Raport științific – proiect PN-III-P1-1.1-TE-2016-0737, nr. 32/2018

Director de proiect,
Prof. dr. ing. Marius Kloetzer

Prezentul document include raportul științific cumulativ al etapelor proiectului PN-III-P1-1.1-TE-2016-0737, finanțat de CNCS – UEFISCDI în cadrul programului PNCDI III. Proiectul intitulat „Navigarea roboților mobili cooperativi în aplicații complexe” s-a desfășurat în trei etape, documentul de față fiind structurat în conformitate cu acestea.

Etapa 1, intitulată „Stabilirea claselor de specificații și proiectarea domeniului de evoluție”, s-a desfășurat în perioada mai-decembrie 2018. Planul său de realizare a inclus activitățile “1.1. Clase de specificații pentru echipe de roboți mobili”, „1.2. Algoritmi pentru planificarea mișcării” și „1.3. Definirea unui domeniu de evoluție 3D”, raportul acestora regăsindu-se pe paginile 2-17 ale prezentului document.

Etapa 2, „Soluții algoritmice și experimente preliminare”, s-a desfășurat pe parcursul anului 2019, fiind structurată în activitățile „2.1. Algoritmi de planificare pentru specificații de mișcare și efectuare de acțiuni”, „2.2. Integrarea algoritmilor dezvoltați într-un mediu software” și „2.3. Rezultate preliminare pentru experimente în timp real”. Raportul științific corespunzător acestei etape este inclus în paginile 18-46.

Etapa 3, „Experimente în timp real”, cu intervalul de desfășurare ianuarie-aprilie 2020, a inclus activitatea „3.1. Experimente pe baza soluțiilor de planificare propuse”, iar raportul științific este cuprins în paginile 47-64 ale documentului de față.

Etapa 1 – Stabilirea claselor de specificații și proiectarea domeniului de evoluție

Activitatea 1.1. Clase de specificații pentru echipe de roboți mobili

***Obiectiv**: Identificarea și definirea specificațiilor complexe pentru roboți mobili. Analiza soluțiilor existente și identificarea unei clase de specificații fezabile pentru rezolvarea problemei propuse.*

Această secțiune a raportului este dedicată prezentării succinte a unor rezultate marcabile pentru domeniul proiectului. Pe viitor, intenționăm să ne inspirăm din unele din aceste rezultate pentru a extinde contribuțiile recente ale membrilor echipei, descrise în secțiunea 1.2.

Planificarea traiectoriei formațiilor de roboți mobili este un domeniu de cercetare care atrage din ce în ce mai multă atenție, rezultatele fiind utilizate în aplicații variate. În acest context, o problemă fundamentală de navigare o constituie cazul unui singur robot mobil care plecând dintr-o poziție inițială trebuie să ajungă într-o poziție finală evitând obstacolele din spațiul de lucru [1,2]. Pentru rezolvarea acestei probleme au fost propuse soluții bazate pe funcții potențial care să poată lucra direct în spațiul continuu sau abordări care să divizeze spațiul continuu într-o descriere finită cu stări (de exemplu descompuneri în celule) [2,3]. Discretizarea spațiului continuu este utilizată preponderent în aplicații complexe, cum ar fi îndeplinirea unor sarcini descrise de specificații boolene [4,5] sau prin formule de logică liniară temporală [6-8].

Descompunerea în celule reprezintă descrierea spațiului de lucru prin regiuni convexe [3]. Această descompunere permite rezolvarea problemelor de planificare prin abstractizarea roboților mobili la puncte [1].

După descompunerea spațiului de lucru se construiește un graf de conectivitate în care nodurile reprezintă celulele iar arcele dintre noduri definesc adiacența celulelor. Graful de conectivitate este parcurs de un algoritm de găsire a traiectoriei care va genera o secvență de celule a căror parcurgere asigură îndeplinirea scopului. În funcție de scop, secvența de celule poate să includă celule libere sau celule de interes în care se pot realiza acțiuni. După stabilirea secvenței de celule care îndeplinește misiunea, traiectoria parcursă de robot este constituită din mijloacele segmentelor care definesc adiacența dintre celule [1,2]. Avantajul acestei abordări este complexitatea computațională redusă deoarece problema identificării traiectoriei dorite se reduce la o căutare într-un graf. Totuși, există un dezavantaj major al acestei metode și anume faptul că pot fi generate soluții cu lungime mare în raport cu distanța necesară de parcurs. Având în vedere acest dezavantaj, distanța minimă poate fi considerată ca restricție dar efectul acestei specificații este formarea unei probleme neliniare de optimizare [9,10]. O soluție propusă [11] o reprezintă trei relaxări suboptime, fiecare necesitând rezolvarea unei probleme de programare liniară (LPP), având complexitate polinomială. Aceste relaxări substituie distanța totală Euclidiană prin suma normelor L_1, L_2 pătrat, L_∞ a segmentelor ce formează traiectoria.

Multitudinea de probleme care pot fi definite pentru sisteme cu roboți mobili pot fi reduse la trei categorii:

1. Se consideră un sistem de roboți mobili modelat dinamic căruia i se atașează un spațiu de lucru limitat cu obstacole. Se dorește proiectarea unor legi de mișcare folosind informații locale accesibile roboților vecini pentru a crea o structura predefinită a formației. În paralel se dorește ca inter-coliziunile dintre roboți precum și coliziunile cu obstacole să fie evitate.

2. Fie un sistem de roboți mobili modelat dinamic în sensul celui mai apropiat vecin. Prin definirea unui task de nivel înalt fiecărui robot se dorește proiectarea unor legi de mișcare descentralizate astfel încât fiecare robot să îndeplinească specificația dorită considerând restricții de timp.
3. Se consideră un sistem de roboți mobili supus perturbațiilor și având eventuale erori de modelare. Prin asignarea unor taskuri specifice fiecărui robot, care pot include și alți roboți din sistem, se dorește proiectarea unor legi de mișcare care să garanteze îndeplinirea sarcinilor.

Fiecare dintre aceste probleme poate fi descrisa prin diferite clase de specificații. În cele ce urmează vom prezenta cele mai utilizate formalisme pentru specificarea sarcinilor de mișcare ale roboților mobili, punctând expresivitatea limbajului si complexitatea problemei de generare a strategiilor de control care sa asigure îndeplinirea unei specificații date.

Formal, dat fiind un mediu de lucru și un set de regiuni de interes *Y*, sarcinile de mișcare ale robotului vor fi definite peste setul tuturor ieșirilor posibile iar problema de rezolvat consta in generarea automata (daca este posibil) a strategiilor de control care asigura satisfacerea unei specificații date. Unele simboluri din setul *Y* pot corespunde la acțiuni robotice ce se pot efectua in regiunile aferente.

1. Expresii regulate

Expresiile regulate [12,13] sunt descrieri scurte si convenabile ale limbajelor regulate, care sunt seturi de secvențe finite acceptate de automatele finite. Astfel, expresiile regulate definite peste $Y=\{y_1, y_2, ..., y_{|Y|}\}$ pot fi utilizate pentru a specifica comportamente de terminare ale roboților. Spre exemplu,

- *vizitarea unor regiuni într-o ordine dorita* ("vizitează y_1, apoi y_2" poate fi exprimat prin $(y_2 + y_{3+...+} y_{|Y|}))^* \cdot y_1 \cdot (y_1 + y_{3+...+} y_{|Y|}))^* \cdot y_2$, unde * denota închiderea Kleene a unui set, + denotă disjuncția (în acest caz uniunea de seturi), iar · denotă concatenarea.
- *urmărirea unui tipar* ("din y_1 mergi in y_2 si apoi in y_3, evitând toate celelalte regiuni" poate fi exprimat cu $y_1 \cdot y_2 \cdot y_3$)

Deși acest formalism este apropiat de limbajul natural, specificarea unor cerințe mai complicate poate devine foarte dificila. Problema generării strategiei de control plecând de la o specificație data sub forma unei expresii regulate peste setul de intrări (sau etichete asociate tranzițiilor) este bine înțeleasă pentru un sistem finit de tranziții [12, 13]. Ea poate fi rezolvata si pentru expresii definite peste setul de ieșiri atât timp cat automatul este deterministic.

2. Expresii ω-regulate

Pentru comportamente continue ale roboților (de exemplu in sarcini de supraveghere), expresiile ω-regulate peste *Y* pot fi o opțiune potrivita. Astfel de expresii codifica seturi de secvențe de lungime infinita ce pot fi acceptate de automate Büchi. Spre exemplu,

- *sarcini de supraveghere* ("vizitează y_1 si y_2 infinit de des" poate fi exprimata prin $((y_1 + y_{3+...+} y_{|Y|}))^* \cdot (y_2 + y_{3+...+} y_{|Y|}))^* \cdot y_2 \cdot (y_2 + y_{3+...+} y_{|Y|}))^*)^\omega$, unde simbolul $^\omega$ denota un număr infinit de repetiții).

Ca si in cazul expresiilor regulate, dificultatea consta in maparea specificațiilor date peste setul de stări sau de regiuni de interes, pentru sintetizarea unu controler.

3. Logici temporale

Logicile temporale au apărut ca framework-uri pentru specificarea si verificarea corectitudinii programelor pe calculator si au devenit utilizate in multe alte domenii datorita apropierii de limbajele naturale. Formulele logicii temporale reprezintă o opțiune buna pentru specificarea mișcării roboților pentru trei motive. In primul rând, sunt expresive si apropiate de limbajul natural. In al doilea rând, sunt deja disponibili algoritmi pentru evaluarea valorilor de adevăr ale expresiilor. In al treilea rând, formulele de logica temporala sunt

definite peste setul de stări sau peste setul de ieșiri ale unui sistem de tranziție, spre deosebire de expresiile regulate si ω-regulate.

Principalele limbaje din aceasta categorie sunt Logica Temporala Lineara (LTL) si Computation Tree Logic (CTL).

Formulele ***LTL*** [14, 15] sunt create folosind un set de propoziții atomice de interes (în cazul nostru, aceste propoziții vor fi regiunile de interes din setul *Y*), operatori logici ("negație" (eng. "not", notat cu ¬), "sau" (disjuncție, eng. "or", notat cu ∨), "și" (conjuncție, eng. and, notat cu ∧), "implicație" (notat ⇒), "echivalență" (notat ⇔)) și operatori temporali ("până când" (eng. "until", notat *U*), "cândva" (eng. "eventually", notat ◊), "mereu" (eng. "always", notat □)). Spre exemplu,

- "ajungi cândva în y_1, evitând tot timpul obstacolele y_2 și y_3" este exprimată prin formula LTL "$\Diamond y_1 \wedge \Box\neg(y_2 \vee y_3)$",
- vizitarea în viitor a oricărei din regiunile y_1 sau y_2 este exprimată prin "$\Diamond (y_1 \vee y_2)$",
- vizitarea în viitor a regiunii y_1 și rămânerea acolo mereu este sugerată de formula "$\Diamond\Box y_1$"

Formulele LTL sunt interpretate pe secvențe de lungime infinită generată de mișcarea robotului (chiar dacă evoluția robotului se oprește într-o anumită poziție, cuvântul generat va fi extins la infinit prin repetarea ultimei poziții). Acest aspect facilitează însă exprimarea prin specificații LTL a unor cerințe care implică o mișcare permanentă a robotului mobil (cum ar fi supravegherea unor regiuni).

In general, modelul robotului utilizat pentru abstractizarea discreta este bazat pe sisteme de tranziții sau procese Markov de decizie, adică pe un model bazat pe grafuri. Sarcina de nivel înalt data sub forma unei formule LTL este transformata într-un automat Büchi sau Rabin. Traiectoriile robotului pot fi calculate prin efectuarea produsului sincron ale modelului echipei cu automatul Büchi sau Rabin si utilizarea unui algoritm de găsire a celui mai scurt drum într-un graf (ce prezintă o complexitate timp polinomiala). Totuși, daca numărul de roboti din echipa creste, numărul de stări ale modelului echipei creste considerabil, fiind necesara efectuarea produsului sincron al diferitelor sisteme de tranziție ca in [16] sau duplicarea automatelor roboților ca in [17].

În [18], specificațiile au fost descrise prin entități LTL, fiind propusă o metodă de planificare în care modelarea spațiului de lucru este construită pe baza situațiilor de deadlock ce pot să apară. Aceste presupuneri pot fi interpretate ca restricții de mișcare a roboților care pot fi relaxate în momente de timp când nu prezintă interes pentru legea de control a sistemului. Aplicațiile acestei abordări au fost considerate în mai multe scenarii printre care și echipe de drone care pot fi utilizate în stingerea incendiilor [19].

Recent, pentru aplicațiile care necesită planificarea de mișcare a unei echipe de roboți precum și transportarea unor obiecte statice a fost propus un framework hibrid folosind specificații LTL [20, 21]. Frameworkul include legi de control pentru navigarea liberă de către roboți și preluarea obiectelor din regiunile specificate fără a exista coliziuni între roboți. Modelarea evoluției roboților și a stării obiectelor se face folosind un sistem de tranziții finite care este utilizat în proiectarea unei abordări de nivel înalt ce satisface specificații LTL.

CTL [14] poate fi de asemenea utilizat pentru specificarea sarcinilor roboților mobili. Adițional operatorilor temporali utilizați in LTL, acest limbaj permite cuantificarea de-a lungul drumurilor posibile dintr-o stare data prin utilizarea operatorilor A („pentru toate rulările") si E („pentru unele rulări"). Totuși, formulele CTL sunt restricționate astfel încât fiecare operator temporal trebuie imediat precedat de un cuantificator de drum. LTL si CTL sunt incomparabile, în sensul că există formule CTL ce nu pot fi exprimate în LTL și vice-versa. Un dezavantaj al CTL este acela că translarea din limbaj natural in formule CTL este

predispusă erorilor, pe când LTL este mai intuitiv. Mai mult, se pare ca există mai multe specificații interesante exprimabile in LTL si nu in CTL.

CTL* este un framework unificator pentru LTL si CTL [14]. Totuși, verificarea modelului pentru CTL* este costisitoare, iar expresivitatea acestui limbaj este prea complicata pentru robotica.

O altă abordare recenta, Metric Interval Temporal Logic (MITL) este descrisă în [22]. Autorii propun un framework de generare automată a secvențelor de control pentru sisteme multi-agent supuse la restricții de timp. Mișcarea agenților în spațiul de lucru este abstractizată în sisteme de tranziție individuale. Fiecărui agent îi este asignată o formulă în MITL, în paralel întregului sistem de agenți fiindu-i asignată o formulă colaborativă. Frameworkul este bazat pe metode de sinteză corecte prin construcție, astfel garantând că sistemul în buclă închisă va satisface specificația.

4. Logica Booleana

Planificarea acțiunilor unei echipe de roboți mobili poate proiectată astfel încât să poată îndeplini specificații Booleene atașate regiunilor de interes [4]. Detaliile de constructie ale unei astfel de specificaţii sunt prezentate in capitolul următor. Acestea sunt construite global pentru echipă fără a permite alocări de tipul robot singular – task. Acest obiectiv poate fi realizat prin modelarea sub formă de reţea Petri a mişcării echipei de roboţi mobili şi a proprietăţilor regiunilor de interes. Formula Booleana ataşată modelului discret este reprezentată printr-un set de inegalităţi liniare ale unor variabile binare, evaluarea acestor variabile fiind strâns legată de posibile secvenţe finite de tranziţii din reţeaua Petri. Se obţine astfel o problemă ce poate fi rezolvată printr-o abordare ILP (integer liniar programming). Soluţia obţinută este optimă în raport cu distanţa parcursă de echipă, evitându-se posibilele cazuri de congestie.

Concluzii A1.1*:*

În opinia noastră, un limbaj de specificare este „bun” dacă este *natural* (apropiat de limbajul uman), *expresiv* (permite specificarea unei clase bogate de sarcini) și algoritmii pentru analiza și sinteza controler-ului au *complexitate mică*.

În cele ce urmează vom adopta o soluție bazată pe logica booleana. Pentru mai mult de un robot aceasta specificație impune o cerință globala asupra vizitării sau evitării regiunilor, fără a permite impunerea de cerințe individuale precum vizitarea a doua regiuni disjuncte de către un același agent. Totuși, aceasta expresivitate mai mica, combinata cu un model bazat pe rețele Petri, permite obținerea de soluții ale căror complexitate este puțin influențată de numărul de roboți, topologia modelului rămânând neschimbată atunci când cardinalitatea echipei de roboți se modifică.

Activitatea 1.2. Algoritmi pentru planificarea mișcării

Obiectiv: *Dezvoltarea de algoritmi pentru planificarea mişcării echipei de roboti mobili pe baza specificaţiilor complexe.*

Secțiunea curentă descrie dezvoltările recente ale membrilor echipei proiectului privind planificarea automată a unei echipe de agenți mobili pe baza specificațiilor complexe. De asemenea, vor fi punctate aspecte pe care ne propunem să le investigăm și să le extindem în etapele următoare.

În cele ce urmează, propunem problema planificării unei echipe de roboti identici astfel încât sa fie îndeplinită o specificaţie Booleana definita peste un set de regiuni de interes. Mediul robotic este cunoscut si static. Specificaţia impune cerinţe Booleene asupra regiunilor vizitate in timpul mişcării echipei precum si asupra poziţiilor finale ale roboţilor. Specificaţia este furnizata global pentru echipa de roboti, fără a permite asignări specifice

robot-sarcina. Pentru a dezvolta o soluţie, propunem modelarea mişcării echipei si a satisfacţiei asociate regiunilor folosind un sistem cu evenimente discrete sub forma unei Reţele Petri cu ieşiri. Astfel, evităm problema dimensiunii spaţiului stărilor pentru modelul echipei de roboti, problema care apare în abordarea bazată pe sisteme de tranziții și modelarea echipei ca un produs al acestor sisteme (conducând la o creștere exponențială a numărului de stări) . Modelele bazate pe reţele Petri sunt scalabile in raport cu dimensiunea echipei de roboti, sub constrângerea ca aceştia sa fie identici. Adăugarea unui robot in echipa nu presupune modificarea structurii reţelei Petri, ci doar adăugarea unui nou jeton.

Astfel, presupunem o echipă de n roboți mobili identici care pot colabora, fiind controlați de o unitate centrală. Domeniul de lucru conține un set de regiuni de interes statice și este partiționat pe baza acestora, folosind algoritmi existenți, de exemplu bazați pe descompuneri în celule. Evoluția posibilă a echipei de roboți în domeniul de lucru este abstractizată folosind un model de tip rețea Petri cu ieșiri, notat în continuare Q. Pentru obținerea lui Q se construiește întâi o rețea Petri cu structura $N = <P, T, F>$, unde P și T sunt mulțimile finite de locații, respectiv tranziții, iar $F \subseteq (P \times T) \cup (T \times P)$ este setul arcelor. Ideea de bază în crearea rețelei N este de a asigna o locație fiecărui element al partiției mediului de evoluție (celulă), iar tranzițiile corespund mișcărilor posibile ale unui robot dintr-o celulă în alta adiacentă. Pentru simplitatea expunerii considerăm că toate arcele au pondere unitară, dar acest lucru este nerestrictiv, putându-se asigna ponderi bazate pe anumite funcții de cost dependente de exemplu de distanța medie parcursă sau de energia consumată pentru mișcarea unui robot din locația curentă într-una adiacentă a partiției, sau pentru efectuarea unei acțiuni.

Fiecare robot din echipă corespunde unui jeton ale rețelei Petri, astfel modelul Q având structura $Q = <N, \boldsymbol{m}_0, \Pi, h>$, unde $\boldsymbol{m}_0$ este marcajul inițial, adică vectorul cu valori întregi arătând în ce locații sunt plasați roboții la momentul inițial. $\Pi=\{\Pi_1, \Pi_2, \ldots, \Pi_{|\Pi|}\} \cup \varnothing$ este de ieșiri (observabile), unde regiunile de interes și eventual acțiunile ce pot fi efectuate în anumite locații sunt notate cu Π_i, iar $\varnothing$ notează observația vidă (o locație din domeniul de evoluție care nu aparține unei regiuni de interes și în care nu pot fi efectuate acțiuni). Funcția de observații $h : P \to 2^{\Pi}$ arată ieșirea fiecărei locații a lui Q, adică atunci când cel puțin un jeton este în locația p, $h(p)$ indică regiunea (sau regiunile) de interes în care se află roboții respectivi și/sau ce acțiuni pot efectua în poziția curentă. Pseudo-codul pentru construcția modelului Q este dat în Algoritmul 1, iar detalii suplimentare pot fi găsite în [4]. Algoritmul 1 este dedicat doar vizitării regiunilor de interes, dar precum a fost menționat unele simboluri Π_i pot referi acțiuni ce pot fi efectuate în anumite regiuni, de exemplu culegerea unor resurse din locații în care astfel de resurse există plasate.

Tranzițiile lui Q corespund mișcărilor robotice, iar starea m (poziția jetoanelor/roboților) obținută în urma execuției tranzițiilor dintr-un așa-numit vector de executare σ (care contorizează numărul de execuții ale fiecărei tranziție) este dată de ecuația de stare a rețelei Petri (1), unde $\boldsymbol{C}$ notează matricea de incidență a modelului:

$$\boldsymbol{m} = \boldsymbol{m}_0 + \boldsymbol{C} \cdot \boldsymbol{\sigma} \tag{1}$$

De menționat că modelul Petri obținut din Algoritmul 1 este o mașină de stare, fiecare tranziție având o singură locație de intrare și o singură locație de ieșire, ceea ce garantează că orice soluție m, σ a ecuației (1) corespunde unei stări ce poate fi atinsă, vectorul σ corespunzând unei secvențe fezabile de tranziții.

Algorithm 1: Construct the PN system $\mathcal{Q}$.

Input: Environment, regions Π, initial team deployment
Output: Team model $\mathcal{Q}$

Construct a cell decomposition of the environment based on the polygonal regions of interest from Π;
Associate each cell from decomposition to a place from P; let $P = \{p_1, p_2, \ldots, p_{|P|}\}$;
Let $T = \emptyset$, $F = \emptyset$, $\boldsymbol{w} = \mathbf{0}$;
for $p_i \in P$ **do**
    for $p_j \in P$, $p_i \neq p_j$ **do**
        if cells p_i and p_j are adjacent **then**
            Add transition $t_{i,j}$ to T;
            $F := F \cup \{(p_i, t_{i,j}), (t_{i,j}, p_j)\}$;
            $\boldsymbol{w}[t_{i,j}]$ = average distance traveled by a robot that moves from cell p_i to p_j;
for $p_i \in P$ **do**
    $\boldsymbol{m}_0[p_i]$ = no. of robots initially deployed in cell p_i;
    $h(p_i) = \{\Pi_j \in \Pi | \text{cell } p_i \text{ is included in region } \Pi_j\}$;

În cele ce urmează vom considera o specificație pentru misiunea echipei dată printr-o *formulă bazată pe logica Booleană*, după cum urmează. Se impune o formulă Booleană pe setul $P = P_t \cup P_f$, unde $P_t = \Pi$, iar $P_f = \{\pi_1, \pi_2, \ldots, \pi_{|\Pi|}\}$. Operatorii Booleeni folosiți sunt $\neg$ (negație), $\wedge$ (conjuncție), $\vee$ (disjuncție) și se consideră că formulele sunt în forma canonică conjunctivă. Seturile P_t și P_f se referă la aceleași regiuni și acțiuni, dar P_t sugerează că regiunile/acțiunile sunt vizitate/executate de-a lungul traiectoriilor roboților, iar P_f include cerințe ce trebuie îndeplinite în starea finală a echipei (atunci când roboții se opresc). Formalismul privind interpretarea semantică a formulelor pe baza secvențelor de observabile ale modelului Q poate fi găsit în [4] și se bazează pe așa-numiții vectori caracteristici v_i ai observației Π_i, i=1,...,n. De exemplu, specificația $\varphi = (\Pi_1 \vee \Pi_2) \wedge \neg\pi_1 \wedge \neg\Pi_3$ impune ca de-a lungul traiectoriilor robotice regiunile sau acțiunile Π_1 sau Π_2 să fie vizitate/efectuate, Π_3 să nu fie evitată totdeauna, iar Π_1 să nu fie adevărată în starea finală (de exemplu niciun robot să nu se oprească în zona notată Π_1).

Lucrarea [4] dezvoltă o soluție algoritmică pentru planificarea echipei de roboți pe baza oricărei specificații Booleene construită având în vedere sintaxa și semantica de mai sus. Ca idee de bază, specificația este translată într-o serie de inegalități lineare pe baza unor variabile binare cuprinse într-un vector $\boldsymbol{x}$, iar pentru rețeaua Petri se consideră o serie de k marcaje (stări) intermediare, între două marcaje succesive un jeton putându-se mișca prin cel mult o tranziție. Problema este apoi translată într-o problemă de programare liniară mixtă cu numere întregi (eng. Mixed Integer Linear Programming - MILP), ecuația (2) formalizând această scriere. Pentru detalierea notațiilor adiționale din (2) cititorul este rugat să consulte referința [4].

$$
\begin{aligned}
& \min \lambda \cdot \boldsymbol{w}^T \cdot \sum_{i=1}^{k} \boldsymbol{\sigma}_i + \mu \cdot b \\
\text{s.t.} \quad & \boldsymbol{m}_i = \boldsymbol{m}_{i-1} + \boldsymbol{C} \cdot \boldsymbol{\sigma}_i, i = 1, \ldots, k \\
& \boldsymbol{m}_{i-1} - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_i \geq 0, i = 1, \ldots, k \\
& \sum_{\gamma \in \mathcal{P}} (\alpha_i(\gamma) \cdot x_\gamma) \geq 1 + \sum_{\gamma \in \mathcal{P}} \min(\alpha_i(\gamma), 0), \quad \forall \varphi_i \\
& N \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_k, \quad \forall \gamma \in \mathcal{P}_f \\
& x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_k, \quad \forall \gamma \in \mathcal{P}_f \\
& N \cdot (k+1) \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \left(\sum_{i=0}^{k} \boldsymbol{m}_i \right), \quad \forall \gamma \in \mathcal{P}_t \\
& x_\gamma \leq \boldsymbol{v}_\gamma \cdot \left(\sum_{i=0}^{k} \boldsymbol{m}_i \right), \quad \forall \gamma \in \mathcal{P}_t \\
& \left(\boldsymbol{Post} \cdot \sum_{i=1}^{k} \boldsymbol{\sigma}_i \right) \leq b \cdot \mathbf{1}^T \\
& \boldsymbol{m}_i \in \mathbb{N}_{\geq 0}^{|P|}, \boldsymbol{\sigma}_i \in \mathbb{N}_{\geq 0}^{|T|}, i = 1, \ldots, k \\
& \boldsymbol{x} \in \{0,1\}^{|\mathcal{P}|}, b \geq 0.
\end{aligned}
\tag{2}
$$

Problema de optimizare (2) este rezolvată folosind rutine software existente, obținându-se un vector de executare fezabil σ pentru modelul Q. Ulterior, σ este translat în planuri de mișcare și executare de acțiuni pentru echipa de roboți mobili. Drept exemplu, considerăm domeniul de evoluție din Figura 1, în care există 3 roboți și 5 regiuni de interes, și specificația $\varphi = \neg\Pi_2 \wedge \Pi_1 \wedge \neg\pi_1 \wedge \pi_3 \wedge \pi_4 \wedge \pi_5$. Specificația impune ca regiunea 2 să fie evitată, regiunea 1 să fie vizitată de-a lungul traiectoriilor, dar niciun robot să nu se oprească în ea, iar în ultimele 3 regiuni să se oprească câte un robot. Traiectoriile robotice obținute sunt reprezentate în Figura 1. Problema MILP (2) a inclus 1891 variabile (din care 1400 întregi și10 binare), 480 de restricții de egalitate și 554 inegalități liniare, fiind rezolvată în aproximativ 0,01 secunde.

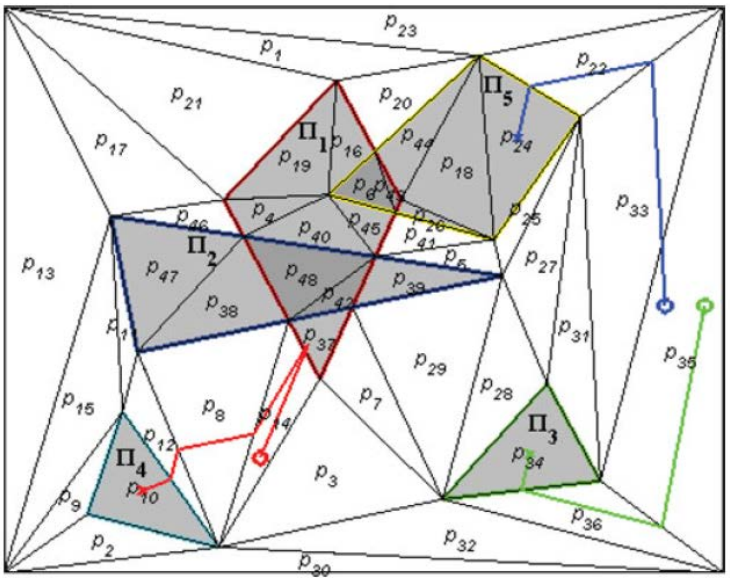


Figura 1. Domeniu de evoluție cu 5 regiuni și 3 roboți plasați inițial în pozițiile marcate cu "○". Modelul Q are 48 de locații și 140 de tranziții, iar traiectoriile obținute sunt reprezentate cu linii frânte, roboții oprindu-se în punctele marcate cu "x".

Concluzii A1.2:

Este clar că o clasă de astfel de specificații Booleene include multe specificații robotice utile, dar are și unele limitări. Pe de o parte, față de formule LTL, utilizatorul nu poate impune o ordine specifică în care regiunile sau acțiunile sunt satisfăcute de-a lungul traiectoriilor. Dar, formulele Booleene cu soluția bazată pe modele de tip rețea Petri aduc importante avantaje computaționale față de formulele LTL și algoritmii de rezolvare bazați pe sisteme de tranziții, formalism în care rezolvarea exemplului anterior ar fi putut dura până la câteva ore. În viitor, ne propunem tratarea formulelor LTL folosind tot modele de tip rețea Petri, în speranța de a obține soluții în timpi mult mai scurți. Pentru aceasta, ne vom baza pe pașii preliminari din [23], unde formulele sunt restricționate la subclasa co-safe LTL și nu includ acțiuni. Pe de altă parte, formulele pe care le considerăm deocamdată sunt globale, ceea ce înseamnă că echipa de roboți trebuie să îndeplinească misiunea, fără a putea impune anumite restricții unui anume robot. Acest fapt limitează deocamdată posibilitatea de a avea acțiuni de genul "ridică piesa A din regiunea 1 și depune-o în regiunea 2", deoarece specificația globală poate impune vizitarea regiunilor 1 și 2 de către roboți diferiți, de exemplu din cauza optimizării numărului total de tranziții. Pe viitor, vom încerca includerea în soluție de atare restricții specifice roboților, folosind pe de o parte modelele de tip rețea Petri și pe de altă parte inspirându-ne din metodele menționate în Secțiunea 1.1 a acestui raport. În plus, cercetările raportate în [24] au condus la soluții pentru probleme simple de planificare fără a utiliza modele de tip rețea Petri, dar transpunând problema într-un formalism apropiat de control predictiv, necesitând tot rezolvarea unor probleme de optimizare intermediare. Ne propunem și investigarea acestor idei pentru specificații mai complexe, în speranța că traiectoriile obținute vor fi mai mult îmbunătățite din punct de vedere al diverselor metrici ce pot fi integrate în funcțiile de cost ale problemelor de optimizare implicate.

Activitatea 1.3. Definirea unui domeniu de evoluție 3D

Obiectiv: *Proiectarea si dezvoltarea unui domeniu de evoluție 3D pentru roboți de tip dronă.*

Specificații generale

Quadcopters (Dronele) sunt sisteme robotice cu patru motoare atractive datorită simplității, agilității acestora cât și gamei largi de aplicații. Aceste platforme robotice sunt utilizate tot mai frecvent în aplicații precum servicii de monitorizare, mentenanță și cartografiere a unei zone de interes, servicii de livrare. Particularitatea quadcopterului este dată de faptul că două elice opuse se rotesc în sensul acelor de ceasornic, iar celelalte două se rotesc în sens contrar. Analiza și proiectarea unor algoritmi de reglare joacă un rol fundamental în aplicațiile cu drone, asigurând astfel stabilitatea, rejecția perturbațiilor, precum și funcționarea optimală cu obținerea performanțelor dorite. În acest context, în ultimii ani, s-a consemnat un interes deosebit pentru integrarea dronelor în aplicații atât pentru medii exterioare, cât și interioare.

Comunicația cu o dronă de mici dimensiuni, Crazyflie 2.0, proiectarea și dezvoltarea unui domeniu de evoluție 3D pentru drone și modelarea dinamicii acesteia [25], [26] precum și controlul și planificarea traiectoriilor cu vizitarea unor regiuni de interes reprezintă subiectele de interes ale acestui studiu. Algoritmul de reglare implică, în primul rând, găsirea unor modele matematice caracterizate printr-un grad ridicat de generalitate, în sensul că trebuie să fie valabile pentru o gamă largă de valori ale mărimilor de intrare, descriind comportarea dronei pe întreaga gamă de funcționare. Mai mult, modelul matematic adoptat trebuie să aibă o structură suficient de complexă pentru a surprinde dinamica, dar în același

timp simplă pentru a facilita proiectarea și implementarea în timp real al algoritmului de control.

Spațiul de lucru în care evoluează quadcopterul este ilustrat în Figura 2 și reprezintă o platformă utilizată pentru controlul roboților mobili înconjurată de o plasă de siguranță și un senzor Kinect amplasat în plafon in centrul platformei. Platforma are dimensiunea 4 x 2.5m, iar înălțimea de 2.8m.

Drona Crazyflie 2.0

Quadcopter-ul Crazyflie 2.0 reprezintă a doua generație a platformei de dezvoltare creată de Bitcraze. Proiectul a fost și este în continuare dezvoltat în paradigma „open source" (licențiere CC BY-SA 3.0) fiind puse la dispoziție atât informațiile despre structura hardware, cât și codul sursă pentru cele două microcontrollere de pe platformă, respectiv pentru aplicațiile de tip client. Caracteristicile generale ale dronei sunt illustrate în Tabel 1.

Tabel 1

Masă	27 g
Dimensiuni (LxÎxA, privire de sus)	92x92x29 mm (fără elicii, cu suporturile de plastic ale motoarelor)
Timp de zbor	Aprox. 7 minute
Timp de încărcare	Aprox. 40 minute

În determinarea ecuațiilor care descriu dinamica quadcopterului este necesară definirea unui sistem de referință neinerțial, în configurație "X", având originea în centrul dronei, și a unui sistem de referință global, sistemul inerțial raportat la centrul Pământului, ilustrate în Figura 2. Menționarea sistemului de referință precizează față de ce sistem se vor raporta măsurătorile preluate de la senzori. Convenția stabilită pentru sistemul de referință inerțial al dronei Crazyflie 2.0 presupune definirea axelor astfel: axa X îndreptată spre Est, axa Y spre Nord și valori pozitive ale altitudinii (axa Z) pentru orientare în sus.

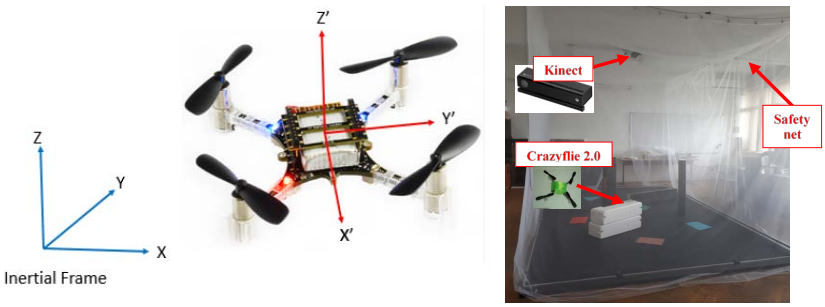


Figura 2. Sistem de referință inerțial CoG, sistem de referință atașat dronei (dreapta) și domeniul de evoluție

Structură hardware

Crazyflie 2.0 este un sistem multiprocesor cu o structură asimetrică ilustrată în Figura 3 și include:

- Un microcontroller cu un nucleu ARM Cortex-M4F (cu unitate de calcul în virgulă mobilă) care rulează algoritmii de control și aplicațiile utilizator;

- Un microcontroller cu un nucleu ARM Cortex-M0 ce include și o interfață radio în banda 2,4 GHz care implementează stiva de comunicație, respectiv mecanismul de control al alimentării dronei;
- O unitate inerțială ce include un accelerometru, un giroscop și un magnetometru. Valorile citite de la cei trei senzori sunt folosite pentru feedback în cadrul algoritmilor de control;
- Driver-ele pentru motoare sunt unidirecționale, fără feedback pentru turație;
- La dronă se pot atașa unul sau mai multe module de expansiune (eng. decks) ce implementează diferite funcționalități - de la afișaje cu LED-uri RGB la sisteme de poziționare relativă, respectiv de detecție a înălțimii de zbor sau a deplasării laterale.

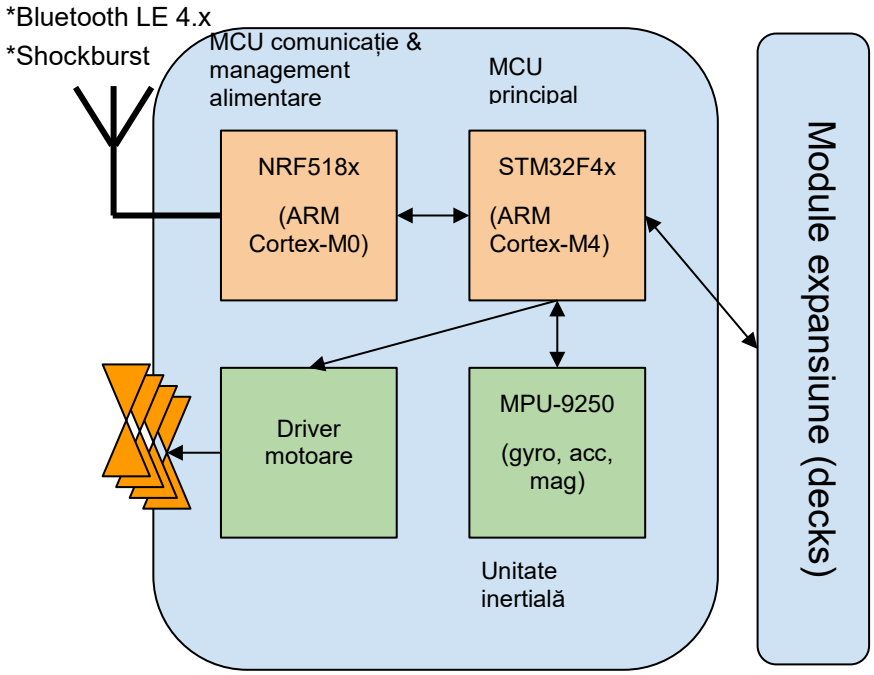


Figura 3 Structura hardware a dronei Crazyflie 2.0

La pornirea dronei, modulele instalate sunt autentificate printr-o interfață 1-wire și în consecință sunt activate modulele software corespunzătoare pentru inițializare și control.

Modelul matematic al dronei Crazyflie 2.0

Determinarea modelului matematic al dronei s-a realizat pe baza ecuațiilor Newton-Euler ce descriu dinamica quadcopterului. Modelul matematic stă la baza implementării unui simulator ce a fost mplementat în mediul Matlab. Parametrii constructivi și mărimile care intervin în modelul matematic sunt în concordanță cu drona Crazyflie 2.0 [26], [28]. Modelul obţinut poate oferi informaţii complete şi despre comportarea subsistemelor componente, în deplină concordanţă cu implementarea fizică a quadcopterului. Astfel, simulatorul construit pornind de la reprezentări matematice își găsește utilizare în testarea prin simulare a unor algoritmi de control pentru specificații de mișcare și efectuare de acțiuni. Limitările pe care acest model le impune rezultă din faptul că relaţiile matematice sunt adesea scrise considerând o serie de ipoteze de lucru și simplificări.

Modelul matematic a fost configurat conform informaţiilor tehnice şi parametrilor reali ai dronei Crazyflie 2.0, fiind validat experimental, iar performanţele sunt apreciate prin comparaţii cu datele reale oferite din funcționarea în buclă deschisă a quadcopterului. În acest scop, este propusă arhitectura din Figura 4, care este alcătuită din următoarele componente: subsistemul Crazyflie 2.0 conținând modelul matematic, blocul algoritmilor de control, un

subsistem pentru interfațarea utilizatorului cu drona și monitorizarea datelor preluate de la senzori și trimiterea de comenzi. Într-o primă etapă, poziționarea dronei și orientarea acesteia într-un domeniu de evoluție indoor, s-a realizat prin intermediul unui senzor extern ce folosește un sistem de evaluare vizuală de tip Kinect. În timpul experimentelor în timp real, s-a observat o întârziere în achiziția datelor prin Kinect și, prin urmare, se dorește utilizarea altor metode de interfațare și control pentru drona Crazyflie 2.0. Determinarea și implementarea unui model matematic facilitează proiectarea și implementarea în timp real a unor algoritmi de control pentru dronă.

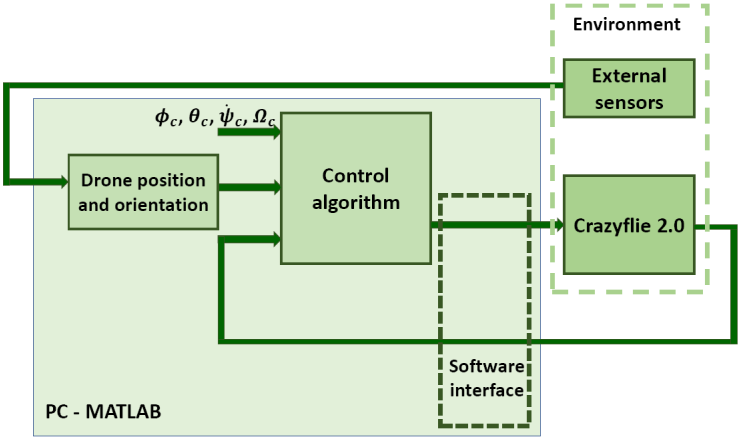


Figura 4 Arhitectura propusă pentru realizarea experimentelor cu drona Crazyflie 2.0

În implementarea modelului matematic s-au folosit 12 stări: pozițiile *x, y* și *z*, exprimate în sistemul de coordonate inerțial (CoG), vitezele liniare ale dronei *u,v,w* de-a lungul celor 3 axe, unghiurile roll (ϕ), pitch (θ) și yaw(ψ) și vitezele unghiulare ale dronei *p*, *q*, *r*. Modelul matematic neliniar este ilustrat în relația (1), în care toate mărimile și parametri constructivi sunt detaliați în articolul [25],[26].

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \\ \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{r} \\ \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} w(s_\psi s_\phi + c_\psi c_\phi s_\theta) - v(s_\psi c_\phi - c_\psi s_\phi s_\theta) + u \times c_\psi c_\theta \\ v(c_\psi c_\phi + s_\psi s_\phi s_\theta) - w(c_\psi s_\phi - s_\psi c_\phi s_\theta) + u \times s_\psi c_\theta \\ w \times c_\psi c_\phi - u \times s_\theta + v \times s_\phi c_\theta \\ p + r \times c_\phi t_\theta + q \times s_\phi t_\theta \\ q \times c_\phi - r \times s_\phi \\ r\frac{c_\phi}{c_\theta} + q\frac{s_\phi}{c_\theta} \\ rv - qw + g \times s_\theta \\ pw - ru - g \times s_\phi c_\theta \\ qu - pv + \frac{U_1}{m} - g \times c_\theta c_\phi \\ \frac{U_2 + pq(I_{xx} - I_{yy})}{I_{zz}} \\ \frac{U_3 + pr(I_{zz} - I_{xx})}{I_{yy}} \\ \frac{U_4 + pq(I_{zz} - I_{yy})}{I_{xx}} \end{bmatrix} \quad (3)$$

Mărimile de intrare considerate pentru modelul matematic al dronei în formalismul intrare-stare-ieșire sunt exprimate în relația (4), pe baza celor 4 viteze unghiulare ale fiecărui motor al quadcopterului:

$$
\begin{aligned}
U_1 &= C_T\left(\omega_1^2+\omega_2^2+\omega_3^2+\omega_4^2\right)\\
U_2 &= dC_T\sqrt{2}\left(-\omega_1^2-\omega_2^2+\omega_3^2+\omega_4^2\right)\\
U_3 &= dC_T\sqrt{2}\left(-\omega_1^2+\omega_2^2+\omega_3^2-\omega_4^2\right)\\
U_4 &= C_D\left(-\omega_1^2+\omega_2^2-\omega_3^2+\omega_4^2\right)
\end{aligned}
\tag{4}
$$

În firmware-ul Crazyflie 2.0, mărimile de intrare sunt semnale PWM transmise fiecărui motor și sunt reprezentate pe 16 biți, în gama [0-65535]. Între viteza unghiulară a fiecărui motor și semnalul PWM există o relație liniară, dată de:

$$\mathrm{RPM} = 0.2685 \times \mathrm{PWM} + 4070.3 \tag{5}$$

În încercarea de a-si menține poziția de echilibru, într-un punct fix la o anumită altitudine (hover mode), forța generată de elicile dronei trebuie să compenseze greutatea dronei. Pornind de la ipoteza că drona este perfect simetrică, toate cele 4 motoare trebuie să se rotească cu aceeași viteză astfel încât să-și mențină poziția de echilibru într-un punct fix. Validarea modelului matematic dezvoltat s-a realizat prin experimente în timp real, folosind structura internă din Figura 5, având regulatoare încorporate pentru controlul dronei.

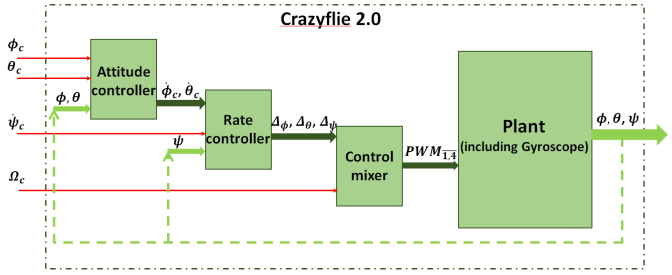


Figura 5 Structura internă cu algoritmi de reglare încorporați

Într-un prim pas se urmăreşte validarea modelului matematic pe setul de date oferit din funcţionarea dronei în circuit deschis, urmând ca apoi performanţele simulatorului să fie testate în circuit închis. În Figura 6 este ilustrată traiectoria dronei pe axa z până la o înălțime de 1.9m, date preluate din simularea numerică, comparativ cu setul de date preluat din funcţionarea în timp real a quadcopterului, rezultatele validând modelul matematic.

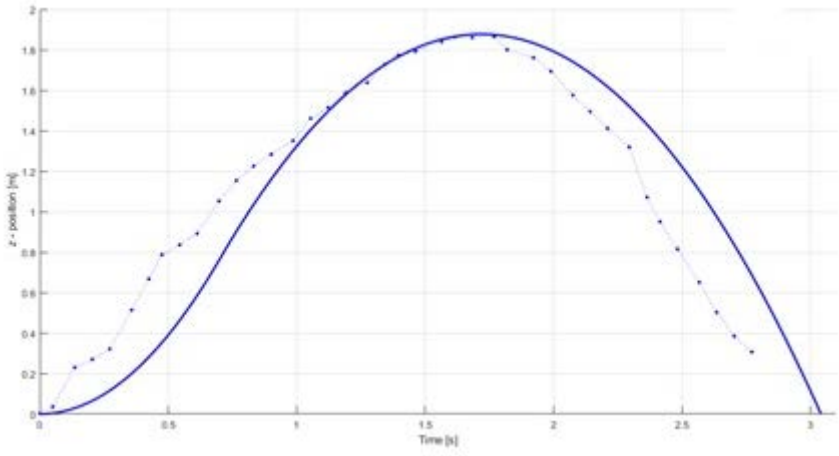


Figura 6 Studiu comparativ: evoluție traiectorie dronă axa z:
simulare numerică (linie continua) și testare în timp real (linie punctată)

În cadrul experimentelor, unghiurile de referință roll și pitch sunt setate la valoarea 0, în contextul în care informaţiile preluate de la sistemul Kinect sunt disponibile cu întârzieri de aproximativ 0.15 s, nefiind fezabile pentru controlul în timp real al dronei, sistemul avand o dinamică rapidă.

Mecanisme de comunicație și interacțiune cu drona

Microcontroller-ul dedicat pentru comunicație implementează o stivă duala bazată pe Bluetooth LE, respectiv Shockburst (protocol proprietar NordicSemi). Comunicația BLE este folosită pentru controlul din aplicațiile mobile (Android, iOS), aplicații ce implementează un control manual, iar comunicația proprietară este folosită pentru implementări mai complexe, prin intermediul unui PC (denumit în continuare client). Astfel, PC-ul trebuie echipat cu un adaptor USB-RF dedicat (CrazyRadio) ce permite comunicarea bidirecțională cu drona (Figura 7).

La nivel aplicație, pentru comunicațiile USB și radio există implementat protocolul CRTP (Crazy Real-Time Protocol), iar pentru comunicația între microcontrollere se folosește protocolul Syslink. Ambele protocoale sunt proprietare Bitcraze.

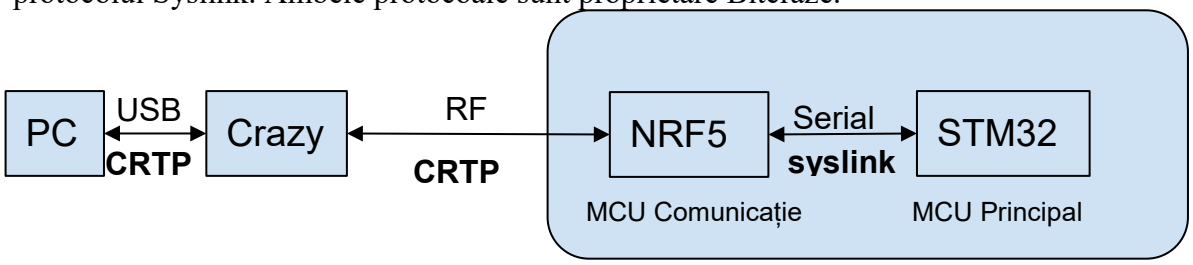


Figura 7 Infrastructura de comunicație între Crazyflie 2.0 și PC

Syslink este un protocol multifuncținal prin intermediul căruia cele două microcontrollere schimbă date, una dintre facilitățile oferite de acesta fiind împachetarea și transmiterea cadrelor CRTP. CRTP definește un mecanism de comunicație bazat pe sarcini (eng. tasks), respectiv funcționalități. Acestea sunt codificate prin intermediul unui antet de 1 octet, după cum urmează:

- Port (4 biți) - identifică sarcina sau funcționalitatea
- Link (2 biți) - nefolosit
- Channel (2 biți) - identifică sub-sarcina sau sub-funcționalitatea

În versiunea curenta, există definite 11 porturi:

Nume	**ID** **(4 biți)**	**Descriere**
Console	0	Port unidirecțional pentru transferul unor șiruri de caractere (stare, depanare etc.) de la dronă către client (similar cu o consolă text).
Parameters	2	Scriere/citire parametri dronă.
Commander	3	Trimitere puncte de control (set-points) pentru regulatoarele implementate pe dronă.
Memory access	4	Accesarea memoriei nonvolatile a dronei (1-Wire și I2C).
Data logging	5	Blocuri de variabile a căror valoare va fi trimisa periodic către PC. Parametrii sunt setați prin intermediul unei macrodefiniții în codul sursa de pe dronă.
Localization	6	Port folosit pentru citirea/scrierea informațiilor de localizare, atunci când există un astfel de sistem disponibil. Definește două

		moduri de localizare: externă și generică.
Generic setpoint	7	Trimitere puncte de control pentru regulatoarele implementate pe dronă (variante de comandă derivate din controlul de bază al port-ului 3)
Setpoint High Level	8	Interfață pentru controlul unor grupuri (eng. swarm) de drone.(variante de comandă derivate din controlul de bază al port-ului 3)
Platform	13	Comenzi diverse pentru dronă (ex. depanare, control alimentare)
Client-side debugging	14	Depanare client - nu se comunică efectiv cu drona.
Link layer	15	Testare legatura radio.

Detalii de implementare

Porturile necesare pentru a controla și pentru a citi informațiile de stare de la dronă sunt *Commander, Parameters* și *Data logging*. Ca interfață, port-urile *Parameters* și *Data logging* sunt similare, însă din punct de vedere funcțional există următoarele diferențe:

- Port-ul *Parameters* este folosit pentru scriere/citirea individuală a variabilelor interne ale dronei. Astfel, pot fi modificate constantele algoritmilor de reglare, pot fi resetați algoritmii, pot fi modificați parametrii de zbor etc.
- Port-ul *Data logging* este folosit pentru citirea uneia sau mai multor variabile în cazul în care frecvența cu care se modifică valorile lor este ridicată. Astfel, poate fi configurată rata de achiziție (cel puțin de ordinul zecilor de milisecunde) și valorile recepționate pot fi salvate la client.

Variabilele care se doresc expuse către interfața CRTP trebuie marcate prin intermediul unui set de macrodefiniții la finalul modulului sursă. Un exemplu din modulul care implementează regulatorul de stabilizare al zborului este prezentat mai jos.

```
PARAM_GROUP_START(pid_attitude)
PARAM_ADD(PARAM_FLOAT, roll_kp, &pidRoll.kp)
PARAM_ADD(PARAM_FLOAT, roll_ki, &pidRoll.ki)
PARAM_ADD(PARAM_FLOAT, roll_kd, &pidRoll.kd)
PARAM_ADD(PARAM_FLOAT, pitch_kp, &pidPitch.kp)
PARAM_ADD(PARAM_FLOAT, pitch_ki, &pidPitch.ki)
PARAM_ADD(PARAM_FLOAT, pitch_kd, &pidPitch.kd)
PARAM_ADD(PARAM_FLOAT, yaw_kp, &pidYaw.kp)
PARAM_ADD(PARAM_FLOAT, yaw_ki, &pidYaw.ki)
PARAM_ADD(PARAM_FLOAT, yaw_kd, &pidYaw.kd)
PARAM_GROUP_STOP(pid_attitude)
```

Pe baza acestui set de macrodefiniții este creată o structură într-o secțiune distinctă (.param) din memoria flash a microcontrollerului, structură ce conține tipul, numele și adresele variabilelor. Accesul in scriere/citire prin intermediul CRTP se realizeaza pe baza numelor stocate in aceste structuri. Prin intermediul CRTP poate fi accesat un cuprins (TOC) al grupurilor de parametri, respectiv al parametrilor, deoarece prin modificarea firmware-ului e posibil sa se adauge sau sa fie eliminati parametri. Mai multe detalii pot fi găsite în referința [28]. Prin intermediul port-ului *Commander* poate fi trimisă o sigură comandă către dronă: *send_setpoint(roll, pitch, yaw_rate, thrust).*

Mărimile *roll* și *pitch* sunt exprimate în grade, *yaw_rate* este exprimat în grade/secundă, iar *thrus*t este exprimat în valoare absolută pe 16 biți - limitată între 1000 și 60000 la recepția comenzilor. În urma testelor efectuate s-a observat că acest mod de control, chiar dacă asigură stabilizarea dronei, suferă din lipsa unui mecanism de reacție pentru a

indica deplasarea dronei pe cele trei axe. Astfel, dacă se dorește zborul la punct fix sau zborul la o anumită înălțime, apare problema reglării (manuale, experimentale) a t*hrust*-ului deoarece drona are nevoie de portanță mai mare la decolare și în apropierea solului. Mai mult, datorită distribuirii inegale a greutății, respectiv a efectelor dinamice (vibrații induse de motoare), drona intră în derivă pe axele x și y, fiind necesar și de această dată un proces de calibrare manuală.

Concluzii A1.3:

Domeniul de lucru dezvoltat este pretabil experimentelor cu dronele de mici dimensiuni Crazyflie 2.0. A fost studiat modelul matematic și structura dronelor, aspecte care vor permite dezvoltarea și testarea ulterioară a diverselor strategii de control. S-au realizat rutine de comunicație cu drona Crazyflie 2.0, urmând extinderea acestora pentru interfațarea cu plăci auxiliare dronei. Pentru controlul în timp real al dronei trebuie asigurat feedback-ul pe poziție și orientare. O încercare în acest sens a fost utilizarea unui senzor Kinect, însă întârzierile aferente achiziției și procesării imaginilor sunt prea mari în raport cu dinamica rapidă a roboților în cauză. Studii viitoare vor fi direcționate spre utilizarea unor echipamente auxiliare pentru preluarea poziției unui quadcopter Crazyflie 2.0.

Bibliografie - etapa 1

[1] H. Choset, K.-M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.-E. Kavraki, and S. Thrun, “Principles of robot motion: Theory, algorithms, and implementations.” Boston, MIT Press, 2005

[2] S. M. LaValle, Planning Algorithms. Cambridge, 2006

[3] M. D. Berg, O. Cheong, and M. van Kreveld, Computational Geometry: Algorithms and Applications, 3rd ed. Springer, 2008

[4] C. Mahulea, M. Kloetzer, "Robot Planning Based on Boolean Specifications Using Petri Net Models", IEEE Transactions on Automatic Control (TAC), vol. 63 (7), pp. 2218 - 2225

[5] E. Vitolo, C. Mahulea, and M. Kloetzer, “A computationally efficient solution for path planning of mobile robots with boolean specifications,” in 21st International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 2017, pp. 63–69.

[6] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, “Symbolic planning and control of robot motion [grand challenges of robotics],” IEEE Robotics & Automation Magazine, vol. 14, no. 1, pp. 61–70, 2007.

[7] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, “Temporal logic motion planning for dynamic robots,” Automatica, vol. 45, no. 2, pp. 343–352, 2009.

[8] M. Guo and D. V. Dimarogonas, “Multi-agent plan reconfiguration under local ltl specifications,” The International Journal of Robotics Research, vol. 34, no. 2, pp. 218–235, 2015.

[9] G. Xue and Y. Ye, “An efficient algorithm for minimizing a sum of Euclidean norms with applications,” SIAM Journal on Optimization, vol. 7, pp. 1017–1036, 1997.

[10] L. Qi, D. Sun, and G. Zhou, “A primal-dual algorithm for minimizing a sum of Euclidean norms,” Journal of Computational and Applied Mathematics, vol. 138, no. 1, pp. 127 – 150, 2002.

[11] R. Gonzalez, M. Kloetzer, and C. Mahulea, “Comparative study of trajectories resulted from cell decomposition path planning approaches,” in 21st International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 2017, pp. 49–54.

[12] C. Cassandras and S. Lafortune. Introduction to Discrete Event Systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. second edition.

[13] R. Kumar and V. K. Garg. Modeling and Control of Logical Discrete Event Systems. Kluwer, Boston, MA, 1995.

[14] E. Clarke, D. Peled, and O. Grumberg. Model checking. MIT Press, 1999.

[15] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, Handbook of Theoretical Computer Science: Formal Models and Semantics, volume B, pages 995–1072. MIT Press, 1990.

[16] M. Kloetzer and C. Mahulea, “LTL-based planning in environments with probabilistic observations,” IEEE Trans. Autom. Sci. Eng., vol. 12, no. 4, pp. 1407–1420, Oct. 2015.

[17] F. Imeson and S.-L. Smith, “Multi-robot task planning and sequencing using the SAT-TSP language,” in Proc. IEEE Conf. Robot. Autom., May 2015, pp. 5397–5402.

[18] J. Alonso-Mora, J. DeCastro, V. Raman, D. Rus, H. Kress-Gazit, “Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles”, Autonomous Robots, 42(4), pp.801-824, 2017.

[19] J. Shaffer, E. Carrillo, H. Xu, “Receding Horizon Synthesis and Dynamic Allocation of UAVs to Fight Fires”, IEEE Conference on Control Technology and Applications (CCTA), 21-24 August 2018.

[20] C. Verginis, D. Dimarogonas, “Multi-Agent Motion Planning and Object Transportation under High Level Goals”, IFAC-PapersOnLine, Vol. 50(1), July 2017, Pages 15816-15821.

[21] C. Verginis, D. Dimarogonas, “Motion and Cooperative Transportation Planning for Multi-Agent Systems under Temporal Logic Formulas”, arXiv:1803.01579, 2018.

[22] S. Andersson, A. Nikou, D. Dimarogonas, “Control Synthesis for Multi-Agent Systems under Metric Interval Temporal Logic Specifications”, IFAC-PapersOnLine, Elsevier, Vol. 50, p. 2397-2402, 2017.

[23] M. Kloetzer and C. Mahulea, "Multi-robot path planning for syntactically co-safe LTL specifications," *13th International Workshop on Discrete Event Systems (WODES)*, Xi'an, China, pp. 452-458, 2016.

[24] E. Vitolo, C. Mahulea, M. Kloetzer, "Path-planning in Discretized Environments with Optimized Waypoints Computation", IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Torino, Italy, 2018.

[25] S. Huștiu, M. Lupașcu, Ș. Popescu, A. Burlacu, M. Kloetzer, "Stable Hovering Architecture for Nanoquadcopter Applications in Indoor Environments", IEEE 22nd International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 2018.

[26] C. Luis and J. Le Ny. “Design of a trajectory tracking controller for a nanoquadcopter”, Technical report, Mobile Robotics and Autonomous Systems Laboratory, Polytechnique Montreal, 2016.

[27] A.A.J. Lefeber. “Controlling of a single drone. Hovering the drone during flight modes.” Technical Report TU/E Eindhoven, Department of Mechanical Engineering, 2015.

[28] Crazyflie 2.0 documentation, available at: https://wiki.bitcraze.io/projects:crazyflie2:index, 2018.

Etapa 2 – Soluții algoritmice și experimente preliminare

Activitatea 2.1. Algoritmi de planificare pentru specificaţii de mişcare şi efectuare de acţiuni

Alocarea optimă de sarcini pentru îndeplinirea unui obiectiv asumat de o echipă de agenţi mobili poate fi realizată în diferite moduri. În cel mai simplu caz, obiectivul echipei se îndeplineşte prin minimizarea sumei tuturor costurilor agenţilor individuali. Cu toate acestea, în funcție de modul în care sunt definite costurile agenţilor, acest lucru ar putea să nu fie mereu de dorit. De exemplu, să presupunem cazul în care costul indică timpul necesar executării sarcinilor atribuite agentului respectiv. Cum agenţii pot evolua în paralel, timpul total necesar pentru finalizarea misiunii complete nu este dat de suma costurilor individuale. Acest timp reprezintă costul maxim dintre agenţi [1].

Planificarea mişcării pentru sisteme în aplicaţii reale implică restricţii suplimentare de resurse [2-4]. O echipă de agenţi mobili trebuie să țină cont de aceste restricţii în avans pentru a putea realiza planificarea mişcării [5]. De exemplu, dacă sunt necesare resurse externe pentru unele dintre sarcini, agenţii trebuie să se coordoneze nu numai în alocarea sarcinilor, ci și alocarea acestor resurse pentru a evita conflictele din planurile de execuție pentru sarcinile individuale.

Problema planificării și controlului sigur pentru echipe de drone prezintă o importanță critică, pe măsură ce domeniul de activitate atribuit acestor sisteme crește. În [6] autorii prezintă o abordare pentru a rezolva această problemă pentru misiuni multi-quadrotor. Având în vedere o misiune exprimată în Semnal Logic Temporal (STL), abordarea generează traiectorii cu robusteţe maximă pentru dronele care satisfac specificația STL în timp continuu. De asemenea,

se demonstrează că restricțiile incluse în optimizare garantează că aceste traiectorii pot fi urmărite aproape perfect de drone folosind sisteme comerciale de control ale poziţiei şi orientării. Soluţia propusă evită simplificarea excesivă a abstractizării întâlnită în multe metode de planificare, păstrând în același timp expresivitatea misiunilor codificate în STL permițând rezolvarea de cerințe complexe spațiale, temporale și reactive. Prin experimente, atât în simulare cât și pe sisteme reale, soluţia propusă arată performanța, scalabilitatea și aplicabilitatea în timp real. Implementarea soluţiei propuse a fost evaluată pentru o echipă de două drone Crazyflie2.0. În definirea problemei de planificare nu au fost incluse în mod distinct acţiuni pe care echipa să le îndeplinească în paralel cu mişcarea.

Proiectarea algoritmilor de planificare cu specificaţii de nivel înalt pentru mişcare şi efectuare de acţiuni reprezintă un subiect de interes în domeniul roboticii. În literatură sunt descrise soluţii pentru mobilitate şi acţiuni doar pentru un agent, în timp ce pentru echipe de agenţi mobili se consideră în special planificarea mişcării. În funcţie de formularea problemei au fost propuse un număr foarte redus de probleme de planificare şi alocare simultană de sarcini pentru echipe de agenţi mobili. În [7,8] autorii propun o soluţie pentru următoarea problemă de tip STAP (Simultaneous Task Allocation and Planning):

Se consideră o descriere de nivel înalt pentru o misiune specifică, o echipă de roboţi mobili cu roţi, costuri pentru acţiuni, resurse iniţiale şi restricţii. Să se găsească o succesiune de acţiuni care să conducă la îndeplinirea misiunii, minimizând costul maxim pe fiecare robot.

Construirea soluţiei porneşte de la extinderea algoritmului Martins [9] a cărui principiu este similar căutării Dijkstra pentru o cale minimă de îndeplinire a unui singur obiectiv. Când se generalizează căutarea căii minime de la un singur obiectiv la probleme multi-obiectiv, pot exista mai multe soluţii pentru a atinge un anumit scop, fiecare putând fi optimă în funcție de ce obiectiv este considerat a fi mai important. O soluţie nu poate fi optimă doar dacă ea este cel mult la fel de bună ca o altă soluţie sub toate criteriile obiectivului. Această noțiune este denumită în mod obișnuit dominanţă Pareto.

Soluţia propusă nu utilizează direct graful de stări, ci consideră pentru fiecare stare un set de etichete ale căror vectori de costuri sunt Pareto optimali. Astfel algoritmul Martins găseşte toate traiectoriile Pareto optimale descrise de aceste etichete. Rezultatele experimentale sunt obţinute din studii de caz efectuate pe o echipă de trei roboţi mobile cu roţi. Spaţiul experimental este modelat pe o schiţă a unui etaj de hotel.

Din punct de vedere al limitărilor, soluţia propusă necesită descompunerea misiunii în submisiuni ce pot fi traduse în formule LTL locale executabile în paralel. Acest fapt nu poate fi îndeplinit în toate formulările LTL. În acelaşi timp, soluţia este valabilă doar pentru formulări LTL „co-safe” [10] (care nu includ sarcini definite de operatorul “always”).

In etapa curenta s-a considerat un task LTL ce exprima cerintele pentru o echipa de roboti identici. Task-ul este exprimat in functie de un set de actiuni ce pot fi efectuate in anumite regiuni de interes, iar scopul este de a gasi in mod automat o strategie pentru echipa de roboti astfel incat acestia sa coopereze pentru indeplinirea misiunii. Pentru aceasta problema, bazele abordarii inovative sunt formulate in articolul [5], ce foloseste abstractii bazate pe retele Petri si formulari in termenii unor probleme de optimizare. Aceste instrumente produc o solutie fezabila computational, spre deosebire de abordarile ce folosesc abstractizari bazate pe grafuri sub forma sistemelor de tranzitie sau automate. Similar abordarii [5] definim un sistem RMPN (Robot Motion Petri Net) ce poate modela intreaga echipa de roboti, cu pastrarea unei topologii fixe, indiferent de numarul de roboti. Problema calcularii unei secvente de executie pentru a atinge o stare (marcaj) finala dorita si a produce o observatie dorita este rezolvata printr-o procedura algoritmica bazata pe doua probleme de optimizare.

Algoritmul iterativ descris in [5] este utilizat pentru a ghida selectia unei rulari acceptate intr-un automat Büchi, rularea optimizarii unei (sau doua) probleme de programare intreaga mixta (MILP), si, in cele din urma, pentru a construi secvente de executie in modelul RMPN ce produc planificari individuale pentru roboti, cu garantia satisfacerii specificatiei date. Desi problemele MILP sunt incluse in clasa de complexitate NP-hard (deci si complexitatea computationala a metodei propuse este NP-hard), prin abordarea propusa se pot rezolva relativ repede unele instante ce nu sunt fezabile pentru metodele bazate pe produse sincrone de modele robotice individuale.

Principala noutate fata de metoda propusa in [5] consta in introducerea in specificarea misiunii pentru echipa de roboti a unui set de actiuni ce trebuie executate de aceasta. Astfel, specificatia LTL este furnizata peste setul de actiuni. Pentru a integra aceasta functionalitate in abstractiile bazate pe Retele Petri si in formularea problemelor de optimizare, propunem definirea modelului RPMN peste un set de regiuni de interes ce au asociate actiuni din setul peste care este definita specificatia LTL. Tranzitiile in reteaua Petri ce modeleaza sistemul corespund atat capabilitatilor de miscare ale robotilor intre regiuni, cat si celor de efectuare a actiunilor asociate.

Presupunem o echipa de R roboti identici ce se deplaseaza intr-un mediu rectangular. Mediul este partitionat intr-un set finit de celule disjuncte folosind o metoda de descompunere precum [12]. O celula a mediului de lucru este notata cu $c_j, j = 1,\ldots,|C|$ unde C reprezinta setul de celule, iar $|C|$ denota cardinalul setului C. In plus, presupunem un set finit de propozitii atomice $\Pi = \{\pi_1, \pi_2, \ldots, \pi_{|\Pi|}\}$. Intr-un scenariu robotic, π_i denota o actiune specifica. O actiune π_i corespunde uneia sau mai multor celule ale mediului de lucru (poate fi efectuata in una sau mai multe celule). Daca cel putin un robot efectueaza actiunea π_i in una din celulele corespunzatoare, spunem ca propozitia π_i este satisfacuta (este Adevarata). Setul Π va fi utilizat pentru furnizarea formulei LTL ce defineste misiunea ce trebuie indeplinita de echipa de roboti.

Pentru a permite formularea unei solutii pentru misiunea specificata ce presupune efectuarea de actiuni introducem urmatoarele ipoteze si notatii aditionale: Fie C^* un subset al C, unde $c^*_j, j \in [1,|C|]$ reprezinta o regiune de interes. Prin regiune de interes intelegem o celula din setul C care indeplineste cel putin una din urmatoarele conditii: este ocupata de cel putin un robot in starea initiala, are asociata o actiune din setul Π.

O actiune π_i din setul Π poate fi efectuata intr-un subset $C_i^* = \{c_j^*, j \in [1, |C^*|]\}$ al C^*. Definim setul P format din elemente $p_k, k = 1, \ldots, |P|$, unde $p_k = (\pi_i, c_j^*)$ a.i. $\pi_i \in \Pi \cup \{\emptyset\}$ si $c_j^* \in C_i^*$. Prin $\emptyset$ intelegem actiunea de vizitare a unei celule. Setul P va include $\sum_{i=1}^{|\Pi|} |C_i^*|$ elemente corespunzatoare regiunilor de interes in care se pot efectua actiuni din Π, si $|C^*|$ elemente corespunzatoare tuturor regiunilor de interes pentru care se asociaza actiunea de vizitare.

Definitie 1. Un sistem RMPN (Robot Motion Petri Net) este un 4-uplu $\mathscr{Q} = \langle \mathscr{N}, m_0, \Pi, h \rangle$, unde:

- $\mathscr{N} = \langle P, T, Post, Pre \rangle$ este o structura de retea Petri cu P setul de pozitii; T reprezinta setul de tranzitii ce modeleaza capabilitatile de miscare ale robotilor intre celule si de efectuare de actiuni; $Post \in \{0,1\}^{|P| \times |T|}$ reprezinta matricea post-incidenta (post-conditii) ce defineste arcele ce unesc o tranzitie cu o pozitie; si $Pre \in \{0,1\}^{|P| \times |T|}$ matricea pre-incidenta (pre-conditii) ce defineste arcele ce unesc o pozitie cu o tranzitie. Spre ex., $Post[p,t] = 1$ daca exista un arc ce conecteaza tranzitia $t \in T$ cu pozitia $p \in P$, altfel $Post[p,t] = 0$ si $Pre[p,t] = 1$ daca exista un arc ce conecteaza pozitia $p \in P$ cu tranzitia $t \in T$, altfel $Pre[p,t] = 0$;

- $\forall t \in T$, $1^T \cdot Pre[\cdot,t] = 1$ si $1^T \cdot Post[\cdot,t] = 1$, unde $1 \in \{1\}^{|P|}$ este un vector cu toate elementele egale cu unu si $Pre[\cdot,t]$ este coloana corespunzatoare tranzitiei t in matricea Pre. Aceasta conditie implica faptul ca toate tranzitiile au o singura pozitie la intrare si o singura pozitie la iesire (in teoria retelelor Petri, modelul este denumit *masina de stare*);

- m_0 denota marcajul initial, unde $m_0[p]$ este numarul de jetoane egal cu numarul de roboti aflati initial in regiunea $p \in P$ (mai precis, asociati cu perechea regiune-actiune vida $p \in P$);

- $\Pi \cup \{\emptyset\}$ reprezinta alfabetul de iesire (set ce contine simbolurile de iesire posibile), unde $\emptyset$ denota observatia vida (in termenii efectuarii de actiuni, $\emptyset$ corespunde vizitarii unei celule fara efectuarea unei alte actiuni);

- $h : P \to 2^\Pi$ realizeaza maparea observatiilor, unde 2^Π este setul tuturor subseturilor lui Π, incluzand setul vid $\emptyset$, iar $h(p_i)$ furnizeaza iesirea pozitiei $p_i \in P$. Daca p_i contine cel putin un jeton, atunci propozitiile din $h(p_i)$ sunt active (satisfacute). ■

Spre deosebire de metoda prezentata in [5], setul P de pozitii ale retelei Petri nu are o corespondenta unu-la-unu cu setul de celule din mediul de lucru. Deoarece intr-o celula pot fi executate mai multe actiuni, aceasta va avea cate o instanta pentru fiecare din aceste actiuni, precum si o instanta asociata actiuni de vizitare ($\emptyset$). Pentru a simplifica structura retelei Petri, celulele care nu au asociate actiuni sau nu reprezinta pozitii initiale ale robotilor nu au pozitii corespondente in P. Tranzitiile T modeleaza miscarea unui robot dintr-o celula in alta apartinand setului C^*, abstractizand (dar nu ignorand) drumul fizic ce trebuie parcurs (acesta poate contine celule ce nu apartin setului C^*). Astfel, o tranzitie va corespunde drumului minim intre doua celule din C^*, calculat pe baza grafului total al mediului de lucru.

Pentru $p \in P$, seturile sale de tranzitii de intrare si de iesire se noteaza ${}^\bullet p = \{t \in T | Post[p,t] > 0\}$ si $p^\bullet = \{t \in T | Pre[p,t] > 0\}$, respectiv. Pentru $t \in T$, setul sau de pozitii

de intrare si de iesire se noteaza ${}^\bullet t = \{p \in P | Pre[p,t] > 0\}$ si $t^\bullet = \{p \in P | Post[p,t] > 0\}$, respectiv. O tranzitie $t_j \in T$ este validata in m daca toate pozitiile de intrare contin cel putin un jeton, adica $\forall p_i \in {}^\bullet t_j, m[p_i] \geq 1$. O tranzitie validata t_j se poate declansa (executa), conducand la o noua stare $\tilde{m} = m + C[\cdot, t_j]$, unde $C = Post - Pre$ reprezinta matricea de evolutie a jetoanelor, iar $C[\cdot, t_j]$ este coloana ce corespunde t_j. Spunem ca $\tilde{m}$ este un marcaj fezabil ce poate fi atins din m prin declansarea t_j si scriem $m[t_j\rangle\tilde{m}$.

Intr-un sistem RMPN, declansarea unei tranzitii t corespunde miscarii unui robot din ${}^\bullet t = \{p_i\}$ in $t^\bullet = \{p_j\}$ si executarii actiunii asociate cu p_j. Se observa ca, prin definitie, fiecare tranzitie are o singura pozitie de intrare si o singura pozitie de iesire, obtinand astfel o masina de stare [13]. Daca $\tilde{m}$ poate fi atinsa din m prin intermediul unei secvente finite de tranzitii $\sigma = t_{i_1} t_{i_2} \ldots t_{i_k}$, este satisfacuta urmatoarea ecuatie de stare (fundamentala):

$$\tilde{m} = m + C \cdot \sigma, \tag{1}$$

unde $\sigma \in \mathbb{N}_{\geq 0}^{|T|}$ este vectorul numarului de executari ale t_j in secventa σ. Se observa ca ecuatia (1) reprezinta doar o conditie necesara pentru atingerea unui marcaj. Solutiile (1) ce reprezinta marcaje ce nu pot fi atinse se numesc *marcaje false*.

O retea Petri este viabila daca, indiferent de marcajul ce a fost atins, este posibil ca, în continuare, să fie executată orice tranziţie a reţelei (posibil dupa executarea prealabila a altor tranzitii). Intr-o masina de stare viabila nu exista marcaje false [14], adica solutiile ecuatiei fundamentale (1) reprezinta setul de marcaje ce pot fi atinse.

Fie $v_i \in \{0,1\}^{1 \times |P|}$ vectorul linie caracteristic al observatiei $\pi_i \in \Pi$ a.i. $v_i[p_k] = 1$ daca $\pi_i \in h(p_k)$ si $v_i[p_k] = 0$ altfel. Se observa cu usurinta ca, pentru un marcaj m ce poate fi atins, daca produsul $v_i \cdot m > 0$, atunci observatia π_i este activa in m

Fie $V \in \{0,1\}^{|\Pi| \times |P|}$ matricea formata de vectorii caracteristici ai tuturor observatiilor (prima linie este vectorul caracteristic al π_1, etc.). Produsul $V \cdot m$ este un vector coloana de dimensiune $|\Pi|$ in care elementul i^{th} este diferit de zero daca observatia π_i este activa. Notam cu $||V \cdot m||$ setul de iesiri corespunzatoare elementelor diferite de zero ale $V \cdot m$, adica $||V \cdot m||$ reprezinta setul de observatii active (element al 2^Π) in marcajul m.

Exemplul 1. Consideram mediul de lucru din Fig. 1 constand din 6 celule $\{c_1, \ldots, c_6\}$, doi roboti identici localizati initial in c_3 si c_6, precum si trei actiuni $\{\pi_1, \pi_2, \pi_3\}$ astfel incat actiunea π_1 corespunde celulelor c_1 si c_2, actiunea π_2 corespunde celulelor c_2 si c_3, in timp ce actinunea π_3 poate fi executata in celula c_4.

Astfel, subsetul regiunilor de interes este $C^* = \{c_1, c_2, c_3, c_4, c_6\}$, iar setul P consta din 7 elemente definite astfel:

$$\begin{array}{l|l} p_1 : (\pi_1, c_1) & p_5 : (\pi_3, c_4) \\ p_2 : (\pi_1, c_2) & p_6 : (\emptyset, c_3) \\ p_3 : (\pi_2, c_2) & p_7 : (\emptyset, c_6) \\ p_4 : (\pi_2, c_3) & \end{array}$$

Setul de tranzitii este reprezentat de $T = \{t_1, \ldots t_{30}\}$, ce contine toate tranzitiile intre oricare doua pozitii distincte din setul P, mai putin cele inspre p_6 si p_7 ce au asociata actiunea

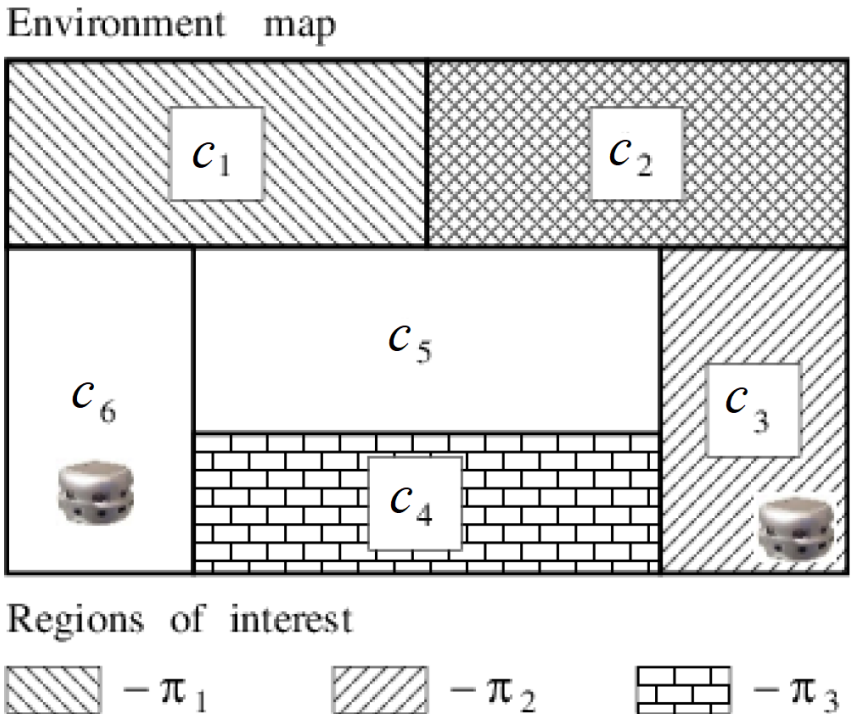


Fig. 1: Harta descompunerii pe regiuni a mediului de lucru in exemplul considerat.

$\emptyset$. Spre exemplu, tranzitia intre p_1 si p_3 are semnificatia deplasarii unui robot intre celulele c_1 si c_2 si efectuarea in c_2 a actiunii π_2. Tranzitia inversa, intre p_3 si p_1 semnifica deplasarea unui robot intre c_2 si c_1 si efectuarea actiunii π_1 in celula c_1. Tranzitia intre p_1 si p_4 echivaleaza cu deplasarea unui robot intre c_1 si c_3, pe drumul cel mai scurt, si efectuarea π_2 in celula c_3.

Marcajul initial al sistemului este $m_0 = [0,0,0,0,0,1,1]^T$, alfabetul de iesire este $\Pi = \{\pi_1, \pi_2, \pi_3\}$ iar functia observatiilor: $h(p_1) = h(p_2) = \{\pi_1\}$, $h(p_3) = h(p_4) = \{\pi_2\}$, $h(p_5) = \{\pi_3\}$, $h(p_6) = h(p_7) = \emptyset$.

Vectorul caracteristic al lui π_1 este $v_1 = [1,1,0,0,0,0,0]$ avand in vedere ca iesirea π_1 poate fi observata in p_1 si p_2, $v_2 = [0,0,1,1,0,0,0]$, iar $v_3 = [0,0,0,0,1,0,0]$. Astfel, $V = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$.

Deoarece $V \cdot m_0 = [0,0,0]^T$, nicio observatie din Π nu este activa in m_0. ■

O *traiectorie* in $\mathscr{Q}$ este o *secventa infinita* $r_{\mathscr{Q}} = m_0[t_{j_1}\rangle\, m_1[t_{j_2}\rangle\, m_2[t_{j_3}\ldots$ ce induce un *cuvant* de iesire, ce reprezinta secventa observata de elemente din 2^{Π}.

Misiunea de miscare si efectuare de actiuni pentru echipa de roboti este furnizata ca formula dintr-o subclasa LTL (Linear Temporal Logic), notata LTL_{-X}. Informal, o formula LTL_{-X} este definita recursiv peste setul de propozitii atomice Π, folosind operatorii Booleeni standard ($\neg$ - negatie, $\vee$ - disjunctie, $\wedge$ - conjunctie, $\Rightarrow$ - implicatie, si $\Leftrightarrow$ - echivalenta) si operatori temporali ($\mathscr{U}$ - until, $\Diamond$ - eventually, si $\Box$ - always). Pentru simplitatea notatiilor, in continuare vom scrie LTL in loc de LTL_{-X}.

Orice formula LTL definita peste setu; Π poate fi transformata intr-un automat Büchi (vezi Def. 2) ce accepta toate si numai siruri de intrare ce satisfac formula [15]. Sunt disponibile instrumente software ce permit astfel de conversii, de ex. [16, 17, 18].

Definitie 2. Automatul Büchi ce corespunde unei formule LTL peste setul Π are structura $B = (S, S_0, \Sigma_B, \rightarrow_B, F)$, unde:

- S este un set finit de stari;
- $S_0 \subseteq S$ este setul de stari initiale;
- $\Sigma_B = 2^{\Pi}$ este setul de intrari;
- $\rightarrow_B \subseteq S \times \Sigma_B \times S$ reprezinta relatia de tranzitie;
- $F \subseteq S$ ieste setul starilor finale. ■

Pentru $s_i, s_j \in S$, notam cu $\rho(s_i, s_j)$ setul tuturor intrarilor lui B ce activeaza tranzitia de la s_i la s_j. Tranzitiile in B pot fi nedeterministe, cu semnificatia ca, dintr-o anumita stare pot exista tranzitii de iesire multiple activate de o aceeasi intrare, adica putem avea $(s, \tau, s') \in \rightarrow_B$ si $(s, \tau, s'') \in \rightarrow_B$, cu $s' \neq s''$. Astfel, o secventa de intrare poate produce mai mult de o secventa de stari.

Un cuvant de intrare de lungime infinita (secventa cu elemente din Σ_B) este *acceptat* de B daca acesta produce cel putin o secventa de stari (*rulare*) ale lui B care viziteaza setul F infinit de des. Concomitent, o rulare a lui B ce viziteaza infinit de des setul F este *acceptata*, si, daca B are cel putin o rulare (infinita) acceptata, atunci, are cel putin o rulare cu o reprezentare prefix-sufix finita [15]. Astfel, rularea acceptata (si cuvantul de intrare corespunzator) poate fi stocata in memoria finita sub forma: (i) un sir de lungime finita denumit *prefix* ce conduce B intr-o stare finala din F, si (ii) un sir de lungime finita denumit *sufix* ce conduce B inapoi in starea finala atinsa de prefix. Rularea lui B este formata din prefix urmat de o infinitate de repetitii ale sufixului, adica *prefix, sufix, sufix,* .... Astfel, ne vom concentra pe gasirea unor rulari ale modelului RMPN $\mathscr{Q}$ cu o structuraprefix-sufix similara care genereaza secvente ce satisfac formulele LTL.

Formularea problemei. *Data fiind o echipa R de roboti identici, un mediu de lucru descompus in setul de celule C, un set de actiuni* Π *si maparea acestora pe celulele din C, precum si o misiune sub forma unei formule LTL peste setul* Π*, gaseste o strategie de miscare si de efectuare a actiunilor de catre echipa de roboti pentru a satisface misiunea data.*

Solutie. Solutia propusa consta din doua etape:

1) Constructia sistemului RMPN plecand de la ipotezele problemei (R, C, Π)
2) Gasirea unei secvente de observatii ce satisface formula LTL si generarea tranzitiilor RMPN ce produc acea secventa

In **prima etapa** se construieste graful celulelor ce formeaza mediul de lucru. Metodele de descompunere in celule partitioneaza mediul in regiuni convexe denumite celule [19]. In urma descompunerii se obtine un graf de conectivitate, in care nodurile reprezinta celulele, iar arcele dintre noduri ilustreaza relatia de adiacenta a acestora.

Definirea setului Π presupune suplimentar si specificarea relatiei dintre fiecare element din Π si celulele grafului de conectivitate. Pe baza acestei relatii si a pozitionarii initiale a robotilor se deduce setul C^* al regiunilor de interes. Pentru acest set se construieste un nou graf de conectivitate in care nodurile sunt reprezentate de regiunile de interes, iar un arc intre doua noduri corespunde drumului cel mai scurt intre cele doua regiuni de interes, determinat din graful de conectivitate initial. Pentru fiecare arc se stocheaza secventa de noduri intermediare corespunzatoare drumului cel mai scurt in graful de adiacenta initial.

Modelul RMPN $\mathscr{Q}$ se construieste definind componentele sale astfel:

1. P: Pentru fiecare regiune de interes, se creeaza o pozitie pentru fiecare actiune ce poate fi efectuata in acea regiune. Pentru regiunile de interes corespunzatoare plasarilor initiale ale robotilor se genereaza pozitii cu actiunea $\emptyset$.

2. T: Intre fiecare doua pozitii din P se defineste o tranzitie, mai putin spre pozitiile ce au asiciata actiunea $\emptyset$. Semnificatia trecerii unui jeton printr-o tranzitie este ca un robot se deplaseaza din celula pozitiei de intrare in celula pozitiei de iesire si efectueaza actiunea asociata pozitiei de iesire. Un aspect important il reprezinta faptul ca cele doua celule asociate pozitiilor de intrare si iesire ale tranzitiei pot fi neadiacente, caz in care tranzitia corespunde unei deplasari pe drumul cel mai scurt intre cele doua celule. De asemenea, celulele pot fi identice, caz in care tranzitia corespunde doar efectuarii actiunii pozitiei de iesire.

3. h: Functia observatiilor va furniza actiunea mapata pe fiecare pozitie, conform datelor de intrare.

Costul unei tranzitii intre doua pozitii este format din costul asociat deplasarii pe drumul cel mai scurt intre celulele corespunzatoare si din costul asociat efectuarii actiunii pozitiei de iesire. Daca actiunea asociata pozitiei de iesire este $\emptyset$, atunci al doilea termen este nul.

Solutia propusa pentru **a doua etapa** se bazeaza pe abordarea descrisa in [5] si consta in 3 pasi principali:

(i) O rulare acceptata r este aleasa din automatul Büchi corespunzator formulei LTL φ;

(ii) Pentru fiecare tranzitie a rularii r, se cauta o secventa de declansari pentru modelul RMPN astfel incat observatiile generate sa produca tranzitia aleasa;

(iii) Strategiile de miscare si efectuare de actiuni ale robotilor sunt obtinute concatenand secventele de declansari din pasul (ii) si impunand momente de sincronizare intre aceste secvente.

Algoritmul 1 descrie constructia iterativa a solutiei conform pasilor de mai sus.

Algorithm 1: Iterative construction of solution

Input: RMPN model $\mathcal{Q}$, Büchi automaton B, R, κ

Output: Solution (firing sequence and synchronizations for each robot)

Run Alg. 2 to find set Γ containing accepted runs;
while $\Gamma \neq \emptyset$ **do**
    Initialize robot sequences as empty and m_0 as initial RMPN marking;
    Pick shortest run $r \in \Gamma$, $r = s_0 s_1 \dots s_p \dots s_{L_r}$;
    for $j = 0, 1, \dots, L_r - 1$ **do**
        Formulate and solve MILP (4);
        if σ *is not spurious* **then**
            Run Alg. 4 to establish if σ is applicable;
            if *solution* σ *is applicable* **then**
                Update robot sequences;
                continue with next j in for loop;
            else
                Formulate and solve MILP (5);
                if *solution* σ *is not spurious* **then**
                    Update robot sequences;
                    continue with next j in for loop;
                else
                    Current transition of r cannot be ensured;
                    break the for loop;
        else
            Current transition of r cannot be ensured;
            break the for loop;
    if *all transitions of r were ensured* **then**
        In the obtained robot sequences, keep repeating the last part ensuring suffix $s_{p+1} \dots s_{L_r}$;
        return solution;
    else
        $\Gamma := \Gamma \setminus \{r\}$;

Pasul (i): Automatul Büchi B corespunzator formulei φ este construit folosind instrumente software existente, ex. [17, 18]. Inainte de a cauta rulari acceptate, se pastreaza pe tranzitiile lui B doar intrarile ce pot fi generate de modelul $\mathscr{Q}$. Acest lucru este realizat in primele patru linii ale Alg. 2, in care initial se construieste setul $O \subseteq 2^{\Pi}$ de observatii (iesiri) ce pot fi generate de un robot (linia 1). Avand in vedere ca $\mathscr{Q}$ contine R jetoane, setul de iesiri $H \subseteq 2^{\Pi}$ ce poate fi generat de intreaga echipa se obtine ca produs cartezian al O cu sine de R ori (linia 2). Pentru fiecare tranzitie a lui B, setul de intrari ce activeaza acea tranzitie este actualizat pe baza setului H deoarece modelul $\mathscr{Q}$ poate genera cel mult aceste iesiri (liniile 3-4). Se observa ca unele tranzitii ale B pot disparea dupa rularea Alg. 2.

In continuare, se construieste un set Γ ce contine un numar finit de rulari acceptate ale lui B. Pentru aceasta, se considera un numar mic $\kappa \in \mathbb{N}$ (de obicei $\kappa \leq 4$) si liniile 5-14 ale Alg. 2. Traseele in B cu structura prefix-sufix sunt gasite prin rularea unor cautari folosind algoritmul k-shortest path [20] pe graful de adiacenta corespunzator tranzitiilor lui B. Din fiecare stare initiala s_0 a lui B sunt gasite cel mult κ cele mai scurte trasee catre fiecare stare finala s_f. Aceste trasee sunt prefixe (liniile 7-8). Trebuie remarcat ca algoritmul de cautare nu include cicluri in traseele returnate, astfel este posibil sa se obtina mai putin de κ trasee intre doua noduri date. In liniile 9-12 se construieste un set de trasee ce ar aduce B inapoi in starea finala s_f. Setul intermediar S_{s_f} contine stari ce pot conduce la s_f. Acest set este necesar deoarece cautarea in graf furnizand acelasi nod ca sursa si destinatie ar returna un drum de lungime 1, chiar daca acel nod nu prezinta o bucla cu el insusi. Rotunjirea din linia 10 este necesara pentru a obtine cel putin un sufix pentru fiecare stare din S_{s_f}. Fiecare rulare adaugata in setul Γ (linia 13) include prefixul (ce conduce B intr-o stare finala) si o iteratie a sufixului (ce conduce B inapoi in acea stare finala). Astfel, fiecare element al Γ este finit, iar lungimea infinita necesara semanticii LTL va aparea prin repetarea infinita a sufixului si repetitiile corespunzatoare ale tranzitiilor in modelul $\mathscr{Q}$ - astfel, in linia 14 se stocheaza elementul fiecarei rulari de unde apar repetitiile.

Urmatorii pasi, (ii) si (iii), ai metodei propuse vor fi iterati pentru fiecare rulare diferita, r din setul Γ. Odata ce o rulare poate fi urmarita prin observatiile modelului $\mathscr{Q}$ (pasul (ii) se executa cu succes), iterarea acestor pasi poate fi terminata si se returneaza solutia obtinuta.

Se noteaza rularea curenta a lui B cu $r = s_0 s_1 \dots s_p \dots s_{L_r}$, unde L_r este numarul de stari din prefixul si sufixul lui r, s_p si s_{L_r} reprezinta aceeasi stare finala ce trebuie vizitata de o infinitate de ori, iar sufixul ce se repeta este $s_{p+1} \dots s_{L_r}$. Daca $L_r = p+1$, atunci sufixul are doar o stare, iar repetitiile acestuia se traduc prin oprirea robotilor in celulele atinse.

Pasul (ii): Pentru activarea tranzitiei $s_j \to s_{j+1}$ in rularea r a lui B, $j = 0, \dots, L_r - 1$, sunt necesare urmatoarele doua conditii:

(a) Sistemul $\mathscr{Q}$ trebuie sa atinga un marcaj final m care genereaza orice observatie din setul $\rho(s_j, s_{j+1}) \subseteq 2^{\Pi}$;

(b) Marcajele intermediare trebuie sa genereze observatii incluse in setul $\rho(s_j, s_j) \subseteq 2^{\Pi}$, a.i. sa nu fie cauzata tranzitia din starea s_j catre o alta stare.

Algorithm 2: Update B and construct set Γ of accepted runs

Input: RMPN $\mathscr{Q}$, Büchi B, R, κ

Output: Trimmed Büchi B, set of runs Γ

Compute $O = \bigcup_{p \in P} h(p)$;

$H = \underbrace{O \times O \times \ldots \times O}_{R\text{-times}}$;

for $s_i, \rho(s_i, s_j), s_j \in \rightarrow_B$ **do**

  $\rho(s_i, s_j) = \rho(s_i, s_j) \cap H$;

$\Gamma = \emptyset$;

for $s_0 \in S_0$ *and* $s_f \in F$ **do**

  Find at most κ paths in graph of B from s_0 to s_f (using k-shortest path algorithm);

  Denote the set of above paths by $Pref$;

  Let $S_{s_f} = \{s \in S | \exists (s, \tau, s_f) \in \rightarrow_B\}$ **for** $s \in S_{s_f}$ **do**

    Find at most $ceil\left(\frac{\kappa}{|S_{s_f}|}\right)$ paths from s_f to s;

    Move s_f from the beginning of each path to the end of path;

    Denote the set of above paths by $Suff$;

  Append to each path from $Pref$ each path from $Suff$ and add the resulted run to set Γ;

  For each run from Γ, store the index for beginning its suffix;

Pentru a impune/verifica o observatie intr-un marcaj fezabil dat m, definim pentru fiecare observatie $\pi_i \in \Pi$ o variabila binara x_i astfel incat:

$$x_i = \begin{cases} 1, & \text{daca } v_i \cdot m > 0 \\ 0, & \text{altfel.} \end{cases} \tag{2}$$

Urmatoarele doua constrangeri asigneaza valoarea corecta lui x_i:

$$\begin{cases} N \cdot x_i & \geq v_i \cdot m \\ x_i & \leq v_i \cdot m \end{cases}, \tag{3}$$

unde N este un numar mare. Se observa ca daca $v_i \cdot m > 0$, prima constrangere din (3) impunes $x_i = 1$, in timp ce daca $v_i \cdot m = 0$, a doua constrangere din (3) asigura $x_i = 0$.

In continuare derivam echivalentele formale in termeni de inegalitati liniare pentru pasul (ii-a). In acest sens, se considera un subset generic $\mathscr{S} \subseteq 2^{\Pi}$. Setul $\mathscr{S}$ (set de subseturi ale lui Π) poate fi vazut ca o disjunctie de conjunctii de propozitii din Π - astfel, ca o formula Booleana φ_S in FND (forma normala disjunctiva). Evident, $\varphi_{\mathscr{S}}$ este *True* daca $\neg(\neg\varphi_{\mathscr{S}})$ este *True*. Se observa ca $(\neg\varphi_{\mathscr{S}})$ este formula ce corespunde $2^{\Pi} \setminus \mathscr{S}$, adica, $\varphi_{2^{\Pi} \setminus \mathscr{S}}$, si dorim $\neg\varphi_{2^{\Pi} \setminus \mathscr{S}}$ sa fie *True*. Totusi, avand in vedere ca $\varphi_{2^{\Pi} \setminus \mathscr{S}}$ este o formula Booleana pentru setul $2^{\Pi} \setminus \mathscr{S}$ este si FND iar $\neg\varphi_{2^{\Pi} \setminus \mathscr{S}}$ va fi o CNF (forma normala conjunctiva). Inspirandu-ne din rezultatele din [21], $\neg\varphi_{2^{\Pi} \setminus \mathscr{S}}$ poate fi scrisa ca un set de constrangeri liniare folosind variabilele x_i. Pentru aceasta, este propus Alg. 3. Informal, odata ce este construit complementul lui $\mathscr{S}$ (line 1),

dorim ca orice element din acest complement sa fie fals (sa nu fie observat). Pentru fiecare element $\bar{s} \in 2^{\Pi} \setminus \mathscr{S}$, inegalitatea din linia 2 nu este afectata daca se observa o propozitie din $\bar{s}$ (termenul din partea dreapta este incrementat cu 1 datorita lui x_i, in timp ce termenul din partea stanga este incrementat datorita cardinalitatii lui $\bar{s}$). Totusi, pentru fiecare propozitie din $\bar{s}$ ce nu este observata, termenul drept nu este incrementat, ajutand astfel satisfacerea inegalitatii. In acelasi timp, termenul drept este decrementat (premiat) cu 1 daca este observata o propozitie din afara setului $\bar{s}$ (datorita x_j). Daca inegalitatea din linia 2 este adevarata, atunci conjunctia din $\varphi_{2^{\Pi} \setminus \mathscr{S}}$ corespunzatoare lui $\bar{s}$ este falsa, deci o disjunctie din FNC $\neg\varphi_{2^{\Pi} \setminus \mathscr{S}}$ devine adevarata. Daca $2^{\Pi} \setminus \mathscr{S}$ include elementul $\emptyset$, atunci trebuie sa fie observata cel putin o propozitie din Π pentru a incalca acest element al setului complement, adica inegalitatea din linia 3 ar trebui sa fie adevarata.

In concluzie, pentru a decide daca observatia curenta a sistemului RMPN $||V \cdot m||$ apartine unui set dat $\mathscr{S}$, trebuie ca mai intai sa se asocieze valorile variabilelor binare x_i cu observatiile sistemului RMPN, ca in inegalitatile (3). Apoi, aceste variabile binare trebuie sa satisfaca si inegalitatile returnate de Alg. 3.

Algorithm 3: Constraints for the set $\mathscr{S}$

Input: Set $\mathscr{S} \subseteq 2^{\Pi}$

Output: A set of linear constraints

1 Compute $2^{\Pi} \setminus \mathscr{S}$ (complement of S);

2 Add constraints

$$\sum_{\pi_i \in \bar{s}} x_i - \sum_{\pi_j \in (\Pi \setminus \bar{s})} x_j \leq |\bar{s}| - 1,\ \forall \bar{s} \in \left(2^{\Pi} \setminus \mathscr{S}\right), \bar{s} \neq \emptyset$$

3 If $\emptyset \in \left(2^{\Pi} \setminus \mathscr{S}\right)$, add constraint

$$\sum_{\pi_i \in \Pi} x_i \geq 1$$

Functia de cost: Pe baza constrangerilor liniare pentru RMPN, ce asigura o tranzitie in automatul Büchi, se dezvolta o formulare de programare intreaga mixta. Pentru aceasta, este necesara stabilirea unei functii de cost. Functia de cost propusa include costurile pentru miscarea robotilor intre celule si pentru efectuarea actiunilor corespunzatoare. Avand in vedere ca σ reprezinta vetorul de declansari, minimizarea costului total poate fi impusa prin minimizarea

$$Cost^T \cdot \sigma,$$

unde *Cost* reprezinta costul tranzitiilor definit in etapa 1 a metodei.

Aspectele de mai sus permit formularea conditiei (a) din pasul (ii) sub forma unei probleme de programare intreaga mixta (4).

$$\begin{aligned}
& \min Cost^T \cdot \sigma & \\
\text{s.t.} \quad & m & = m_0 + C \cdot \sigma \\
& N \cdot x_i & \geq v_i \cdot m, \forall \pi_i \in \Pi \\
& x_i & \leq v_i \cdot m, \forall \pi_i \in \Pi \\
& \text{Lin. ineq. in } x_i \text{ given} & \text{by Alg. 3 for set } \rho(s_j, s_{j+1}) \\
& m_0 - m & \leq w \\
& -m_0 + m & \leq w \\
& Pre \cdot \sigma + m & \leq \gamma \cdot 1 \\
& m \in \mathbb{N}_{\geq 0}^{|P|}, \sigma \in \mathbb{N}_{\geq 0}^{|T|}, & x_i \in \{0,1\}, i = 1, \ldots, |\Pi| \\
& w \in \mathbb{R}^{|P|}, & \gamma \in \mathbb{R}
\end{aligned} \tag{4}$$

Se observa ca in loc sa se introduca constrangeri asupra tuturor variabilelor binare x_i, $i = 1, \ldots, |\Pi|$, se pot include doar constrangerile asupra propozitiilor ce apar in $\rho(s_j, s_{j+1})$ (acest lucru nu este cuprins in problema (4) pentru a mentine simplitatea notatiilor).

Problema (4) este rezolvata colosind rutine specifice de optimizare [22, 23]. Vectorul de declansari obtinut σ este apoi proiectat pe tranzitii individuale ale robotilor si secvente de pozitii vizitate/actiuni efectuate, prin declansarea fiecarei tranzitii validate in marcajul curent (vezi algoritmul din [24]). In cazuri rare, aceasta proiectie poate fi imposibila datorita functie de cost si constrangerilor asociate w si γ, cand se spune ca σ este *falsa*. Daca se obtine o solutie, inseamna ca prin declansarea tranzitiilor din σ, modelul RMPN atinge marcajul final m in care observatia activeaza tranzitia $s_j \rightarrow s_{j+1}$ in B.

Totusi, trebuie verificata daca observatiile intermediare ale modelului RMPN nu au declansat o tranzitie in B ce paraseste rularea curenta, adica intr-o alta stare decat s_j sau s_{j+1}. Pentru aceasta, se foloseste Alg. 4. Evident, daca nu se declanseaza nicio tranzitie in σ, nu este necesara nicio verificare suplimentara (linile 1-2 din Alg. 4) - robotii nu se deplaseaza si nu efectueaza nicio actiune, insemnand ca tranzitia $s_j \rightarrow s_{j+1}$ este deja activata cat timp in m_0. Linile 3-4 reprezinta procedura din [24] pentru a proiecta σ in tranzitii individuale si secvente de pozitii vizitate in RMPN. Apoi, in liniile 5-9 este gasit un set de iesiri intermediare generate de fiecare robot. In acest timp, presupunem ca robotii se deplaseaza individual (fara o sincronizare intermediara) si se sincronizeaza la ultima tranzitie de declansat, astfel incat vor patrunde sincron in pozitiile finale. Din acest motiv nu includem ultima iesire (liniile 8-9), din moment ce toate ultimile iesiri ale echipei vor valida cu siguranta tranzitia $s_j \rightarrow s_{j+1}$ dorita a lui B. Produsul Cartezian din linia 10 furnizeaza setul $Interm_{obs} \subseteq 2^{\Pi}$ ce include observatiile intermediare posibile ale sistemului RMPN ce pot fi generate in timp ce robotii evolueaza conform σ, sincronizarea avan loc doar la ultima tranzitie declansata. Testul din linia 11 este validat daca iesirie intermediare posibile valideaza o bucla $s_j \rightarrow s_j$ in B, caz in care spunem ca *solutia data de σ este aplicabila*. Altfel, (linia 13), starea s_j ar fi putut fi parasita catre o stare diferita de s_{j+1}, inainte ca RMPN sa atinga marcajul final m impus de MILP (4).

Daca σ data de problema (4) nu este aplicabila, se adauga mai multe constrangeri celor din problema (4), pentru conditia (ii-b). Concret, se considera vectorul cumulativ de marcaje m_t definit anterior pentru a impune restrictii asupra observatiilor traiectoriei, fara a include in

Algorithm 4: Check if σ returned by MILP (4) is applicable

Input: σ, RMPN model $\mathscr{Q}$, set $\rho(s_j, s_j)$
Output: Applicability of σ

if $\sigma = 0$ **then**
  σ is applicable;
Project σ to individual robot firing sequences;
Let $seq_i = seq_i[1], seq_i[2], \ldots, seq_i[|seq_i|]$ be the sequence of places visited by i^{th} robot, according to its firing sequence;
for each robot *i,* $i = 1, \ldots, R$ **do**
  if $|seq_i| = 1$ **then**
    $Obs_i = h(seq_i[1])$;
  else
    $Obs_i = \bigcup_{k=1}^{|seq_i|-1} h(seq_i[k])$;
$Interm_{obs} = Obs_1 \times Obs_2 \times \ldots \times Obs_R$;
if $Interm_{obs} \subseteq \rho(s_j, s_j)$ **then**
  σ is applicable;
else
  σ is not applicable;

m_t marcajul final m. Astfel, obtinem problema (5), ce contine de doua ori mai multe variabile binare decat (4), variabilele $x_{i(t)}$ corespunzand valorilor de adevar ale propozitiilor date de m_t.

$$
\begin{aligned}
& \min Cost^T \cdot \sigma \\
\text{s.t.} \quad & m = m_0 + C \cdot \sigma \\
& N \cdot x_i \geq v_i \cdot m, \forall \pi_i \in \Pi \\
& x_i \leq v_i \cdot m, \forall \pi_i \in \Pi \\
& \text{Lin. ineq. in } x_i \text{ given by Alg. 3 for set } \rho(s_j, s_{j+1}) \\
& N \cdot x_{i(t)} \geq v_i \cdot (Pre \cdot \sigma), \forall \pi_i \in \Pi \\
& x_{i(t)} \leq v_i \cdot (Pre \cdot \sigma), \forall \pi_i \in \Pi \\
& \text{Lin. ineq. in } x_{i(t)} \text{ given by Alg. 3 for set } \rho(s_j, s_j) \\
& m_0 - m \leq w \\
& -m_0 + m \leq w \\
& Pre \cdot \sigma + m \leq \gamma \cdot 1 \\
& m \in \mathbb{N}^{|P|}_{\geq 0}, \sigma \in \mathbb{N}^{|T|}_{\geq 0}, \quad x_i, x_{i(t)} \in \{0,1\}, i = 1, \ldots, |\Pi| \\
& w \in \mathbb{R}^{|P|}, \gamma \in \mathbb{R}
\end{aligned}
\tag{5}
$$

Informal, problema (5) include conditiile problemei (4) si conditia (b) din pasul (ii). Astfel, problema (4) poate fi computational mai rapid de rezolvat decat problema (5) (datorita numarului mai mic de variabile si constrangeri), in timp ce problema (5) nu poate returna un vector de declansari neaplicabil.

$$
\begin{aligned}
& & \min\ & Cost^T \cdot \textstyle\sum_{i=1}^{k} \sigma_i & & \\
& \text{s.t.} & m_i \quad & = m_{i-1} + C \cdot \sigma_i, i = 1, \ldots, k & & \\
& & m_{i-1} - Pre \cdot \sigma_i \quad & \geq 0, i = 1, \ldots, k & & \\
& & N \cdot x_i \quad & \geq v_i \cdot m_k, \forall \pi_i \in \Pi & & \\
& & x_i \quad & \leq v_i \cdot m_k, \forall \pi_i \in \Pi & & \\
& & \text{Lin. ineq. in } x_i \text{ given} \quad & \text{by Alg. 3 for set } \rho(s_j, s_{j+1}) & & \\
& & N \cdot x_{i(t)} \quad & \geq v_i \cdot \left(\textstyle\sum_{i=0}^{k-1} m_i \right), \forall \pi_i \in \Pi & & (6) \\
& & x_{i(t)} \quad & \leq v_i \cdot \left(\textstyle\sum_{i=0}^{k-1} m_i \right), \forall \pi_i \in \Pi & & \\
& & \text{Lin. ineq. in } x_{i(t)} \text{ given} \quad & \text{by Alg. 3 for set } \rho(s_j, s_j) & & \\
& & m_0 - m_k \quad & \leq w & & \\
& & -m_0 + m_k \quad & \leq w & & \\
& & m_i \quad & \leq \gamma_i \cdot 1, i = 1, \ldots, k & & \\
& m_i \in \mathbb{N}_{\geq 0}^{|P|}, \sigma_i \in \mathbb{N}_{\geq 0}^{|T|}, i = 1, \ldots, k & & x_i, x_{i(t)} \in \{0,1\}, i = 1, \ldots, |\Pi| & & \\
& w \in \mathbb{R}^{|P|}, & & \gamma_i \in \mathbb{R}, i = 1, \ldots, k & &
\end{aligned}
$$

Pasul (iii) al solutiei propuse converteste vetorul σ in secvente de declansari pentru fiecare robot, folosind o metoda din [24]. Apoi, se adauga aceste secvente secventelor anterioare ale robotilor si se impune ca robotii ce efectueaza o deplasare/actiune (aceia care efectueaza cel putin o tranzitie) sa se sincronizeze la ultima tranzitie din σ. Fiecare declansare va corespunde parcurgerii drumului cel mai scurt intre cele doua celule corespunzatoare pozitiilor si efectuarea actiunii de iesire.

Activitatea 2.2. Integrarea algoritmilor dezvoltaţi într-un mediu software

Robot Motion Toolbox (RMTool) [11] oferă o colecţie de instrumente dedicate modelării, planificării căilor şi controlului mişcării roboţilor mobili. RMTool este încorporat în mediul MATLAB, care oferă avantajul considerabil de a crea instrumente algebrice, statistice şi grafice puternice care exploatează rutinele de înaltă calitate disponibile în MATLAB. Interacţiunea utilizator mediu de simulare se realizează printr-o interfaţă grafică (Fig. 2) care permite introducerea de valori pentru diferiţi parametri aferenţi algoritmilor de planificare pentru roboţi mobile cu roţi. Interfaţa poate fi împărţită în cinci zone principale:

1. Meniu
2. Zona de reprezentare (spaţiu de lucru, traiectorie, informaţii despre stări interne ale roboţilor mobili)
3. Panou planificare misiuni
4. Panou definire misiuni
5. Panou execuţie simulări

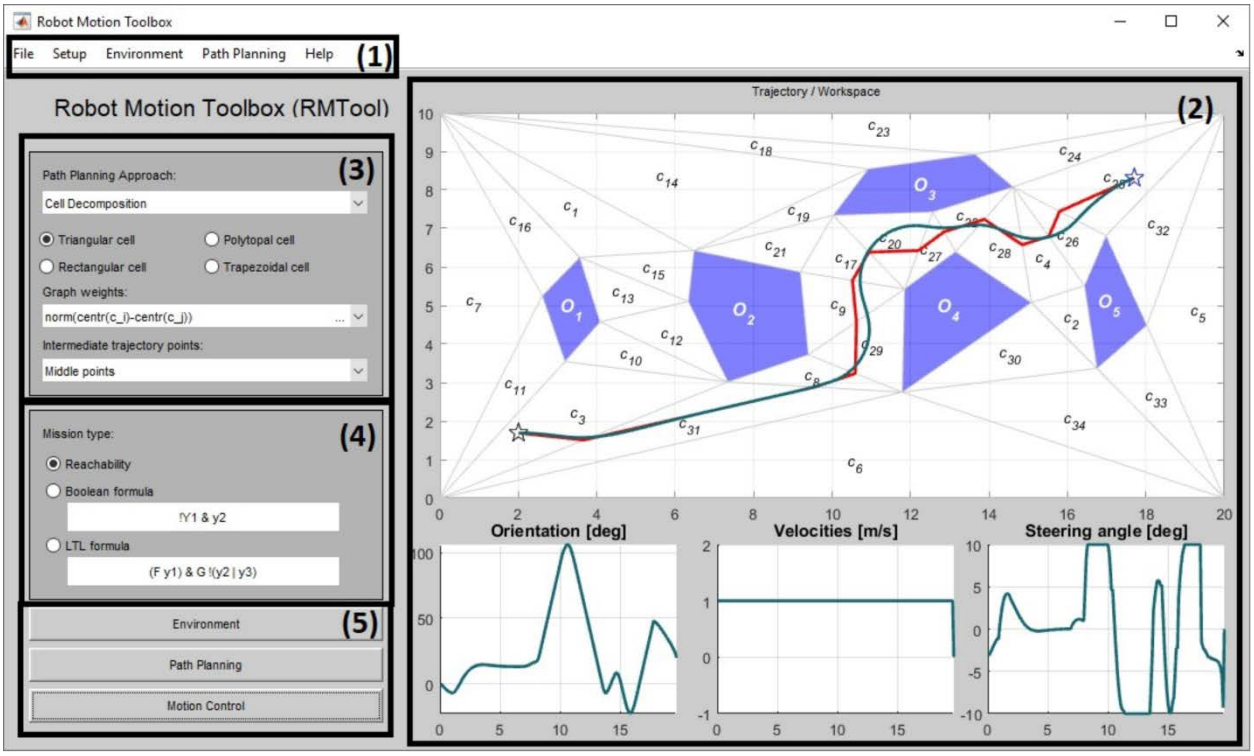


Fig. 2: Interfaţă grafică RMTool

Conform algoritmilor propuşi în această etapă a proiectului [5], RMTool a fost extins prin includerea în zona 4 a descrierii misiunii prin formule boolene şi LTL (Fig. 3). În noua versiune RMTool permite planificarea căilor de mişcare folosind specificaţii expresive sau de nivel înalt, în principal pentru un grup de roboţi mobili identici.

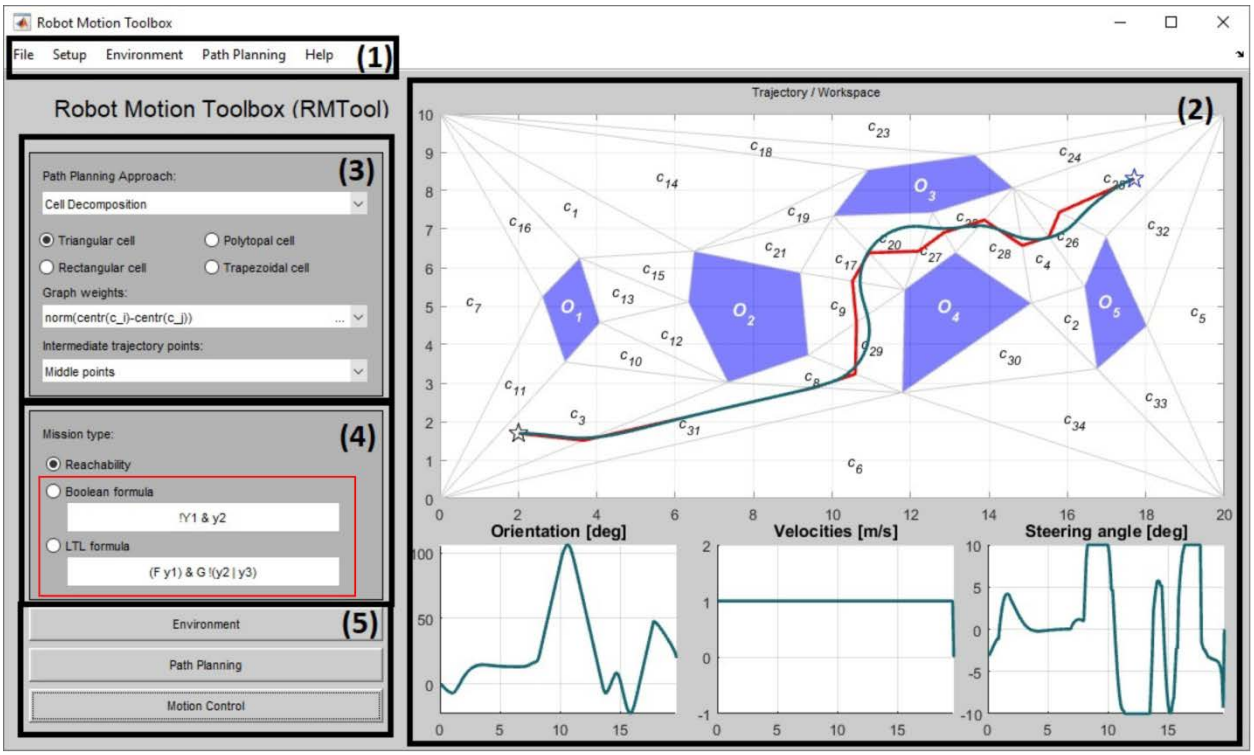


Fig. 3: Extindere RMTool prin includerea descrierii misiunii prin formule boolene şi LTL

Pentru exemplificare considerăm un spaţiu de lucru cu doi roboţi mobili şi şase regiuni de interes notate O1, O2, O3, O4, O5 şi O6. În Fig. 4 este ilustrat rezultatul planificării unei misiuni descrise prin formula LTL:

$$\phi = \mathrm{F}(\mathrm{y}_2)\,\&\,\mathrm{F}(\mathrm{y}_3)\,\&\,\mathrm{G}(!\mathrm{y}_4)\,\&\,(!\mathrm{y}_2 \cup \mathrm{y}_1)\,\&\,\mathrm{F}(\mathrm{y}_5\,\&\,\mathrm{y}_6)$$

Îndeplinirea formulei este echivalentă cu:

- regiunile O2 şi O3 trebuie vizitate eventual
- regiunea O4 trebuie evitată pe parcursul misiunii
- regiunea O2 poate fi vizitată după ce a fost vizitată O1
- regiunile O5 şi O6 trebuie ocupate eventual în acelaşi moment.

Fig. 5 ilustrează rezultatul planificării unei misiuni descrise prin formula booleană

$$\phi = \mathrm{y}_1\,\&\,!\mathrm{Y}_2\,\&\,\mathrm{y}_3\,\&\,\mathrm{Y}_4\,\&\,\mathrm{Y}_5\,\&\,\mathrm{Y}_6$$

Îndeplinirea formulei este echivalentă cu:

- regiunile O1 şi O3 trebuie ocupate în starea finală
- regiunile O4 şi O5 trebuie traversate pe parcursul mişcării
- regiunile O2 şi O6 nu trebuie traversate pe parcursul mişcării

Scopul acestui experiment este de a vedea cum algoritmul de planificare generează schimbări în misiunea roboților în funcție de sarcină și de distanța dintre roboți și regiunile respective de interes pentru a fi vizitate sau evitate. Asta înseamnă că distanța globală parcursă de roboți este cea mai scurtă posibil. Se observă că în acest caz robotul 2 ajunge în trei regiuni (O3, O4, O5) iar robotul 1 vizitează o singură regiune (O1).

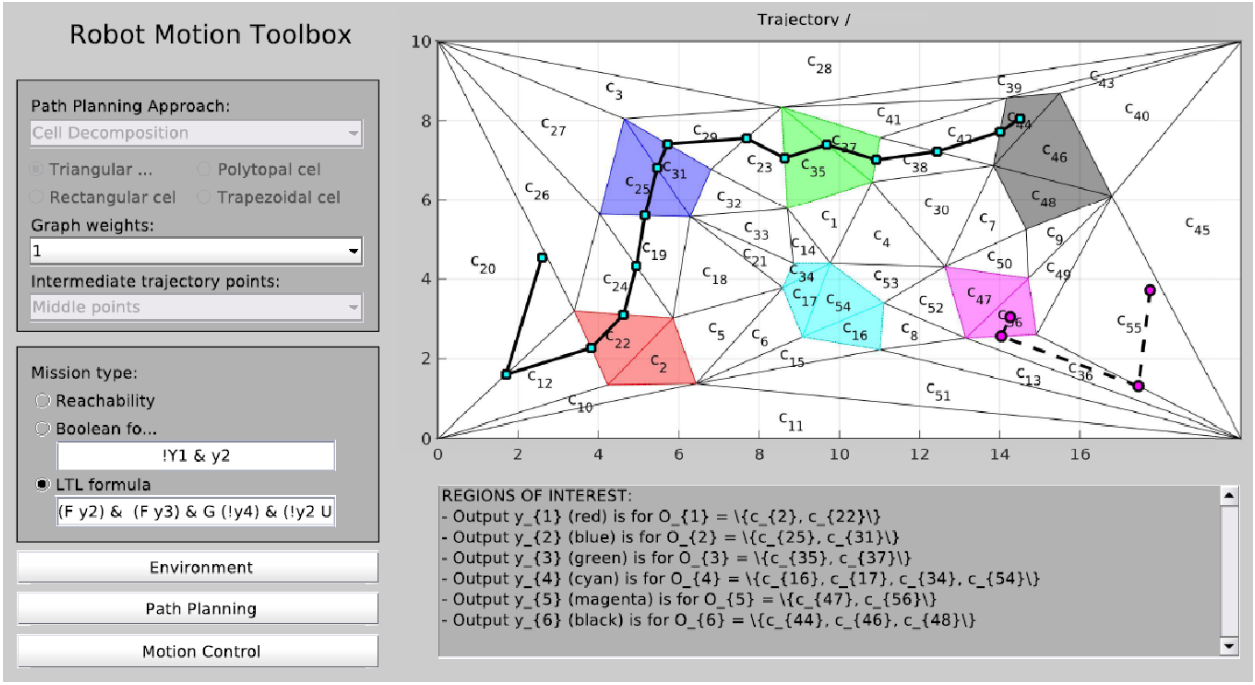


Fig. 4: Rezultat de planificare bazată pe formulă LTL

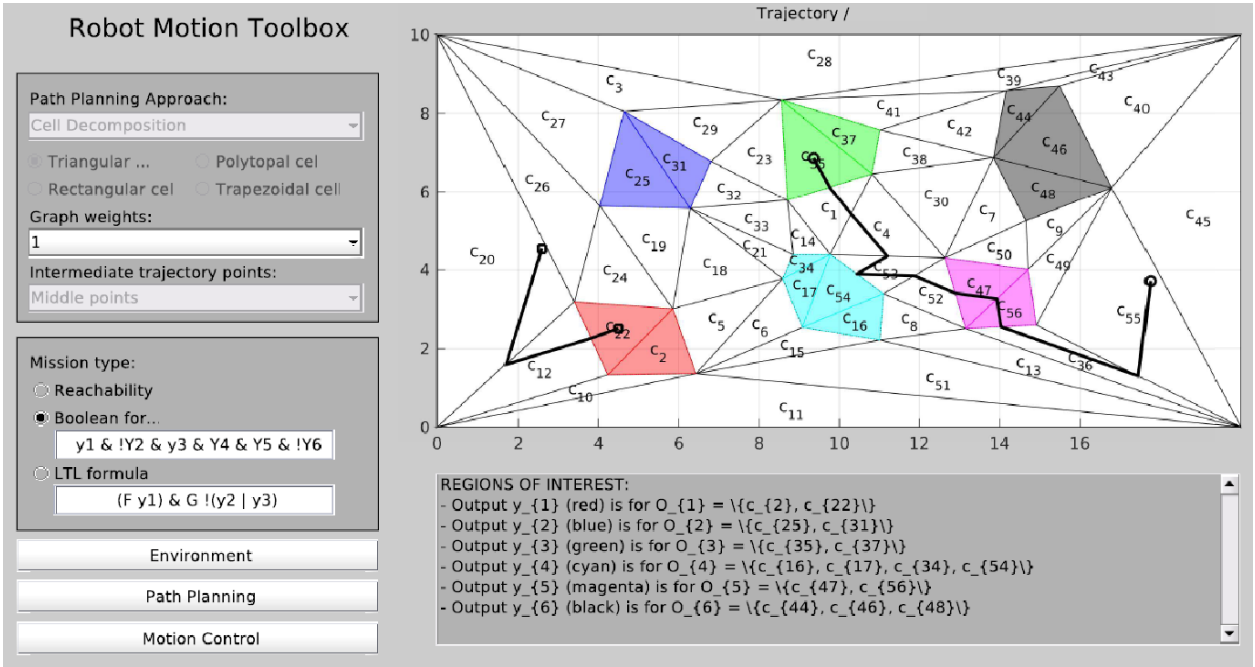


Fig. 5: Rezultat de planificare bazată pe formulă Booleană

Activitatea 2.3. Rezultate preliminare pentru experimente în timp real

Configurații de test

Platforma Crazyflie 2.0 implementează un mecanism configurabil de estimare a poziției. Astfel, agregarea și filtrarea datelor poate fi realizată pe baza unui filtru Kalman extins (EKF) sau pe baza unui filtru complementar [25]. De asemenea, plaja de intrări ale filtrelor este extensibilă - în varianta de bază, estimatorul folosește date de la unitatea inerțială integrată (MPU-9250), iar prin adăugarea unor module de extensie, acesta poate primi la intrare date de poziționare relativă sau absolută.

Platforma Crazyflie 2.0 este echipată cu doi conectori de extensie și poate găzdui simultan unul sau mai multe module suplimentare (eng. decks). Trei astfel de module, Z-Ranger v2 (I), Flow deck v2 (II) și Loco Positioning Deck (III) sunt utilizate pentru a îmbunătăți performanța de estimare a poziției și orientării dronei, deoarece în mod implicit estimatorul folosește doar date de la senzorii integrați [25, 26]. Toate cele trei module testate în acest proiect oferă informații despre poziția (și orientarea dronei), mărimile citite de la senzori fiind utilizate în estimatorul implementat pe dronă [3].

I. Z-Ranger v2 - este echipat cu un senzor laser VL53L1X ce este utilizat pentru a determina distanța până la sol. Distanța maxim de măsurare este de 400 cm, însă performanțele scad în condițiile puternice de iluminare.

II. Flow Deck v2 – este echipat cu un senzor VL53L1X (similar cu Z-Ranger) și un senzor optic de urmărire a mișcării, PMW3901MB. Acest senzor optic este similar în construcție cu un senzor de mouse optic și poate urmări mișcarea începând de la o înălțime de 8 cm. De menționat este faptul că în foaia de catalog a senzorului nu sunt disponibile informații legate de modul în care este procesată și interpretată ieșirea. De asemenea, producătorul dronei CF 2.0 afirmă din cauza lipsei de informații, integrarea senzorului s-a realizat experimental, prin încercări repetate.

III. Loco Positioning Deck - este echipat cu un modul de comunicație DW1000. Prin intermediul unor secvențe de comunicație specifice, drona determină poziția absolută în domeniul de evoluție 3D. Acest modul este asemănător cu un sistem de tip GPS în miniatură [28].

Instalarea și configurarea elementelor hardware

Spațiul de lucru în care evoluează drona este reprezentat de o platformă de 4 x 2,5 metri, ce poate fi înconjurată cu o plasă de siguranță. Aceasta dispune și de un senzor Kinect montat pe plafon, în centrul platformei, senzor ce a fost utilizat în procesul de evaluare. Senzorul este montat la o înălțime de 2,8 m și poate acoperi o suprafață de aproximativ 2,1 x 1,3 m la o înălțime de 1 m [3], [27].

Configurațiile de test care au presupus folosirea modulelor Z-Ranger (I) și Flow Deck (II) nu au ridicat probleme deosebite din punct de vedere hardware, deoarece orientarea sistemului de coordonate în care evoluează drona este relativă la poziția de start din care decolează [3]. Folosirea unor astfel de configurații ridică două probleme când există mai multe drone care evoluează în paralel și care necesită coordonare în acțiuni:

- (1) diferențele de poziție și orientare trebuie cunoscute de toate dronele (i.e. configurare manuală pentru fiecare experiment) și
- (2) erorile de estimare sunt acumulate individual de fiecare dronă (i.e. de la IMU și PMW3901MB).

Astfel, s-a decis folosirea unui sistem de poziționare absolută (Loco Positioning System - LPS (III)), ce presupune instalarea unui set de module radio (ancore) în exteriorul domeniului de evoluție, acestea fiind folosite de drone pentru determinarea poziției. Funcțional, sistemul se bazează pe măsurarea timpului de propagare a unor pachete radio (eng. *Time-of-Flight*) provenite de la ancore aflate la poziții fixe, principiu folosit și cazul altor tehnologii de poziționare trecute și actuale (Loran-C, Omega, RSDN-20, GPS, GLONASS, Beidou etc.).

Modulele LPS implementează standardul IEEE 802.15.4a-2011 (i.e. comunicații radio cu consum redus și rate de transfer mici), ce definește 16 canale UWB (Ultra-Wide Band), cu următoarele proprietăți [28]:

- canalele au o lățime de bandă de cel puțin 500 MHz și prin distribuirea uniformă a energiei în banda de emisie nu ridică probleme de coexistență cu alte tehnologii radio;
- codificările la nivel fizic presupun folosirea unor impulsuri scurte pentru transmisia datelor astfel reducându-se influența interferențelor datorate căilor de propagare multiple (eng. multipath fading);
- timpul de propagare (eng. time-of-flight) al semnalelor poate fi determinat cu o acuratețe mai mare decât în cazul tehnologiilor ce folosesc lățimi de bandă mai mici.

Tehnica de bază folosită pentru determinarea timpului de propagare nu necesită o referință de timp comună pentru sistemele care comunică. TWR (Two-way ranging) presupune generarea unei secvențe de 4 mesaje și atașarea unor mărci de timp la transmisia, respectiv recepția mesajelor așa cum este ilustrat în Fig. 6). Prin măsurarea timpilor de transmisie dus-întors (Tround2, Tround2) și a timpilor de procesare dintre o recepție și o transmisie (Treply1, Treply2), stația Tag care a inițiat secvența de comunicație (i.e. drona) poate extrage timpul de propagare al mesajelor, valoarea lui fiind direct proporțională cu distanța parcursă. Prin comunicarea succesivă cu mai multe ancore și prin cunoașterea pozițiilor absolute ale ancorelor, stația Tag poate să-și estimeze poziția prin triangularizare.

Existența mai multor stații Tag implică o accesare concurentă a mediului de comunicație (fiecare va iniția secvențe de comunicație cu ancorele), astfel apărând necesitatea folosirii unui mecanism de control al accesului la mediu, fie de tip CSMA/CA, fie de tip TDMA. Ambele prezintă dezavantaje, CSMA/CA nu se pretează aplicațiilor de localizare deoarece implică retransmisii, respectiv amânarea unor transmisii cu perioade aleatoare de timp, iar TDMA nu se pretează aplicațiilor ce includ un număr variabil de ancore, respectiv stații Tag (i.e. drone) deoarece prezintă o scalabilitate limitată. Astfel, producătorii LPS pun la dispoziție modul de

funcționare TDoA2 (Time-difference of arrival), descris de Ledergerber [28], mod ce presupune implementarea comunicației TWR între maxim 8 ancore printr-un mecanism TDMA și recepția pachetelor TWR de un număr nelimitat de stații Tag. Pe baza marcării timpilor de recepție a mesajelor provenite de la ancore diferite, o stație Tag poate să determine indirect timpii de propagare, respectiv distanțele față de ancore.

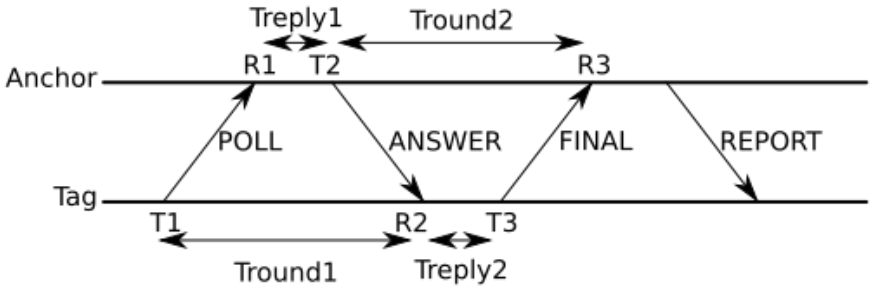


Fig. 6 Secvența de comunicație TWR [28]

Configurația de referință descrisă de producător [29], presupune folosirea a cel puțin șase ancore poziționate la înălțimi diferite (în cel puțin două planuri) și la cel puțin doi metri distanță pe oricare două axe, ca în Fig. 7 (a) și (b). După instalarea inițială a ancorelor, au fost derulate experimente pentru determinarea acurateții de poziționare și s-a constatat existența unei variații în poziționare (referit mai departe ca *jitter*), de 30-50 cm pentru axele X, Y, respectiv 15-20 cm pentru axa Z.

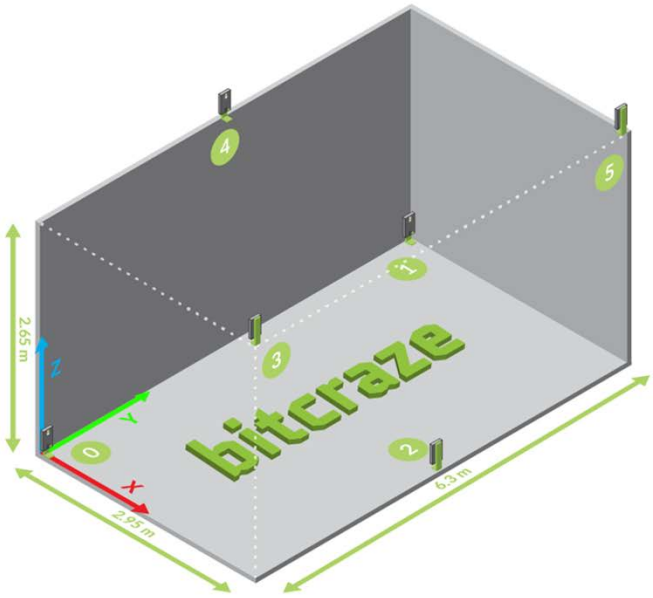


Fig. 7. (a) Configurația de referință

(b) Poziționarea ancorelor la nivelul spațiului de lucru

Pentru a facilita achiziția semnalelor de la ancore, a fost dezvoltată o aplicație de monitorizare a mesajelor vehiculate de ancore, interfață ilustrată în Fig. 8. Această aplicație oferă informații despre media și dispersia distanțelor raportate (A), numărul și secvențierea mesajelor generate (B), respectiv pozițiile raportate de ancore (C). Astfel, au fost observate discrepanțe mari între distanțele determinate în cele două direcții de comunicație pentru anumite perechi de ancore. În Fig. 8, se pot observa diferențe (marcate cu roșu) pentru ancorele A3, A4, A6, respectiv A7, ceea ce sugerează o poziționare defectuoasă a lor deoarece în benzile de frecvență folosite (3-6 GHz) nu poate fi eliminată complet influența obiectelor de dimensiuni mari (pereți, tavan, dulapuri, etc.) nici măcar prin intermediul tehnologie UWB.

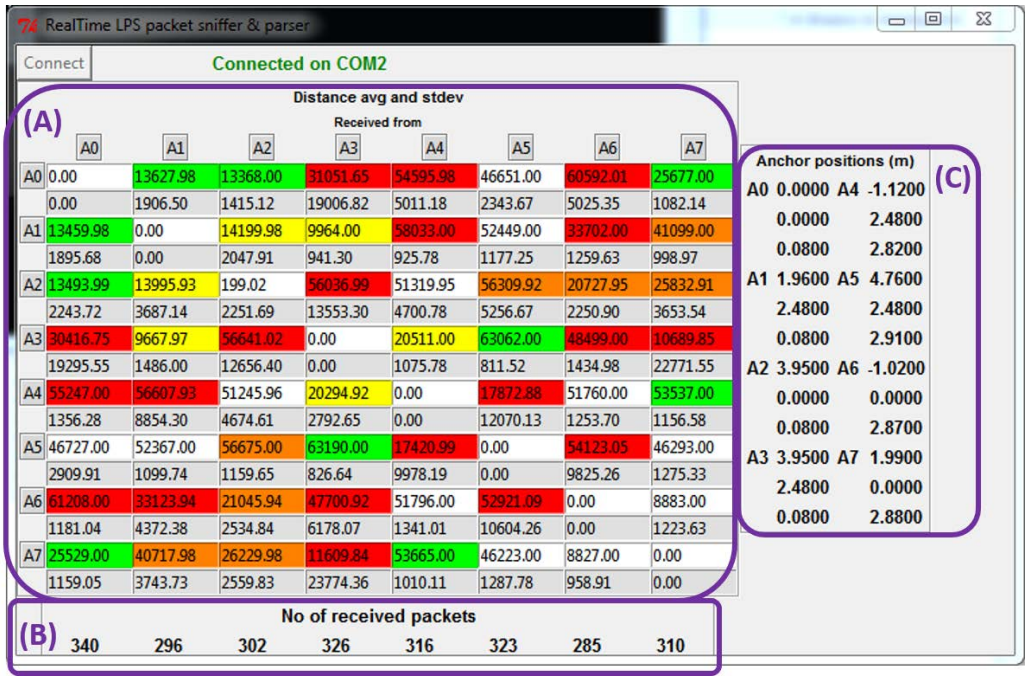
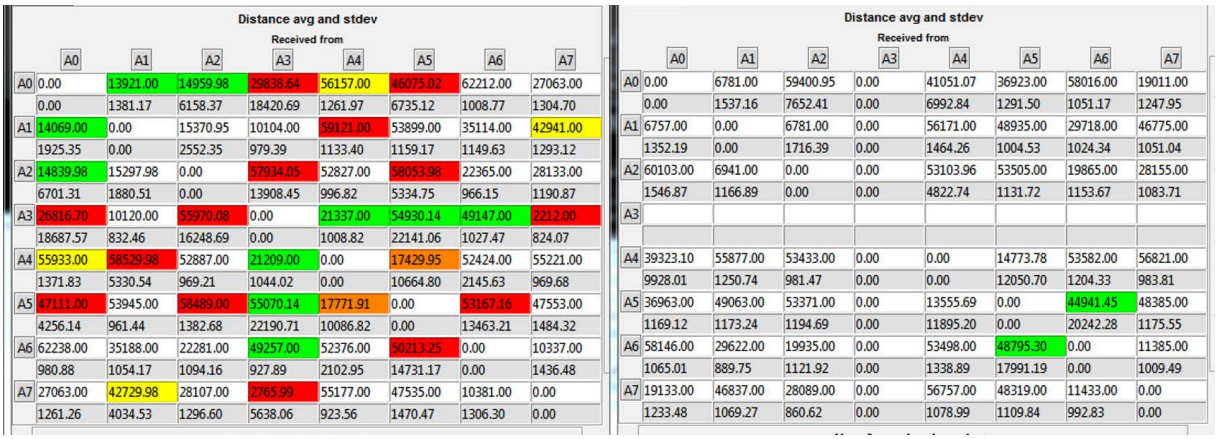


Fig. 8. Aplicația dezvoltată pentru monitorizarea comunicației dintre ancore

Pentru îmbunătățirea acurateții, ancorele au fost montate pe suporturi de plastic cu o înălțime de 20 cm, printate 3D, ce permit fixarea atât în plan orizontal, cât și vertical. Prin monitorizare continuă a mesajelor, conform cu Fig. 9 (a), au fost determinate noi poziții pentru ancore, valorile distanțelor după repoziționare fiind ilustrate în Fig. 9 (b), pentru care *jitter*-ul de poziționare a fost redus la 5 cm, precizia de poziționare declarată de producător fiind de 5 cm [29].

Aplicațiile UWB au fost studiate și de alți autori în diverse scenarii pentru monitorizarea de obiecte, persoane, echipamente, etc. În [30], autorii analizează efectul de ecranare reciprocă și impactul acestuia, precum și de metodele de îmbunătățire a urmăririi multiple a persoanelor în mișcare prin radarele UWB, propunând 3 abordări complementare. Autorii din [31] propun un algoritm de urmărire a pietonilor bazat pe ancore a căror poziție nu se cunoaște, precizia obținută fiind de 0,77 metri în 90% din cazuri.



Distance avg and stdev

Received from

	A0	A1	A2	A3	A4	A5	A6	A7
A0	0.00	13921.00	14959.98	29838.64	56157.00	46075.02	62212.00	27063.00
	0.00	1381.17	6158.37	18420.69	1261.97	6735.12	1008.77	1304.70
A1	14069.00	0.00	15370.95	10104.00	59121.00	53899.00	35114.00	42941.00
	1925.35	0.00	2552.35	979.39	1133.40	1159.17	1149.63	1293.12
A2	14839.98	15297.98	0.00	57934.05	52827.00	58053.98	22365.00	28133.00
	6701.31	1880.51	0.00	13908.45	996.82	5334.75	966.15	1190.87
A3	26816.70	10120.00	55970.08	0.00	21337.00	54930.14	49147.00	2212.00
	18687.57	832.46	16248.69	0.00	1008.82	22141.06	1027.47	824.07
A4	55933.00	58529.98	52887.00	21209.00	0.00	17429.95	52424.00	55221.00
	1371.83	5330.54	969.21	1044.02	0.00	10664.80	2145.63	969.68
A5	47111.00	53945.00	58489.00	55070.14	17771.91	0.00	53167.16	47553.00
	4256.14	961.44	1382.68	22190.71	10086.82	0.00	13463.21	1484.32
A6	62238.00	35188.00	22281.00	49257.00	52376.00	50213.25	0.00	10337.00
	980.88	1054.17	1094.16	927.89	2102.95	14731.17	0.00	1436.48
A7	27063.00	42729.98	28107.00	2765.99	55177.00	47535.00	10381.00	0.00
	1261.26	4034.53	1296.60	5638.06	923.56	1470.47	1306.30	0.00

Distance avg and stdev

Received from

	A0	A1	A2	A3	A4	A5	A6	A7
A0	0.00	6781.00	59400.95	0.00	41051.07	36923.00	58016.00	19011.00
	0.00	1537.16	7652.41	0.00	6992.84	1291.50	1051.17	1247.95
A1	6757.00	0.00	6781.00	0.00	56171.00	48935.00	29718.00	46775.00
	1352.19	0.00	1716.39	0.00	1464.26	1004.53	1024.34	1051.04
A2	60103.00	6941.00	0.00	0.00	53103.96	53505.00	19865.00	28155.00
	1546.87	1166.89	0.00	0.00	4822.74	1131.72	1153.67	1083.71
A3								
A4	39323.10	55877.00	53433.00	0.00	0.00	14773.78	53582.00	56821.00
	9928.01	1250.74	981.47	0.00	0.00	12050.70	1204.33	983.81
A5	36963.00	49063.00	53371.00	0.00	13555.69	0.00	44941.45	48385.00
	1169.12	1173.24	1194.69	0.00	11895.20	0.00	20242.28	1175.55
A6	58146.00	29622.00	19935.00	0.00	53498.00	48795.30	0.00	11385.00
	1065.01	889.75	1121.92	0.00	1338.89	17991.19	0.00	1009.49
A7	19133.00	46837.00	28089.00	0.00	56757.00	48319.00	11433.00	0.00
	1233.48	1069.27	860.62	0.00	1078.99	1109.84	992.83	0.00

Fig. 9. (a) Valorile distanțelor după repoziționarea parțială a ancorelor;

(b) Valorile distanțelor după repoziționarea ancorelor

Descrierea experimentelor și analiza performanțelor

Cercetarea noastră vizează evaluarea performanței zborului dronei în diferite configurații hardware, deoarece acestea sunt direct legate de precizia estimatorului de poziție [3], [32]. În acest studiu sunt efectuate câteva experimente, considerând 3 configurații ale dronei CF 2.0 cu modulele Z-Ranger și Flow Deck și Loco Positioning Deck și analiza performanțelor pentru fiecare configurație. Această necesitate decurge din faptul că elementele hardware nu sunt bine documentate nici dintr-o perspectivă de implementare, nici din una funcțională. Pentru fiecare dintre configurațiile HW și scenariile de test, experimentele conțin două faze principale: (i) rularea testului și achiziția de date și (ii) prelucrarea datelor offline și analiza rezultatelor, așa cum este ilustrat în Fig. 10.

Prima fază include zborul CF2 în diferite scenarii și monitorizarea dinamicii zborului. Configurația experimentală include atât monitorizarea în timp real a datelor de la senzori și procesarea acestora, cât și date obținute de la senzorul Kinect montat la nivelul platformei de lucru. Pentru prima fază, CF2 a fost marcată cu un marker de culoare roșie pentru a-i îmbunătăți vizibilitatea pentru senzorul Kinect. Apoi, comenzile de poziționare (decolare, deplasare și aterizare) sunt trimise secvențial către CF2. În a doua fază, CF2 este detectată de senzorul Kinect, obținându-se astfel poziția dronei pe cele 3 coordonate.

Parametrii de interes monitorizați de pe drona CF2 sunt unghiurile roll (ϕ), pitch (θ) și yaw (ψ), vitezele unghiulare ale dronei *p, q, r*, vitezele liniare ale dronei *u, v, w* corespunzătoare celor 3 axe, valorile PWM ale motoarelor, date preluate de la senzori (accelerație, viteză unghiulară, mișcare XY) [3].

Fig. 10. Rularea testelor, achiziția de date și prelucrarea datelor offline

Au fost utilizate drone CF2, considerând cele 3 module de extensie, iar experimentele au fost rulate de cel puțin 4 ori, fiind raportate cele mai bune rezultate. Pentru fiecare configurație HW, am considerat 4 scenarii de testare. Experimentele s-au desfășurat impunând o înălțime minimă constantă de 1 m față de sol, restricție motivată de precizia senzorului Kinect. În Tabel 1 sunt descrise configurațiile HW utilizate în experimente [3].

Tabel 1: Configurații hardware pentru experimente cu drona CF2

Configurație HW	**Descriere**	**Intrările estimatorului**
Drona CF2	Kit-ul standard, fără module suplimentare. Conține senzor IMU	Accelerație și viteză unghiulară
Drona CF2 și Z-Ranger v2 deck	Z - Ranger utilizează senzorul laser VL53L1X ToF pentru măsurarea distanței față de sol. Interval maxim pentru distanța până la sol: 400cm	Accelerație, viteză unghiulară, distanță pe verticală (axa Z).
Drona CF2 și Flow Deck v2	Flow Deck încorporează senzorul laser VL53L1X și un senzor optic PMW3901MB pentru detecția mișcării pe axele XY. Senzorul optic detectează mișcarea de la cel puțin 8 cm înălțime.	Accelerație, viteză unghiulară, distanță pe verticală (axa Z), deplasare pe axele X,Y.
Drona CF2 și Loco Positioning Deck	Sistemul de ancore este configurat în modul TDoA2, deck-ul determină poziția absolută pe baza pachetelor recepționate.	Accelerație, viteză unghiulară, poziția absolută în sistemul de coordonate LPS.

Pentru fiecare configurație hardware au fost considerate patru scenarii de testare, descrise în Tabelul 2.

Tabel 2 Scenarii de testare

ID	Descriere test
1.	Zbor la punct fix la înălțimea de 1 metru timp de 5 secunde
2.	Deplasare liniară, 2m, pe axa X la înălțimea de 1 metru
3.	Deplasare de-a lungul unui pătrat de 1x1 metru, la înălțimea de 1m
4.	Deplasare în forma unui cerc cu raza de 1 metru (20 de segmente liniare) la înălțimea de 1m

Analiza performanțelor obținute pentru controlul poziției dronei în spațiul 3D

Structura de reglare implicită livrată împreună cu drona CF 2.0 implică existența a 9 regulatoare discrete PID conectate în cascadă, implementarea acestora realizându-se în limbajul de programare C. Bucla exterioară dedicată pentru controlul de atitudine, iar bucla internă cu regulatoare PID pentru controlul vitezelor unghiulare ale dronei.

Experimentele în timp real care implică configurațiile hardware CF2 și CF2 cu Z - Ranger au subliniat faptul că filtrul Kalman extins, implementat pe dronă nu este capabil să estimeze poziția cu o eroare rezonabilă. Încercările de testare din prima configurație, Kit-ul standard, fără module suplimentare, au eșuat din faza de decolare. Pentru configurația care a inclus modulul Z - Ranger, a existat o derivă semnificativă pe axele X-Y. În testele noastre în timp real, folosind CF2 și Flow Deck, am folosit structura de reglare automată în cascadă cu 9 regulatoare PID pornind de la parametrii impliciți ai regulatoarelor. În Tabel 3 sunt ilustrate valorile parametrilor regulatoarelor PID utilizate în experimentele în timp real [3].

Tabel 3: Parametrii PID dronă

Mărimi controlate	Kr	Ti	Td
x	2	0	0
y	2	0	0
z	2	0.5	0
roll	3.5	3	0
pitch	6	3	0
yaw	6	1	0.35
p	25	1	0
q	25	1	0
r	25	15	0

Un comportament stabil al dronei se poate obține dacă, pentru regulatorul PID pentru controlul poziției pe axa Z, factorul proporțional (Kr) ia valori în intervalul [1.8 - 2.3], iar componenta integratoare (Ti) în intervalul [0.1 - 0.5]. Testele au fost efectuate, luând în considerare semnale de referință pentru X (deplasarea înainte) și deplasarea pe axa Z (deplasarea pe înălțime) conform cu scenariile de test din Tabel 2, [3].

În experimentele în timp real, senzorul Kinect nu a putut detecta CF2 atunci când zbura la înălțimi sub 90 de centimetri, drona nefiind detectată din cauza rezolutiei senzorului (512x424 pixeli). Fig. 10 ilustrează traiectoria dronei, mișcarea de-a lungul axei X („forward movement") pentru o distanță de 2m, axei Y („lateral movement") la o înălțime de 1 metru („height"). În Fig. 11 este ilustrată comparativ traiectoria dronei, reprezentarea cu roșu fiind dată de estimatorul Kalman, ce preia date de la senzori, iar cu albastru datele preluate din procesarea imaginilor de la senzorul Kinect. Perioada de eșantionare utilizată în experimentele în timp real este de 50 ms. Din figură, se pot observa oscilații ale dronei cu amplitudinea de 0,1 m în jurul axei Y față de valoarea de referință. Oscilații cu amplitudinea de 0.1 m apar în jurul punctului de referință, de asemenea, în modul „hovering".

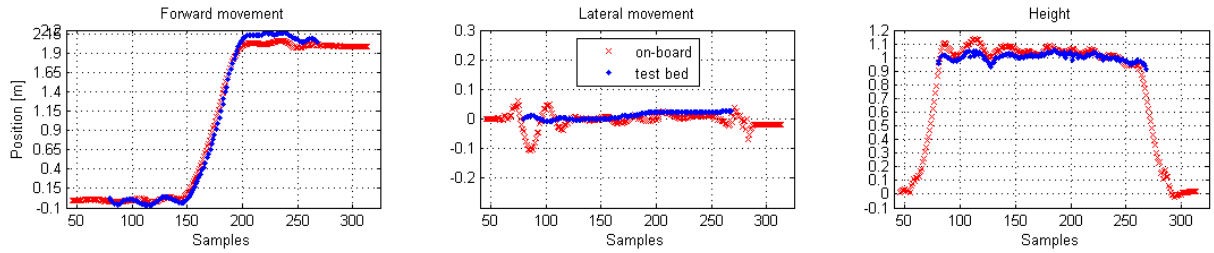


Fig. 11. Deplasarea dronei pe cele 3 axe, considerând scenariul 2

Fig. 12 ilustrează traiectoria parcursă de dronă pentru descrierea unui pătrat. De asemenea, se observă că CF2 urmărește referința impusă, cu mici oscilații pe axa Z [Budaciu et. al, 2019].

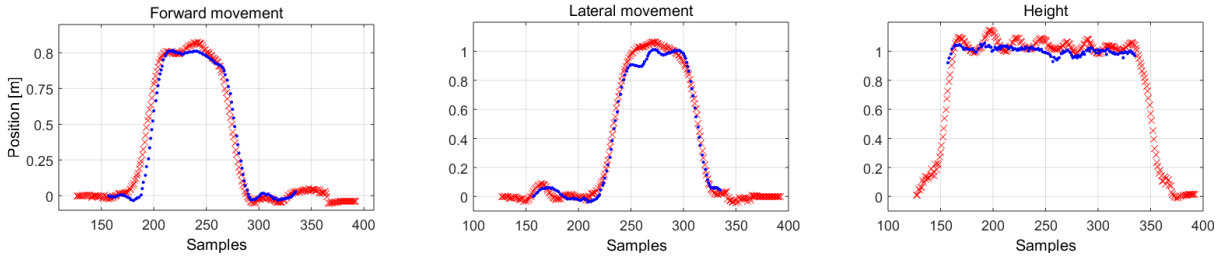


Fig. 12 Deplasarea dronei pe cele 3 axe, considerând scenariul 3

Fig. 13 ilustrează traiectoria circulară parcursă de dronă. De asemenea, se observă că CF2 urmărește referința impusă, un cerc cu rază de 1 metru, cu oscilații mici pe axa Z, timpul de răspuns fiind de aproximativ 2.5 secunde.

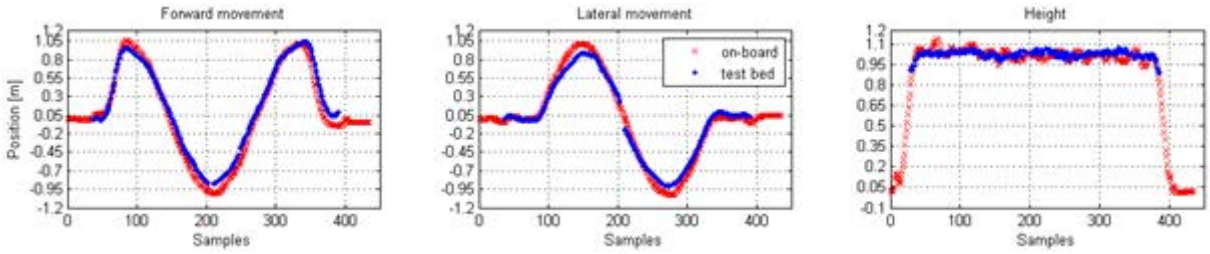


Fig. 13. Deplasarea dronei pe cele 3 axe, considerând scenariul 4

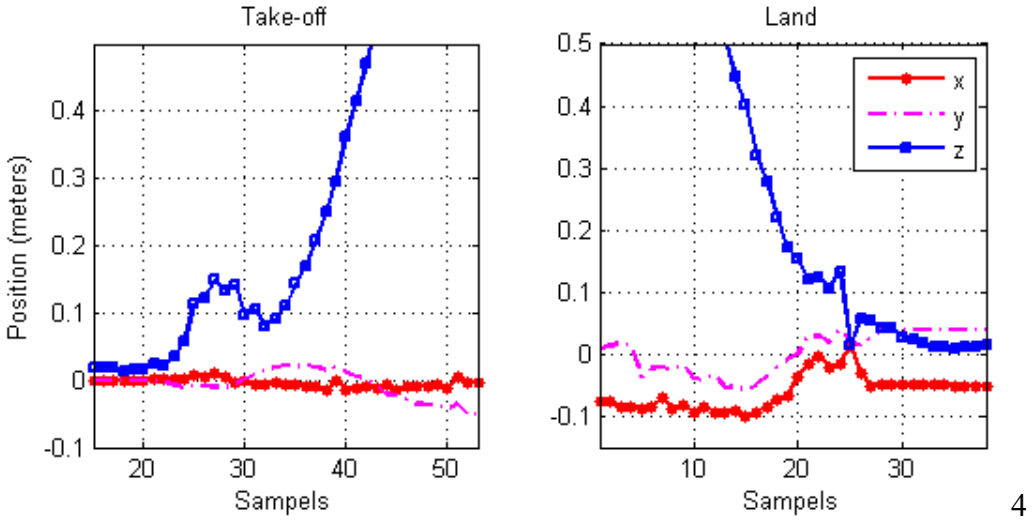


Fig. 14 Teste efectuate la decolare (Take off) și aterizare (Land)

De asemenea, s-a urmărit efectuarea unor teste de decolare, ilustrate în Fig. 14 (“Take off”) și aterizare (“Land”) în timpul cărora fiind monitorizate variabilele de ieșire și date de la senzorului optic. Datele înregistrate sugerează faptul că există o legătură între distanța minimă impusă de senzorul PWM3901MB (8 cm) și o ușoară derivă care a fost observată și ilustrată în Fig. 14 [3]. La aterizare, deriva s-a menținut sub pragul menționat. Deși nu există date de la producător cu privire la comportamentul senzorului în afara limitelor sale de lucru sau în ceea ce privește interpretarea datelor de ieșire, s-a observat că rezultate similare au fost obținute indiferent de suprafața solului folosită (mată, lucioasă, culoare uniformă, etc).

În cadrul acestui studiu s-au efectuat și experimente în timp real cu Drona CF2 și Loco Positioning Deck, cu poziționarea ancorelor la nivelul spațiului de lucru conform cu Fig. 7 (b). Sistemul de ancore este configurat în modul TDoA2, deck-ul determinând poziția absolută a dronei pe baza pachetelor recepționate. Rezultatele preliminare obținute cu această configurație HW demonstrează o poziționare a dronei cu o acuratețe comparabilă cu cea obținută cu Flow Deck. Avantajele oferite de configurația cu Loco Positioning Deck pot fi evidențiate în problemele de planificare în care sunt implicate mai multe drone cu sarcini diferite. Pe baza acestor rezultate, etapa următoare va valida algoritmii de planificare descriși anterior, considerând configurația HW cu Loco Positioning Deck. În acest context, algoritmii de reglare implementați implicit pe drona CF 2.0 vor fi reevaluați în vederea obținerii unui zbor cât mai stabil conform cu o referință impusă.

Bibliografie - etapa 2

[1] A. Burlacu, M. Kloetzer, C. Mahulea: Numerical Evaluation of Sample Gathering Solutions for Mobile Robots, *Applied Sciences*, vol.9, pp.791-809, 2019.

[2] M. Lupascu, S. Hustiu, A. Burlacu, M. Kloetzer: Path Planning for Autonomous Drones using 3D Rectangular Cuboid Decomposition, *23rd International Conference on System Theory, Control and Computing (ICSTCC)*, Sinaia, Romania, 2019.

[3] C. Budaciu, N. Botezatu, M. Kloetzer, A. Burlacu: On the Evaluation of the Crazyflie Modular Quadcopter System, *IEEE 24th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain, 2019.

[4] M. Kloetzer, A. Burlacu, G. Enescu, S. Caraiman, C. Mahulea: Optimal Indoor Goods Delivery Using Drones, *IEEE 24th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain, 2019.

[5] M. Kloetzer, C. Mahulea: Path Planning for Robotic Teams based on LTL Specifications and Petri Net Models, *Discrete Event Dynamic Systems - Theory and Applications*, accepted.

[6] Y.V. Pant, H. Abbs, R. Quaye, R. Mangharam: Fly-by-Logic: Control of Multi-Drone Fleets with Temporal Logic Objectives, *9th ACM/IEEE International Conference on Cyber-Physical Systems*, pp.186-197, 2018.

[7] P. Schillinger: Specification Decomposition and Formal Behavior Generation in Multi-Robot Systems, *PhD Thesis,* KTH, Stockholm, Sweden, 2017.

[8] P. Schillinger, M. Burger, D. Dimarogonas: Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems, *The International Journal of Robotics Research*, 2018.

[9] E. Q. V. Martins: On a multicriteria shortest path problem. *European Journal of Operational Research*, vol. 16, pp. 236-245, 1984.

[10] O. Kupferman, M. Y. Vardi: Model checking of safety properties, *Formal Methods in System Design*, Vol 19(3), pp. 291-314, 2001.

[11] R. Gonzalez, M. Kloetzer, C. Mahulea: Robot Motion Toolbox - RMTool, available at http://webdiis.unizar.es/RMTool/

[12] S.M. LaValle: *Planning algorithms*, Cambridge, 2006.

[13] T. Murata: Petri nets: Properties, analysis and applications, *Proceedings of the IEEE*, vol 77(4), pp. 541-580, 1989.

[14] M. Silva, E. Teruel, J.M. Colom: Linear algebraic and linear programming techniques for the analysis of P/T net systems, *Lecture on Petri Nets I: Basic Models*, vol. 1491, pp. 309-373, 1998.

[15] P. Wolper, M. Vardi, A. Sistla: Reasoning about infinite computation paths, *24th IEEE Symposium on Foundations of Computer Science*, pp. 185-194. Tucson, AZ, 1983.

[16] G. Holzmann: The Spin model checker, Primer and reference manual, *Addison-Wesley*, Reading, MA, 2004.

[17] P. Gastin, D. Oddoux: Fast LTL to Büchi automata translation, *13th Conference on Computer Aided Verification (CAV)*, LNCS 2102, pp. 53-65, 2001.

[18] A. Duret-Lutz, A. Lewkowicz, A. Fauchille, T. Michaud, E. Renault, L. Xu: Spot 2.0 - A framework for LTL and ω-automata manipulation, *Proc. of ATVA'16*, LNCS 9938, pp. 122-129, 2016.

[19] M.D. Berg, O. Cheong, M. van Kreveld: *Computational Geometry: Algorithms and Applications*, 3rd ed., Springer, 2008.

[20] J.Y. Yen: Finding the k shortest loopless paths in a network, *Management Science*, vol. 17(11), pp. 712-716, 1971.

[21] C. Mahulea, M. Kloetzer: Planning mobile robots with Boolean-based specifications, *53rd IEEE Conference on Decision and Control (CDC)*, Los Angeles, USA, 2014.

[22] A. Makhorin: GNU linear programming kit (2012), http://www.gnu.org/software/glpk/

[23] IBM: Ibm Ilog Cplex optimization studio software (2016), available at http://www.ibm.com/products/ilog-cplex-optimization-studio/.

[24] C. Mahulea, M. Kloetzer: Robot planning based on Boolean specifications using Petri net models, *IEEE Transactions on Automatic Control*, vol. 63(7), pp. 2218-2225, 2018.

[25] Crazyflie 2.0 documentation page, available at https://bit.ly/2vhTDy0

[26] M. Greiff: Modelling and control of the Crazyflie quadrotor for aggressive and autonomous flight by optical flow driven state estimation, *MSc. Thesis*, Lund University, 2017.

[27] S. Hustiu, M. Lupaşcu, S. Popescu, A. Burlacu and M. Kloetzer: Stable hovering architecture for nanoquadcopter applications in indoor environments, *22nd International Conference on System Theory, Control and Computing (ICSTCC)*, 2018.

[28] E. Karapistoli, F.N. Pavlidou, I. Gragopoulos, I. Tsetsinas: An overview of the IEEE 802.15.4a Standard, *IEEE Communications Magazine*, vol. 48(1), pp. 47-53, 2010.

[29] Loco-positioning-system for Crazyflie 2/0, available at https://www.bitcraze.io/getting-started-with-the-loco-positioning-system/

[30] A. Alarifi, A. Al-Salman, M.Alsaleh, A. Alnafessah, S. Al-Hadhrami, M.A. Al-Ammar, H.S. Al-Khalifa: Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances, *Sensors*, vol. 16(5), 2016.

[31] C. Gentner and M. Ulmschneider: Simultaneous Localization and Mapping for Pedestrians using Low-Cost Ultra-Wideband System and Gyroscope, *IEEE Int. Conf. on Indoor Positioning and Indoor Navigation*, Saporo, Japan, 2017.

[32] Shi G. and Ming Y.: Survey of Indoor Positioning Systems Based on Ultra-wideband (UWB) Technology, *Wireless Communications, Networking and Applications*, Springer, pp. 1269–1278, 2016.

Etapa 3 – Experimente în timp real

Activitatea 3.1. Experimente pe baza soluțiilor de planificare propuse

Descrierea platformei experimentale

Experimentele în timp real s-au desfășurat într-un spațiu de lucru reprezentat de o platformă din PAL melaminat negru cu dimensiunea de 4 x 2,5 metri, platformă ce poate fi utilizată pentru cercetări privind planificarea traiectoriilor atât în domeniul de evoluție 2D cât și 3D. La nivelul platformei sunt instalate două sisteme de poziționare pentru drone:

- Microsoft Kinect v2 - senzor de mișcare compus din trei surse de iluminare IR și două camere video, una RBG și una IR. Acesta folosește principiul ToF (time-of-flight) pentru a determina distanța până la obiectele din cadru. Senzorul este montat la o înălțime de 2,8 metri, în centrul platformei și având un câmp vizual de 70x60 grade acoperă o suprafață de aproximativ 2,1 x 1,3 metri la o înălțime de 1 metru față de sol.
- Bitcraze Loco Positioning System (LPS) - sistem de poziționare bazat pe o tehnologie radio UWB (ultra wide band) ce exploatează timpul de propagare al pachetelor radio (time-of-flight) [1]. Componentele sistemului sunt poziționate la limita sau în exteriorul platformei, astfel realizându-se o acoperire completă a domeniului de evoluție 3D, așa cum este ilustrat în Fig. 2.

Sisteme pentru poziționarea absolută

Pentru urmărirea mișcării dronelor, acestea au avut montate markere color în partea superioară, pentru a facilita detecția deplasării pe axele X, Y. Detecția înălțimii de zbor a fost realizată prin intermediul unui senzor RGB-D tip Kinect v2. Acest tip de senzor prezintă următoarele caracteristici: rezoluție cameră 1920x1080 pixeli; rezoluție senzor IR 512x424 pixeli; câmp de vizualizare 70x60 grade; achiziție 30fps; rezoluție adâncime 0.5-4.5m. Senzorul RGB-D plasat deasupra platformei (Fig. 1) poate fi utilizat pentru detectarea poziției și orientării dronelor. Interacțiunea cu sistemele de calcul se poate realiza prin propriul API care poate fi apelat prin scripturi dezvoltate în diferite medii precum Matlab sau Python. Utilizarea unui senzor RGB-D are ca scop facilitarea monitorizării continue în medii indoor, simularea accesului la GPS și, în plus, posibilitatea de a calcula orientarea dronei.

În cadrul testelor realizate, folosind senzorul Kinect v2, au fost observate următoarele dezavantaje:

- Algoritmul de detecție a poziției este relativ lent (5-10 fps), rata de actualizare nefiind adecvată pentru detecția în timp real a posturii dronelor;
- Înălțimea minimă de zbor nu poate fi mai mică de 80 centimetri, deoarece caracteristicile senzorului de adâncime nu permit detectarea sub acest prag a obiectelor

de dimensiuni reduse. Astfel, scenariile de lucru nu ar fi putut include decolări și aterizări;
- Domeniul de evoluție 3D este limitat și de câmpul vizual al camerei.

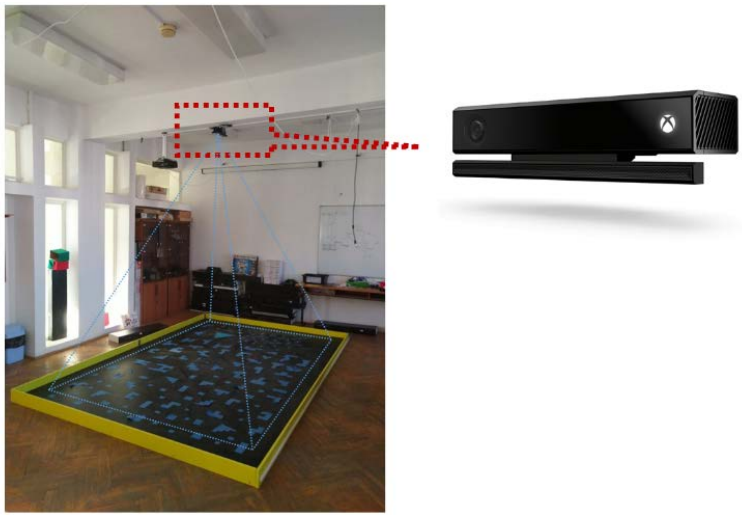


Fig. 1: Senzor RGB-D utilizat în detecția poziției și orientării dronelor în medii indoor

Sistemul LPS implementează standardul IEEE 802.15.4a-2011 (i.e. comunicații radio cu consum redus și rate de transfer mici), ce definește 16 canale UWB (Ultra-Wide Band), cu următoarele proprietăți [2]:

- canalele au o lățime de bandă de cel puțin 500 MHz și prin distribuirea uniformă a energiei în banda de emisie nu ridică probleme de coexistență cu alte tehnologii radio;
- codificările la nivel fizic presupun folosirea unor impulsuri scurte pentru transmisia datelor astfel reducându-se influența interferențelor datorate căilor de propagare multiple (eng. multipath fading);
- timpul de propagare (eng. time-of-flight) al semnalelor poate fi determinat cu o acuratețe mai mare decât în cazul tehnologiilor ce folosesc lățimi de bandă mai mici.

Tehnica de bază folosită pentru determinarea timpului de propagare nu necesită o referință de timp comună pentru sistemele care comunică. TWR (Two-way ranging) presupune generarea unei secvențe de 4 mesaje și atașarea unor mărci de timp la transmisia, respectiv recepția mesajelor. Prin măsurarea timpilor de transmisie dus-întors și a timpilor de procesare dintre recepție și transmisie, robotul care a inițiat secvența de comunicație (i.e. drona) poate extrage timpul de propagare al mesajelor, valoarea lui fiind direct proporțională cu distanța față de nodurile de referință (i.e. ancore). Prin comunicarea succesivă cu mai multe ancore și prin cunoașterea pozițiilor absolute ale ancorelor, drona poate să-și estimeze poziția prin triangularizare.

În cazul aplicațiilor complexe, existența mai multor roboți ridică probleme deosebite deoarece pentru păstrarea consistenței secvențelor de comunicație (i.e. absența întârzierilor induse) trebuie implementate mecanisme de acces la mediu de tipul CSMA/CA sau TDMA. Aceste tipuri de mecanisme au marele dezavantaj de a nu fi scalabile pentru un număr mare de noduri de comunicație - duc la creșterea numărului de coliziuni, creșterea neuniformă a întârzierilor sau la saturarea cuantelor de timp disponibile). În consecință, sistemul LPS pune la dispoziție modul de funcționare TDoA2 (Time-difference of arrival), descris de Ledergerber [2], mod ce presupune implementarea comunicației TWR între maxim 8 ancore printr-un

mecanism TDMA și recepția pachetelor TWR de un număr nelimitat de drone. Pe baza marcării timpilor de recepție a mesajelor provenite de la ancore diferite, o dronă poate să determine indirect timpii de propagare, respectiv distanțele față de ancore.

Configurația folosită este compusă din 8 ancore notate cu A0 - A7 ca în Fig. 2, dispuse la înălțimi diferite (în trei planuri), la limita sau în exteriorul platformei experimentale:

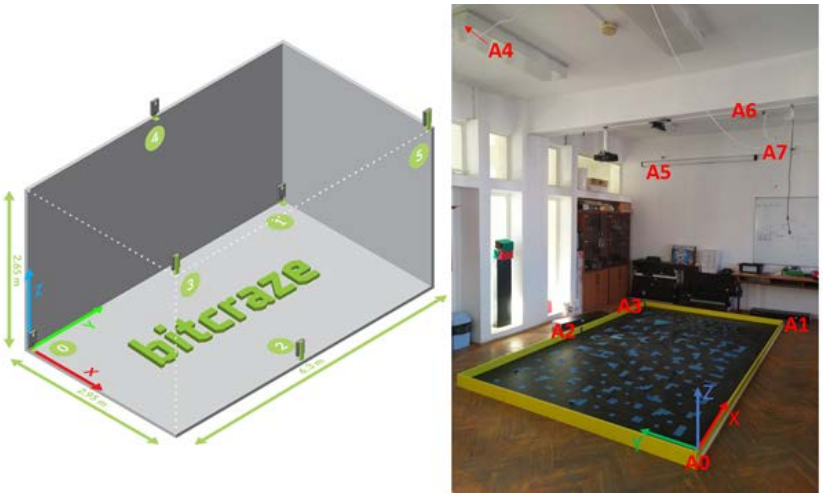


Fig. 2: Poziționarea ancorelor LPS la nivelul spațiului de lucru

În Tabelul 1 sunt ilustrate coordonatele la care sunt poziționate cele 8 ancore montate ca în Fig. 2, această configurație a ancorelor fiind aleasă în urma mai multor experimente.

Tabel 1

Ancoră	Poziție X (metri)	Poziție Y (metri)	Poziție Z (metri)
A0	0.0	0.0	0.15
A1	1.94	2.48	0.15
A2	3.91	0.0	0.15
A3	0.0	2.48	0.15
A4	-1.13	2.48	2.72
A5	1.86	0.0	2.72
A6	4.63	2.48	2.65
A7	4.63	0.0	2.65

În cadrul testelor statice, sistemul LPS are o acuratețe bună, de aproximativ 5 cm [3], însă în cadrul testelor dinamice rata de actualizare a poziției este relativ redusă și cu abateri ce pot depăși 5 centimetri, ceea ce duce la un zbor neuniform, cu oscilații (i.e. ușor observabil în cazul testelor de zbor la punct fix).

Sistem relativ de poziționare

Dronele Crazyflie 2.x pot fi echipate cu o placă de extensie (Flowdeck v2), ce poate fi utilizată pentru urmărirea mișcării. Extensia dispune de un senzor laser VL53L1X ce este utilizat pentru determinarea distanței până la sol, distanța maximă de măsurare fiind de 400 cm (performanțele scăzând în condiții de iluminare puternică). De asemenea, extensia dispune și de un senzor optic de urmărire a mișcării, PMW3901MB. Acesta este similar în construcție cu un senzor de mouse optic și poate urmări mișcarea începând cu o înălțime de 8 cm. De menționat este faptul că în foaia de catalog a senzorului nu sunt disponibile informații legate de modul în care este procesată și interpretată ieșirea, motiv pentru care producătorul dronelor Crazyflie a realizat integrarea senzorului într-o manieră experimentală, prin încercări repetate.

Testele efectuate au relevat un comportament stabil al dronei, cu ușoare oscilații doar în vecinătatea punctelor de destinație, respectiv cu o ușoară derivă în cazul zborului la punct fix. Cele două efecte se explică prin capacitatea redusă de detecție a mișcării pe axele X, Y în cazul suprafețelor colorate uniform. Stabilitatea a fost îmbunătățită prin aplicarea unor marcaje albastre la nivelul platformei (Fig. 2). Pentru acest sistem de poziționare au fost identificate următoarele neajunsuri:

- Imposibilitatea deplasării la o înălțime mai mică de 8 centimetri, datorită limitărilor constructive ale senzorului PMW3901MB;
- Drone au ca referință punctul de decolare. Astfel, în cazul scenariilor de lucru cu mai mulți roboți, sincronizarea deplasării poate fi realizată doar manual (prin cunoașterea poziției absolute a punctelor de decolare), cu riscul crescut de coliziune în cazul traiectoriilor care se intersectează datorită erorilor ce se acumulează la nivelul estimatorului de poziție;
- La deplasarea peste obstacole, dronele își modifică altitudinea de zbor. Pentru măsurarea distanței, se consideră ca referință drona și nu poate fi realizată o diferențiere între nivelul 0 al spațiului de lucru (i.e. platformă) și obiectele de pe traseu.

Configurația HW utilizată în experimentele de timp real

Pentru experimentele bazate pe soluțiile de planificare propuse a fost aleasă o soluție de poziționare mixtă, bazată pe sistemele LPS și Flowdeck. Pe lângă datele primite implicit de la unitatea inerțială, estimatorul de poziție al dronei acceptă simultan date de poziție de la cele două sisteme amintite anterior. Astfel, apar următoarele două efecte: (1) abaterea de calcul la eșantionări succesive (jitter) LPS este redusă prin influența senzorului optic prezent pe Flowdeck și (2) deriva datorată aceluiași senzor prezent pe Flowdeck este corectată periodic prin intermediul datelor de poziție de la LPS. Singurul efect nedorit rămas este cel al modificării altitudinii de zbor la deplasarea peste obiecte, însă prin fuzionarea datelor de poziție absolută și relativă se obține o amplitudine mai mică a deplasamentului pe axa Z.

Considerente de testare și calibrare

În urma testelor realizate putem să tragem următoarele concluzii referitoare la platforma experimentală bazată pe roboții Crazyflie:

- *Dronele sunt sensibile la șocurile mecanice.*

  Corpul lor este reprezentat de circuitul imprimat, la care motoarele se atașează prin intermediul unor suporturi/picioare de plastic semirigid. Suporturile și motoarele se

fixează prin presare. A fost observat faptul că în urma testelor nereușite (impact cu obiecte, pereți, podea sau aterizare bruscă de la o înălțime mare) uneori dronele își modifică comportamentul în zbor, apărând oscilații fie la decolare, fie la deplasare pe oricare din cele 3 axe. În urma impacturilor apar torsionări, deplasări, microfisuri la nivelul suporturilor ce au ca rezultat amplificarea vibrațiilor de la motoare, prin urmare suporturile trebuie înlocuite sau ajustate.

Soluție: Realizarea unor teste pentru detecția vibrațiilor înaintea fiecărui experiment. Astfel, motoarele sunt pornite/oprite în secvență și pe durata de funcționare a fiecărui motor se eșantionează accelerometrul integrat pe platformă.

- *Poziția ancorelor LPS influențează performanțele sistemului*

  În cadrul testelor inițiale de evaluare a sistemului LPS au fost observate variații de poziționare (jitter) de 30-50 cm pentru axele X, Y, respectiv 15-20 cm pentru axa Z, cu mult peste valorile specificate de producător. În urma unei analize detaliate a fost constatată influența căilor multiple de propagare ale semnalelor radio pentru unele perechi de ancore. Astfel, la nivelul secvențelor de comunicație TWR timpii de propagare ai pachetelor radio în cele două sensuri ajungeau să difere cu mai mult de 1000 cuante de timp (măsurarea distanțelor se realizează indirect pe baza unor numărătoare care contorizează durata transmisiilor și care aplică mărci de timp la nivelul pachetelor).

  Soluție: A fost dezvoltată o aplicație pentru monitorizarea duratelor de propagare ale pachetelor (Fig. 3) și s-a procedat la repoziționarea ancorelor pentru minimizarea diferențelor dintre valorile obținute pentru pachetele comunicate în cele două sensuri.

Distance avg and stdev

Received from

	A0	A1	A2	A3	A4	A5	A6	A7
A0	0.00	13921.00	14959.98	29838.64	56157.00	46075.02	62212.00	27063.00
	0.00	1381.17	6158.37	18420.69	1261.97	6735.12	1008.77	1304.70
A1	14069.00	0.00	15370.95	10104.00	39121.00	53899.00	35114.00	42941.00
	1925.35	0.00	2552.35	979.39	1133.40	1159.17	1149.63	1293.12
A2	14839.98	15297.98	0.00	37934.03	52827.00	48053.98	22365.00	28133.00
	6701.31	1880.51	0.00	13908.45	996.82	5334.75	966.15	1190.87
A3	26816.70	10120.00	55070.08	0.00	21337.00	54930.14	49147.00	7212.00
	18687.57	832.46	16248.69	0.00	1008.82	22141.06	1027.47	824.07
A4	55933.00	50529.08	52887.00	21209.00	0.00	17429.95	52424.00	55221.00
	1371.83	5330.54	969.21	1044.02	0.00	10664.80	2145.63	969.68
A5	47111.00	53945.00	56469.00	55070.14	17771.91	0.00	53167.18	47553.00
	4256.14	961.44	1382.68	22190.71	10086.82	0.00	13463.21	1484.32
A6	62238.00	35188.00	22281.00	49257.00	52376.00	46213.25	0.00	10337.00
	980.88	1054.17	1094.16	927.89	2102.95	14731.17	0.00	1436.48
A7	27063.00	42729.98	28107.00	3765.99	55177.00	47535.00	10381.00	0.00
	1261.26	4034.53	1296.60	5638.06	923.56	1470.47	1306.30	0.00

Fig. 3 (a): Valorile distanțelor după repoziționarea parțială a ancorelor

Distance avg and stdev

Received from

	A0	A1	A2	A3	A4	A5	A6	A7
A0	0.00	6781.00	59400.95	0.00	41051.07	36923.00	58016.00	19011.00
	0.00	1537.16	7652.41	0.00	6992.84	1291.50	1051.17	1247.95
A1	6757.00	0.00	6781.00	0.00	56171.00	48935.00	29718.00	46775.00
	1352.19	0.00	1716.39	0.00	1464.26	1004.53	1024.34	1051.04
A2	60103.00	6941.00	0.00	0.00	53103.96	53505.00	19865.00	28155.00
	1546.87	1166.89	0.00	0.00	4822.74	1131.72	1153.67	1083.71
A3								
A4	39323.10	55877.00	53433.00	0.00	0.00	14773.78	53582.00	56821.00
	9928.01	1250.74	981.47	0.00	0.00	12050.70	1204.33	983.81
A5	36963.00	49063.00	53371.00	0.00	13555.69	0.00	44941.45	48385.00
	1169.12	1173.24	1194.69	0.00	11895.20	0.00	20242.28	1175.55
A6	58146.00	29622.00	19935.00	0.00	53498.00	48795.30	0.00	11385.00
	1065.01	889.75	1121.92	0.00	1338.89	17991.19	0.00	1009.49
A7	19133.00	46837.00	28089.00	0.00	56757.00	48319.00	11433.00	0.00
	1233.48	1069.27	860.62	0.00	1078.99	1109.84	992.83	0.00

Fig. 3 (b): Valorile distanțelor după repoziționarea ancorelor

Algoritmi suplimentari și exemple

Evitarea coliziunilor în etapa de planificare

Soluțiile de planificare propuse anterior returnează secvențe de mișcare și de efectuare a acțiunilor de către roboții echipei, împreună cu momente necesare de sincronizare, acestea fiind prezentate pe larg în monografia [4]. În timpul mișcării trebuie evitate coliziunile între roboți, pentru acest lucru putându-se apela la procedee relaționate cu domeniul de alocare a resurselor. Ca noutate, cercetările raportate în lucrarea [5] încorporează evitarea coliziunilor direct în problema de planificare pe baza unei specificații Booleene. Astfel, sunt extinse soluțiile anterioare din [6], unde nu se putea garanta evitarea coliziunilor, ci doar se încerca minimizarea posibilității de apariție a acestora.

Pe scurt, se presupune o echipă cu N_r roboți, un set de propoziții atomice (acțiuni și regiuni) $Y = \{y_1, y_2 \dots, y_{|Y|}\}$ și o specificație Booleană pentru întreaga echipă. Specificația este impune cerințe intermediare (de-a lungul traiectoriilor) folosind setul $Y_i = \{Y_1, Y_2 \dots, Y_{|Y|}\}$ și cerințe pentru starea finală a agenților folosind setul $Y_f = Y$. Formula Booleană este presupusă de formă normal conjunctivă, precum $\varphi = \varphi_1 \wedge \dots \varphi_n$, fiecare termen φ_i fiind o disjuncție cu elemente din setul Y_i sau Y_f. Pentru echipa robotică se consideră un model RMPN (Robot Motion Petri Net) notat cu Q, definit în studii anterioare precum [6-7]. Pentru formularea unei probleme de planificare care să garanteze lipsa coliziunilor, rezultatele din [5] necesită următoarele ipoteze adiționale față de [6]:

(i) Fiecare disjuncție φ_i conține elemente fie din setul Y_i, fie din Y_f, dar nu din ambele.

(ii) Partea formulei referitoare la specificații intermediare poate fi satisfăcută de o singură configurație a celor N_r roboți, adică există cel puțin un marcaj al modelului Q care îndeplinește cerințele intermediare. Bineînțeles, acest lucru este adevărat pentru cerințele în starea finală, deoarece altfel formula nu ar putea fi îndeplinită.

(iii) Fiecare disjuncție cu elemente din Y_i (cerințe intermediare) poate conține fie mai multe elemente ne-negate, fie este egală doar cu un singur element negat.

(iv) La momentul inițial roboții sunt plasați în celule diferite din partiția P, adică marcajul inițial al rețelei Petri Q satisface $m_0[p] \leq 1, \forall p \in P$.

Se observă că presupunerile (i)-(iii) implică misiuni Booleene mai restrictive decât în [6]. De exemplu, din cauza ipotezei (i) nu putem avea formule conținând $(Y_1 \vee y_2)$, iar în cazul unui robot ipoteza (ii) implică imposibilitatea cerinței $(Y_1 \wedge Y_2)$, pe când soluția din [6] ar fi implicat un plan în care robotul satisfăcea pe rând Y_1, respectiv Y_2. Însă, precum a fost menționat, aceste ipoteze permit o soluție de planificare care garantează evitarea coliziunilor, chiar dacă traiectoriile se intersectează.

Metoda propusă în [5] se bazează pe rezolvarea a două probleme de programare liniară cu necunoscute întregi și reale (MILP), una pentru cerințele intermediare, iar cealaltă pentru cerințele finale. Aceste probleme sunt prezentate în Fig. 4, iar pentru detalierea notațiilor aferente direcționăm cititorul spre studiile [4-5].

$$\min \mathbf{1}^T \cdot \sum_{j=1}^{N_r+1} j \cdot \boldsymbol{\sigma}_j$$
$$\text{s.t. } \boldsymbol{m}_j = \boldsymbol{m}_{j-1} + \boldsymbol{C} \cdot \boldsymbol{\sigma}_j,\ j = 1, 2, \ldots, N_r + 1$$
$$\sum_{\gamma \in \mathcal{Y}_i \setminus \bar{\mathcal{Y}}} (\alpha_j(\gamma) \cdot x_\gamma) \geq 1 + \sum_{\gamma \in \mathcal{Y}_i \setminus \bar{\mathcal{Y}}} min\left(\alpha_j(\gamma), 0\right), \forall \varphi_j$$
$$N_r \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+1}, \forall \gamma \in \mathcal{Y}_i \setminus \bar{\mathcal{Y}}$$
$$x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+1}, \forall \gamma \in \mathcal{Y}_i \setminus \bar{\mathcal{Y}}$$
$$\boldsymbol{\eta} \cdot \boldsymbol{\sigma}_j = 0,\ j = 1, 2, \ldots, N_r + 1$$
$$\boldsymbol{Post} \cdot \boldsymbol{\sigma}_j + \boldsymbol{m}_{j-1} \leq \mathbf{1},\ j = 1, 2, \ldots, N_r + 1$$
$$\boldsymbol{m}_j \in \mathbb{R}_{\geq 0}^{|P|},\ j = 1, 2, \ldots, N_r + 1,$$
$$\boldsymbol{\sigma}_j \in \mathbb{N}_{\geq 0}^{|T|},\ j = 1, 2, \ldots, N_r + 1, \boldsymbol{x} \in \{0, 1\}^{|\mathcal{Y}_i \setminus \bar{\mathcal{Y}}|}.$$

(a)

$$\min \mathbf{1}^T \cdot \sum_{j=1}^{N_r+2} j \cdot \boldsymbol{\sigma}_j$$
$$\text{s.t. } \boldsymbol{m}_j = \boldsymbol{m}_{j-1} + \boldsymbol{C} \cdot \boldsymbol{\sigma}_j, j = 1, 2, \ldots, N_r + 2,$$
$$\sum_{\gamma \in \mathcal{Y}_f} (\alpha_j(\gamma) \cdot x_\gamma) \geq 1 + \sum_{\gamma \in \mathcal{Y}_f} min\left(\alpha_j(\gamma), 0\right), \forall \varphi_j$$
$$N_r \cdot x_\gamma \geq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+2}, \forall \gamma \in \mathcal{Y}_f$$
$$x_\gamma \leq \boldsymbol{v}_\gamma \cdot \boldsymbol{m}_{N_r+2}, \forall \gamma \in \mathcal{Y}_f$$
$$\boldsymbol{\eta} \cdot \boldsymbol{\sigma}_j = 0,\ j = 1, 2, \ldots, N_r + 1$$
$$\boldsymbol{Post} \cdot \boldsymbol{\sigma}_j + \boldsymbol{m}_{j-1} \leq \mathbf{1},\ j = 1, 2, \ldots, N_r + 2$$
$$\boldsymbol{m}_{N_r+1} - \boldsymbol{Pre} \cdot \boldsymbol{\sigma}_{N_r+2} \geq \mathbf{0}$$
$$\boldsymbol{m}_j \in \mathbb{R}_{\geq 0}^{|P|},\ j = 1, 2, \ldots, N_r + 2,$$
$$\boldsymbol{\sigma}_j \in \mathbb{N}_{\geq 0}^{|T|},\ j = 1, 2, \ldots, N_r + 2, \boldsymbol{x} \in \{0, 1\}^{|\mathcal{Y}_f|}.$$

(b)

Fig. 4: Formulări MILP asigurând evitarea coliziunilor, corespunzătoare cerințelor Booleene: (a) intermediare, (b) finale.

Pe scurt, problema MILP (a) generează strategii robotice la sfârșitul cărora roboții satisfac cerințele intermediare, iar MILP (b) generează continuări către pozițiile finale. Sunt necesare sincronizări în marcajele intermediare ale celor două probleme MILP pentru a se asigura evitarea coliziunilor. Funcțiile de cost sunt gândite astfel încât (atunci când este posibil) unele marcaje succesive să rezulte identice, reducând astfel momente de sincronizare.

Pseudo-codul soluției de planificare Booleană cu evitarea coliziunilor este dat în Algoritmul 1 de mai jos. Se pornește de la testarea valorii de adevăr a ipotezelor (i)-(iv) – în caz negativ se utilizează metoda din [6] (care nu garantează evitarea coliziunilor), iar în caz afirmativ se continuă cu soluțiile corespunzătoare MILP (a) și (b), asigurându-se sincronizările necesare.

Algorithm 1. Solution pseudo-code
Input: RMPN $\mathcal{Q}$, set $\mathcal{Y}$, formula φ, N_r
Output: Robot movement strategies
  if *Any assumption (i)-(iv) is False* **then**
    Use the more complex and general method from [6] and Return solution;
  Solve MILP (a);
  Construct movement plans from non-empty $\boldsymbol{\sigma}_j$, $j = 1, \ldots, N_r + 1$;
  Robots synchronize in each marking $\boldsymbol{m}_j$ for which $\boldsymbol{\sigma}_j \neq \mathbf{0}$, $j = 1, \ldots, N_r + 1$;
  Set $\boldsymbol{m}_0 := \boldsymbol{m}_{N_r+1}$, to use in MILP (b);
  Solve MILP (b);
  Robots further move based on non-empty $\boldsymbol{\sigma}_j$, $j = 1, \ldots, N_r + 2$;
  Robots synchronize in each marking $\boldsymbol{m}_j$ for which $\boldsymbol{\sigma}_j \neq \mathbf{0}$, $j = 1, \ldots, N_r + 2$

Formulările MILP (a) și (b) oferă soluții complete pentru problema de planificare. Din punct de vedere al complexității (judecată prin numărul necunoscutelor), problema (a) are $(N_r + 1) \cdot |P|$ variabile reale, $(N_r + 1) \cdot |T|$ variabile întregi și $|Y \setminus \underline{Y}|$ variabile binare, unde $|P|$ și $|T|$ reprezintă numărul de poziții, respectiv tranziții ale modelului Q, iar $|Y \setminus \underline{Y}|$ este numărul de propoziții atomice care apar fără a fi negate în specificația φ. Problema (b) are $(N_r + 2) \cdot |P|$ variabile reale, $(N_r + 2) \cdot |T|$ variabile întregi și $|Y|$ variabile binare. În contrast, soluția din [6] avea un număr de variabile dependent de un parametru k ales de utilizator, valori

prea mici ale acestui parametru putând duce la infezabilitatea problemei de optimizare, iar valori mari crescând complexitatea optimizării.

Pentru exemplificarea procedurii de planificare de mai sus, considerăm scenariul din Fig. 5 și specificația $\varphi = (\Lambda_{i=1}^{10} \neg Y_i) \Lambda (\Lambda_{i=11}^{20} y_i)$, care impune atingerea în starea finală a regiunilor din partea dreaptă, evitând de-a lungul traiectoriilor regiunile din mijloc.

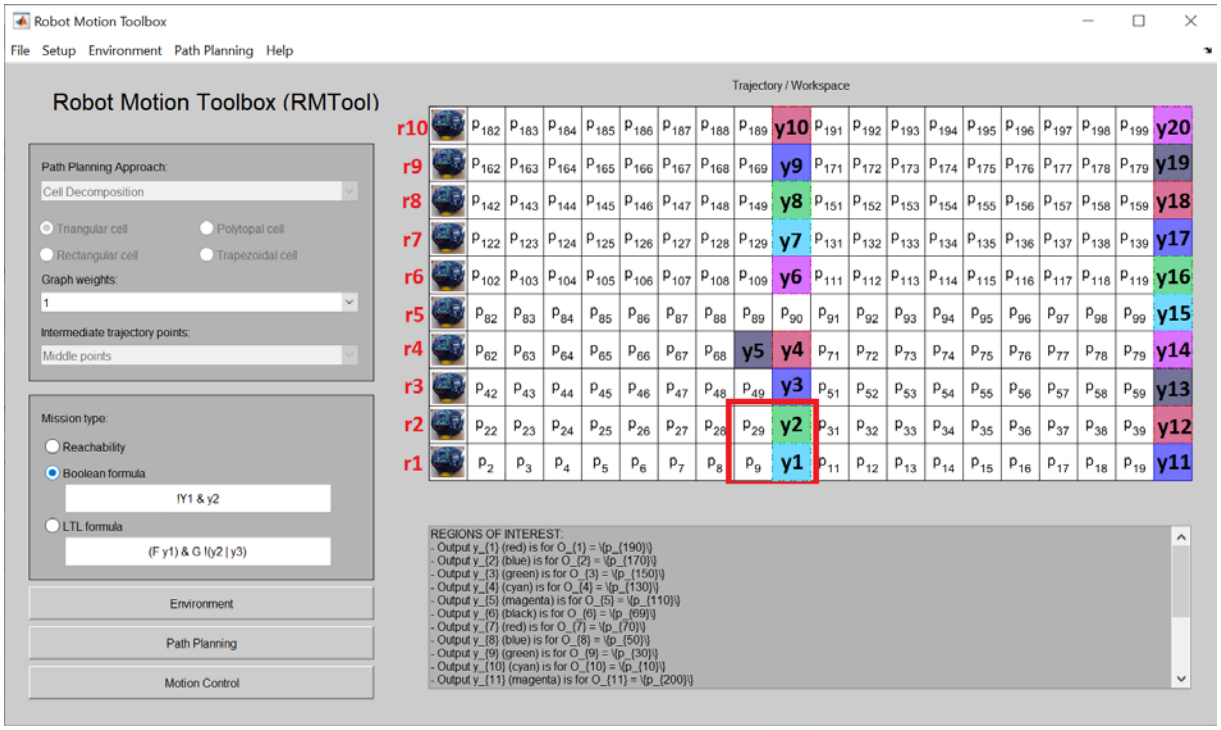


Fig. 5: Domeniu de evoluție cu 10 roboți, 200 de celule în partiție și 20 de regiuni de interes.

Implementarea a fost realizată în Matlab și integrată în RMTool (Robot Motion Toolbox) [7]. O soluție conform Algoritmului 1 a fost obținută în aproape o secundă (pe un calculator cu procesor i7 gen. 8), problemele de optimizare având (a) 10360 variabile, respectiv (b) 11300 variabile. Coliziunile sunt evitate, în acest sens fiind necesare 10 momente de sincronizare, ilustrate în Fig. 6 împreună cu traiectoriilor robotice obținute. Dacă s-ar fi utilizat metoda din [6], ar fi fost posibilă apariția coliziunilor (deoarece unele marcaje intermediare ale modelului Q ar fi avut mai mult de două jetoane în aceeași poziție / celulă a partiției), pe când acest lucru nu este posibil folosind metoda menționată anterior [5]. Devine clară necesitatea sincronizărilor între roboți, acest lucru fiind transpus în practică prin experimente cu drone, prezentate spre sfârșitul documentului curent.

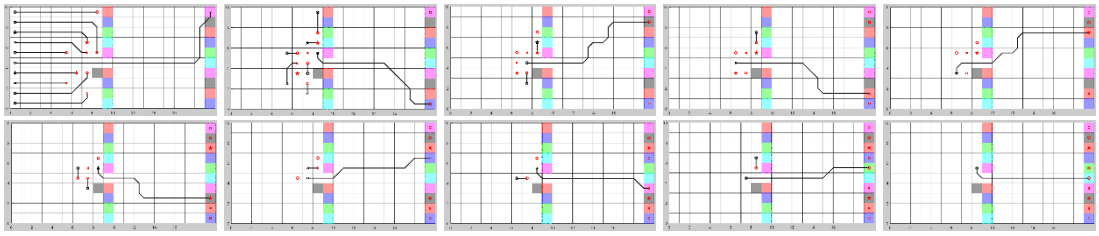


Fig. 6: Momente de sincronizare necesare evitării coliziunilor robotice.
Traiectoriile obținute satisfac specificația φ.

Pentru a extinde sfera aplicabilității specificațiilor de nivel înalt, lucrarea [8] consideră o specificație Booleană globală doar pentru stări finale, dar într-un scenariu distribuit, unde roboții pot comunica între ei doar într-o anumită rază. În plus, roboții au doar informații parțiale asupra domeniului de evoluție și estimează zonele de interes din vecinătatea lor în timpul mișcării. Fiecare robot rulează un algoritm local prin care își actualizează estimările pe baza comunicațiilor cu roboții apropiați și încearcă satisfacerea specificației globale prin rezolvarea unei instanțe locale a unei probleme de optimizare.

Problemă de planificare pentru livrarea unor resurse

Un alt scenariu exemplificat pe o platforma de timp real se referă la echipe de drone care au acțiuni de preluare și livrare bunuri în regiuni de interes având restricții de energie [9]. Pentru acest scenariu se consideră următoarele ipoteze de lucru:

I. Se consideră un spațiu indoor în care sunt poziționate un număr finit de depozite de stocare bunuri (Fig. 7);
II. Se consideră un set de drone cu număr egal cu cel al depozitelor de bunuri;
III. Fiecare dronă poate transporta un singur tip de bun, un număr egal cu *c* din aceste bunuri, are energie egală cu *E* și se poate reîncărca în depozitele de stocare.
IV. În spațiul de lucru există un număr finit de centre de colectare bunuri, fiecare centru putând accepta un număr predefinit din fiecare categorie de bunuri.

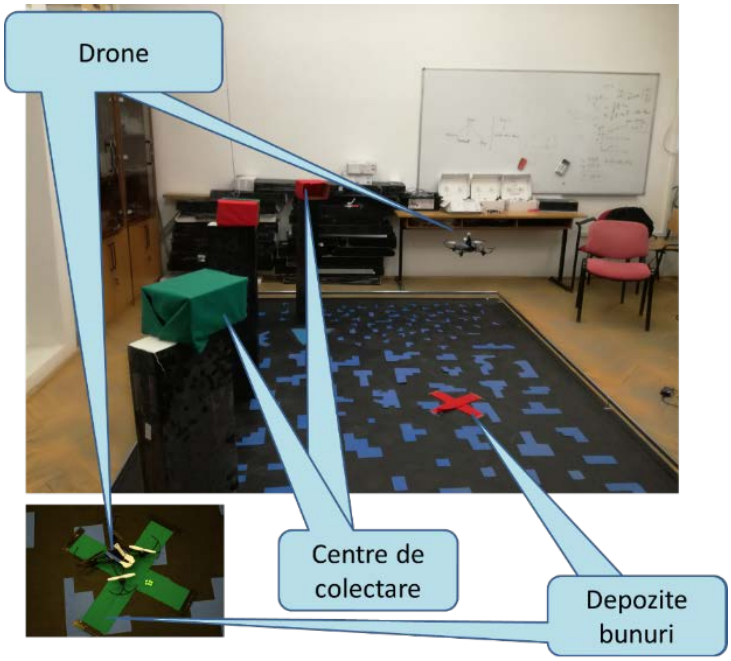


Fig. 7: Spațiul de lucru în platforma experimentală de timp real

Problema care modelează scenariul descris are ca scop calcularea secvențelor de preluare și transport de bunuri pentru fiecare dronă, optimizând consumul de energie. Adițional, toate transporturile trebuie realizate într-o durată maximă egală cu T. Soluția constă în construirea de formulări BIP pentru fiecare dronă (Fig. 8). Depozitul de bunuri și numărul de bunuri pe care trebuie să le livreze $|G|$ sunt elemente ale mulțimii $P = \{0, 1, 2, \ldots, |G|\}$.

Pentru realizarea zborului și finalizarea acțiunilor de livrare cu drone se consumă unități de energie și timp. Drona poate realiza maxim N_t tururi pentru a realiza toate livrările.

$$(i) \quad \min_{x_g^t, y_{i,j}^t, z^t} \sum_{t\in\{1,2,\ldots,N_t\}} \Big(\sum_{g\in G} u^e \cdot x_g^t + \sum_{i,j\in P,\, i\neq j} m^e \cdot d(i,j) \cdot y_{i,j}^t \Big)$$

subject to:

$$(ii) \quad \sum_{g\in G} u^e \cdot x_g^t + \sum_{i,j\in P,\, i\neq j} m^e \cdot d(i,j) \cdot y_{i,j}^t \leq E\,,\ \forall t \in \{1,2,\ldots,N_t\}$$

$$(iii) \quad \sum_{t\in\{1,2,\ldots,N_t\}} \Big(\sum_{g\in G} u^\tau \cdot x_g^t + \sum_{i,j\in P,\, i\neq j} m^\tau \cdot d(i,j) \cdot y_{i,j}^t + f^\tau \cdot z^t \Big) \leq T$$

$$(iv) \quad \sum_{g\in G} x_g^t \leq c\,,\ \forall t \in \{1,2,\ldots,N_t\}$$

$$(v) \quad \sum_{t\in\{1,2,\ldots,N_t\}} x_g^t = 1\,,\ \forall g \in G$$

$$(vi) \quad \sum_{g\in G} x_g^t \geq z^t\,,\ \forall t \in \{1,2,\ldots,N_t\}$$

$$(vii) \quad \sum_{g\in G} x_g^t \leq c \cdot z^t\,,\ \forall t \in \{1,2,\ldots,N_t\}$$

$$(viii) \quad \sum_{i\in P,\, i\neq g} y_{i,g}^t = x_g^t\,,\ \forall t \in \{1,2,\ldots,N_t\},\ \forall g \in G$$

$$(ix) \quad \sum_{j\in P,\, j\neq g} y_{g,j}^t = x_g^t\,,\ \forall t \in \{1,2,\ldots,N_t\},\ \forall g \in G$$

$$(x) \quad \sum_{g\in G} y_{0,g}^t = z^t\,,\ \forall t \in \{1,2,\ldots,N_t\}$$

$$(xi) \quad \sum_{g\in G} y_{g,0}^t = z^t\,,\ \forall t \in \{1,2,\ldots,N_t\}$$

$$(xii) \quad \sum_{\alpha\in P\setminus M} \sum_{\beta\in M} y_{\alpha,\beta}^t \geq x_g^t\,,\ \forall t \in \{1,2,\ldots,N_t\},\ \forall M \subset G,\ \forall g \in M$$

$$(xiii) \quad x_g^t, y_{i,j}^t, z^t \in \{0,1\}\,,\ \forall g \in G,\ \forall i,j \in P\,(i \neq j),\ \forall t \in \{1,2,\ldots,N_t\}$$

Fig. 8: Formulare BIP scenariu individual drona

Validarea soluției propuse a fost realizată în două faze. Prima fază folosind un simulator, în timp ce în a doua fază soluția a fost transpusă într-o aplicație de timp real. Pentru simulare (Fig. 9) se consideră un spațiu paralelipipedic în care evoluează două drone (una verde una roșie) având capacitate c=3. Depozitele de bunuri sunt considerate la nivelul solului (înălțime 0) și centrele de colectare sunt la o înălțime egală cu 3 unități de măsură. Formularea BIP pentru drona verde are 72 de necunoscute si 188 de restricții, în timp ce pentru drona roșie sunt 16 necunoscute și 25 de restricții. Formulările BIP au fost generate automat și rezolvate în mai puțin de 0.15 secunde folosind mediile Matlab și CPLEX.

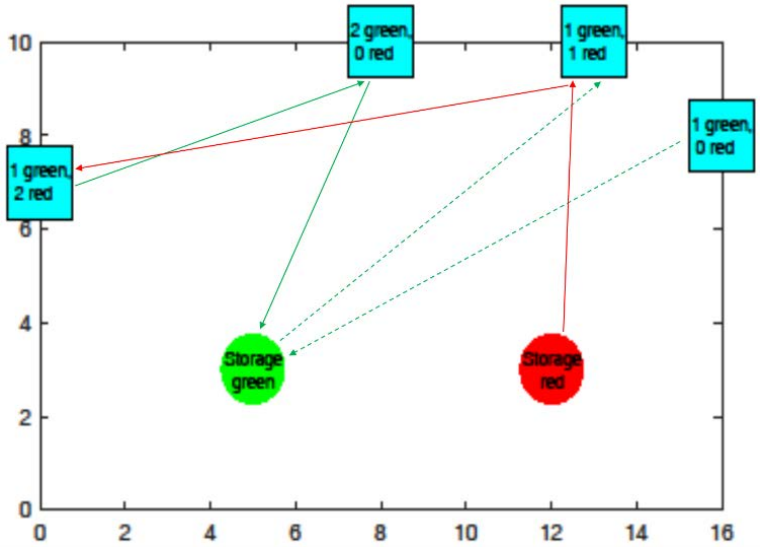


Fig. 9 Simularea soluției propuse

După validarea prin simulare a fost considerată o aplicație de timp real. Două drone sunt utilizate pentru a îndeplini sarcinile de transportare a bunurilor către centrele de colectare în spațiul descris în Fig. 10. Planificarea traiectoriilor este realizată off-line în mediul Matlab după o identificare prealabilă a dronelor, centrelor de colectare și a depozitelor. Identificarea se realizează folosind informații de culoare și adâncime achiziționate de un sensor RGB-D tip kinect v2 și procesate cu OpenCV. După construirea traiectoriilor ce implică și acțiuni de preluare și depunere bunuri, acestea sunt transmise via bluetooth dronelor utilizând un script Python.

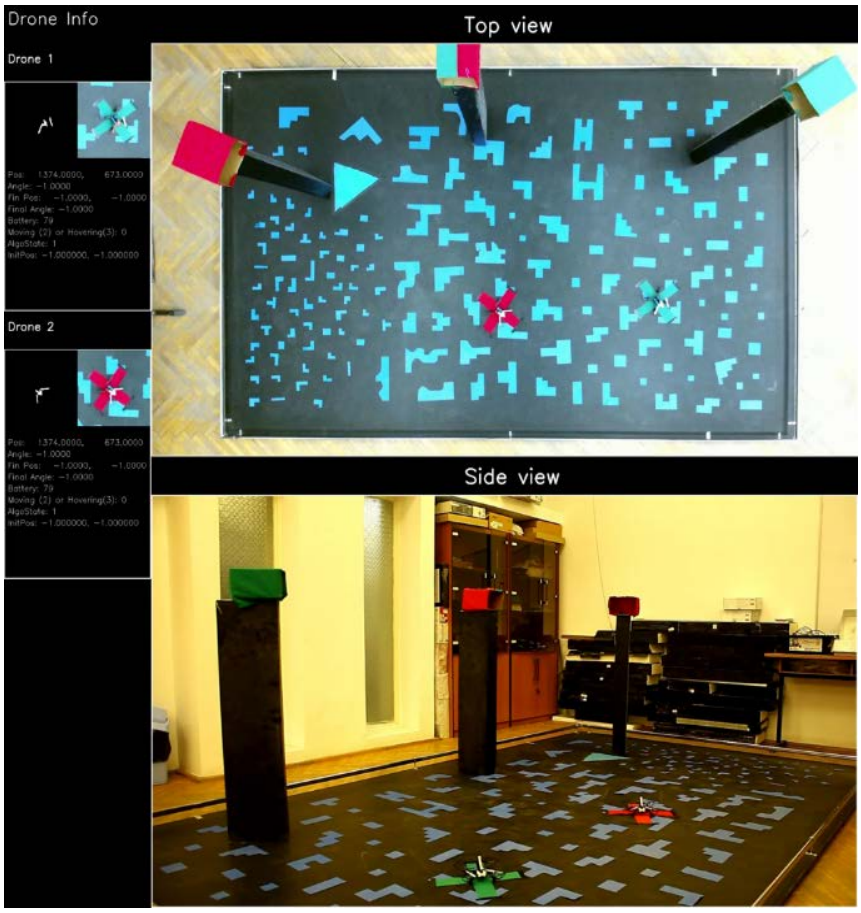


Fig. 10: Platformă experimentală

Planificarea traiectoriilor asigură realizarea sarcinilor cu un consum minim de energie și într-un timp sub 10 secunde. Experimentele în timp real pot fi vizualizate accesând link-ul: https://www.youtube.com/watch?v=SDDywcx6rms. Scenariul anterior poate fi extins de exemplu inserând pauze de alimentare cu energie a dronelor prin aterizarea pe stații de încărcare wireless, în funcție de energia curentă și cea necesară pentru preluările și livrările asignate.

Simulare soluții de planificare cu specificații bazate pe acțiuni

Exemplul 1. Se considera o echipa de roboti $R = \{r_1, r_2\}$, cu capabilitati identice, prezenti intr-un mediu industrial automatizat reprezentat de mediul de lucru rectangular din Fig. 11. Mediul de lucru consta din trei zone de interes: zona de productie (marcata cu rosu), zona depozit materiale (marcata cu albastru), zona de mentenanta echipamente (marcata cu verde).

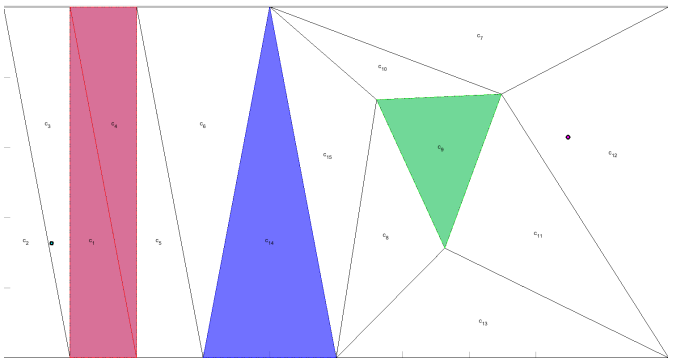


Fig. 11: Harta descompunerii pe regiuni a mediului de lucru in exemplul 1.

Mediul de lucru este descompus in celule obtinandu-se setul $C = \{c_i, i \in [1,15]\}$, si presupunem ca robotii sunt plasati in starea initiala in regiunile c_3 si c_{12}.

Se considera actiunea π_1 de "verificare proces productie", actiunea π_2 de "verificare a stocului de materiale" si actiunea π_3 de "reparare echipament" astfel incat setul $\Pi = \{\pi_1, \pi_2, \pi_3\}$.

Actiunea π_1 se poate efectua in regiunile c_1 si c_4 corespunzand zonei de productie, actiunea π_2 in c_{14} repezentand zona de depozitare, si actiunea π_3 in c_9 corespunzatoare zonei de mentenanta, astfel ca setul C^* al regiunilor de interes este definit ca $C^* = c_1, c_3, c_4, c_9, c_{12}, c_{14}$, incluzand si regiunile in care sunt plasati initial robotii.

Consideram o specificatie pentru echipa de roboti mobili, de forma "Verifica proces productie, verifica stoc materiale si asigura repararea echipamentelor defecte". Formal, specificatia LTL va avea forma $\phi_1 = (F\pi_1)\&(F\pi_2)\&(F\pi_3)$. Aceasta specificatie impune realizarea cooperativa de catre echipa de roboti a actiunilor specificate in regiunile in care acestea se pot efectua.

Simularea acestei specificatii in RMTool [4] furnizeaza solutia din Fig. 12. Traiectoriile robotilor sunt reprezentate prin conectarea punctelor mediane ale segmentelor partajate de celule adiacente.

Obtinerea acestei solutii folosind metoda bazata pe retele Petri descrisa in raportul etapei anterioare de proiect presupune definirea setului de pozitii si de tranzitii ale retelei pe baza ipotezelor de lucru si reducerea grafului de adiacenta corespunzator celulelor mediului de lucru la graful de regiuni de interes, retinand drumul cel mai scurt in graful complet intre oricare doua noduri din graful redus. Figura de mai sus ilustreaza traiectoriile robotilor corespunzatoare detalierii acestor drumuri.

Ipotezele de lucru conduc la definirea setului P de pozitii in reteaua Petri prin $P = \{p_i, i \in [1,6]\}$.

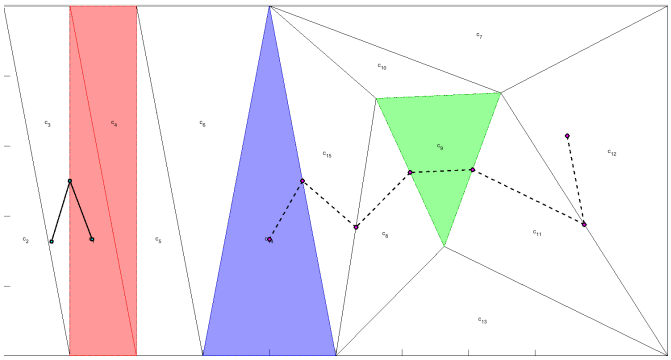


Fig. 12: Traiectoriile ce satisfac specificatia ϕ_1 din exemplul 1.

$$\begin{array}{l|l} p_1 : (\pi_1, c_1) & p_4 : (\pi_3, c_9) \\ p_2 : (\pi_1, c_4) & p_5 : (\emptyset, c_3) \\ p_3 : (\pi_2, c_{14}) & p_6 : (\emptyset, c_{12}) \end{array}$$

Setul de tranzitii este reprezentat de $T = \{t_1, \ldots t_{20}\}$, ce contine toate tranzitiile intre oricare doua pozitii distincte din setul P, mai putin cele inspre p_5 si p_6 ce au asociata actiunea $\emptyset$. Spre exemplu, tranzitia intre p_1 si p_3 are semnificatia deplasarii unui robot intre celulele c_1 si c_{14} si efectuarea in c_{14} a actiunii π_2. Tranzitia inversa, intre p_3 si p_1 semnifica deplasarea unui robot intre c_{14} si c_1 si efectuarea actiunii π_1 in celula c_1. Tranzitia intre p_1 si p_4 echivaleaza cu deplasarea unui robot intre c_1 si c_9, pe drumul cel mai scurt, si efectuarea π_3 in celula c_9.

Marcajul initial al sistemului este $m_0 = [0,0,0,0,1,1]^T$, alfabetul de iesire este $\Pi = \{\pi_1, \pi_2, \pi_3\}$ iar functia observatiilor: $h(p_1) = h(p_2) = \{\pi_1\}$, $h(p_3) = \{\pi_2\}$, $h(p_4) = \{\pi_3\}$, $h(p_5) = h(p_6) = \emptyset$.

Vectorul caracteristic al lui π_1 este $v_1 = [1,1,0,0,0,0]$ avand in vedere ca iesirea π_1 poate fi observata in p_1 si p_2, $v_2 = [0,0,1,0,0,0]$, iar $v_3 = [0,0,0,1,0,0]$.

Astfel, $V = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$.

Deoarece $V \cdot m_0 = [0,0,0]^T$, nicio observatie din Π nu este activa in m_0.

Exemplul 2. In acest exemplu vom ilustra situatia in care se doreste combinarea unei specificatii pe actiuni cu specificatii de miscare pentru robotii mobili. Consideram mediul de lucru si setul de roboti din exemplul 1 si adaugam in misiunea echipei constrangerea evitarii unei regiuni, de exemplu: "Verifica stocul de materiale din depozit si efectueaza reparatiile echipamentelor defecte, evitand trecerea prin zona de productie". Pentru a formaliza evitarea unei regiuni in cadrul unei specificatii pe actiuni vom defini o noua actiune de tipul "evita regiune". Vom obtine astfel setul de actiuni $\Pi = \{\pi_1, \pi_2, \pi_3\, \pi_4\}$, unde π_4 corespunde actiunii de evitare a unei regiuni, iar restul actiunilor sunt cele din exemplul 1:

$$\begin{array}{l|l} p_1 : (\pi_1, c_1) & p_5 : (\pi_4, c_1) \\ p_2 : (\pi_1, c_4) & p_6 : (\pi_4, c_4) \\ p_3 : (\pi_2, c_{14}) & p_7 : (\emptyset, c_3) \\ p_4 : (\pi_3, c_9) & p_8 : (\emptyset, c_{12}) \end{array}$$

iar specificatia LTL va fi formulata sub forma $\phi_2 = (F\pi_2)\&(F\pi_3)\&G!(\pi_4)$. Simularea misiunii in RMTool furnizeaza solutia din Fig. 13.

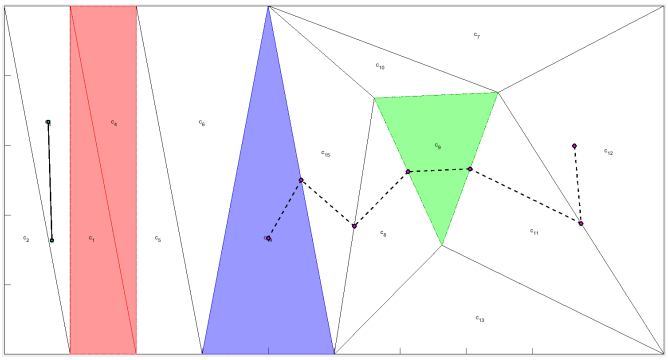


Fig. 13: Traiectoriile ce satisfac specificatia ϕ_2 din exemplul 2. Robotul r_1 nu participa la indeplinirea misiunii datorita restrictiei de vizitare a regiunii de productie (rosu).

Se observa ca intreaga misiune este satisfacuta cu un singur robot, avand in vedere ca robotul r_1 nu poate contribui datorita restrictiei de vizitare a zonei de productie.

Daca se elimina din misiune cerinta de reparare a echipamentelor defecte iar restrictia de vizitare se introduce si asupra zonei de mentenanta, misiunea poate fi specificata formal prin $\phi_3 = (F\pi_2)\&G!(\pi_4)$ avand solutia ilustrata in Fig. 14.

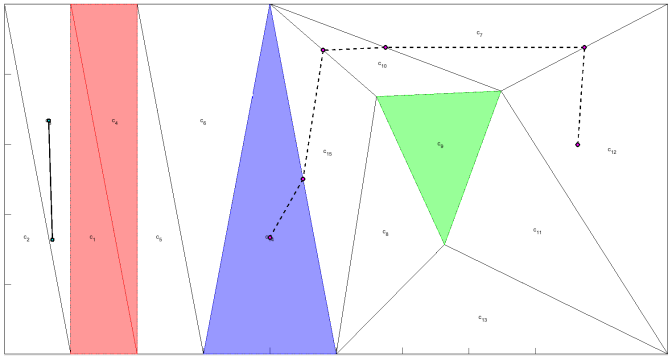


Fig. 14: Traiectorii ce satisfac specificatia ϕ_3. Robotul r_1 nu participa la indeplinirea misiunii datorita restrictiei de vizitare a regiunii de productie (rosu), iar robotul r_2 indeplineste misiunea de verificare a stocului de materiale evitand trecerea prin zona de mentenanta.

Pentru aceasta configuratie, setul de pozitii in reteaua Petri este definit de:

$$\begin{array}{l|l} p_1 : (\pi_1, c_1) & p_6 : (\pi_4, c_4) \\ p_2 : (\pi_1, c_4) & p_7 : (\pi_4, c_9) \\ p_3 : (\pi_2, c_{14}) & p_8 : (\emptyset, c_3) \\ p_4 : (\pi_3, c_9) & p_9 : (\emptyset, c_{12}) \\ p_5 : (\pi_4, c_1) & \end{array}$$

Descrierea experimentelor adiționale și analiza performanțelor

În cadrul experimentelor adiționale s-au realizat teste în timp real cu mai multe drone Crazyflie 2.0, folosind configurația hardware descrisă în prima secțiune a documentului. În acest context, s-au realizat teste experimentale în timp real cu 3 și 4 drone efectuând diferite acțiuni sincronizate. Avantajele oferite de configurația mixtă LPS plus Flowdeck pot fi evidențiate în problemele de planificare în care sunt implicate mai multe drone cu sarcini diferite.

În etapa anterioară a acestui proiect, s-au realizat teste cu o singură dronă CF2, considerând configurații diferite pentru sistemele de poziționare: *Drona CF2 (fără sistem de poziționare), Drona CF2 și Z-Ranger (feedback poziție doar pe axa Z)* și *Drona CF2 și Flow Deck v2,* fiind raportate cele mai bune rezultate [10]. Pentru fiecare configurație, au fost considerate 4 scenarii de testare. Experimentele s-au desfășurat impunând o înălțime minimă constantă de 1 m față de sol, restricție motivată de precizia senzorului Kinect, care a fost folosit pentru evaluarea preciziei estimatorului de poziție. Datorită acestui neajuns, experimentele din această a III-a etapă care au implicat acțiuni de decolare și aterizare a dronelor Crazyflie, nu au putut fi validate prin folosirea senzorului Kinect.

Experimentele în timp real au implicat câteva scenarii de testare, cu 2 echipe a câte 2 drone (Fig. 15) și 3 drone (Fig. 16) ce realizează acțiuni cu unul sau mai multe puncte de sincronizare. În Tabel 2 sunt prezentate atât scenariile de test rulate în care echipele de drone colaborează la îndeplinirea unei specificații globale, cât și scenariile cu evitarea de obstacole. Traiectoriile pot fi planificate pe baza algoritmilor dezvoltați, iar momentele de sincronizare returnate asigură evitarea coliziunilor.

Tabel 2. Scenarii de testare

Scenarii de test	**Descriere acțiuni (decolare, survolare/evitare regiuni/obstacole și aterizare)**
1.	Două echipe a câte 2 drone (R1, R2) și (R3, R4) poziționate la puncte fixe (rezultate din planificare).
2.	Decolarea simultană a celor 4 drone la punct fix la înălțimea de 2 metri timp de 5 secunde.
3.	Coborâre simultană a unei echipe de 2 drone la înălțimea de 1 metru (R1, R2), cealaltă echipă de 2 drone (R3, R4) menținându-și zborul la punct fix la înălțimea de 2 metri, acțiunile fiind sincronizate.
4.	Deplasare liniară aproximativ 3.5 metri a dronelor R1 și R2, pe axa X la înălțimea de 1 metru, acțiune sincronizată cu deplasarea liniară a celorlalte 2 drone R3 și R4 la înălțimea de 2 metri.
5.	Aterizare simultană în regiuni prestabilite.
6.	Deplasarea a 3 drone cu survolare de regiuni (1, 2, respectiv 4 regiuni), cu moment de sincronizare în ultima regiune vizitată (Fig. 17). Regiunile vizitate de drona R2 sunt poziționate la înălțimi diferite.
7.	Deplasare liniară urmată de un moment de sincronizare comun pentru toate cele 4 drone, urmat de survolarea unor regiuni (la înălțimi diferite față de înălțimea de decolare) și deplasarea către locurile de aterizare, cu traiectorii care se intersectează de câte două ori (Fig. 18).

8.	Scenariu ce include evitarea de obiecte (câte 1 obiect pentru R2 și R3, respectiv 2 obiecte pentru R1) și survolare de regiuni. Sunt descrise două momente de sincronizare, ce au ca scop prioritizarea deplasării pe segmentele de intersecție ale traiectoriilor (Fig. 19).

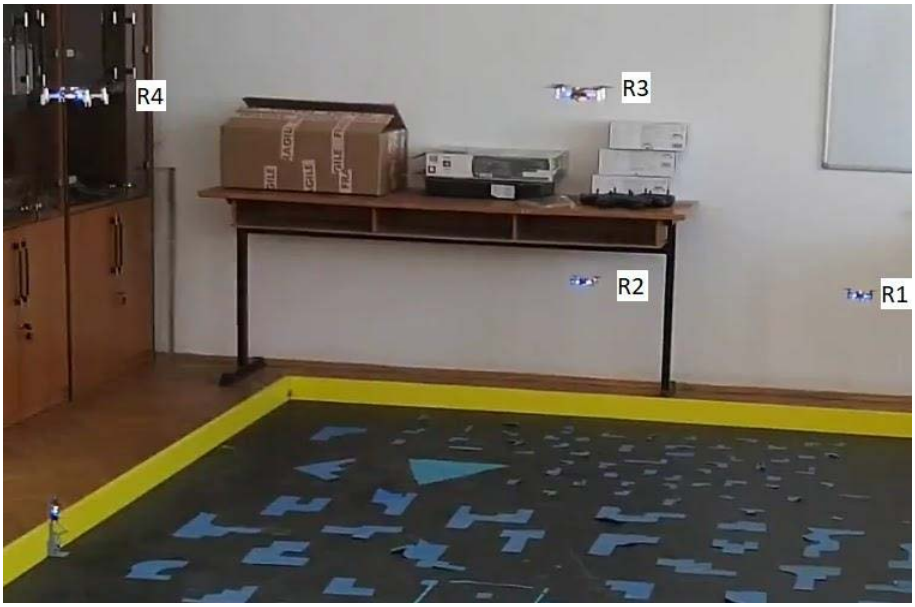


Fig. 15: Testare acțiuni sincronizate cu 2 echipe a câte 2 drone

Deplasarea între regiuni se realizează liniar, centrul regiunii ce trebuie survolată fiind la capătul unui segment care marchează deplasarea. Scenariile în care există mai multe obstacole de diferite forme implică parcurgerea unor traiectorii prestabilite de către 3 drone Crazyflie. Obstacolele au fost realizate din polistiren expandat de culoare neagră (Fig. 16).

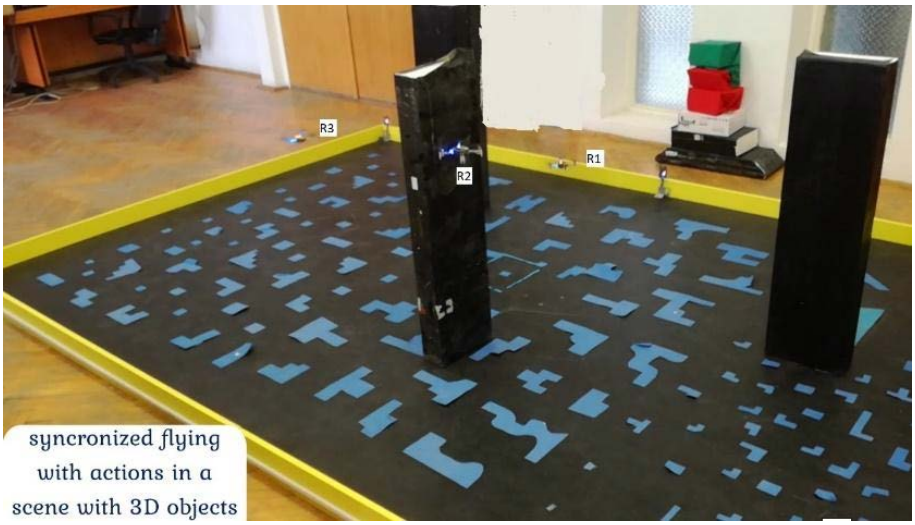


Fig. 16: Poziționarea obstacolelor în scenariul cu evitare de obstacole

Experimentele descrise mai sus pot fi vizualizate accesând link-ul: https://www.youtube.com/watch?v=tig0uLlNRXo.

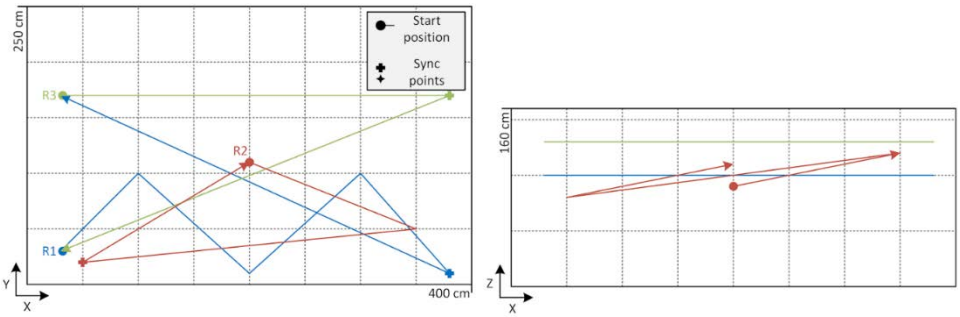


Fig. 17: Scenariu de test cu intersecții multiple și un singur moment de sincronizare

În Fig. 17 sunt reprezentate traiectoriile parcurse de drone în spațiul de lucru, fiind marcate punctele de plecare și punctele de sincronizare, conform descrierii din Tabelul 2. Traiectoria parcursă de fiecare dronă este descrisă prin segmente de drepte.

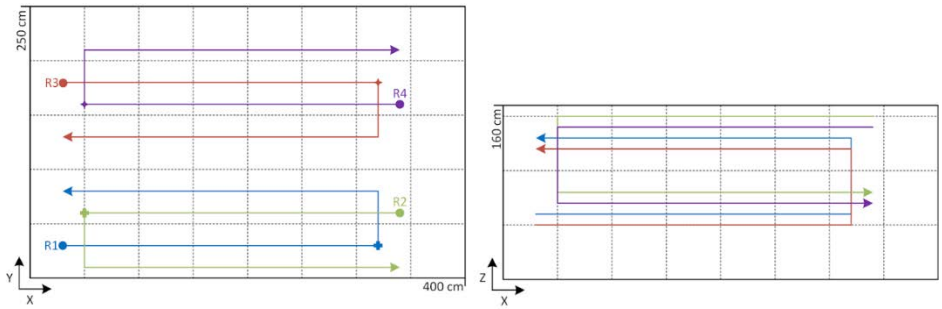


Fig. 18: Scenariu de test cu survolare de regiuni la înălțimi diferite

Au fost efectuate mai multe încercări cu poziționări diferite ale obstacolelor în spațiul de lucru, traiectoriile parcurse de cele 3 drone fiind reprezentate în Fig. 19, împreună cu două momente de sincronizare.

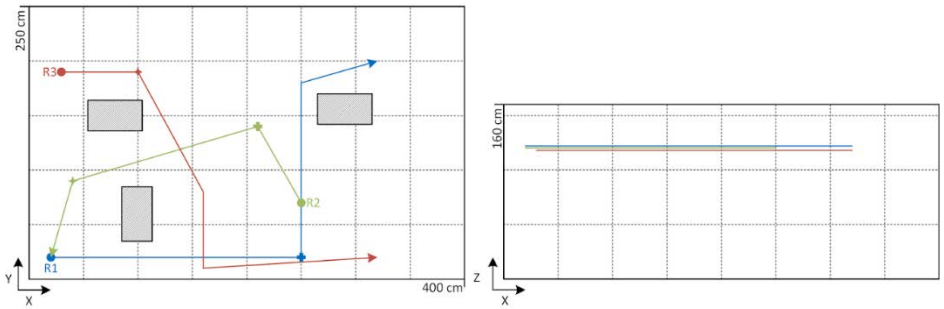


Fig. 19: Scenariu de test cu evitare de obstacole și două momente de sincronizare

Soluțiile de planificare propuse returnează secvențe de mișcare și de efectuare a acțiunilor de către roboții echipei, împreună cu momente necesare de sincronizare. Pentru experimentele bazate pe soluțiile de planificare propuse a fost aleasă o soluție de poziționare mixtă, bazată pe sistemele LPS și Flowdeck. Testele efectuate au relevat un comportament stabil al dronelor, cu ușoare oscilații doar în vecinătatea punctelor de destinație, respectiv cu o ușoară derivă în cazul zborului la punct fix. Integrarea algoritmilor de planificare într-un instrument software și validarea în timp real contribuie la creșterea impactului cercetărilor efectuate, permițând totodată dezvoltări ulterioare ce îmbină aspectele teoretice și experimentale.

Bibliografie - etapa 3

[1] Loco-positioning-system for Crazyflie 2/0, available at https://www.bitcraze.io/getting-started-with-the-loco-positioning-system/.

[2] E. Karapistoli F. N. Pavlidou I. Gragopoulos I. Tsetsinas "An overview of the IEEE 802.15.4a Standard" IEEE Communications Magazine vol. 48 no. 1 pp. 47-53 Jan. 2010.

[3] Measurements of accuracy and precision of the Loco Positioning system, available at https://wiki.bitcraze.io/misc:investigations:lps-precision.

[4] Cristian Mahulea, Marius Kloetzer, Ramón González, *Path Planning of Cooperative Mobile Robots Using Discrete Event Models*, Wiley-IEEE Press, IEEE Press Series on Systems Science and Engineering, series editor MengChu Zhou, ISBN 978-1-119-48632-9, 2020.

[5] Cristian Mahulea, Marius Kloetzer, Jean-Jacques Lesage, *Multi-robot Path Planning with Boolean Specifications and Collision Avoidance*, 15th Workshop on Discrete Event Systems (WODES'20), Rio de Janeiro, Brazil, 2020 (accepted).

[6] Cristian Mahulea, Marius Kloetzer, *Robot Planning Based on Boolean Specifications Using Petri Net Models*, IEEE Transactions on Automatic Control (TAC), vol. 63 (7), pp. 2218-2225, 2018.

[7] Marius Kloetzer, Cristian Mahulea, *Path Planning for Robotic Teams based on LTL Specifications and Petri Net Models*, Discrete Event Dynamic Systems, vol. 30 (1), pp. 55-79, 2020.

[8] Cristian Mahulea, Eduardo Montijano, Marius Kloetzer, *Distributed Multirobot Path Planning in Unknown Maps Using Petri Net Models*, 21st IFAC World Congress, Berlin, Germany, 2020 (accepted).

[9] M. Kloetzer, A. Burlacu, G. Enescu, S. Caraiman, C. Mahulea, Optimal Indoor Goods Delivery Using Drones, 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1579-1582, 2019.

[10] C. Budaciu, N. Botezatu, M. Kloetzer, A. Burlacu: On the Evaluation of the Crazyflie Modular Quadcopter System, *IEEE 24th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Zaragoza, Spain, 2019.