta = \tan^{-1}\left(\frac{y_r - y_o}{x_r - x_o}\right) \quad (4)$$

وبالتالي يعطى شعاع الحقل المنفر بالعلاقات التالية:

$$\vec{v}_{rep} = \begin{pmatrix} x_{rep} \\ y_{rep} \end{pmatrix} \quad (5)$$



جدول 2.2: عملية حساب المسار في طريقة الحقل الكمومي

$$x_{rep} = \frac{1}{d_o} \cos(\theta_o), if d_o < s_{red} = 0, otherwise \quad (6)$$

$$y_{rep} = \frac{1}{d_o} \sin(\theta_o), if d_o < s_{red} = 0, otherwise \quad (7)$$

حيث s هي قطر المنطقة الفعالية المحيطة بمركز الحاجز. جمع شعاعي الحركة الجاذب والمنفر يحصل على شعاع الحركة النهائي:

$$\vec{v}_{final} = \vec{v}_{attr} + \vec{v}_{rep} \quad (8)$$

2.2 تعميم الحل للأسراب

يمكن تصور تعميم الخوارزميات السابقة لسرب من الروبوتات بشكل مبسط بالشكل الآتي: نقوم بحساب المسار الأمثل لكل روبوت على حدٍ باعتبار كل بقية الروبوتات على أنها عوائق، وإعادة توليد هذه المسار بشكل مستمر باستخدام التقنية الراجعة من الكاميرا، ولكن وبسبب طبيعة هذه خوارزميات لا يمكن تطبيق هذا المفهوم البسيط على أرض الواقع. فكما نجد في [34] عند استخدام خوارزمية *A^{*} لتوليد مسار لسرب من الروبوتات، تطلب التطبيق عدة عمليات معالجة لاحقة وتنظيم لحركة الروبوتات لمنع التصادم، كما تمت الحاجة لتصميم وحل مسألة برمجة خطية لتقليل طول المسار و زمن الوصول. بالحالة العامة وفي معظم مسائل تخطيط المسار لسرب روبوتات، الهدف هو إيصال كل روبوت من نقطة البداية إلى نقطة الهدف بدون حدوث أي تصادم. كما وجدنا في الدراسة السابقة، فإن الكثير من الدراسات السابقة تعتمد تحسينات على خوارزميات تخطيط المسار التقليدية. في [34] تم تقديم خوارزمية جديدة تسمى البحث المعتمد على التصادم (CBS) وهي طريقة متميّزة تتكون من مرحلتين، مرحلة عالية المستوى ومرحلة منخفضة المستوى. في المرحلة المرتفعة المستوى، يتم البحث في ما يسمى شجرة التصادم (Conflict Tree) CT وهي شجرة تُبني بناءً على التصادم بين الروبوتات والعوائق، كل عقدة في الشجرة تمثل مجموعة قيود على حركة الروبوتات. في المرحلة المنخفضة المستوى،

2.2. تعميم الحل للأسراب

25

تم عملية بحيث سريع بما يحقق القيود المطروحة في الطبقة العالية المستوى. في معظم الحالات تحقق هذه الخوارزمية مسحًا لعدد أقل من العقد بالمقارنة مع A^* وتحافظ في نفس الوقت على الأمثلية.

1.2.2 CBS تعريف مسألة

نعرف كل قيد على أنه ثلاثة بالشكل (a_i, v, t) حيث أن الروبوت a_i ممنوع من المرور بالعقدة v في الزمن t خلال إيجاد الحل سيتم ربط كل روبوت بعدد من القيود لاحقاً يجب إيجاد مسار لكل روبوت بحيث يتحقق كل قيوده تعرف التصادم على أنه رباعية بالشكل (a_i, a_j, v, t) مما يدل على أن الروبوتات a_i و a_j تشغلا نفس العقدة v في الزمن t . يكون الحل المكون من k مسار (عدد الروبوتات) هو k صحيحاً عندما تكون كل المسارات خالية من أي تصادمات. المرحلة عالية المستوى:

حيث يتم بناء شجرة CT وهي شجرة ثنائية كل عقدة منها تتكون من:

• مجموعة قيود: (N.constraints) كل قيد ينتمي لروبوت، عقدة الجذر لا تحتوي على أية قيود وكل عقدة ابن ترث قيود العقدة الأم وتضيف قيداً جديداً.

• حل: (N.solution) مجموعة مسارات بعدد k بحيث أن مسار كل روبوت يحقق القيود المفروضة عليه. يتم إيجاد هذه المسارات من خلال بحث في المرحلة منخفضة المستوى.

• تكلفة كلية: (N.cost) تكلفة مسارات كل الروبوتات. تسمى هذه التكلفة f-value للعقدة N.

إيجاد المسار وفق شجرة CT :

تم عملية البحث عن مسار كل روبوت في المرحلة منخفضة المستوى وذلك بمعرفة جميع القيود، عند إيجاد المسارات (المسار الخاص بكل روبوت وفق قيوده) يتم اختبار صحتها. يتم الاختبار من خلال المرور على كل المواقع في كل الأزمنة والتحقق من عدم وجود أي روبوتين في نفس المكان والزمن. في حال تم العثور على تصادم، تتوقف العملية وتنقل إلى مرحلة حل التصادم. حل التصادم: عند حدوث تصادم من الشكل (a_i, a_j, v, t) ، نجد بدليهياً أنه يجب لإضافة إما (a_i, v, t) أو (a_j, v, t) إلى قائمة القيود N.constraints. لضمان أمثلية الحل، يتم اختبار الحالتين وتتقسم العقدة N إلى عقدتين ترثان قيود العقدة الأم وتضيفان قيد الروبوت الأول والروبوت الثاني

على الترتيب. يوضح الجدول 1 خوارزمية طريقة CBS للمرحلة عالية المستوى.

Algorithm 1: High Level CBS

Input: MAPF instance

Output: Your output

```

1 Root.constraints  $\leftarrow \Phi$ 
2 Root.solution  $\leftarrow$  find individual paths by low level procedure
3 Root.cost = SIC(Root.solution)
4 insert RoottoOpen
5 while Open  $\neq \Phi$  do
6   P  $\leftarrow$  best node from Open;
7   Validate paths from open until a conflict occurs
8   if P has no conflict then
9     Return P.solution
10  C  $\leftarrow$  firstconflict(ai, aj, v, t)inP
11  foreach agenti in C do
12    A  $\leftarrow$  newnode
13    A.constraints  $\leftarrow$  P.constraints + (ai, v, t)
14    A.solution  $\leftarrow$  P.solution
15    Update A.solution invoking low level ai
16    A.cost  $\leftarrow$  SIC(A.solution)
17    if A.cost < inf then
18      Insert A to Open
  
```

3.2 تخطيط المسار عن طريق المعادلات التفاضلية الجزئية لحركة المواقع

1.3.2 الخوارزمية المقترحة لروبوت واحد

تعتمد الخوارزمية المقترحة لإنتاج طريق يسلكه الروبوت في بيته على إنشاء حقل سرعة يمكن للروبوت الانحدار فيه مشكلاً طريقه الخاص. نعتمد هنا على إنشاء هذا الحقل عن طريق حل المعادلة التفاضلية الجزئية لتدفق السوائل الغير قابلة للانضغاط من نقطة إلى أخرى. أولاً، نكتب معادلات نافير-ستوكس بشكلها المتجمعي الخاصة بالمواقع غير القابلة للانضغاط:

$$\nabla \cdot \vec{v} = 0 \quad (9)$$

$$\frac{\partial v}{\partial t} + (\vec{v} \cdot \nabla) \vec{v} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \vec{v} \quad (10)$$

حيث $\vec{v} = (u, v, w)^T$ مركبات السرعة على المحاور الإحداثية الثلاث.

المعادلة الأولى تمثل احتفاظ الكتلة عند ثبات الكثافة (عدم الانضغاطية)، وتمثل المعادلة الثانية احتفاظ العزم. في الموضع غير الانضغاطية، معادلة احتفاظ الكتلة تخلق قيد كينيماطيكي يتطلب من حقل الضغط أن يتغير بشكل يسمح ل معدل تغير السرعة $\nabla \cdot \vec{v}$ بالانعدام في كل نقاط الفضاء. يمكن بناء حقل الضغط بأخذ تباع (divergence) معادلة العزم، بعدأخذ هذه الخطوة تظهر معادلة بواسون للضغط.

نعيد كتابة معادلات نافير-ستوكس بشكل مريح للتقطيع وذلك بعد تعويض $w = 0$:

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = -\frac{1}{\rho} \frac{\partial p}{\partial x} + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (11)$$

$$\frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} = -\frac{1}{\rho} \frac{\partial p}{\partial y} + \nu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (12)$$

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -\rho \left(\frac{\partial u}{\partial x} \frac{\partial u}{\partial x} + 2 \frac{\partial u}{\partial y} \frac{\partial v}{\partial x} + \frac{\partial v}{\partial y} \frac{\partial v}{\partial y} \right) \quad (13)$$

نوجه الان لإنشاء mesh خاص بالمكان يسمح لنا بحل هذه المعادلات ببيسر. اتجهنا لاستخدام تقطيع المكان الى مربعات سهولة برمجتها. تظهر الصورة في الملحق رقم 1 تقطيع لحبلة مولدة عشوائية. بعد تقطيع الحبلة يمكن بسهولة نقل 12 ، 13 ، 14 الى المقطع بالشكل:

$$\begin{aligned} & \frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} + u_{i,j}^n \frac{u_{i,j} - u_{i-1,j}^n}{\Delta x} + v_{i,j}^n \frac{u_{i,j} - u_{i,j-1}^n}{\Delta y} = \\ & -\frac{1}{\rho} \frac{p_{i+1,j}^n - p_{i-1,j}^n}{2\Delta x} + \nu \left(\frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} + \frac{u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n}{\Delta y^2} \right) \end{aligned} \quad (14)$$

$$\begin{aligned} & \frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta t} + u_{i,j}^n \frac{v_{i,j} - v_{i-1,j}^n}{\Delta x} + v_{i,j}^n \frac{v_{i,j} - v_{i,j-1}^n}{\Delta y} = \\ & -\frac{1}{\rho} \frac{p_{i,j+1}^n - p_{i,j-1}^n}{2\Delta y} + \nu \left(\frac{v_{i+1,j}^n - 2v_{i,j}^n + v_{i-1,j}^n}{\Delta x^2} + \frac{v_{i,j+1}^n - 2v_{i,j}^n + v_{i,j-1}^n}{\Delta y^2} \right) \end{aligned} \quad (15)$$

$$\begin{aligned} & \frac{p_{i+1,j}^n - 2p_{i,j}^n + p_{i-1,j}^n}{\Delta x^2} + \frac{p_{i,j+1}^n - 2p_{i,j}^n + p_{i,j-1}^n}{\Delta y^2} = \\ & \frac{\rho}{\Delta t} \left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right) - \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} - \\ & 2\rho \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta x} - \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \end{aligned} \quad (16)$$

بإعادة ترتيب المعادلات السابقة يمكن الحصول على الأطراف المتقدمة زمنياً:

$$\begin{aligned} u_{i,j}^{n+1} = & u_{i,j}^n - u_{i,j}^n \frac{\Delta t}{\Delta x} (u_{i,j}^n - u_{i-1,j}^n) - v_{i,j}^n \frac{\Delta t}{\Delta y} (u_{i,j}^n - u_{i,j-1}^n) \\ & - \frac{\Delta t}{\rho 2 \Delta x} (p_{i+1,j}^n - p_{i-1,j}^n) \\ & + \nu \left(\frac{\Delta t}{\Delta x^2} (u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n) + \frac{\Delta t}{\Delta y^2} (u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n) \right) \end{aligned} \quad (17)$$

$$\begin{aligned} v_{i,j}^{n+1} = & v_{i,j}^n - u_{i,j}^n \frac{\Delta t}{\Delta x} (v_{i,j}^n - v_{i-1,j}^n) - v_{i,j}^n \frac{\Delta t}{\Delta y} (v_{i,j}^n - v_{i,j-1}^n) \\ & - \frac{\Delta t}{\rho 2 \Delta y} (p_{i,j+1}^n - p_{i,j-1}^n) \\ & + \nu \left(\frac{\Delta t}{\Delta x^2} (v_{i+1,j}^n - 2v_{i,j}^n + v_{i-1,j}^n) + \frac{\Delta t}{\Delta y^2} (v_{i,j+1}^n - 2v_{i,j}^n + v_{i,j-1}^n) \right) \end{aligned} \quad (18)$$

$$\begin{aligned} p_{i,j}^{n+1} = & \frac{(p_{i+1,j}^n + p_{i-1,j}^n) \Delta y^2 + (p_{i,j+1}^n + p_{i,j-1}^n) \Delta x^2}{2(\Delta x^2 + \Delta y^2)} - \frac{\rho \Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)} \\ & \times \left[\frac{1}{\Delta t} \left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right) - \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \right. \\ & \left. - 2 \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} \frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} - \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \right] \end{aligned} \quad (19)$$

شروط المسألة:

- السرعة الابتدائية والضغط الابتدائي معرومان.
- السرعة معدومة عند الجدران .
- مشتق الضغط المكاني معدوم عند الجدران.
- الضغط +a عند نقطة انطلاق الروبوت -a عند الهدف.
- الضغط صفر عند الجدران.

بالحل الرقمي **iterative** يمكن الحصول على الحالة الثابتة **state steady** للمائع بعد عدد معين من التكرارات حيث يتم حساب السرعات المتقدمة بزمن قدره Δt . يمكن اختبار الوصول الى الحالة الثابتة من عدمه عن طريق معاينة $\frac{dv}{dt}$ ، $\frac{du}{dt}$ في كل iteration وانهاء الحساب عند هبوطه عن حد معين ϵ . عند الحصول على الحل النهائي نقوم بجعل الروبوت ينحدر باستخدام الحقل المتجهي الناتج موصلاً نفسه الى الهدف بشكل مباشر. عند انشاء حقل السرعة لحريطة بأبعاد 100×100 يستغرق الحاسوب الموصوف في ما

3.2. تخطيط المسار عن طريق المعادلات التفاضلية الجزئية لحركة المواقع

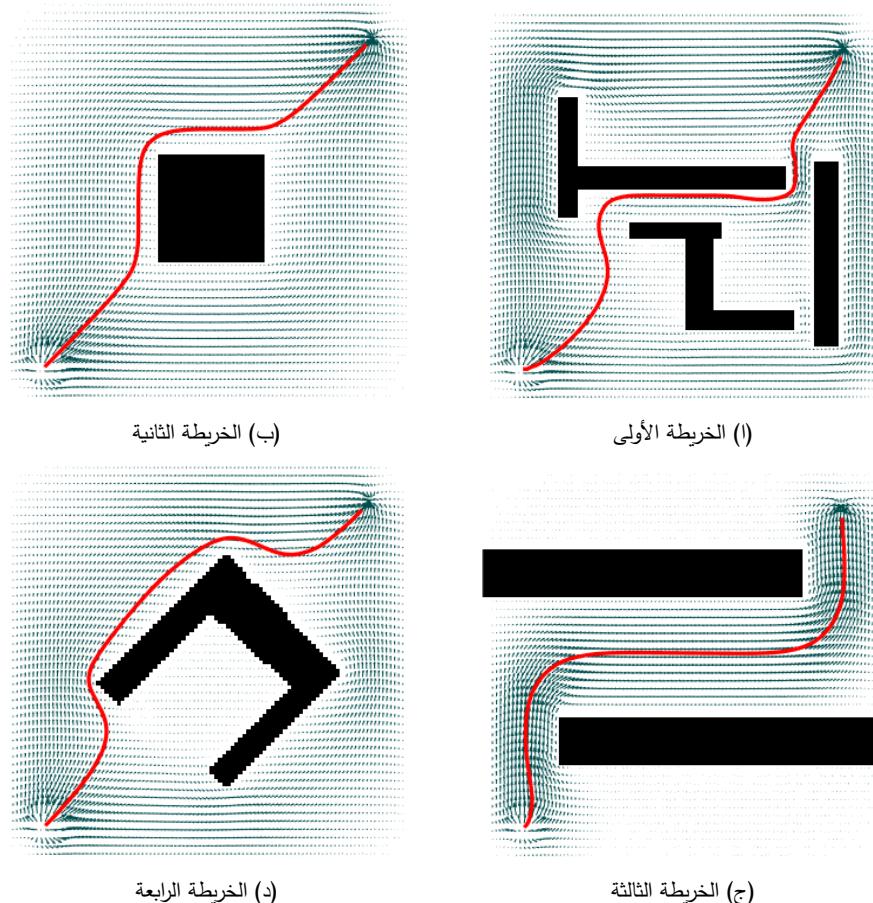
متوسطه 500 ملي ثانية. نذكر هنا أن هذا الزمن لا يعبر عن أصغر زمن يمكن للنظام انتاج حقل به، فهو لم يتعرض لأمثلة من هذه النواحي بعد:

- شكل التقاطع.

- المعاملات الفيزيائية للمائع.

- التسريع باستخدام العتاد الصلب (Hardware Acceleration).

نوضح فيما يلي أمثلة عن حل الخوارزمية لعدة فضاءات عمل:



جدول 3.2: أداء الخوارزمية لعدة خرائط

2.3.2 الخوارزمية المقترحة للسرب

أفضت الطريقة المتبعة في إنتاج مسار لروبوت واحد في فضاء اختياري ناتج عالي الوثيقية والأمان، التجارب أثبتت بشكل كبير قدرة هذه الطريقة على إنتاج طريق ناعم حال وجود اتصال بين المبدأ والهدف. يعد تعليم هذه الطريقة لضمان وصول مجموعة روبوتات إلى هدفها عملية غاية في التعقيد، وذلك خصوصاً لطبيعة الروبوتات الغير هولنومية المدروسة في هذا المشروع. يمكن الاستدلال إلى تهجين بين عدة طرق لإيجاد ما يناسب حالات خاصة من فضاءات اعمل في الدراسات المرجعية. لنتمكن من إنتاج حل يتسم بالبساطة الاستدلالية وبالاحتمالية والكمالية، يجب العودة إلى الآلية التي تتصرف فيها جزيئات سائل وجدت نفسها في حقل كموني. إن آلية التفكير هذه تقضي إلى نوعين من الحلول:

- الروبوتات غير واعية لبعضها

- الروبوتات (الجزيئات) واعية لبعضها

يمكن مقاربة الحل الأول بجزيئات سائل لا نيوتوني تسير في حقل السرعة المنتج، والطريقة الثانية بعدد من الأجسام صلبة تهوي في هذا الحقل.

الروبوتات غير الوعية لوجود السرب

هنا يتم احتساب حقل كموني صغير خارج من كل روبوت في السرب وإضافة هذه الحقول إلى حقل السرعة منهاً للتتصادم، يتم هنا احتساب جميع قيم السرعة في كل نقاط الفضاء عن طريقأخذ جريان المائع من جهة والجيران من جهة أخرى. يمكن التحكم بقوة الحقل الكموني النافر الصادر عن كل روبوت بتغيير بaramترات الحساب. آلية الحساب:

يتم أولاً ملء الخريطة فارغة بأجسام تتناسب أبعادها مع أبعاد الروبوتات الباقية في السرب، ثم يتم حساب تحويل المسافة للمصفوفة الناتجة (distance transform). يتبع هذه الخطوة القيام بعملية مشرحة في الجدول 4.2.

ثم يتم جمع الحقل الناتج إلى حقول السرعة وإنتاج الحقل الخاص بكل روبوت. انطلاق الروبوت على الحقل الناتج يضمن وصوله إلى الهدف بسلامة. انظر الشكل 2.2 كمثال.

الروبوتات الوعية لوجود السرب

يتم في هذه الطريقة برمجة وجود نابض ومحمد بين كل روبوتين متباينين من روبوتات السرب. وجود هذه النظم الميكانيكي سيتضمن حركة السرب ضمن شكلية معينة نجمية تحدد عناصرها من بaramترات البرمجة. يمكن حساب القوة المتبادلة بين روبوتين متباينين كالتالي:

$$F_{ij} = F_k + F_c = k(d_{ij} - d_0) + c \frac{d}{dt}(d_{ij} - d_0) \quad (20)$$

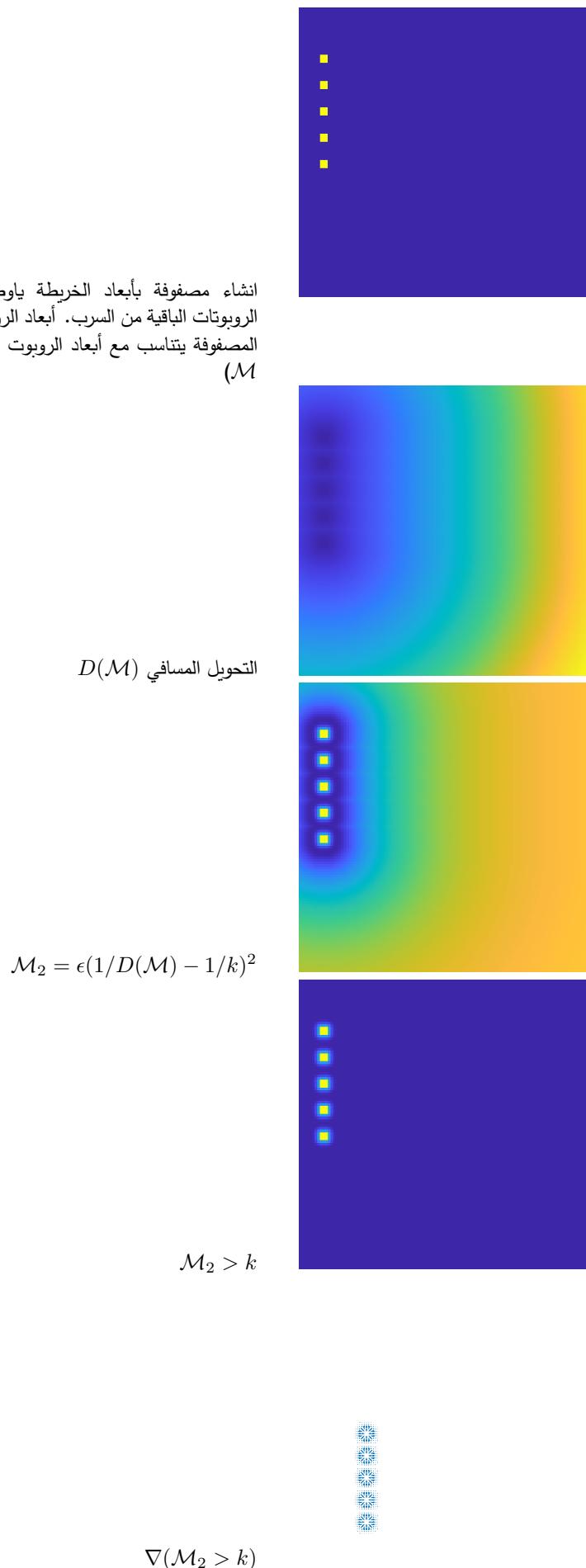
وبعدها يتم تحصيل القوى باستخدام:

$$\vec{F}_i = \sum_{j \in N_i} \vec{F}_{ij} \quad (21)$$

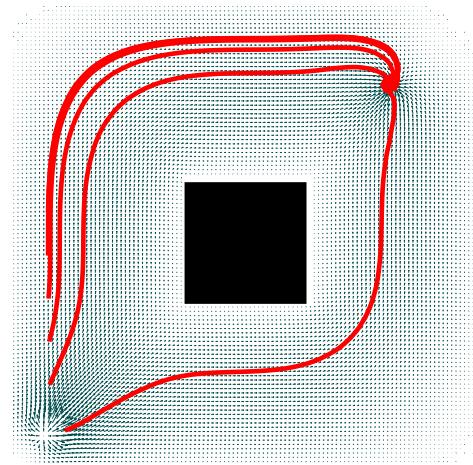
تجمع القوة السابقة مع القوة الناتجة عن وجود حقل سرعة مائع لإنتاج خطوة المشي الجديدة. انظر الشكل 3.2

3.2. تخطيط المسار عن طريق المعادلات التفاضلية الجزئية لحركة المواقع

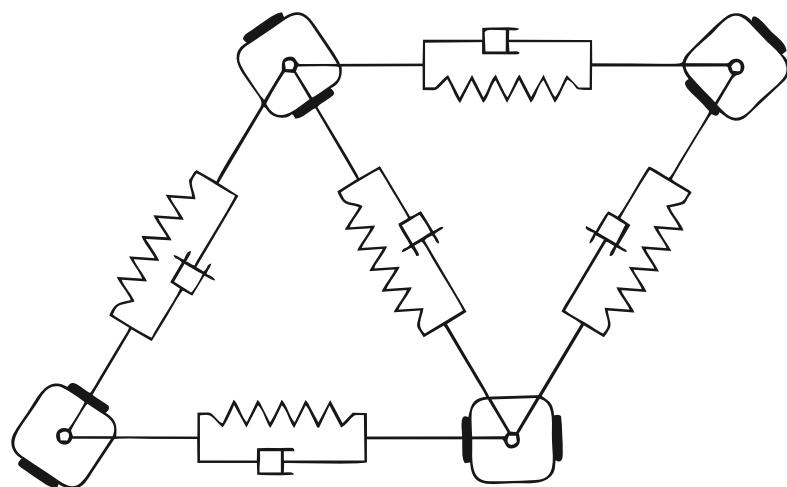
31



جدول 4.2: عملية حساب المسار في الحالة الأولى من الحل



شكل 2.2: مثال عن الطريقة المقترحة



شكل 2.3: القوى المتبادلة بين الروبوتات ممثلة كعناصر ميكانيكية

4.2 التغذية الراجعة لشعاع حالة الروبوتات

لإيجاد موقع الروبوتات سوف يتم استخدام نظام التعريف، ArUco، حيث تتم طباعة عدة codes QR ولصقها على سطح كل روبوت. بعد القيام بعملية معايرة لبارامترات الكاميرا \mathbb{C} يمكن إيجاد المصفوفة \mathbb{C} المستخدمة في 23.

$$\mathbb{P} = \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} = \mathbb{C} \times \begin{pmatrix} r_x \\ r_y \\ r_z \\ 1 \end{pmatrix} = \mathbb{C} \times \mathbb{R} \quad (22)$$

حيث يمثل \mathbb{P} موضع الروبوت في إحداثيات حساس الكاميرا، و \mathbb{R} إحداثيات الروبوت في الإحداثيات الأرضية، و \mathbb{C} مصفوفة الكاميرا (Camera Matrix).

منها يمكن نقل كل بكسل من بكسلات الكاميرا إلى الإحداثيات الأرضية باستخدام pseudo-inverse \mathbb{C} . نذكر هنا أن المصفوفة \mathbb{C} :

$$\mathbb{C} = \mathbb{K} \times (\mathbb{R} | T) \quad (23)$$

حيث تمثل المصفوفة $(T | \mathbb{R})$ التحويل المتجانس الذي ينقل \mathbb{R} إلى إحداثيات الكاميرا، والمصفوفة \mathbb{K} تقام من بارامترات العدسة والحساس وتشكل كالتالي:

$$\mathbb{K} = \begin{pmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{pmatrix} \quad (24)$$

يمثل كل من f_x, f_y البعد البؤري للعدسة، و s معامل انحراف الحساس، و c_x, c_y يمثلان مركز الحساس بوحدات البكسل. نذكر هنا أنه من الواجب القيام بعملية معايرة تسبق مرحلة نقل كل نقطة من نقاط الكاميرا إلى الإحداثيات الواقعية وذلك لتقريب الكاميرا الموجودة أقرب ما يمكن إلى نموذج camera pinhole الموجود في المعادلات السابقة.

5.2 النموذج الرياضي العام للروبوت

النموذج الديناميكي للروبوت التقاضلي مدروس بشكل كامل مع استنتاجه في أعلى الكتب النصية الخاصة بأساسيات علم الروبوت. نكتفي هنا بإدراج معالة $\ddot{\mathbb{S}}$

$$\dot{\mathbb{S}} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \frac{(u_1+u_2)}{2} \cos \theta \\ \frac{(u_1+u_2)}{2} \sin \theta \\ \frac{(u_1-u_2)}{l} \end{pmatrix} \quad (25)$$

حيث يمثل u_1, u_2 السرعة الخطية للعجلات، و l المسافة بين العجلتين.

6.2 دراسة العجلات

تخدم العجلات وظيفتين رئيسيتين: الأولى هي دورها الأساسي لتحريك الروبوت، والثانية هي كأقراص مشقة لتسجيل حركة المحركات. نناقش في هذا القسم الأدوات الرياضية والشروط التصميمية المطلوبة لما سبق. العجلات للحركة عند اجراء التجارب المخبرية على سرعات المحركات الموجودة متبوعة بطلب سرعتها، وجدنا أن السرعة عند $7.5V$ (أي السرعة الأعظمية) كانت 135 دورة في الدقيقة. عند وضع شرط أن الروبوت يجب أن يكون قادر على قطع الحلبة شاقولياً أو أفقياً خلال مدة أقصاها عشر ثوان، وجدنا أن السرعة الخطية يجب أن تكون أكبر من $0.24 m.s^{-1}$. هذا جعل من السهل حساب أن نصف قطر الدوّلاب المطلوب يجب أن يكون أكبر من $1.7E - 2 m$. إن اختيار نصف قطر الدوّلاب أيضاً يخضع أيضاً لقيد آخر هو شقوق مسجلات الحركة.

العجلات كمسجلات حركة

من المعلوم أنه لإنشاء نظام تحكم دقيق يجب وجود نظام تحسس للزاوية وسرعة حركة المحركات. أول طريقة وأكثرها شيوعاً هي استخدام قرص مشقق بفتحات ذات ابعاد متساوية. يقوم مقطع ضوئي بمقاطعة المتحكم عند تسجيل مرور فتحة أمامه مما يسمح للمتحكم بتسجيل عبور هذه الفتحة، مما يؤدي لتسجيل تغير في الزاوية مقداره الزاوية المصممة بين الفتحتين⁶.

$$\vartheta = \frac{2\pi}{n} \quad (26)$$

حيث يمثل n عدد الفتحات الموجودة على القرص. نعرف ϕ بأنها الزاوية التي يمسحها المقطع الضوئي لعبور شق واحد. ومنه يمكن تعريف $\frac{\phi}{\vartheta} = \eta$ ، وهي نسبة عرض الشق إلى مجمل الزاوية⁷. يمكن مضاعفة الدقة عن طريق جعل وحدة المقاطعة الخارجية الموجودة في المتحكم AVR تقوم بتسجيل الحافة الصاعدة والهابطة في إشارة المقطع. فتصبح Δd (أصغر مسافة يمكن للمسجل قراءتها):

$$\Delta d = \frac{\vartheta}{2} r = \frac{\pi}{n} r \quad (27)$$

يسمح وجود مقطع ضوئي واحد بتسجيل تغيرات الزاوية دون معرفة اتجاه الدوران. لقراءة اتجاه الدوران يجب إضافة مقطع ضوئي ثان متواضع مع فرق طور مقداره $4/\eta$ بالنسبة للمقطع الأول. ان تحريك المقطع الضوئي الثاني بعد صحيح من مضاعفات η لن يغير من أداء عمله، لكن سيمكن المصمم من وضع كلًا للحساسين في مكان مريح للفك والتركيب السريعين، مما يتواافق مع أحد أهم شروط الهيكل الخارجي للروبوت. نسمي المسافة بين الحساسين η ، ونعتبر أن الحساس الأول موضوع في المبدأ.

بنقل هذه الشروط الى معادلات رياضية:

$$\vartheta = \frac{2\pi}{n} \quad (28)$$

$$\rightarrow \psi = \frac{\vartheta}{4} + k\vartheta = \left(\frac{\eta}{2} + k\right)\vartheta \quad (29)$$

حيث $k \in \mathbb{Z}$. إذا وضعنا شروط التالية:

- دقة تسجيل الحركة يجب أن تكون أصغر من الدقة التي نستطيع الحصول عليها من الكاميرا.
- الزاوية بين المقطعين (ψ) يجب أن تكون بين $\frac{\pi}{4}$ و $\frac{3\pi}{4}$.
- الزاوية بين المقطعين يجب أن تكون عدد صحيح من مضاعفات $5 \times \frac{\pi}{180}$ لتسهيل عملية التصميم.
- عرض الشق لا يقل عن $1.2E - 3m$.

تولد الشروط مسألة البرمجة الصحيحة الآتية:

إيجاد k, η, n, r حيث:

$$\psi = \frac{\phi}{2} + \|\vartheta = \left(\frac{\eta}{2} + \|\right)\vartheta \quad (30)$$

$$\Delta d < 2.5E - 3 \rightarrow \frac{r}{n} < 2 \times 3.979E - 4 \quad (31)$$

$$\psi = \text{II} \left(\frac{\pi}{180} \times 5 \right), \text{ where } \text{II} \in \mathbb{Z} \quad (32)$$

$$\frac{\pi}{4} \leq \psi \leq \frac{3\pi}{4} \quad (33)$$

تسمى المسألة السابقة بأنها مسألة برمجة صحيحة بسبب وجود شرط انتماء k الى مجموعة الأعداد الصحيحة. تم اثبات ان مسائل البرمجة الصحيحة هي مسائل NP-Hard ويحتاج حلها الى استخدام احدى تقنيات البحث مع القليل من الأمثلة. لن ننطرق الى استخدام هذه الطرق بسبب قلة عدد القيود على المسألة وصغر فضاء الحل، اذ يمكن عبور كل مجال الحل وتسجيل أفضل نقطة.

بعد حل المسألة السابقة يمكن استخلاص ما يلي:

$$\eta = 0.5 \quad (34)$$

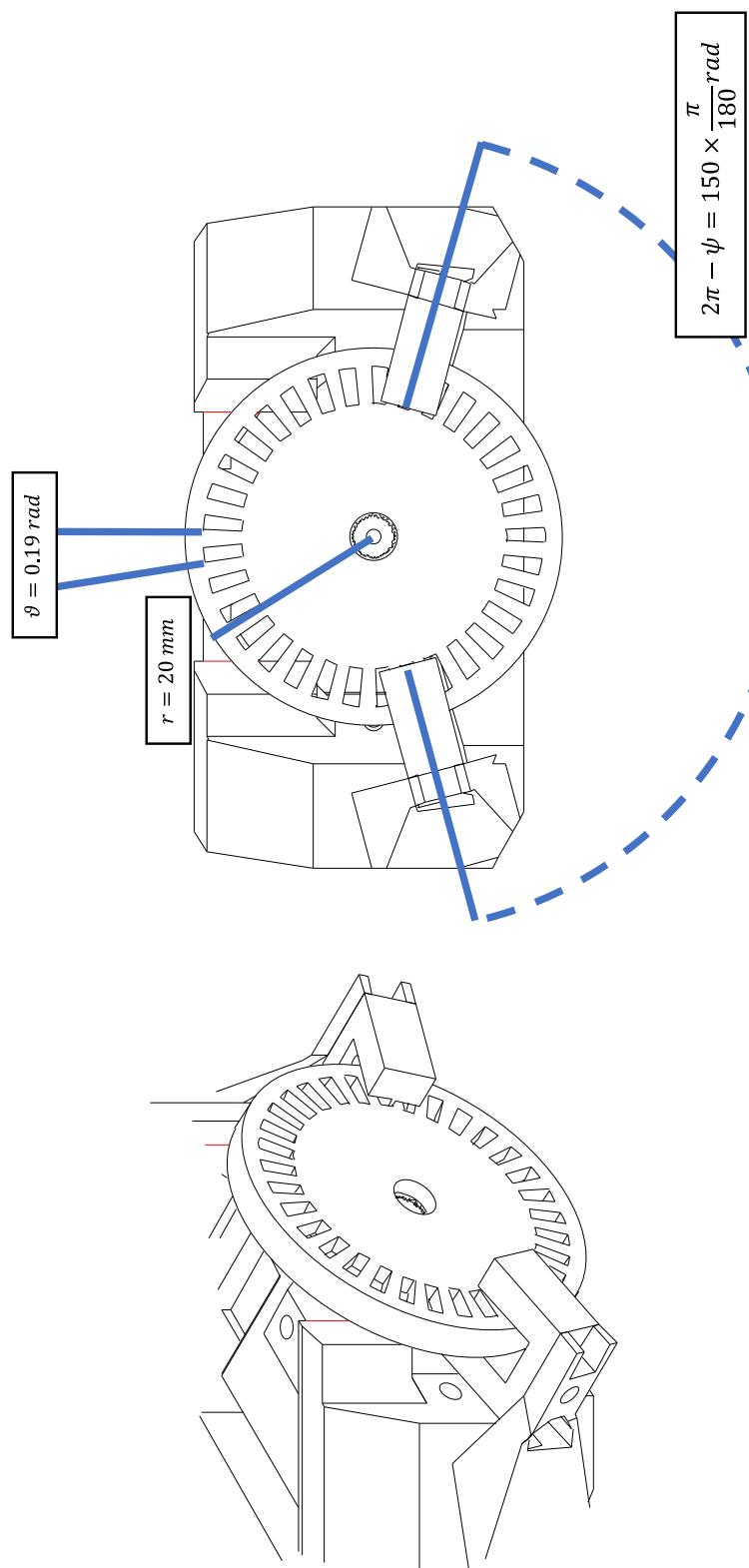
$$r = 2E - 2 \text{ } m \quad (35)$$

$$\psi = 210 \times \frac{\pi}{180} \text{ rad} \quad (36)$$

$$n = 33 \quad (37)$$

$$\Delta d = 1.9E - 3 \text{ } m \quad (38)$$

ومنه ينبع تصميم العجل الموضح في الشكل 4.2.



شكل 4.2: تصميم عجلات الروبوت الناتج

باب 3

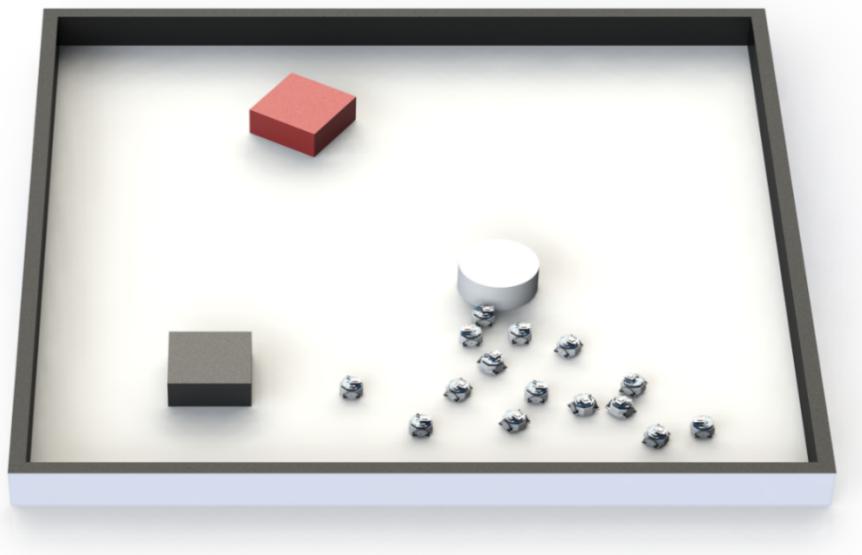
بنية النظام الصلبة والبرمجية

1.3 مقدمة

تتألف منصة العمل من: 1) فضاء عمل و 2) الروبوتات. يشرح هذا الفصل الأسس الرياضية والمعايير المتبعة في تصميم ما سبق. نبدأ بشرح فضاء العمل والشروط التصميمية التي يفرضها على تصميم الروبوتات، ثم ننتقل إلى عرض تصميم الروبوتات بشقيه الرياضي والحسوبي.

2.3 فضاء العمل

يعرف فضاء عمل أي نظام روبوتى بمجموعة النقاط التي يستطيع الروبوت التواجد فيها بشكل آمن. ذكر هنا أن فضاء العمل لا يكون بالضرورة معرف بأساس الاحداثيات المكانية، على سبيل المثال، يمكن تعريف فضاء عمل ذراع روبوت بدلالة قيم مفاصل هذه الذراع، ويكون عدد أبعاده بعدد درجات حرية الذراع. يستخدم فضاء العمل بشكل أساسى للبحث عن مسارات يستطيع الروبوت عبرها بغية انجاز عمل معين. قد يكون هذا العمل غير مقيد، كإيجاد مسار بين موضع الروبوت الحالى ونقطة هدف محددة مسبقاً، وقد يكون مقيد بشروط معينة كإيجاد أقصر مسار أو أقلها استهلاكاً للطاقة. يمكننا التعريف الجيد لفضاء العمل من إيجاد قيود المسألة التي يحاول الروبوت حلها بغية إيجاز المهمة الموكلة له، بمعنى أكثر دقة، تشكيل فضاء العمل هو أول خطوة من حل مسألة تخطيط المسار، التي تعتبر مسألة أمثلة، بين الشكل 1.3 فضاء عمل سرب الروبوتات. يتكون من مربع بضلع 4.2 متر وحواف مرتفعة. يحتوى المربع على عدة عوائق بأشكال مكعب أو أسطوانة وبأبعاد مختلفة. يمكن للعوائق أن تتحرك بالشكل المطلوب على الشبكة لتشكيل فضاء عمل ديناميكى.



شكل 1.3: إخراج ثلاثي الأبعاد لمنصة العمل

3.3 بنية النظام الصلبة

1.3.3 نظرة عامة

2.3.3 الأدوات

التغذية الراجعة

إن تحديد نظام التغذية الراجعة feedback كان الخطوة الأساسية لنا لوضع تصور أساسى للمشروع ولبدء عملية التصميم واختيار القطع الإلكترونية بحيث نحصل على أفضل أداء ممكن وفق تصميم مناسب وأسعار مقبولة. اعتمدنا في هذه المرحلة على نظام نقاط بين 1 و 5 - حيث يشير الرقم 5 إلى الأفضلية العليا - للمفاضلة بين الخيارات المتاحة وفق معاير محددة كما هو موضح في الجدول أدناه.

الجداول	الاحتياج للاختبار	فعالية التطبيق	سهولة التصميم	الحاجة	السعر	
24	5	3	5	4	4	3 Reckoner Dead
20	4	5	3	2	5	1 Sensor Mouse
25	5	3	5	2	5	5 Camera

بعد عملية المقارنة وبناءً على النتائج الموضحة أعلاه كان خيار استخراج موقع الروبوتات من كاميرا مثبتة فوق حلبة عمل الروبوتات هو الخيار الأفضل. أما عملية اختيار الكاميرا فكانت ترتكز على معيارين أساسين وهما الحصول على تقنية كاملة للمنصة وعدد الأطر التي يمكن الحصول عليها في الثانية الواحدة. لذلك قمنا باختيار كاميرا ستريو واسعة الرؤية التي تعد خياراً مناسباً لنا. حيث أنَّ أبعاد حساس كل كاميرا 960×2560 بكسل. يمكن بسهولة إيجاد أنَّ أصغر مسافة يمكن قياسها باستخدام الكاميرا عن طريق تقسيم عرض فضاء العمل (على دقة حساس الكاميرا و النتيجة هي 5.2 mm . بالإضافة إلى إمكانية الحصول على 16 إطار بالثانية وهو تردد كافٍ

لمتابعة سير الروبوتات وتحديد مواقعها.

المحرك

المعايير الأساسية التي تم اعتمادها لاختيار المحركات: طريقة قياس لموضع وطريقة قيادة الروبوت وقد اعتمدنا لتحديد ذلك النظام النقطي السابق الذي اعتمدناه لتحديد نظام التغذية الراجعة في الفقرة السابقة.

الحاجة للاختبار	فعالية التحكم	سهولة التصميم	الحاجة	السعر			
15	4	3	1	1	5	1	BLDC ODD
22	1	3	5	5	3	5	Servo Micro
13	3	2	1	1	4	2	DC Gear

كما نرى في الجدول السابق فإن محرك servo Micro كان الخيار الأكثر ملائمة ولاسيما مع سهولة استخدامه وتغليفه الذي سوف يسهل عملية التصميم بشكل كبير بالإضافة إلى سعره المقبول. ولكنه كان بحاجة لعملية اختبار للتأكد من مدى قدرتنا على التحكم به وعزمه عند جهد تغذية 8.4 فولت $1.3Kg.cm$ وسرعة الدوران الأعظمية 360 Continuous Degree. يبلغ وزن المotor 9 غرام وعزمه عند جهد تغذية 8.4 فولت 110 دورة في الدقيقة. لكن نتائج الاختبار لم تكن مرضية بشكل كامل، حيث بينت الاختبارات وجود منطقة ميتة في خرج الاكوادر الداخلي للمotor dead band. لكن الخيارات الأخرى كانت غير مناسبة من ناحية التوفير؛ لذلك لجأنا إلى حل آخر، وهو فصل محرك التيار المستمر الموجود داخل المotor SG90 عن دارة قيادته ووصله مع المقاومة المتغيرة الداخلية إلى دراتنا السفلية و وبالتالي يمكن استخدام محركات تيار مستمر بحجم مناسب وضمن تغليف مناسب مما سهل علينا عملية تصميم الجسم وتصغير حجمه بالإضافة إلى السعر المناسب. كما أثنا سوف نقوم باستخدام العجلات كمسجلات للحركة من أجل تحكم أكثر دقة.

قيادة المحرك

بالنسبة لقيادة محركات التيار المستمر، فتم ذلك باستخدام الجسر L298. يعني الجسر من عدة مشاكل أهمها قدم تكنولوجيا TTL المستخدمة في صناعته، وأيضاً كبر حجمه. مما فرض قيوداً في الحجم والشكل على تصميم الدارات لذلك استخدمنا Dual – L298N IC وسنو لاحقاً كيف أن تصميم الدارات وتجميعها مكّناً من تنفيذ الشروط المطلوبة للتصميم ضمن القيود المفروضة عليه.

3.3.3 وحدة التحكم

لتغطية كافة احتياجات النظام الإلكتروني تم وضع عدة نقاط رئيسية يجب على النظام المدمج في الروبوت شملها:

- اتصال آمن وبتأخير (latency) منخفضة بين الروبوت والنظام المركزي.
- القدرة على قيادة محركي DC بجهد أقصى 5.7 V وتيار 5.0 A أقصى.
- عرض نبضة بتردد متوافق من الثابت الزمني للمحرك.
- نظام تحديد سرعة وموضع كل من محركي التيار المستمر.
- نظام تغذية.

- وحدات ادخال وإخراج بدائية بغية debugging وتبديل البرام捷ات الداخلية on-the-run.

اعتمدنا على سلسلة متحكمات الا 8 bit الخاصة بشركة Atmel ذات Instruction set RISC يوجد في هذه المتحكمات 35 تعلیمة بالإضافة لعدة وحدات توقيت ومقاطعات داخلية وخارجية تجعل المهمة المطلوبة ممكناً. ولقد اخترنا المتحكم Atmega328p tqfp.

يمتلك المتحكم Atmega328p الموصفات الآتية:

- Two 8-bit Timer/Counter
- One 16-bit Timer/Counter
- UART Connection Unit
- Two External Interrupt Unit
- I2C Unit
- Three 8-bit I/O units
- One ADC Unit with MUX to 8-bit I/O Unit

يضمّن وجود ثلاثة مؤقتات/عدادات قدرة النظام على جدولة المهام حسب أولوياتها وحسب الوقت المطلوب لها. كما أنها توفر العتاد المطلوب لتشكيل إشارة بعرض نبضة متغير PWM تستخدم لقيادة محركات التيار المستمر. (يتم تشكيلها باستخدام hardware). توفر وحدة UART الاتصال المباشر بين وحدة الاتصال وبين المتحكم وبشكل سهل و مباشر وآمن من الأخطاء. أما بالنسبة للمقاطعات الخارجية: فهي تسمح للمتحكم بتسجيل الحركة التي تسجلها المقاطعات الضوئية المستخدمة في قراءة دوران المحرك.

4.3.3 التغذية

باختلاف التصورات المتحملة لسرب الروبوتات، فإن كل روبوت ضمن السرب يحتاج إلى إمداده بطاقة لإكمال المهمة الموكلة له. ولقد قمنا باختبار بطاريات Lithium-Ion القابلة لإعادة الشحن؛ حيث يمكن إعادة شحن هذا النوع من البطاريات مئات المرات مع الحفاظ على ثباتها وأدائها مقارنةً بغيرها من البطاريات. تمثل بطاريات Lithium-Ion إلى أن تكون ذات كثافة طاقة أعلى، معدل تفريغ ذاتي أقل من البطاريات الأخرى القابلة لإعادة الشحن؛ مما يؤدي إلى تحسين كفاءة الطاقة حيث تتمتع الخلية الواحدة باحتفاظ بالشحن لفترة أطول من أنواع البطاريات الأخرى. أما بالنسبة للشكل: فقد اخترنا بطاريات بأبعاد (cm 5× cm 4 mm× 4) لكي تتناسب مع التصميم. ونقدم هذه البطاريات جهدًا مقداره 7.3 V وتتوافق عليها دارة حماية.

إن استخدام مصدر تغذية قابل لإعادة الشحن يفرض علينا استخدام وحدة شحن وتغريغ. وقد استخدمنا وحدات TP4056A المستخدمة لشحن وتغريغ بطاريات الليثيوم أيون. ونظراً إلى أنّ البطاريات المستخدمة تقدم جهد 7.3 V؛ احتجنا إلى استخدام وحدة رافع جهد DC-DC convertor. إن دخل دارة رافع الجهد يتراوح بين 3 إلى 15 فولت، بينما يكون الخرج من 4 إلى 35 فولت، وبينة ضبطها لاعطاء الخرج المطلوب بشكل يدوى.

وحدات مطرزة		وحدات بدائية					
Wemos-D1-Mini	NodeMcu-Lua-WIFI-Board	NodeMcu-DevKit-V1.0	ESP-12e	ESP-01			
11	11	11	11	2	GPIO	المداخل والمخارج للأغراض العامة	
1	1	1	1	-	ADC	م Howell تمايزي رقمي	
مطبوعة على الدارة		مطبوعة على الدارة		Antenna		هوائي	
مطبوعة على الدارة		مطبوعة على الدارة		USB-to-Serial		إمكانية الوصول المباشر	
ممكن	ممكن	غير ممكن	غير ممكن			عامل التشكيل	
متوسط	كبير	صغير	متوسط	Form		وحدة لا ESP8266	
ESP12E	ESP12E	ESP12E	-	Module ESP8266		Serial	
CH340G	CP2102	CH340G	-			شريحة الوصول المباشر	
يوجد	يوجد	يوجد	يوجد	لا يوجد	RF-shield	طاء لحجب الترددات الراديوية	

جدول 1.3: جدول مقارنة بين وحدات الاتصال المختلفة

5.3.3 نظام الاتصال

بعد قراءة الدراسات المرجعية واستشارة المختصين تم اعتماد تقنية الواي فاي لربط الروبوتات مع بعضها ومع الحاسوب المركزي. ولاختيار وحدة الواي فاي المناسبة.

وتم التقييم وفق معايير معينة وهي الوثوقية، الحجم، التوفيرية، السعر، وامكانية الوصول المباشر (USB-to-Serial) . بناءً على هذه المعايير والجدول السابق قمنا باختيار وحدة الاتصال Wemos ؛ حيث أنها تحقق ذات وثائقية عالية مقارنة بساعتها وحجمها مناسب بالإضافة إلى توافرها.

6.3.3 الدارة الالكترونية المطبوعة PCB

يتم قيادة محرك التيار المستمر باستخدام الجسر L298. يعاني الجسر من عدة مشاكل أهمها قدم تكنولوجيا TTL المستخدمة في صناعته، وأيضاً كبر حجمه. مما فرض قيوداً في الحجم والشكل على تصميم الدارات. سوف نعرض في الفقرتين التاليتين تجميعة الدارات التي مكنتنا من تنفيذ الشروط السابقة ضمن قيود الحجم والشكل المفروضة عليه. حيث سنقوم في القسم الأول بشرح عملية تصميم أمّا في القسم الثاني سوف نقوم باستعراض عملية التصنيع.

التصميم باستخدام الحاسوب

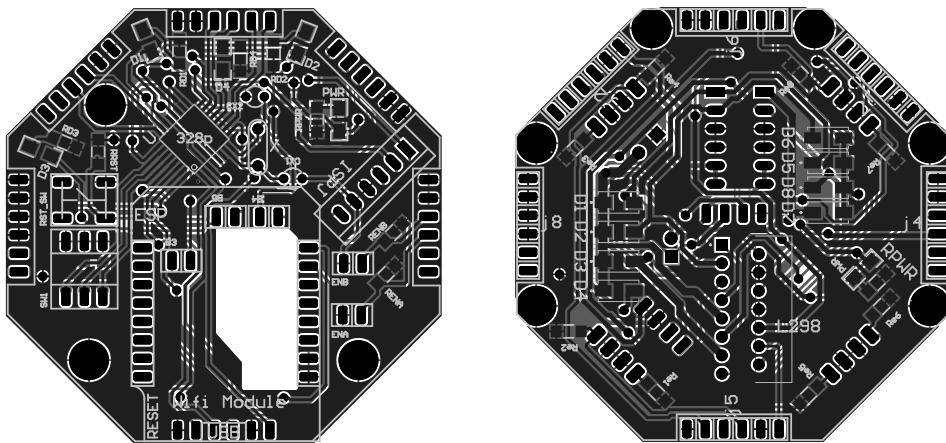
لقد استخدمنا البرنامج التصميمي Altium لتصميم الطبقة العلوية والسفلى. حيث أن الطبقة العلوية تحتوي على وحدة الاتصال والمحكم الصغرى ووحدات الادخال والإخراج، بينما يحتوي بضم القسم السفلي جسر القيادة، وحدات التغذية وشحن المدخلة، ودارات المقطعات الضوئية.

الطبقة العلوية: طبقة المعالجة والإشارة

هذه الطبقة مسؤولة عن أي إجراء على الروبوت القيام به حسابياً أو لتلقي أوامر الحركة من حاسوب مركزي. تحوي الطبقة متحكمين اثنين: الأول بمعمارية AVR RISC bit 8 والثاني bit 32. يوفر المعالج الأول الوسائل المنطقية والتوقيقية للتحكم بالمحركات بدقة وكذلك لإجراء حسابات المعادلات الفرقوقية بدور تقطيع بالغ الدقة، أي للقيادة منخفضة المستوى. أما المحكم الثاني فيوفر آلية اتصال ببروتوكول Wi-Fi بين روبوتات السرب وبين القاعدة، وكذلك بعض الحسابات عالية المستوى كتقطيع المسار وموائمة التراسل بين الروبوت والهاسوب المركزي وملاحقة حركة الروبوت. يوجد المنتج النهائي للدارة في الشكل 2.3

الطبقة السفلية: طبقة التضخيم والإشارة

إن العزل الحاصل بين كل من التضخيم والإشارة يضمن إمكانية ملاحة الأخطاء وعزل السبب إن كان كهربائي أو الكتروني-برمجي. تضمن هذه الطبقة تحسس ومراقبة كل من موضع وتيار المحركات باستخدام مفاهيم كهربائية بسيطة، وكذلك توفر إمكانية الرابط الكهربائي بين كل من داري الشحن ورفع الجهد من جهة، وبين المدخلة من جهة ثانية. إن استخدام الجسر L298 لم يكن الخيار الأقرب كهربائياً، خصوصاً مع هذا الحمل المنخفض نسبياً. لكن عدم توفر بديل مناسب ضمن شروط الحجم الموضوعة مسبقاً جعله الخيار السليم. يوجد على الدارة أيضاً ديوانات حماية كون المحرك المستخدم من النوع المستمر. لمراجعة الملفات التصميمية انظر الشكل 2.3



شكل 2.3: التصميم باستخدام الحاسوب لكل من الطبقتين العلوية والسفلية

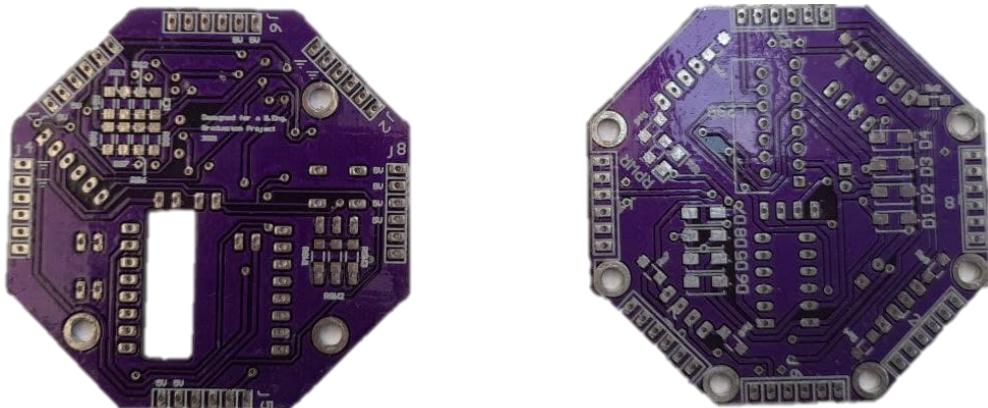
عملية تصنيع الدارات 7.3.3

بعد عملية تصميم الدارات انتقلنا إلى مرحلة التصنيع، حيث أن كل طبقة (السفليّة والعلوّيّة) عبارة عن دارة ثنائية الجوانب أي كل أنّ منها تتّألف من طبقي نحاس مطليتين بمادة الفاير غلاس بسمكها (FR4) 1.8mm. ليأتي بعد ذلك غطاء اللّحام وهي التي تمنح الدارة لونها الأرجوانيّ (أو ألوان آخر)، حيث يتمّ وضع هذه الطبقة على طبقة النّحاس لعزل المسارات التّحاصية من الاتصال الغير مقصود مع أي معدن أو لحام أو أطراف موصلة أخرى وهذه الطبقة تساعد في وضع اللّحام في مكانه الصحيح وتحجب اللّحام الخاطئ مما سهل عملية لحام القطع الإلكترونيّة السطحية (SMD) يلي طبقة غطاء اللّحام طبقة بيضاء من الحرير تدعى الشاشة الحريريّة (Silkscreen)، وتستخدم هذه الطبقة في إضافة الأحرف والرموز والأرقام إلى ألواح PCB مما يسهل عملية تجميع المكوّنات على الألواح وفهمها بشكل أفضل. يوضح الشكل أدناه الدارات بعد عملية التصنيع.

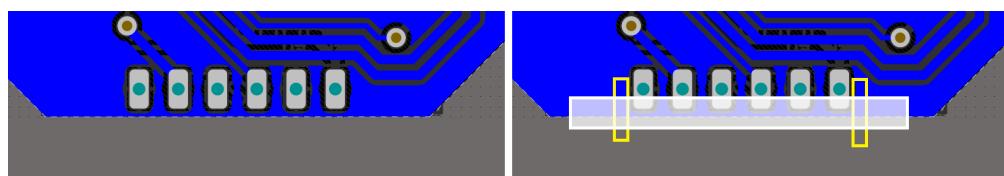
اختبار المنتج 8.3.3

بعد عملية التصنيع قمنا بعملية فحص للدّارات للتأكد من التوصيلات باستخدام الأدوات المختلفة. ووجدنا بعد عملية الفحص الأولى حدوث قصر في الدّارة ناتج عن انقطاع في الأرضي نتيجة القص الزائد عند الأطراف، حيث أن أرضي الدّارة يتواجد في الأطراف والصورة أدناه توضح منطقة حدوث الانقطاع.

لم يقتصر حدوث الانقطاع على منطقة الأرضي بل كان هناك انقطاعات في المناطق الموصولة عند الانتقال من جزء واسع إلى جزء ضيق (*necks*). بعد ذلك قمنا بإحصاء أماكن حدوث هذه المشكلة ووجدنا وجود انقطاعين في الطبقة السفلية وانقطاعين في الطبقة



شكل 3.3: المنتج النهائي بعد تصنيع الدارة



شكل 4.3: العطب الناتج عن التصنيع

العليا وقمنا بوصول جميع المناطق المقطوعة بأسلاك خارجية.

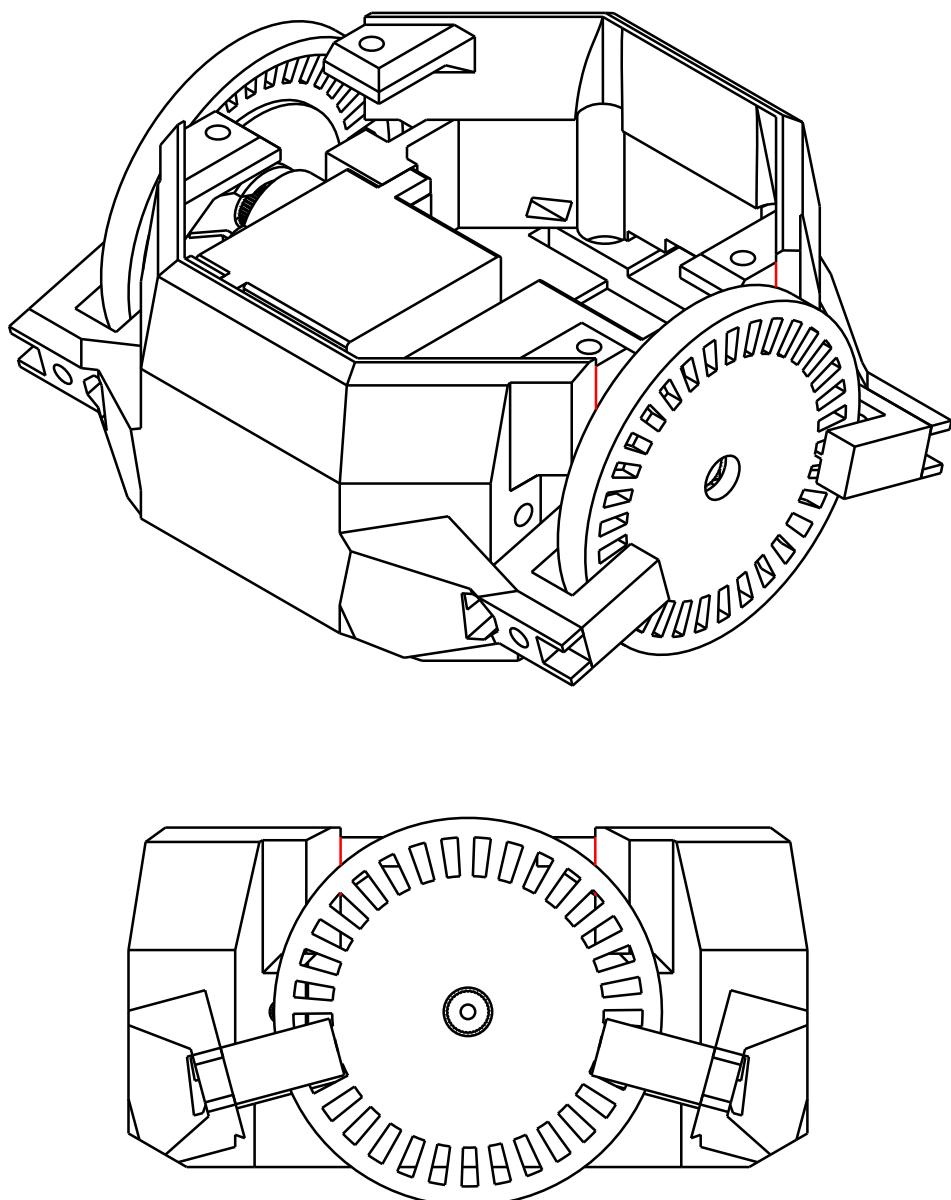
9.3.3 الجسم الخارجي والتحريك

تم تصميم جسم الروبوت باستخدام برنامج SolidWorks واعتماد تصميم بسيط صغير الحجم بأبعاد $6 \times 7 \times 6$ بـشكل مثمن. ثبتت البطارية من الجهة السفلية للجسم ويوجد أماكن مخصصة لثبت المحركات والعجلات والمقاطعات الضوئية وثبت الدرatinين فوق الجسم باستخدام أربع براغي. كما هو موضح في الشكل

تم تصنيع الجسم باستخدام تقنية الطباعة ثلاثية الأبعاد من مادة PLA مع Acid Polylactic (PLA) خصائص طباعة تتلخص في ارتفاع الطبقة 0.16 ملم وسمك الجدران 0.80 ملم. بعد تجربة ثلاثة ورشات متوفرة وسهلة الوصول جغرافيًا، توصلنا إلى أفضل طباعة بشكل يضمن الجودة المطلوبة ولاسيما في تصنيع العجلات حيث أن أي اختلاف أو عدم انتظام في الثقوب سيؤثر مباشرة على قيم التغذية الراجعة التي سيعتمدتها الروبوت لتتبع المسار الناتج. توضح الصورة 6.3 مقارنة بين عجلة مثالية وعجلة ذات جودة تصنيع منخفضة. لضمان حركة مستقرة للروبوت على الأرض ونظرًا لوزن وزن الروبوت، قمنا باختيار مادة مناسبة من الكاوتشوك لإحاطة إطارات كل العجلات وزيادة الاحتكاك وبعد التجربة تم تحقيق الغاية المرجوة حيث أن انزلاق العجلات على الأرض كان شبه معادوم. سنعرض النتائج النهائية لتركيب الروبوتات في فصل النتائج.

بالنسبة دور العجلات كمسجلات حركة فكان يجب اختبار دقة الطباعة ثلاثية الأبعاد في إخراج الشفوق الموجودة على العجلات، ولاختبار ذلك قمنا بتحريك العجلات بأقصى سرعة يمكن للروبوت السير بها فكانت استجابة المقاطعات الضوئية لعبور الشفوق عبرها كالتالي (الشكل 7.3):

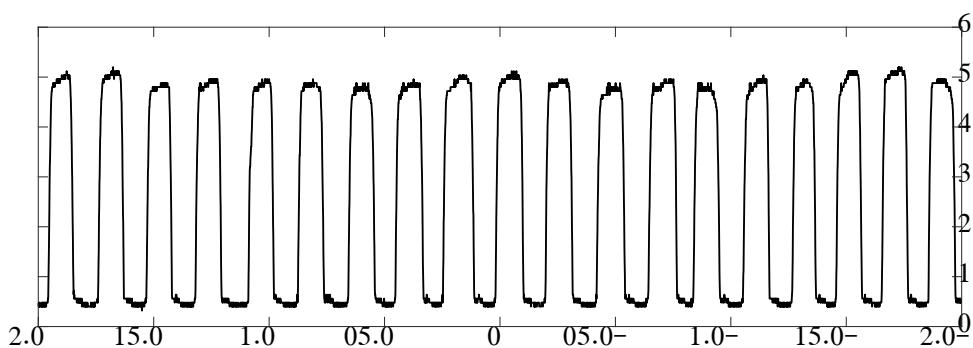
أما الخطوة التالية فكانت اختبار استجابة AVR للمقاطعات الضوئية، وكما نلاحظ من قراءة الاستجابة الموضحة في الشكل 9.3



شكل 5.3: الهيكل الخارجي للروبوت



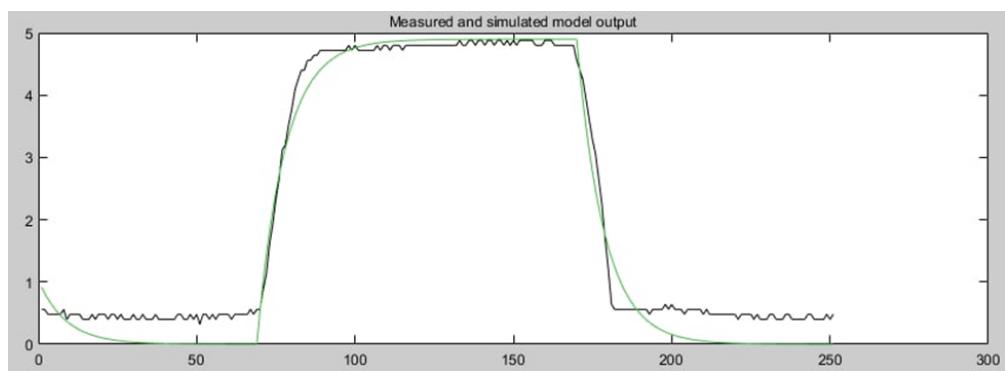
شكل 6.3: مثال يوضح مقارنة بين سوء وجودة تصنيع العجلات



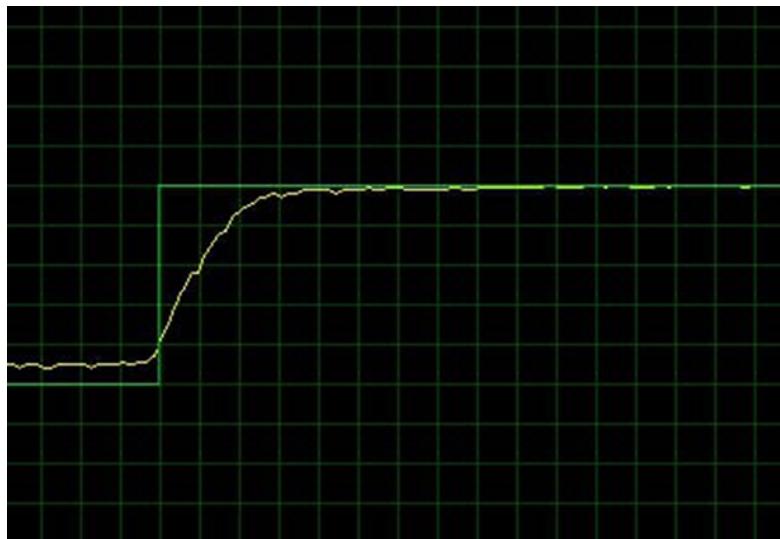
شكل 7.3: استجابة المقاطعات الصوتية

وجود تأخير بسيط في استجابة AVR.

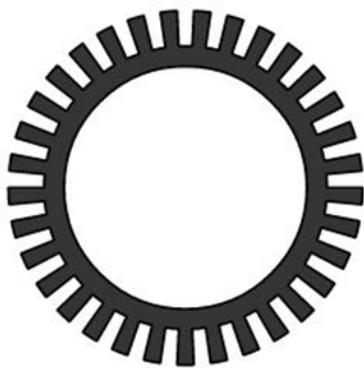
ولتحسين قراءة المقاطعات الصوتية قمنا بتصميم encoder وتصنيعه للحصول على حواف شفوق حادة ومن ثم إطباقه على شفوق العجلات للحصول على أفضل استجابة ممكنة.



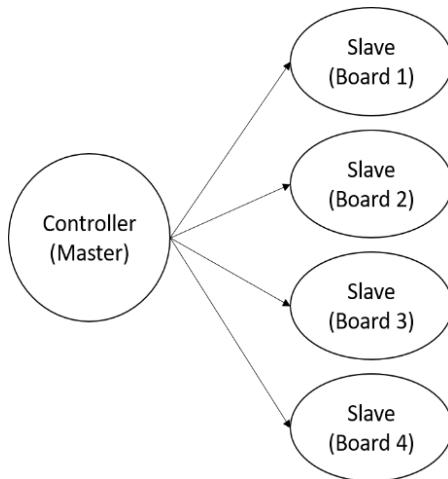
شكل 8.3: التقرير إلى نظام درجة أولى



شكل 9.3: محاكاة استجابة الـ AVR للمقاطعة الصوتية



شكل 10.3: طبقة بلاستيكية لتحسين قراءة المقطعات



شكل 11.3

10.3.3 نظام الاتصال

طوبولوجيا الشبكة

على مستوى طبقة الـ Data Link، قمنا في البداية بالعمل على بروتوكول ESP-NOW المطور من قبل Espressif، وإن الاتصال اعتمدًا على هذا البروتوكول هو اتصال موثوق TCP من دون الحاجة إلى عمليات المصادقة. نظرًا لأهمية موثوقية الاتصال فقد قمنا باختبار البروتوكول many to one ESP-NOW على خمس عقد، حيث تكون أحد العقد هي الـ Master والبقية Slaves كما هو موضح في الشكل 11.3.

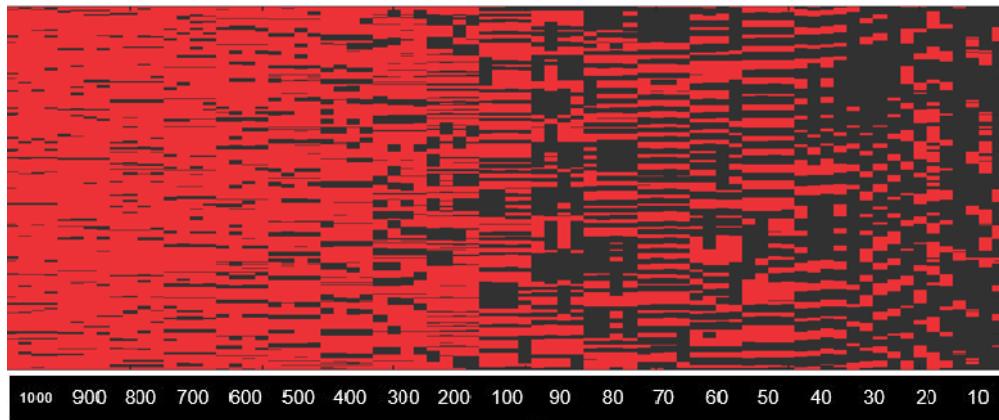
يُث قمنا في البداية بتعريف عنوان الـ MAC الخاص بكل شريحة اتصال، ومن ثم إرسال إحداثيين x و y مولدين عشوائيًّا لجميع العقد ألف مرة وإعادة التجربة من أجل أ زمنة تأخير مختلفة كما هو موضح في الشكل 12.3، حيث يمثل المحور الأفقي أ زمن التأخير من أجل الشرائح الأربع، أما المحور الشاقولي فهو عبارة عن الرسائل الأربع المرسلة مرتبة من الرسالة الأولى في الأسفل إلى الرسالة الأخيرة في الأعلى. يُعبر اللون الأحمر عن وصول الرسالة فيما يشير الأسود إلى فشل تسلیم الرسالة.

11.3.3 بروتوكول MQTT

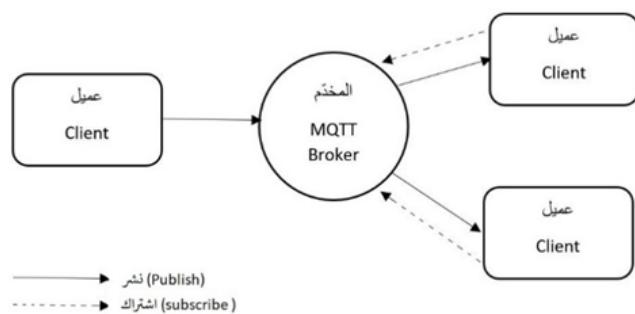
هو بروتوكول نقل رسائل بين مخدم زبون Client/Server بنمط نشر/اشتراك publish/subscribe قائم على الموضوع (Topic) لأنها ذات تطبيق مستقر لا ESP8266Mosquitto (استخدام مخدم MQTT) [18]. وهنا يجب التنويه لوجود ثلاث أنماط أساسية الاشتراك: النمط القائم على الموضوع والنمط القائم على المحتوى والنمط القائم على النوع.

إن بروتوكول MQTT (الموضح في الشكل 13.3) يتكون من ثلاثة أجزاء أساسية: المخدم، الناشر، والمشترك. يُعد المخدم مسؤولاً عن إدارة شبكة من العملاء والذين هم عبارة عن مزيج من الناشرين والمشتركين؛ حيث أن الناشر هو الجهاز الذي يقوم بإرسال الرسائل إلى المخدم بحيث تكون هذه الرسائل معنونة بعنوان موضوع (topic) بينما المشترك هو الجهاز الذي يستمع لموضوع أو مواضيع معينة. يمكن تشبيه الموضوع بوسم خاص بكل رسالة، ويمكن أن يكون وحيد الطبقة أو متعدد الطبقات ويسمح هذا ب التقسيم المنطقي حيث للمعطيات على كل جهاز يرغب باستقبال موضوع معين أن يشتراك به وذلك بإخبار المخدم.

ليس هناك اتصال مباشر بين المشترك والنادر وإنما وببساطة يقوم المشترك بإخبار المخدم أنه مهتم بموضوع محددة وبعد ذلك يقوم



شكل 12.3



شكل 13.3: شكل توضيحي للأجزاء الأساسية في بروتوكول نقل الرسائل MQTT

المخدم بإرسال الرسائل إلى المشتركين عند توافرها. إن نمط نشر/اشتراك مختلف تماماً عن نمط طلب/رد (request/response) المتبع في بروتوكول الا HTTP، كما أن بروتوكول MQTT يعمل بالاعتماد على TCP/IP وعلى خلاف الا HTTP فهو بروتوكول Binary وليس ASCII، وهذه إحدى ميزاته فكما نعلم أن بروتوكولات الا binary تسهل حمولة نقل بيانات أقل من الشبكة. بالإضافة إلى ذلك هناك خيارات في بروتوكول الا MQTT تتعلق بتوصيل الرسالة بين المخدم والعميل يطلق عليها QoS (Quality of Service)، وهناك ثلث خيارات بخصوص هذا الشأن:

- مستوى 0 (QoS = 0): توصيل الرسالة مرة واحدة في أفضل حالة، أي أنه في حال لم يستلم المشترك الرسالة فإن المخدم لن يقوم بإرسالها مرة أخرى حيث لا يتوقع المخدم من المشترك تأكيد وصول الرسالة (Acknowledgment).
- مستوى 1 (QoS = 1): توصيل الرسالة مرة واحدة على الأقل. هذا يعني أن المخدم يستمر بإرسال الرسالة إلى أن يقوم المشترك باستلام الرسالة ويرسل تأكيد باستلامها، وهذا يؤدي إلى إمكانية وصول أكثر من رسالة واحدة إلى نفس المشترك.
- مستوى 2 (QoS=2): توصيل الرسالة مرة واحدة لزوماً، وهذا يعني أن الرسالة يجب أن تصل لمرة واحدة دون تكرار المشترك. وفي حال عدم اتصال المشترك أساساً فيمكننا ضبط الإعدادات بحيث يبدأ الاتصال بجلاسة اتصال نظيفة؛ حيث يقوم المخدم بتجاهل الرسائل القديمة ويبدا جلسة جديدة. بينما في حال تعطيل هذه الخاصية فإن الرسائل (من المستويين 0 و 1) التي تم إرسالها إلى أحد المشتركين أثناء انقطاعه عن الشبكة سوف تترافق وسيقوم المخدم بإرسالها له فور اتصاله بالشبكة.

4.3 بنية النظام البرمجية

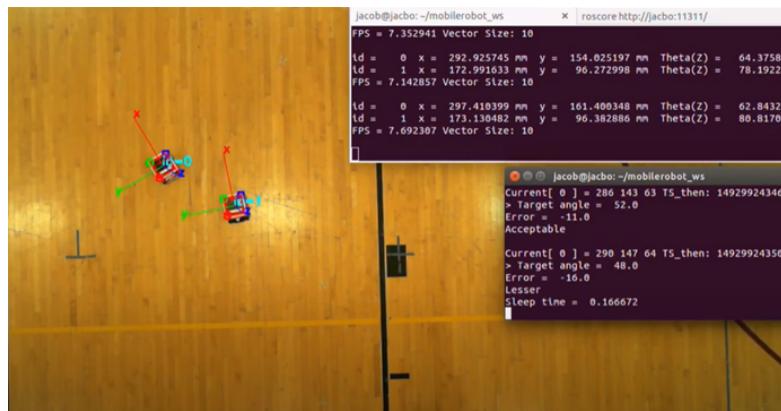
1.4.3 نظرة عامة

كما أشرنا سابقاً فإن عملية التحكم مرکزة؛ حيث يوجد نظام تشغيل ROS على الحاسوب المركزي. يوجد على نظام ROS ثلاثة عقد أساسية. العقدة الأولى : تتصل مع الكاميرا المتباينة أعلى المنصة، حيث تقوم الكاميرا بإرسال الأطر لها لتقوم باستخلاص معلومات عن الخريطة وموقع الروبوتات. ولتحقيق القراءة الدقيقة لموقع الروبوتات في الزمن الحقيقي على الحاسوب المركزي سنعتمد مكتبة ArUco وهي مكتبة تستخدم في تطبيقات الواقع المعزز وتعتمد على مكتبة OpenCV وتساعد في عملية معایرة الكاميرا. سيتم تثبيت علامات QR-markers على الروبوتات ومنه تقوم عقدة الا ArUCo في الا ROS تقوم باكتشاف وتحديد موقع العلامات وترسلها للـ . ROS-TOPIC

العقدة الثانية: تقوم هذه العقدة باستقبال إحداثيات الروبوتات والعوائق من العقدة الأولى عن طريق اشتراكها بالـ TOPIC ROS ذاته. بعد ذلك تتم على هذه العقدة عملية توليد المسار الأمثل وفق الخوارزمية المختارة وإرسال نقاط المسارات الكاملة لجميع الروبوتات إلى العقدة الثالثة. العقدة الثالثة: بعد استقبال هذه العقدة لنقطات المسارات الكاملة لجميع الروبوتات، تقوم بإرسال كل مسار إلى الا خاص به (إلى الروبوت الخاص به المشترك بنفس الا Topic) عن طريق مخدم الا mqtt لتكون هذه العقدة بمثابة عقدة وصل بين نظام ROS و مخدم بروتوكول الا mqtt. وبعد أن يستقبل كل روبوت الرسائل الخاصة به تكرر هذه العملية بشكل دوري، ثم يقوم كل روبوت بتتبع المسار وفق خوارزمية تصحيح PID.

2.4.3 البنية البرمجية للروبوت منخفضة المستوى

كتب برنامج تشغيل الروبوت باستخدام لغة C لغبيها من العتاد الصلب ولدعمها من قبل منظومة GCC. كون المعالج بمعمارية 8 بت، هناك العديد من القيود على سرعة تنفيذ العمليات الحسابية عليه، فكلفة حساب متتحول صحيح 32 بت أضعاف كلفة حساب ذات الرقم



شكل 14.3: صورة توضيحية لآلية عمل محددات ArUco

إذا كان مخزنًا في متحول 8 بت. لا تواجه المعالجات الحاسوبية ذات معمارية 64 بت هذه المشاكل بشكل عام. يمكن القول بأن نقطة الضعف الرئيسية لهذه المعالجات تكمن في عرض مسجلاتها. حيث تميز بطيف واسع من المزايا المناسبة بشكل مطلق للتطبيق الحالي.

صمم نظام تشغيل الروبوت بطريقة البرمجة التابعة لخلو لغة C من الأغراض والصفوف. نجد في الملحق توصيف شامل لكل أجزاء الكود البرمجي.

يتبع الروبوت المسار الواسط إلى عن طريق مصحح متقطع PID. تتكون حلقة الحساب من مصححين: الأول يتحكم بسرعة المحركات لعدم المسافة بين مركز الروبوت والنقطة الهدف، والثاني يقوم بعدم الزاوية بي شعاع سرعة الروبوت والشعاع الواسط بين مركز الروبوت والنقطة الهدف. عمل هذين المصححين ينتج مسارات مناسبة بين النقاط المتتابعة.

باب 4

النتائج

بعد برمجة كل طرق توليد المسار المذكورة في الملف سيتم إجراء ما تبقى من تجارب تجريبية خلال الأسبوع القادم، حيث يجب إجراء جميع التجارب في ظروف عمل أقرب إلى المثالية وتتضمن التطابق بين التكرارات. يتم حالياً تحضير النتائج لكتابة الدراستين التاليتين:

- الدراسة الأولى عن تصميم وتنفيذ منصة سرب روبوتات لأغراض بحثية
- الدراسة الثانية عن مقارنة بين طريقة تخطيط المسار المقترحة والطرق الشهيرة في الدراسات المرجعية باستخدام المنصة المنجزة

Bibliography

- [1] Yogeswaran M, S G P. In: Swarm Robotics: An Extensive Research Review; 2010. .
- [2] Canny J, Reif J. New lower bound techniques for robot motion planning problems. In: 28th Annual Symposium on Foundations of Computer Science (sfcs 1987); 1987. p. 49-60.
- [3] Canny JF. The Complexity of Robot Motion Planning. Cambridge, MA, USA: MIT Press; 1988.
- [4] Farinelli A, Iocchi L, Nardi D. Multirobot systems: a classification focused on coordination [Journal Article]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). 2004;34(5):2015-28.
- [5] Burgard W, Moors M, Stachniss C, Schneider FE. Coordinated multi-robot exploration [Journal Article]. Trans Rob. 2005;21(3):376□386. Available from: <https://doi.org/10.1109/TR0.2004.839232>.
- [6] Konur S, Dixon C, Fisher M. Analysing robot swarm behaviour via probabilistic model checking [Journal Article]. Robotics and Autonomous Systems. 2012;60(2):199-213.
- [7] Liu W, Winfield AFT, Sa J, Chen J, Dou L. Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots [Journal Article]. Adaptive Behavior. 2007;15(3):289-305. Available from: <https://journals.sagepub.com/doi/abs/10.1177/1059712307082088>.
- [8] Khatib O. Real-time obstacle avoidance for manipulators and mobile robots. In: Proceedings. 1985 IEEE International Conference on Robotics and Automation. vol. 2;. p. 500-5.
- [9] Saska M, Macas M, Preucil L, Lhotska L. Robot Path Planning using Particle Swarm Optimization of Ferguson Splines. In: 2006 IEEE Conference on Emerging Technologies and Factory Automation;; p. 833-9.
- [10] Sullivan J, Waydo S, Campbell M. In: Using Stream Functions for Complex Behavior and Path Generation;. Available from: <https://arc.aiaa.org/doi/abs/10.2514/6.2003-5800>.

- [11] Wang H, Lyu W, Yao P, Liang X, Liu C. Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system [Journal Article]. Chinese Journal of Aeronautics. 2015;28(1):229-39. Available from: <https://www.sciencedirect.com/science/article/pii/S1000936114002301>.
- [12]
- [13] Bae H, Kim G, Kim J, Qian D, Lee S. Multi-Robot Path Planning Method Using Reinforcement Learning. Applied Sciences. 2019;9(15). Available from: <https://www.mdpi.com/2076-3417/9/15/3057>.
- [14] Das PK, Behera HS, Panigrahi BK. Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity. Engineering Science and Technology, an International Journal. 2016;19(1):651-69. Available from: <https://www.sciencedirect.com/science/article/pii/S2215098615001548>.
- [15] Lin HY, Huang YC. Collaborative Complete Coverage and Path Planning for Multi-Robot Exploration. Sensors. 2021;21(11). Available from: <https://www.mdpi.com/1424-8220/21/11/3709>.
- [16] You W, Yin Y, Qian X, Zhang X. A Path Planning Algorithm Based on Fluid Simulation [Journal Article]. 2017 10th International Symposium on Computational Intelligence and Design (ISCID). 2017;1:502-6.
- [17] Sabry Hassouna M, Abdel-Hakim AE, Farag AA. PDE-based robust robotic navigation [Journal Article]. Image and Vision Computing. 2009;27(1):10-8. Available from: <https://www.sciencedirect.com/science/article/pii/S0262885607000790>.
- [18] ;..
- [19] Hadidi R, Cao J, Woodward M, Ryoo MS, Kim H. Distributed perception by collaborative robots [Journal Article]. IEEE Robotics and Automation Letters. 2018;3(4):3709-16.
- [20] Arjadi RH, Candra H, Prananto HD, Wijanarko TAW. RSSI Comparison of ESP8266 Modules. In: 2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS). IEEE; p. 150-3.
- [21] Anoop AS, Kanakasabapathy P. Review on swarm robotics platforms. In: 2017 International Conference on Technological Advancements in Power and Energy (TAP Energy); 2017. p. 1-6.
- [22] Arvin F, Samsudin K, Ramli AR. Development of a miniature robot for swarm robotic application. International Journal of Computer and Electrical Engineering. 2009;1(4):422-36.

- [23] Rubenstein M, Ahler C, Nagpal R. Kilobot: A low cost scalable robot system for collective behaviors. In: 2012 IEEE International Conference on Robotics and Automation; 2012. p. 3293-8.
- [24] McLurkin J, McMullen A, Robbins N, Habibi G, Becker A, Chou A, et al. A robot system design for low-cost multi-robot manipulation. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2014:912-8.
- [25] Dorigo M. SWARM-BOT: an experiment in swarm robotics. In: Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.; 2005. p. 192-200.
- [26] Mondada F, Bonani M, Raemy X, Pugh J, Cianci C, Klaptoz A, et al. The e-puck, a robot designed for education in engineering. Proceedings of the 9th conference on autonomous robot systems and competitions. 2009;1(1):59-65.
- [27] Bonani M, Longchamp V, Magnenat S, Réturnaz P, Burnier D, Roulet G, et al. The marXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings. 2010:4187-93.
- [28] Wilson S, Gameros R, Sheely M, Lin M, Dover K, Gevorkyan R, et al. Pheeno, A Versatile Swarm Robotic Research and Education Platform. IEEE Robotics and Automation Letters. 2016;1(2):884-91.
- [29] Liu C, Mao Q, Chu X, Xie S. An Improved A-Star Algorithm Considering Water Current, Traffic Separation and Berthing for Vessel Path Planning. Applied Sciences. 2019 mar;9:1057.
- [30] Mehlhorn K, Sanders P. Algorithms and Data Structures: The Basic Toolbox. 1st ed. Springer Publishing Company, Incorporated; 2008.
- [31] Yun X, Tan KC. A wall-following method for escaping local minima in potential field based motion planning. In: 1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97; 1997. p. 421-6.
- [32] Chengqing L, Ang M, Krishnan H, Lim S. Virtual obstacle concept for local-minimum-recovery in potential-field based navigation. Proceedings 2000 ICRA Millennium Conference IEEE International Conference on Robotics and Automation Symposia Proceedings (Cat No00CH37065). 2000;2:983-8 vol.2.
- [33] Koren Y, Borenstein J. Potential field methods and their inherent limitations for mobile robot navigation. In: Proceedings. 1991 IEEE International Conference on Robotics and Automation; 1991. p. 1398-404 vol.2.

- [34] Yu J, LaValle SM. Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics. *IEEE Transactions on Robotics*. 2016;32(5):1163-77.

قائمة الأشكال

15	1.1	مثال على الدوال التتفقية
16	2.1	التدفق المولد باستخدام الخوارزمية <i>latticeBoltzman</i> . يمكن ملاحظة السلوك غير الفيزيائي للسائل بالنظر إلى تداخل السائل مع الجدران.
17	3.1	الأمواج المتقدمة والمتأخرة في العمل [17]
19	4.1
22	1.2	تحويل فضاء العمل المقطع إلى بيان غير موجه
32	2.2	مثال عن الطريقة المقترحة
32	3.2	القوى المتبادلة بين الروبوتات مماثلة لعناصر ميكانيكية
37	4.2	تصميم عجلات الروبوت الناتج
40	1.3	إخراج ثلاثي الأبعاد لمنصة العمل
44	2.3	التصميم باستخدام الحاسب لكل من الطبقتين العلوية والسفلية
45	3.3	المنتج النهائي بعد تصنيع الدارة
45	4.3	العطب الناتج عن التصنيع
46	5.3	الهيكل الخارجي للروبوت
47	6.3	مثال يوضح مقارنة بين سوء وجودة تصنيع العجلات
47	7.3	استجابة المقاطعات الضوئية
47	8.3	التقريب إلى نظام درجة أولى
48	9.3	محاكاة استجابة AVR للمقاطعة الضوئية
48	10.3	طبقة بلاستيكية لتحسين قراءة المقاطعات
49	11.3
50	12.3
50	13.3	شكل توضيحي للأجزاء الأساسية في بروتوكول نقل الرسائل MQTT
52	14.3	صورة توضيحية لآلية عمل محددات ArUco

قائمة الجداول

14	الأبعاد التسقية وأبعاد النظام	1.1
22	خرائط حل كل من خوارزميتي البيان.	1.2
24	عملية حساب المسار في طريقة الحقل الكموني	2.2
29	أداء الخوارزمية لعدة خرائط	3.2
31	عملية حساب المسار في الحالة الأولى من الحل	4.2
43	جدول مقارنة بين وحدات الاتصال المختلفة	1.3

باب 5

الملحقات

Manual Reference Code C AVR 1.5

Swarm Firmware AVR

Using by Doxygen 1.9.1

page numbering is the reference manual is relative to this page for ease of indexing, hyperef is present.

1 License	1
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 _i_pos Struct Reference	7
4.1.1 Detailed Description	7
4.2 _omega Struct Reference	7
4.2.1 Detailed Description	8
4.3 _theta Struct Reference	8
4.3.1 Detailed Description	8
4.4 mat2 Struct Reference	8
4.4.1 Detailed Description	8
4.5 PID_DATA Struct Reference	9
4.5.1 Detailed Description	9
4.6 point Struct Reference	9
4.6.1 Detailed Description	10
4.6.2 Field Documentation	10
4.6.2.1 x	10
4.6.2.2 y	10
4.7 pos Struct Reference	10
4.7.1 Detailed Description	11
4.7.2 Field Documentation	11
4.7.2.1 th	11
4.7.2.2 x	11
4.7.2.3 y	11
4.8 pos_dot Struct Reference	11
4.8.1 Detailed Description	12
4.8.2 Field Documentation	12
4.8.2.1 th_dot	12
4.8.2.2 x_dot	12
4.8.2.3 y_dot	12
5 File Documentation	13
5.1 src/_adc__.h File Reference	13
5.1.1 Detailed Description	13
5.1.2 Function Documentation	14
5.1.2.1 _adc_init()	14
5.1.2.2 _adc_read()	14
5.2 src/_dc_control__.h File Reference	14

5.2.1 Detailed Description	15
5.2.2 Function Documentation	15
5.2.2.1 _break_motor()	15
5.2.2.2 _command()	16
5.2.2.3 _dc_controller_loop()	16
5.2.2.4 _init_dc_control()	16
5.2.2.5 _ref()	16
5.2.2.6 _sens()	17
5.2.2.7 _set_speed()	17
5.2.2.8 _update_controller()	18
5.3 src/_format__.h File Reference	18
5.3.1 Detailed Description	18
5.3.2 Function Documentation	19
5.3.2.1 _float_to_printable()	19
5.4 src/_INT_0_1__.h File Reference	19
5.4.1 Detailed Description	19
5.4.2 Function Documentation	20
5.4.2.1 _interrupt0_enable()	20
5.5 src/_kinematics__.h File Reference	20
5.5.1 Detailed Description	21
5.5.2 Function Documentation	21
5.5.2.1 _dead_reckon()	21
5.5.2.2 _inertial_to_pos_dot()	22
5.5.2.3 _omega_to_intertial()	22
5.5.2.4 _thetaLR_to_pos()	22
5.5.2.5 _update_omega()	23
5.5.2.6 _update_thetaLR()	23
5.6 src/_odometry__.h File Reference	23
5.6.1 Detailed Description	24
5.6.2 Function Documentation	24
5.6.2.1 _insertion_sort()	24
5.6.2.2 _omega_comp_A()	25
5.6.2.3 _omega_comp_B()	25
5.6.2.4 _omega_from_encA()	25
5.6.2.5 _omega_from_encB()	25
5.6.2.6 _omega_from_PMA()	25
5.6.2.7 _omega_from_PMB()	26
5.6.2.8 _thetaA()	26
5.6.2.9 _thetaB()	26
5.6.2.10 _ticksA()	26
5.6.2.11 _ticksB()	26
5.7 src/_pwm__.h File Reference	26

5.7.1 Detailed Description	27
5.7.2 Function Documentation	27
5.7.2.1 _set_pwm_0A()	27
5.7.2.2 _set_pwm_0B()	27
5.7.2.3 _set_pwm_1A()	28
5.7.2.4 _set_pwm_1B()	28
5.8 src/_swarm_wold__.h File Reference	28
5.8.1 Detailed Description	29
5.9 src/_timer0__.h File Reference	29
5.9.1 Detailed Description	30
5.9.2 Function Documentation	30
5.9.2.1 _micros0()	30
5.9.2.2 _millis0()	31
5.9.2.3 _timer0_init()	31
5.9.2.4 _timer0_init_prescaler()	31
5.10 src/_timer1__.h File Reference	31
5.10.1 Detailed Description	32
5.10.2 Function Documentation	32
5.10.2.1 _timer1_init()	32
5.11 src/_timer2__.h File Reference	32
5.11.1 Detailed Description	33
5.11.2 Function Documentation	33
5.11.2.1 _timer2_init()	33
5.12 src/_uart__.h File Reference	33
5.12.1 Detailed Description	34
5.12.2 Function Documentation	34
5.12.2.1 usart_init()	34
5.13 src/_pid_.c File Reference	35
5.13.1 Detailed Description	35
5.13.2 Function Documentation	36
5.13.2.1 pid_Controller()	36
5.13.2.2 pid_Init()	36
5.13.2.3 pid_Reset_Integrator()	36
5.14 src/_pid_.h File Reference	37
5.14.1 Detailed Description	38
5.14.2 Macro Definition Documentation	38
5.14.2.1 MAX_INT	38
5.14.3 Typedef Documentation	38
5.14.3.1 pidData_t	38
5.14.4 Function Documentation	39
5.14.4.1 pid_Controller()	39
5.14.4.2 pid_Init()	39

5.14.4.3 pid_Reset_Integrator()	39
5.15 src/asf.h File Reference	40
5.15.1 Detailed Description	40
Index	41

Chapter 1

License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.
4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

_i_pos	7
_omega	7
_theta	8
mat2	8
PID_DATA	
PID Status	9
point	9
pos	10
pos_dot	11

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

src/adc.h	Header file for adc.c	13
src/dc_control.h	Header file for dc_control.c	14
src/format.h	Header file for format.c	18
src/INT_0_1.h	Header file for INT_0_1.c	19
src/kinematics.h	Header file for kinematics.c	20
src/odometry.h	Header file for odometry.c	23
src/pin_map.h		??
src/pwm.h	Header file for pwm.c	26
src/swarm_wold.h	Header file for dummy purposes	28
src/timer0.h	Header file for timer0.c	29
src/timer1.h	Header file for timer1.c	31
src/timer2.h	Header file for timer2.c	32
src/usart.h	Header file for pid.c	33
src/pid.c	General PID implementation for AVR	35
src/pid.h	Header file for pid.c	37
src/asf.h	Autogenerated API include file for the Atmel Software Framework (ASF)	40

Chapter 4

Data Structure Documentation

4.1 _i_pos Struct Reference

```
#include <__kinematics__.h>
```

Data Fields

- float **v**
linear velocity V
- float **w**
rotational velocity W

4.1.1 Detailed Description

Inertial pos vector V and W

The documentation for this struct was generated from the following file:

- src/_kinematics_.h

4.2 _omega Struct Reference

```
#include <__kinematics__.h>
```

Data Fields

- float **wl**
left motor
- float **wr**
right motor

4.2.1 Detailed Description

Struct to hold wheels' velocity rad/s

The documentation for this struct was generated from the following file:

- src/[__kinematics__.h](#)

4.3 _theta Struct Reference

```
#include <__kinematics__.h>
```

Data Fields

- float **left**
left motor
- float **right**
right motor

4.3.1 Detailed Description

struct to hold wheel absolute rotation

The documentation for this struct was generated from the following file:

- src/[__kinematics__.h](#)

4.4 mat2 Struct Reference

```
#include <__kinematics__.h>
```

Data Fields

- float **a**
11
- float **b**
12
- float **c**
21
- float **d**
22

4.4.1 Detailed Description

square matrix struct [

Parameters

a	
---	--

The documentation for this struct was generated from the following file:

- src/[kinematics.h](#)

4.5 PID_DATA Struct Reference

PID Status.

```
#include <_pid_.h>
```

Data Fields

- int16_t lastProcessValue
Last process value, used to find derivative of process value.
- int32_t sumError
Summation of errors, used for integrate calculations.
- int16_t P_Factor
The Proportional tuning constant, multiplied with SCALING_FACTOR.
- int16_t I_Factor
The Integral tuning constant, multiplied with SCALING_FACTOR.
- int16_t D_Factor
The Derivative tuning constant, multiplied with SCALING_FACTOR.
- int16_t maxError
Maximum allowed error, avoid overflow.
- int32_t maxSumError
Maximum allowed sumerror, avoid overflow.

4.5.1 Detailed Description

PID Status.

Setpoints and data used by the PID control algorithm

The documentation for this struct was generated from the following file:

- src/[pid.h](#)

4.6 point Struct Reference

```
#include <__kinematics__.h>
```

Data Fields

- float [x](#)
- float [y](#)

4.6.1 Detailed Description

point struct (x and y card. coordinates

4.6.2 Field Documentation

4.6.2.1 x

float x

Parameters

x	<input type="text"/>
---	----------------------

4.6.2.2 y

float y

Parameters

y	<input type="text"/>
---	----------------------

The documentation for this struct was generated from the following file:

- src/[__kinematics__.h](#)

4.7 pos Struct Reference

```
#include <__kinematics__.h>
```

Data Fields

- float [x](#)
- float [y](#)
- float [th](#)

4.7.1 Detailed Description

position struct to hold robot's pos vector

4.7.2 Field Documentation

4.7.2.1 th

float th

Parameters

<i>Theta</i>	(\theta)
--------------	----------

4.7.2.2 x

float x

Parameters

x	
---	--

4.7.2.3 y

float y

Parameters

y	
---	--

The documentation for this struct was generated from the following file:

- src/_kinematics_.h

4.8 pos_dot Struct Reference

```
#include <__kinematics__.h>
```

Data Fields

- float `x_dot`
- float `y_dot`
- float `th_dot`

4.8.1 Detailed Description

position derivative vector struct ($\frac{d \text{ pos}}{dt}$)

4.8.2 Field Documentation

4.8.2.1 `th_dot`

float `th_dot`

Parameters

<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------

4.8.2.2 `x_dot`

float `x_dot`

Parameters

<code>x'</code>	<input type="checkbox"/>
-----------------	--------------------------

4.8.2.3 `y_dot`

float `y_dot`

Parameters

<code>y'</code>	<input type="checkbox"/>
-----------------	--------------------------

The documentation for this struct was generated from the following file:

- src/[kinematics.h](#)

Chapter 5

File Documentation

5.1 src/_adc_.h File Reference

Header file for **adc.c**.

```
#include <__swarm_wold__.h>
```

Functions

- void _adc_init (void)
- int _adc_read (uint8_t channel)

5.1.1 Detailed Description

Header file for **adc.c**.

- File: **adc.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: ADC module driver

Author

Swarm robot graduation project workgroub
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/19/2021 9:30:13 AM

5.1.2 Function Documentation

5.1.2.1 _adc_init()

```
void _adc_init (
    void )
```

Initialize the ADC module

5.1.2.2 _adc_read()

```
int _adc_read (
    uint8_t channel )
```

Reads from

Parameters

<i>channel</i>	
----------------	--

Parameters

in	<i>channel</i>	adc channel from PortC.
----	----------------	-------------------------

5.2 src/_dc_control__.h File Reference

Header file for **dc_control.c**.

```
#include <__swarm_wold__.h>
```

Macros

- #define **MA** 1
- #define **MB** 2
- #define **DEBUG_CONTROLLER** 0
- #define **K_P** 4.0
- #define **K_I** 0.3
- #define **K_D** 0.5

Functions

- int16_t _ref (uint8_t motor)
- void _command (uint8_t motor, int16_t inputValue)
- float _sens (uint8_t motor)
- int _set_speed (uint8_t motor, int value)
- void _break_motor (uint8_t motor)
- void _init_dc_control (void)
- int16_t _update_controller (uint8_t motor)
- int16_t _dc_controller_loop (void)

5.2.1 Detailed Description

Header file for **dc_control.c**.

- File: **dc_control.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: PID for DC motor control

Author

Swarm robot graduation project workgroup
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/26/2021 9:18:13 PM

5.2.2 Function Documentation

5.2.2.1 _break_motor()

```
void _break_motor (
    uint8_t motor )
```

breaks the

Parameters

<i>motor</i>	<input type="checkbox"/>
--------------	--------------------------

Parameters

in	<i>motor</i>	motor MA or MB.
----	--------------	-----------------

5.2.2.2 _command()

```
void _command (
    uint8_t motor,
    int16_t inputValue )
```

Output command module for

Parameters

<i>motor</i>	of value
<i>inputValue</i>	

Parameters

in	<i>motor</i>	motor MA or MB.
in	<i>inputValue</i>	input value

5.2.2.3 _dc_controller_loop()

```
int16_t _dc_controller_loop (
    void )
```

PID DC controller loop

5.2.2.4 _init_dc_control()

```
void _init_dc_control (
    void )
```

Initialize the DC PID control module

5.2.2.5 _ref()

```
int16_t _ref (
    uint8_t motor )
```

Reference for

Parameters

<i>motor</i>	<input type="text"/>
--------------	----------------------

Parameters

in	<i>motor</i>	motor MA or MB.
----	--------------	-----------------

5.2.2.6 _sens()

```
float _sens (
    uint8_t motor )
```

sensor module for

Parameters

<i>motor</i>	<input type="text"/>
--------------	----------------------

Parameters

in	<i>motor</i>	motor MA or MB.
----	--------------	-----------------

5.2.2.7 _set_speed()

```
int _set_speed (
    uint8_t motor,
    int value )
```

L298 driver module using PWM signals and motor orientation

Parameters

<i>value</i>	can be negative or positive for direction
--------------	---

Parameters

in	<i>motor</i>	motor MA or MB.
in	<i>value</i>	speed value

5.2.2.8 _update_controller()

```
int16_t _update_controller (
    uint8_t motor )
```

Update a control iteration for a discrete PID controller for the DC motor

Parameters

in	<i>motor</i>	motor MA or MB.
----	--------------	-----------------

5.3 src/_format__.h File Reference

Header file for **format.c**.

```
#include <__swarm_wold__.h>
```

Functions

- char * [_float_to_printable](#) (float input)

5.3.1 Detailed Description

Header file for **format.c**.

- File: **format.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: string formatter

Author

Swarm robot graduation project workgroup
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/5/2021 12:34:35 AM

5.3.2 Function Documentation

5.3.2.1 _float_to_printable()

```
char* _float_to_printable (
    float input )
```

Convert a float variable to string

Parameters

in	<i>input</i>	input value of float to convert to string
----	--------------	---

5.4 src/_INT_0_1_.h File Reference

Header file for **INT_0_1.c**.

```
#include <__swarm_wold__.h>
```

Macros

- #define _INT_LOW_LEVEL 0
- #define _INT_CHANGE_LEVEL 1
- #define _INT_FALLING_EDGE 2
- #define _INT_RISING_EDGE 3

Functions

- void [_interrupt0_enable](#) (uint8_t trigger)

5.4.1 Detailed Description

Header file for **INT_0_1.c**.

- File: **INT_0_1.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: external interrupt driver

Author

Swarm robot graduation project workgroub
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/11/2021 9:18:06 AM

5.4.2 Function Documentation

5.4.2.1 _interrupt0_enable()

```
void _interrupt0_enable (
    uint8_t trigger )
```

Enable interrupt of pin INT0 (see _pin_map.h)

Parameters

in	<i>trigger</i>	trigger mode, see file _INT_0_1_.h
----	----------------	--

5.5 src/_kinematics_.h File Reference

Header file for **kinematics.c**.

```
#include <__swarm_wold__.h>
```

Data Structures

- struct [point](#)
- struct [pos](#)
- struct [pos_dot](#)
- struct [i_pos](#)
- struct [mat2](#)
- struct [omega](#)
- struct [theta](#)

Macros

- #define L 0.06
- #define r 0.02
- #define R_over_L 0.333
- #define R_over_2 0.01

Functions

- struct _i_pos_omega_to_intertial (struct _omega *input)
- struct pos_dot_inertial_to_pos_dot (struct _i_pos *inertial, struct pos *_pos)
- uint16_t _dead_reckon (struct pos *_pos, struct pos_dot *_pos_dot)
- struct pos_thetaLR_to_pos (struct _theta *th)
- void _update_thetaLR (struct _theta *input)
- void _update_omega (struct _omega *input)

5.5.1 Detailed Description

Header file for **kinematics.c**.

- File: **kinematics.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: Kinematics calculations for the mobile robot

Author

Swarm robot graduation project workgroup
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

5/29/2021 8:04:54 PM

5.5.2 Function Documentation

5.5.2.1 _dead_reckon()

```
uint16_t _dead_reckon (
    struct pos * _pos,
    struct pos_dot * _pos_dot )
```

a dead-reckoner for spatial pos estimation, returns debug info

Parameters

in	<i>_pos</i>	pos
in	<i>_pos_dot</i>	derivative of pos

5.5.2.2 *_inertial_to_pos_dot()*

```
struct pos_dot _inertial_to_pos_dot (
    struct _i_pos * inertial,
    struct pos * _pos )
```

A function to convert inertial velocities to *pos_dot*

Parameters

in	<i>inertial</i>	struct of inertial pos
in	<i>_pos</i>	struct of current pos for \theta

5.5.2.3 *_omega_to_intertial()*

```
struct _i_pos _omega_to_intertial (
    struct _omega * input )
```

A function to convert wheels rotation velocity to inertial velocities

Parameters

in	<i>input</i>	struct of wheels velocity
	<i>_omega</i>	

5.5.2.4 *_thetaLR_to_pos()*

```
struct pos _thetaLR_to_pos (
    struct _theta * th )
```

A function to convert wheels' absolute rotation angles to pos (not proper)

Parameters

in	<i>th</i>	wheels' absolute rotation
----	-----------	---------------------------

5.5.2.5 _update_omega()

```
void _update_omega (
    struct _omega * input )
```

A function to update the wheels' rotation velocity or each wheel from hardware modules

Parameters

in	input	wheel's rotational velocity
----	-------	-----------------------------

5.5.2.6 _update_thetaLR()

```
void _update_thetaLR (
    struct _theta * input )
```

A function to update the absolute rotation angle of each wheel from hardware modules

Parameters

in	input	wheels' absolute rotation
----	-------	---------------------------

5.6 src/_odometry_.h File Reference

Header file for **odometry.c**.

```
#include <__swarm_wold__.h>
```

Macros

- #define __ENC_TICK_THETA_FOR_OMEGA 190399
- #define __ENC_TICK_THETA 0.1904
- #define __PM_lower_bound 200
- #define __PM_upper_bound 800
- #define __PM_SAMPLE_COUNT 5
- #define __PM_SLOPE 1
- #define FORWARD 1
- #define BACKWARD -1

Functions

- float `_thetaA` (void)
- float `_thetaB` (void)
- float `_omega_from_encA` (void)
- float `_omega_from_encB` (void)
- float `_omega_from_PMA` (void)
- float `_omega_from_PMB` (void)
- float `_omega_comp_A` (void)
- float `_omega_comp_B` (void)
- int32_t `_ticksA` ()
- int32_t `_ticksB` ()
- void `_insertion_sort` (uint16_t arr[], int n)

5.6.1 Detailed Description

Header file for `odometry.c`.

- File: `__odometry__h`
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: Odometry calculations for the mobile robot

Author

Swarm robot graduation project workgroub
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/19/2021 9:43:59 AM

5.6.2 Function Documentation

5.6.2.1 `_insertion_sort()`

```
void _insertion_sort (
    uint16_t arr[],
    int n )
```

Insertion sort algorithm (fastest for small array sizes)

Parameters

in	<i>arr</i>	pointer to start of sorting
in	<i>n</i>	offset from
	<i>arr[]</i>	

5.6.2.2 _omega_comp_A()

```
float _omega_comp_A (
    void )
```

Complementary filter for wheel A angular velocity (encoder ticks and internal potentiometer)

5.6.2.3 _omega_comp_B()

```
float _omega_comp_B (
    void )
```

Complementary filter for wheel B angular velocity (encoder ticks and internal potentiometer)

5.6.2.4 _omega_from_encA()

```
float _omega_from_encA (
    void )
```

Returns the A wheel's angular velocity from encoder readings by measuring he time between consecutive encoder ticks

5.6.2.5 _omega_from_encB()

```
float _omega_from_encB (
    void )
```

Returns the B wheel's angular velocity from encoder readings by measuring he time between consecutive encoder ticks

5.6.2.6 _omega_from_PMA()

```
float _omega_from_PMA (
    void )
```

Returns the A wheel's angular velocity using the potentiometer readings

5.6.2.7 _omega_from_PMB()

```
float _omega_from_PMB (
    void )
```

Returns the B wheel's angular velocity using the potentiometer readings

5.6.2.8 _thetaA()

```
float _thetaA (
    void )
```

Returns the absolute rotation angle of motor A

5.6.2.9 _thetaB()

```
float _thetaB (
    void )
```

Returns the absolute rotation angle of motor B

5.6.2.10 _ticksA()

```
int32_t _ticksA ( )
```

returns the number of ticks happened since boot to wheel A

5.6.2.11 _ticksB()

```
int32_t _ticksB ( )
```

returns the number of ticks happened since boot to wheel B

5.7 src/_pwm_.h File Reference

Header file for **pwm.c**.

```
#include <__swarm_wold__.h>
```

Functions

- void [_set_pwm_0A](#) (int input)
- void [_set_pwm_0B](#) (int input)
- void [_set_pwm_1A](#) (int input)
- void [_set_pwm_1B](#) (int input)

5.7.1 Detailed Description

Header file for **pwm.c**.

- File: **pwm.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: PWM driver

Author

Swarm robot graduation project workgroup
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/19/2021 9:30:13 AM

5.7.2 Function Documentation

5.7.2.1 _set_pwm_0A()

```
void _set_pwm_0A (
    int input )
```

sets the PWM output of OCR0A (Motor A rear)

Parameters

in	<i>input</i>	
	<i>input</i>	a number (0 --> 255)

5.7.2.2 _set_pwm_0B()

```
void _set_pwm_0B (
```

```
int input )
```

sets the PWM output of OCR0B (Motor B rear)

Parameters

in	<i>input</i>	
	<i>input</i>	a number (0 --> 255)

5.7.2.3 _set_pwm_1A()

```
void _set_pwm_1A (
    int input )
```

sets the PWM output of OCR1A (Motor A front)

Parameters

in	<i>input</i>	
	<i>input</i>	a number (0 --> 255)

5.7.2.4 _set_pwm_1B()

```
void _set_pwm_1B (
    int input )
```

sets the PWM output of OCR1B (Motor B front)

Parameters

in	<i>input</i>	
	<i>input</i>	a number (0 --> 255)

5.8 src/_swarm_wold__.h File Reference

Header file for dummy purposes.

```
#include <stdio.h>
#include <asf.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <board.h>
#include <conf_board.h>
```

```
#include <util/delay.h>
#include "stdint.h"
#include <__adc__.h>
#include <__INT_0_1__.h>
#include <__pin_map.h>
#include <__timer0__.h>
#include <__timer1__.h>
#include <__timer2__.h>
#include <__USART__.h>
#include <__dc_control__.h>
#include <__odometry__.h>
#include <__pwm__.h>
#include <__pid__.h>
#include <__format__.h>
```

Macros

- #define F_CPU 16000000

5.8.1 Detailed Description

Header file for dummy purposes.

- File: **swarm_world.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: Discrete PID controller

Author

Swarm robot graduation project workgroub
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/19/2021 9:31:42 AM

5.9 src/_timer0__.h File Reference

Header file for **timer0.c**.

```
#include <__swarm_wold__.h>
```

Functions

- void `_timer0_init` (void)
- void `_timer0_init_prescaler` (uint16_t prescaler)
- unsigned long `_micros0` (void)
- unsigned long `_millis0` (void)

5.9.1 Detailed Description

Header file for `timer0.c`.

- File: `timer0.h`
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: Discrete PID controller

Author

Swarm robot graduation project workgroup
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/19/2021 9:03:51 AM

5.9.2 Function Documentation

5.9.2.1 `_micros0()`

```
unsigned long _micros0 (
    void )
```

Returns the microseconds passed since timer initialize

5.9.2.2 _millis0()

```
unsigned long _millis0 (
    void )
```

Returns the milliseconds passed since timer initialize

5.9.2.3 _timer0_init()

```
void _timer0_init (
    void )
```

Initialize Timer/Counter 0 with clock prescaler of 1024 (244 Hz overflow at 16 MHz)

5.9.2.4 _timer0_init_prescaler()

```
void _timer0_init_prescaler (
    uint16_t prescaler )
```

Initialize Timer/Counter 0 with costume prescaler (see the datasheet)

Parameters

in	<i>prescaler</i>	prescaler 1,2,4,8,255,1024
----	------------------	----------------------------

5.10 src/_timer1__.h File Reference

Header file for **timer1.c**.

```
#include <__swarm_wold__.h>
```

Macros

- #define **_TICK_US_1** 0.0625
- #define **_TICK_MS_1** 0.0000625
- #define **_TICK_US_0** 16
- #define **_TICK_MS_0** 0.016

Functions

- void **_timer1_init** (void)
- uint64_t **_micros1** (void)
- uint64_t **_millis1** (void)

Variables

- volatile uint8_t **_controler_flag_A**
- volatile uint8_t **_controler_flag_B**

5.10.1 Detailed Description

Header file for **timer1.c**.

- File: **timer1.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: Discrete PID controller

Author

Swarm robot graduation project workgroup
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/9/2021 10:17:41 AM

5.10.2 Function Documentation

5.10.2.1 **_timer1_init()**

```
void _timer1_init (
    void )
```

Initialize Timer/Counter 1 with 1 prescaler (244 MHz overflow frequency at 16 MHz)

5.11 **src/_timer2__.h** File Reference

Header file for **timer2.c**.

```
#include <__swarm_wold__.h>
```

Functions

- void [_timer2_init](#) (void)

5.11.1 Detailed Description

Header file for **timer2.c**.

- File: **timer2.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: Discrete PID controller

Author

Swarm robot graduation project workgroub
Mechatronics Program for the Distinguished
\$Name\$

Revision

1

\$RCSfile\$

Date

/20/2021 12:04:32 AM

5.11.2 Function Documentation

5.11.2.1 _timer2_init()

```
void _timer2_init (
    void )
```

Initialize Timer/Counter 2 with 1 prescaler (244 Hz overflow frequency at 16 MHz)

5.12 src/_usart__.h File Reference

Header file for pid.c.

```
#include <asf.h>
#include <stdio.h>
#include <board.h>
#include <conf_board.h>
#include <util/delay.h>
```

Macros

- #define **BAUD** 57600
- #define **RX_BUFSIZE** 120
- #define **BRC** ((F_CPU/(16UL*BAUD)) - 1)

Functions

- void [usart_init](#) (void)

5.12.1 Detailed Description

Header file for pid.c.

- File: **usart.h**
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: USART module driver

Author

Swarm robot graduation project workgroup
Mechatronics Program for the Distinguished
\$Name\$

Revision

456

\$RCSfile\$

Date

021-5

5.12.2 Function Documentation

5.12.2.1 usart_init()

```
void usart_init (
    void )
```

Initialize the USART module with

Parameters

BAUD	
------	--

5.13 src/_pid_.c File Reference

General PID implementation for AVR.

```
#include <_pid_.h>
```

Functions

- void **pid_Init** (int16_t p_factor, int16_t i_factor, int16_t d_factor, struct **PID_DATA** *pid)
Initialisation of PID controller parameters.
- int16_t **pid_Controller** (int16_t setPoint, int16_t processValue, struct **PID_DATA** *pid_st)
PID control algorithm.
- void **pid_Reset_Integrator** (pidData_t *pid_st)
Resets the integrator.

5.13.1 Detailed Description

General PID implementation for AVR.

Discrete PID controller implementation. Set up by giving P/I/D terms to **Init_PID()**, and uses a struct **PID_DATA** to store internal values.

- File: pid.c
- Compiler: IAR EWAAVR 4.11A
- Supported devices: All AVR devices can be used.
- AppNote: AVR221 - Discrete PID controller

Author

Atmel Corporation: <http://www.atmel.com>
Support email: avr@atmel.com

\$Name\$

Revision

456

\$RCSfile\$

Date

2006-02-16 12:46:13 +0100 (to, 16 feb 2006)

5.13.2 Function Documentation

5.13.2.1 pid_Controller()

```
int16_t pid_Controller (
    int16_t setPoint,
    int16_t processValue,
    struct PID_DATA * pid_st )
```

PID control algorithm.

Calculates output from setpoint, process value and PID status.

Parameters

<i>setPoint</i>	Desired value.
<i>processValue</i>	Measured value.
<i>pid_st</i>	PID status struct.

5.13.2.2 pid_Init()

```
void pid_Init (
    int16_t p_factor,
    int16_t i_factor,
    int16_t d_factor,
    struct PID_DATA * pid )
```

Initialisation of PID controller parameters.

Initialise the variables used by the PID algorithm.

Parameters

<i>p_factor</i>	Proportional term.
<i>i_factor</i>	Integral term.
<i>d_factor</i>	Derivate term.
<i>pid</i>	Struct with PID status.

5.13.2.3 pid_Reset_Integrator()

```
void pid_Reset_Integrator (
    pidData_t * pid_st )
```

Resets the integrator.

Calling this function will reset the integrator in the PID regulator.

5.14 src/_pid_.h File Reference

Header file for pid.c.

```
#include "stdint.h"
```

Data Structures

- struct **PID_DATA**
PID Status.

Macros

- #define **SCALING_FACTOR** 128
- #define **MAX_INT** INT16_MAX
Maximum values.
- #define **MAX_LONG** INT32_MAX
- #define **MAX_I_TERM** (MAX_LONG / 2)
- #define **FALSE** 0
- #define **TRUE** 1

Typedefs

- typedef struct **PID_DATA** pidData_t
PID Status.

Functions

- void **pid_Init** (int16_t p_factor, int16_t i_factor, int16_t d_factor, struct **PID_DATA** *pid)
Initialisation of PID controller parameters.
- int16_t **pid_Controller** (int16_t setPoint, int16_t processValue, struct **PID_DATA** *pid_st)
PID control algorithm.
- void **pid_Reset_Integrator** (pidData_t *pid_st)
Resets the integrator.

Variables

- volatile uint8_t **flag**

5.14.1 Detailed Description

Header file for pid.c.

- File: pid.h
- Compiler: GCC-AVR
- Supported devices: Tested on 328p
- AppNote: Discrete PID controller

Author

Swarm robot graduation project workgroup
Mechatronics Program for the Distinguished
\$Name\$

Revision

456

\$RCSfile\$

Date

021

5.14.2 Macro Definition Documentation

5.14.2.1 MAX_INT

```
#define MAX_INT INT16_MAX
```

Maximum values.

Needed to avoid sign/overflow problems

5.14.3 Typedef Documentation

5.14.3.1 pidData_t

```
typedef struct PID_DATA pidData_t
```

PID Status.

Setpoints and data used by the PID control algorithm

5.14.4 Function Documentation

5.14.4.1 pid_Controller()

```
int16_t pid_Controller (
    int16_t setPoint,
    int16_t processValue,
    struct PID_DATA * pid_st )
```

PID control algorithm.

Calculates output from setpoint, process value and PID status.

Parameters

<i>setPoint</i>	Desired value.
<i>processValue</i>	Measured value.
<i>pid_st</i>	PID status struct.

5.14.4.2 pid_Init()

```
void pid_Init (
    int16_t p_factor,
    int16_t i_factor,
    int16_t d_factor,
    struct PID_DATA * pid )
```

Initialisation of PID controller parameters.

Initialise the variables used by the PID algorithm.

Parameters

<i>p_factor</i>	Proportional term.
<i>i_factor</i>	Integral term.
<i>d_factor</i>	Derivate term.
<i>pid</i>	Struct with PID status.

5.14.4.3 pid_Reset_Integrator()

```
void pid_Reset_Integrator (
    pidData_t * pid_st )
```

Resets the integrator.

Calling this function will reset the integrator in the PID regulator.

5.15 src/asf.h File Reference

Autogenerated API include file for the Atmel Software Framework (ASF)

```
#include <user_board.h>
#include <board.h>
#include <interrupt.h>
#include <compiler.h>
#include <status_codes.h>
#include <parts.h>
```

Macros

- #define **F_CPU** 16000000

5.15.1 Detailed Description

Autogenerated API include file for the Atmel Software Framework (ASF)

Copyright (c) 2012 Atmel Corporation. All rights reserved.

\asf_license_start

Index

__INT_0_1__.h
 _interrupt0_enable, 20
__adc__.h
 _adc_init, 14
 _adc_read, 14
__dc_control__.h
 _break_motor, 15
 _command, 16
 _dc_controller_loop, 16
 _init_dc_control, 16
 _ref, 16
 _sens, 17
 _set_speed, 17
 _update_controller, 17
__format__.h
 _float_to_printable, 19
__kinematics__.h
 _dead_reckon, 21
 _inertial_to_pos_dot, 22
 _omega_to_intertial, 22
 _thetaLR_to_pos, 22
 _update_omega, 23
 _update_thetaLR, 23
__odometry__.h
 _insertion_sort, 24
 _omega_comp_A, 25
 _omega_comp_B, 25
 _omega_from_PMA, 25
 _omega_from_PMB, 25
 _omega_from_encA, 25
 _omega_from_encB, 25
 _thetaA, 26
 _thetaB, 26
 _ticksA, 26
 _ticksB, 26
__pwm__.h
 _set_pwm_0A, 27
 _set_pwm_0B, 27
 _set_pwm_1A, 28
 _set_pwm_1B, 28
__timer0__.h
 _micros0, 30
 _millis0, 30
 _timer0_init, 31
 _timer0_init_prescaler, 31
__timer1__.h
 _timer1_init, 32
__timer2__.h
 _timer2_init, 33
__uart__.h
 _usart_init, 34
__adc_init
 __adc__.h, 14
__adc_read
 __adc__.h, 14
_break_motor
 __dc_control__.h, 15
_command
 __dc_control__.h, 16
_dc_controller_loop
 __dc_control__.h, 16
_dead_reckon

MAX_INT, 38
 pid_Controller, 39
 pid_Init, 39
 pid_Reset_Integrator, 39
 pidData_t, 38
 _ref
 __dc_control__.h, 16
 _sens
 __dc_control__.h, 17
 _set_pwm_0A
 __pwm__.h, 27
 _set_pwm_0B
 __pwm__.h, 27
 _set_pwm_1A
 __pwm__.h, 28
 _set_pwm_1B
 __pwm__.h, 28
 _set_speed
 __dc_control__.h, 17
 _theta, 8
 _thetaA
 __odometry__.h, 26
 _thetaB
 __odometry__.h, 26
 _thetaLR_to_pos
 __kinematics__.h, 22
 _ticksA
 __odometry__.h, 26
 _ticksB
 __odometry__.h, 26
 _timer0_init
 __timer0__.h, 31
 _timer0_init_prescaler
 __timer0__.h, 31
 _timer1_init
 __timer1__.h, 32
 _timer2_init
 __timer2__.h, 33
 _update_controller
 __dc_control__.h, 17
 _update_omega
 __kinematics__.h, 23
 _update_thetaLR
 __kinematics__.h, 23

 mat2, 8
 MAX_INT
 pid.h, 38

 pid_Controller
 pid.c, 36
 pid.h, 39
 PID_DATA, 9
 pid_Init
 pid.c, 36
 pid.h, 39
 pid_Reset_Integrator
 pid.c, 36
 pid.h, 39

pidData_t
 pid.h, 38
 point, 9
 x, 10
 y, 10
 pos, 10
 th, 11
 x, 11
 y, 11
 pos_dot, 11
 th_dot, 12
 x_dot, 12
 y_dot, 12

 src/__adc__.h, 13
 src/__dc_control__.h, 14
 src/__format__.h, 18
 src/__INT_0_1__.h, 19
 src/__kinematics__.h, 20
 src/__odometry__.h, 23
 src/__pwm__.h, 26
 src/__swarm_wold__.h, 28
 src/__timer0__.h, 29
 src/__timer1__.h, 31
 src/__timer2__.h, 32
 src/__uart__.h, 33
 src/_pid_.c, 35
 src/_pid_.h, 37
 src/asf.h, 40

th
 pos, 11
 th_dot
 pos_dot, 12

 uart_init
 __uart__.h, 34

x
 point, 10
 pos, 11
 x_dot
 pos_dot, 12

y
 point, 10
 pos, 11
 y_dot
 pos_dot, 12