UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

# PROIECT DE DIPLOMĂ

Sistem de Localizare Robust pentru un Robot Mobil Holonomic prin Fuziunea Datelor IMU și Viziune Computerizata

Alexandru Toader

**Coordonator științific:**
Șl. dr. ing. Alexandru-Jan Văduva

**BUCUREŞTI**

2025

# CUPRINS

## SINOPSIS

Performanța roboților mobili autonomi în sarcini complexe, precum cele din competițiile internaționale de robotică, este direct dependentă de acuratețea și robustețea sistemului de localizare. Platformele holonomice cu roți Mecanum sunt preferate în aceste scenarii pentru manevrabilitatea superioară, însă introduc provocări semnificative: alunecarea roților generează erori cumulative rapide în odometrie, iar senzorii inerțiali (IMU) suferă de un drift care le degradează fiabilitatea pe termen lung. Fără un mecanism de corecție, aceste erori combinate fac imposibilă navigarea precisă. Pentru rezolvarea acestei probleme, lucrarea propune să dezvolte un sistem de localizare de înaltă precizie, pentru un robot cu roți mecanum, capabil să funcționeze fiabil într-un mediu cunoscut și structurat, specific unui teren de competiție. Obiectivul central este implementarea unui filtru Kalman extins ce fuzionează datele de la giroscopul și accelerometrul unui IMU, cu un sistem extern de localizare bazat pe markere fudiciale ArUco[10], cât și o platformă hardware complet funcțională.

## ABSTRACT

The performance of autonomous mobile robots in complex tasks, such as those in international robotics competitions, is directly dependent on the accuracy and robustness of the localization system. Holonomic platforms with Mecanum wheels are preferred in these scenarios for their superior maneuverability, but they introduce significant challenges: wheel slippage generates rapid cumulative errors in odometry, and inertial sensors (IMUs) suffer from a drift that degrades their long-term reliability. Without a correction mechanism, these combined errors make precise navigation impossible. To solve this problem, this paper proposes the development of a high-precision localization system for a Mecanum-wheeled robot, capable of operating reliably in a known and structured environment, typical of a competition field. The central objective is the implementation of an Extended Kalman Filter that fuses data from an IMU's gyroscope and accelerometer with an external localization system based on ArUco[10] fiducial markers and complete functional robotic platform.

# 1 INTRODUCERE

## 1.1 Context

Avansul tehnologic al ultimelor decenii a facilitat introducerea roboților mobili autonomi în tot mai multe domenii. În centrul acestei revoluții se află nevoia acestora de a percepe mediul înconjurător, acest factor jucând un rol cheie în procesul de a lua decizii, pentru a putea îndeplini, în mod fiabil, sarcini complexe.

Roboții autonomi au pătruns și în domenii unde manevrabilitatea precisă în spații restrânse și aglomerate este un factor critic, fiind nevoie ca aceștia să dețină capabilitatea de deplasare holonomică (omnidirecțională). Odată cu această revoluție a roboților autonomi în industrie, au apărut și competiții internaționale, unde participanții se confruntă cu provocări dificile pentru a proiecta, construi și programa astfel de roboți; una dintre acestea este Eurobot.

În acest context, un robot cu capabilități holonomice este ideal pentru tipul de competiții precum Eurobot[1], unde capacitatea de a se deplasa cu ușurință printre obstacole reprezintă un avantaj semnificativ. Motivația principală a acestei lucrări este de a dezvolta un robot omnidirecțional, dar și un sistem performant de localizare a acestuia, componente cheie pentru a asigura succesul viitoarelor echipe ale universității, ce vor participa la competiții internaționale de prestigiu.

## 1.2 Problema

roblema fundamentală abordată de această lucrare o reprezintă lipsa de acuratețe a unui cadru robotic holonomic. Deși acestea prezintă capabilități excepționale, esențiale pentru astfel de concursuri, sunt susceptibile la derapaje, alunecări pe suprafața de contact, făcând ca senzorii pe bază de encodere să nu fie eficienți în astfel de implementări.

Pe de altă parte, unitățile de măsură inerțială (IMU), deși oferă date cu o frecvență înaltă, nu este fiabilă o implementare doar pe baza acestui tip de senzor din cauza zgomotului datelor.

Astfel, problema principală a lucrării este de a obține o estimare continuă, precisă și robustă a poziției unui robot holonomic, pe un teren de joc predefinit, depășind limitele impuse de șasiu și de imperfecțiunile senzorilor.

## 1.3 Obiective

Obiectivul principal al acestei lucrări este proiectarea unei arhitecturi de fuziune a datelor, creând astfel un estimator robust, capabil de predicții precise, cu o frecvență a datelor cât mai înaltă, absolut necesar pentru controlul autonom al unui astfel de robot, cât și integrarea acestuia cu o platformă mecanică modulară, capabilă de mișcare holonomică, ce are atât rolul de validator pentru implementarea sistemului de localizare, cât și ca fundament pentru proiectarea și construirea roboților ce vor reprezenta universitatea în cadrul competițiilor internaționale.

## 1.4 Structura lucrării

Capitolul 2 prezintă tehnologiile folosite în prezent pe partea de șașiuri capabile de mișcare holonomică, senzori folosiți în general, cât și algoritmi de viziune computerizată, pentru poziționarea unui robot în spațiu, cât și algoritmi de fuziune ai datelor obținute de la acești senzori.

Capitolele 3 și 4 prezintă implementarea propusă pentru a rezolva problema descrisă mai sus, capitolul 3 fiind mai mult axat pe aspectele teoretice, pe când capitolul 4 este axat pe implementarea efectivă.

Capitolul 5 prezintă rezultatele obținute în urma implementării și a testelor, iar capitolul 6 reprezintă concluziile trase.

# 2 MODELE EXISTENTE

## 2.1 Roboți cu capabilități holonomice

Un robot holonomic este un robot al cărui număr de grade de libertate controlabile este egal cu numărul de grade de libertate ale spațiului în care se deplasează. În cazul unui teren plat, acest lucru înseamnă capacitatea de a se mișca independent de-a lungul axelor x și y (translație) și de a se roti în jurul propriului său centru, rotație în jurul axei z.

### 2.1.1 Roțile Mecanum[7]

Acest tip de roți a fost patentat de inginerul suedez Bengt Erland Ilon în anul 1972, pe când lucra la compania Mecanum AB. Roata mecanum prezentată în figura 1, este construită având role dispuse de-a lungul circumferinței sale, amplasate la un unghi de 45 de grade față de axul principal de rotație.

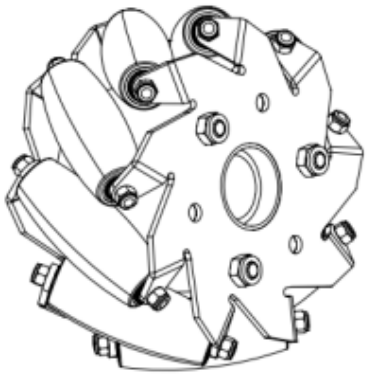


Figura 1 Reprezentare grafică a unei roți mecanum[7]

Totuși, roțile mecanum alunecă în timpul mișcării datorită rolelor amplasate de-a lungul circumferinței, în plus, acestea produc vibrații datorită formei neregulate.

### 2.1.2 Roți Omnidirectionale[12]

Acestea au o construcție similară cu roțile mecanum, tot bazată pe role pentru a obține mișcarea holonomică, diferența de construcție constând în amplasarea acestora perpendicular pe axul roții.

Un mod de construcție al unui șasiu bazat pe acest tip de roți îl reprezintă dispunerea acestora în colțurile unui triunghi echilateral, model prezentat în lucrarea "Kinematics and Control A Three-wheeled Mobile Robot with Omni-directional Wheels"[13].

Dezavantajele acestui tip de șasiu sunt similare cu cele ale unui robot cu roți mecanum: alunecarea și vibrațiile ce au loc în timpul deplasării, plus, un astfel de șasiu este mai dificil de integrat în proiectarea unui robot și cu alte capacități.

### 2.1.3 Platforme sferice

Reprezintă o abordare complet diferită de modelele prezentate anterior pentru a obține o mișcare holonomică. Aceasta propune ca, în loc de un șasiu convențional, bazat pe trei roți,

precum implementarea roților omnidirecționale, sau patru, necesare unui șasiu bazat pe roți mecanum, robotul să se balanseze pe o sferă controlată de motoare interne. Platformele sferice oferă o manevrabilitate extraordinară, cu costul unei implementări mult mai complexe și limitări serioase în proiectare. Un astfel de robot este descris în lucrarea "899: BALLBOT"[14].

Roțile mecanum reprezintă cea mai comună abordare în vederea obținerii mișcării holonomice, iar acestea au fost folosite în cadrul realizării proiectului, deoarece, deși prezintă dezavantaje, forma triunghiulară a șasiului impusă de roțile omnidirecționale ar putea impune limitări viitoare în funcționalitățile necesare ale robotului. Roțile mecanum au fost alese în defavoarea unei platforme sferice, datorită complexității implementării și instabilității acesteia.

## 2.2 Tipuri de senzori

Pentru implementarea oricărui algoritm descris în secțiunea trecută, nu doar EKF, trebuie obținute informații referitoare la starea actuală a robotului. Există mai multe tipuri de senzori ce pot fi folosiți tocmai cu acest scop.

### 2.2.1 IMU

Senzorii IMU integrează mai multe componente:

- Accelerometru: care calculează accelerația senzorului pe toate cele trei axe, x, y și z. Totuși aceștia suferă din cauza zgomotului, pentru a putea calcula poziția robotului, datele de la senzor ar trebui integrate de două ori după timp, astfel, amplificând perturbațiile date de zgomot.
- Giroscop: ce calculează viteza unghiulară în jurul celor trei axe, x, y și z. La fel ca la accelerometru, datele trebuie integrate după timp, iar nici acest tip de senzor nu este perfect, fiind și el afectat de zgomot
- Magnetometru: ce măsoară intensitatea și direcția câmpului magnetic local, principiul de funcționare este similar cu al unei busole digitale, calculând astfel orientarea în jurul axei Z. Principala problemă o reprezintă perturbațiile magnetice generate de metale feroase din apropiere.

### 2.2.2 Encoderele de rotație

Acest tip de senzor măsoară rotațiile axului unui motor în timpul deplasării. Spre deosebire de componentele unui IMU, acestea nu suferă de zgomot în sine, dar acumulează erori, atât din imperfecțiunile procesului de construcție al robotului, cum ar fi diferențe între diametrele roților, sau faptul că acestea ar putea să nu fie perfect aliniate, dar și din erori ce nu țin de deficiențe în procesul de fabricație sau construcție a robotului, ci de alunecări ale acestuia. Astfel, pentru un robot cu capacități holonomice, ce folosește roți mecanum, ca în cazul celui construit în cadrul acestei lucrări, encoderele nu reprezintă o alegere optimă.

### 2.2.3 Sisteme de poziționare globala

Calculează poziția pe glob pe baza timpului de parcurgere a semnalelor de la o rețea de sateliți, deși oferă o poziție absolută, suferă de deficiențe:

- Funcționează doar în exterior
- Acuratețe limitată
- Rată de actualizare scăzută

În implementarea lucrării am ales să folosesc un senzor de tipul IMU, dar și o alternativă a GPS-ului, folosind un sistem de localizare a robotului pe baza unui sistem de viziune computerizată, care, având o transmisie în timp real a robotului în cadrul terenului, acesta poate calcula poziția acestuia raportată la sistemul de coordonate al terenului.

## 2.3 Sisteme de viziune computerizata

Aceste sisteme se pot baza pe markere, care sunt folosite pe post de repere, pentru a identifica pozitionarea în spațiu a unui obiect, sau folosind mai multe markere, pentru a efectua transformări asupra întregului spațiu.

### 2.3.1 Markere fiduciale

Markerele fiduciale sunt repere artificiale folosite de algoritmii de viziune computerizată, pentru a identifica poziția unei camere în raport cu acestea.

Markerele ArUco[10] reprezintă un tip de markere fiduciale ce au fost dezvoltate de cercetătorii de la Universitatea din Columbia, acestea reprezintă o matrice de pătrate, cu exteriorul negru, și interiorul alcătuit din pătrate negre și albe, fiecare codificând un identificator.

În jurul acestui tip de markere au fost dezvoltate biblioteci mature, care folosesc mecanisme de detecție robuste, cu rate mici de fals pozitive, exemplu OpenCV.

Markerele AprilTag[11] reprezintă un competitor direct al markerelor ArUco[10]. Această implementare este considerată mai robustă, fiind mai eficientă la deformări de perspectivă și mai puțin sensibilă la condițiile de iluminare.

Totuși, în implementarea lucrării au fost folosite markere ArUco[10], motivul principal fiind imposibilitatea schimbării terenului, acesta având deja amplasate acest tip de markere.

## 2.4 Metode de estimare a poziționării unui robot în spațiu

Prima descoperire majoră în domeniul ingineriei sistemelor de control referitor la metodele de localizare este reprezentată de estimatorul liniar descoperit de Rudolf Kalman în anul 1960, numit astăzi în literatură filtru Kalman[2] . Acesta a prezentat o metodă recursivă de estimare a poziției unui sistem bazat pe informații imperfecte, afectate de zgomot, generate de senzori. Algoritmul propus are la bază un principiu bazat pe două etape, predicție și actualizare. În etapa de predicție, folosindu-se de starea anterioară a sistemului (poziție, viteză, rotație) și de un model matematic al legii mișcării robotului, acesta estimează poziția curentă a acestuia.

În cadrul etapei de actualizare, filtrul se folosește de informațiile de la senzori pentru a-și actualiza predicția, precum și parametrii interni, ce dictează nivelul de încredere pe care acesta îl are în modelul matematic, precum și în informațiile de la senzori.

Limitarea esențială a acestui filtru este presupunerea făcută că legea de tranziție a sistemului este liniară, ceea ce este rar întâlnit în sistemele reale.

Acest tip a reprezentat sursa de inspirație, fundamentul teoretic pentru viitoarele descoperiri în domeniul ingineriei controlului, pentru a rezolva problema estimării stării unui sistem.

### 2.4.1 Filtrul Kalman Extins(EKF)[3]

Majoritatea sistemelor robotice, inclusiv modelul cinematic al roților Mecanum, este unul neliniar. Filtrul Kalman extins reprezintă o soluție pentru limitarea principală a filtrului Kalman clasic, abordarea acestuia fiind de a liniariza sistemul non-liniar în jurul estimării curente a stării, la fiecare pas al algoritmului. Liniarizarea sistemului este realizată cu ajutorul matricelor Jacobiene, ce reprezintă derivatele parțiale de ordin întâi ale funcțiilor de transfer ale sistemului.

Acesta a fost publicat pentru prima dată în lucrarea "Application of Space-State Methods to Navigation Problems.", șase ani mai târziu de la publicarea lucrării lui R. E. Kalman.

Totuși, aproximarea sistemului prin liniarizare poate duce la acumularea de erori semnificative în timp, iar calculul Jacobienilor poate fi complex și predispus la erori.

### 2.4.2 Filtrul Kalman “Unscented”[4]

Filtrul Kalman „Unscented” (UKF) a apărut ca o soluție la limitările EKF, evitând liniarizarea analitică. În schimb, acesta propagă un set de puncte prin funcțiile non-liniare ale sistemului, apoi rezultatele sunt obținute prin recombinarea valorilor acelor puncte, astfel obținându-se o nouă estimare pentru starea sistemului. Folosind direct funcțiile neliniare pentru a realiza estimările, acesta reduce complexitatea EKF, oferind predicții în general mai bune.

Totuși, conform comparației realizate de Mathieu St-Pierre în lucrarea sa[], deși acuratețea EKF nu este la fel de bună ca a unui UKF, acesta este mai eficient din punct de vedere al costului computațional.[6]

### 2.4.3 Filtrul de particule[5]

Acesta se folosește de o metodă de tip Monte Carlo pentru a reprezenta distribuția de probabilitate a stării, printr-un set de eșantioane ponderate. Acestea pot modela orice tip de distribuție, pe când EKF și UKF sunt limitate la distribuția gaussiană. Aceasta este folosită în probleme de localizare complexe, unde EKF și UKF ar eșua.

Comparat cu celelalte două metode prezentate anterior, filtrul de particule are un cost computațional mult mai mare.[6]

În cadrul lucrării a fost implementat un EKF, tocmai din cauza costului computațional redus, comparat cu celelalte două metode.

# 3 SOLUȚIA PROPUSĂ

Implementarea mea propune atât proiectarea și construirea unui robot, cât și a unui sistem performant de localizare bazat pe informațiile oferite de o filmare în timp real a terenului, cât și pe un senzor inerțial (IMU) poziționat pe robot.

## 3.1 Modelul mecanic al robotului

Robotul este alcătuit din două componente principale: șasiul, ce se folosește de un sistem alcătuit din patru roți Mecanum al căror scop este de a oferi capabilități de deplasare holonomică, cât și rotație în jurul propriei axe.

A doua componentă o reprezintă un lift, acționat de două motoare și ridicat cu ajutorul unui sistem de scripeți, reprezentat în Figura 2. De acesta este atașat un mecanism de prindere pentru ca robotul să poată interacționa cu mediul exterior, putând muta diverse obiecte dintr-un punct în altul.

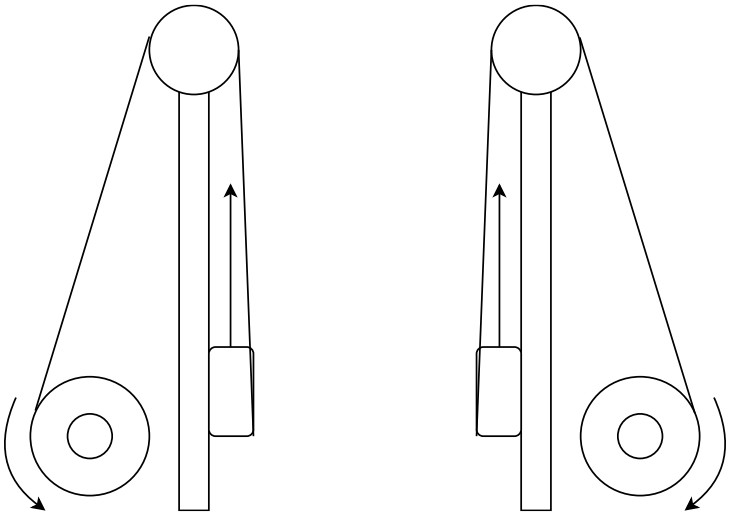


Figura 2 Schematica liftului

### 3.1.1 Proprietățile roților mecanum[8]

Construcția roților Mecanum este una inovatoare, distinctivă prin prezența unor role amplasate de-a lungul circumferinței, la un unghi de 45 de grade față de axa principală a roții. Roțile Mecanum vin în seturi de câte două, având rolele amplasate în direcții opuse una față de cealaltă.

Pentru a obține mișcarea holonomică, robotul este echipat cu patru roți Mecanum, amplasate astfel încât să formeze un model de „X” privit de deasupra, așa cum se poate observa în Figura 3. Tot din Figura 3, observăm că roțile Mecanum vin în perechi, acestea fiind de două tipuri:

- A, unde rolele se află la 45 de grade în sensul acelor de ceasornic față de axa principală a roții.
- B, unde rolele se află la 45 de grade în sensul invers acelor de ceasornic față de axa principală a roții.

Principiul de funcționare al roților Mecanum este reprezentat de forțele generate în timpul rotației, conform cu Figura 3, generând o forță la un unghi de 45 de grade față de direcția de mers, vector ce poate fi descompus în două forțe: una paralelă cu direcția de rotație, cât și o forță perpendiculară pe direcția de rotație. Prin controlul independent al vitezei și direcției de rotație se poate obține un vector de forță netă în orice direcție dorită, fără a fi nevoie de o mișcare de rotație a robotului, astfel neschimbând nici orientarea robotului.

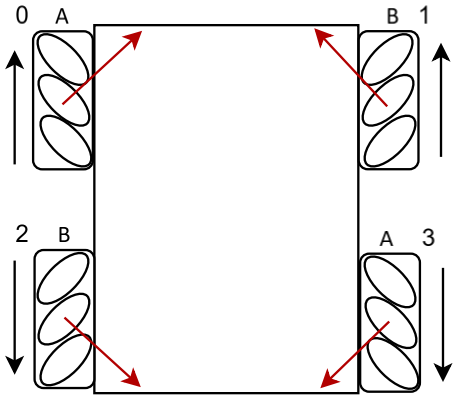



Figura 3 Vectorii generați de roți raportat la direcția de mișcare

Deși avantajele sunt evidente, un astfel de șasiu are și dezavantaje. În primul rând, roțile Mecanum sunt predispuse la alunecări, mulțumită rolelor, mai ales pe suprafețe mai puțin aderente. De asemenea, mulțumită construcției roților, acestea nefiind perfect circulare, produc vibrații.

Apoi, acest tip de șasiu necesită o complexitate software, cât și hardware crescută, deoarece, fiecare roată mișcându-se individual, sunt necesare patru motoare pentru controlul robotului.

### 3.1.2 Modul de control al sasiului[8]

Pentru a obține o mișcare holonomică, trebuie ca fiecare motor să se învârtă cu o anumită viteză. Pentru a calcula aceste viteze vom folosi următoarea formulă:

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \frac{1}{r}\begin{bmatrix}1 & -1 & -(l_x+l_y)\\ 1 & 1 & (l_x+l_y)\\ 1 & 1 & -(l_x+l_y)\\ 1 & -1 & (l_x+l_y)\end{bmatrix}\begin{bmatrix}v_x\\ v_y\\ \omega_z\end{bmatrix} \quad (1)$$

Unde $\omega_i$ (i = 0, 1, 2 ,3), numerotate conform cu Figura 3, reprezintă viteza de rotație a roții „i”, $v_x$ si $v_y$ reprezintă vitezele de deplasare de-a lungul axei x, respectiv axei y, iar $\omega_z$ reprezintă viteza de rotație în jurul propriei axe. Astfel, cunoscând viteza și direcția de deplasare a robotului, respectiv viteza de rotație, se pot calcula vitezele pentru fiecare roată, deci și comanda necesară fiecărui motor.

## 3.2 Arhitectura sistemului de control distribuit

Schema bloc din Figura 4 prezintă arhitectura robotului și sensul de transmisie a informațiilor și semnalelor. Sistemul principal de calcul al robotului este un Raspberry Pi 4, la care sunt conectate două microcontrolere ESP32. Rolul celui din stânga este de a controla motoarele șasiului, asigurând mișcarea robotului, iar cel de-al doilea este responsabil de controlul ansamblului liftului. De asemenea, este conectat și la senzorul inerțial, acesta colectând date și trimițându-le către Raspberry Pi.

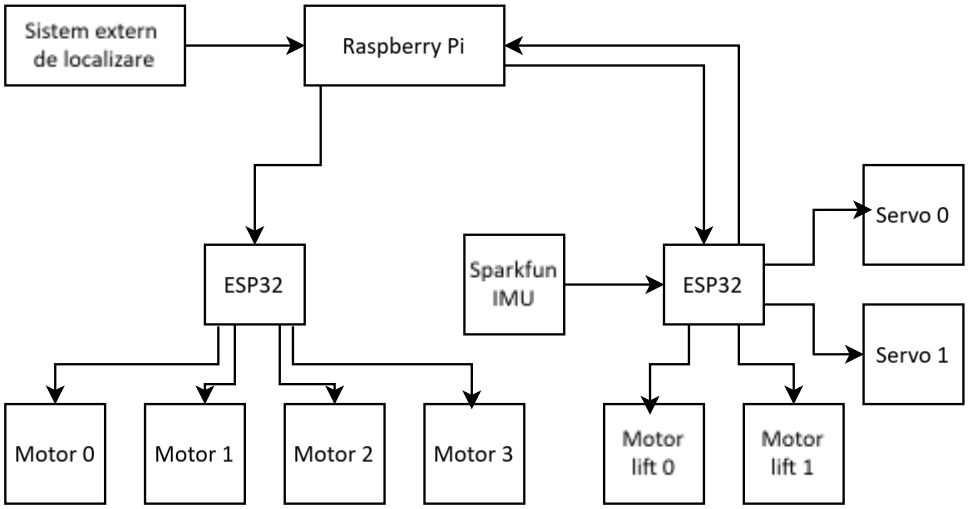



Figura 4 Arhitetura sistemului robotul

## 3.3 Sistemul de localizare

Pentru localizarea robustă a robotului este folosit un sistem de fuziune senzorială bazat pe un filtru Kalman extins (EKF). Acest algoritm estimează, în timp real, poziționarea și unghiul de rotație al robotului în interiorul terenului.

Filtrul se bazează pe informațiile oferite de două componente principale: senzorul inerțial amplasat pe robot și un sistem extern ce oferă coordonatele aproximative ale robotului, raportate la sistemul de coordonate al terenului, conform cu intrările Raspberry Pi-ului din Figura 4.

Cu ajutorul telefonului mobil, este transmisă spre un server filmarea în timp real a terenului. În cadrul terenului se află patru markere ArUco[10], ce sunt folosiți pentru realizarea proiecției omografice[22] a acestuia. De asemenea, pe robot este amplasat un alt marker, coordonatele acestuia fiind înregistrate și transmise către robot.

### 3.3.1 Principiul de funcționare al EKF

Filtrul Kalman extins are scopul de a prezice cu o acuratețe cât mai bună poziția robotului în planul de coordonate cartezian al terenului (coordonatele x și y ale acestuia), cât și rotația (yaw). Acesta se folosește de accelerațiile pe axele x și y ale robotului, cât și de rotația acestuia, date preluate de la senzorul inerțial, cât și de poziționarea acestuia în spațiu, date oferite de server.

Este importantă agregarea tuturor acestor date, deoarece, datele oferite de IMU, deși vin la o frecvență crescută, acestea suferă de zgomot, iar integrarea de două ori a accelerațiilor după timp duce la acumularea erorilor, iar, după un timp, datele devin complet inutilizabile. La fel, localizarea doar prin datele de la server este imposibilă, deoarece vin lent, iar și acestea suferă de zgomot.

Componentele principale ale filtrului sunt: starea actuală a robotului, notată cu x, și matricele de covariație P, Q și R, ce reprezintă incertitudinile sistemului raportate la starea actuală x, la modelul de mișcare al robotului, respectiv pentru valorile oferite de senzori.

Starea robotului, pentru modelul definit aici, este un vector cinci-dimensional:

$$x = \begin{bmatrix} x_{robot} \\ y_{robot} \\ \theta \\ v_x \\ v_y \end{bmatrix} \quad (2)$$

Unde $x_{robot}$ și $y_{robot}$ reprezintă poziția robotului, $\theta$ reprezintă unghiul de rotație al acestuia, iar $v_x$ și $v_y$ vitezele de-a lungul axelor x, respectiv y.

Astfel, EKF funcționează, conform cu Figura 5, în două etape:

- Predicție: etapa la care filtrul estimează poziția actuală a robotului.
- Actualizare: etapa la care filtrul ajustează predicția și parametrii săi interni, ținând cont de intrările sistemului.

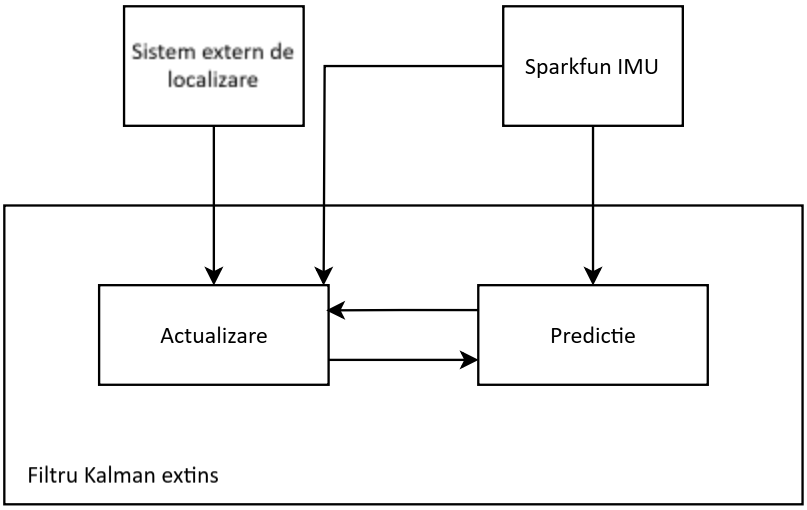



Figura 5 Modul de funcționare al EKF

Pentru etapa de predicție se folosesc ecuațiile legii de mișcare:

$$\dot{x} = v_x \tag{3}$$

$$\dot{y} = v_y \tag{4}$$

$$\begin{bmatrix} \dot{v_x} \\ \dot{v_y} \end{bmatrix} = R(\theta) * \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{5}$$

Unde $R(\theta)$reprezintă matricea de rotație cu unghiul $\theta$în sensul invers acelor de ceasornic. Valorile acesteia sunt date de următoarea formulă:

$$\theta$$

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{6}$$

Accerelațiile date de senzor sunt în reperul cartezian al acestuia, pe când, pentru a calcula corect mișcarea robotului, sunt necesare accelerațiile în reperul cartezian al terenului.

Astfel, pentru pasul de predicție, se folosește funcția f, ce are următoarea ecuație:

$$f(x_k, u_k) = F * x_{k-1} + B * R(\theta) * u_{k-1} \tag{7}$$

$$u_k = \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{8}$$

Unde valorile pentru F și B sunt următoarele:

$$F = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$B = \begin{bmatrix} \Delta t^2 & 0 \\ 0 & \Delta t^2 \\ 0 & 0 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \tag{10}$$

Filtrul Kalman Extins încearcă să liniarizeze o ecuație de mișcare neliniară, astfel, acesta, tot la pasul de predicție, actualizează valorile lui P folosindu-se de Jacobianul$J_f x_k$.

$$f(x_k, u_k) = f_i(x_k, u_k), \text{ i } \epsilon \{1, 2, 3, 4, 5\} \quad (11)$$

$$J_f = \left[\frac{\delta f_i(x_k)}{\delta x_k}\right] \quad (12)$$

$$P_k = J_{f,k} * P_{k-1} * J_{f,k}^T + Q \quad (13)$$

Pentru pasul de actualizare, întâi se calculează inovarea sistemului y și matricea de inovare a covariației S, acestea reprezintă nesiguranța sistemului în datele pe care le are, conform formulelor

$$y = z - H * x_k \quad (14)$$

$$S = H * P * H^T + R \quad (15)$$

Unde H reprezintă o matrice de dimensiune $m * n$, unde $m \leq n$, al cărui scop este de a transforma vectorul de stare x, de lungime $n$ într-un vector ce conține doar elementele relevante pasului de actualizare, de lungime $m$. Astfel, actualizând datele de la server, legate de pozitionare, produsul $H * x\_k$ ar returna un vector de lungime 2 ce conține doar pozițiile absolute prezise la pasul anterior.

Apoi, se actulizează câștigul filtrului:

$$K = P * H^T * S^{-1} \quad (16)$$

Acest câștig este folosit pentru a actualiza starea actuală a robotului, un câștig mare influențează o schimbare mai mare, deci o încredere mai mare în noile valori oferite de senzor, pe când, un câștig mic nu modifică la fel de mult valoarea predicției, deci modelul are mai multă încredere în prezicerile făcute după modelul de mișcare:

$$x_{k+1} = x_k + K * y \quad (17)$$

$$P_{k+1} = (I - K * H) * P \quad (18)$$

Unde $I$ reprezintă matricea unitate.

# 4 DETALII DE IMPLEMENTARE

Capitolul anterior a fost prezentată soluția propusă, arhitectura generală a sistemului, ce componentă se folosește de celelalte și cum, precum și modul teoretic în care lucrurile funcționează, acest capitol urmărește implementarea efectivă a soluției și tehnologiile folosite în realizarea proiectului.

## 4.1 Constructia robotului

Partea mecanică a robotului a fost proiectată și construită de la zero folosind:

- Profile de aluminiu, pentru șasiu
- Prinderi pentru axele motoarelor confecționate din aluminiu
- Bare de suport din aluminiu pentru structura liftului și rulmenți liniari pentru prinderea elementului mobil al acestuia
- Piese printate 3D folosind imprimanta Prusa Mk3s, pentru suporți de motoare, prinderi între profilele de aluminiu, suport pentru lift, scripeți, sistem de prindere, etc.

Au fost folosite profile de aluminiu pentru construcția robotului, în defavoarea unor placaje create special la CNC pentru a putea oferi robotului modularitate, astfel prin crearea unor prinderi speciale la 3D printer, piesele ce vin prinse de-a lungul profilelor pot fi amplasate oriunde de-a lungul acestora, comparat cu un model rigid, ce, o dată gândit și proiectat, nu mai poate fi schimbat atât de ușor. A fost nevoie de această flexibilitate, deoarece, robotul a suferit schimbări iterative, în urma deficiențelor constatate în timpul testărilor de-a lungul anului.

După cum se poate vedea în figura 5, piesa făcută la imprimanta 3D intră perfect în canalul profilului, la capătul acesteia putând fi atașat orice alt modul separat, cum ar fi prinderi pentru motoare, lift, suporți pentru senzori. Aceasta este fixată folosind șuruburi, strângându-le piesa rămâne fixă pe profilul de aluminiu.

Figură 6 Piesă prinsă de profilul de aluminiu; vedere de sus

Șasiul are un model în formă de „U”, permițând deplasarea robotului deasupra unor obiecte din mediul exterior, apucarea acestora prin mecanismul interior, ridicarea acestora de la sol folosind capabilitățile liftului, permițând astfel mutarea acestora de-a lungul terenului.

Pentru a putea controla și motoarele din față, dar a și permite această capabilitate și formă a șașiunului, motoarele roților frontale au fost montate la un unghi de 90 de grade, transmisia fiind realizată prin angrenaje conice. Roțile dințate folosite la aceste angrenaje au fost, de asemenea, create folosind imprimanta 3D. Acestea fiind destul de mari, nu au suferit uzură în urma testelor.

Liftul funcționează folosind un sistem de scripeți, fiecare braț al liftului folosind câte un motor DC, pentru a ridica mecanismul de prindere. De fiecare motor este atașat un scripete, ce este înfășurat cu sfoară. Sfoara trece peste un alt scripete și după este prinsă de baza mecanismului de prindere a liftului. Prin rotirea motorului spre exteriorul robotului, sfoara este strânsă de scripete iar liftul se ridică, iar prin rotirea în sensul celălalt, sfoara este eliberată, iar, mulțumită forței gravitaționale și forței de frecare scăzută între barele de aluminiu și rulmenții liniari, mecanismul de prindere este coborât, funcționând după principiul descris în figura 1.

Mecanismul de prindere este alcătuit din două servomotoare, câte unul atașat de fiecare braț al liftului. De fiecare motor este atașat câte un braț de care este prinsă o piesă creată special din TPU, un material cauciucat, ce are o aderență crescută, dar pe care imprimanta 3D îl poate folosi.

Poze ale robotului se afla în anexa 1.

## 4.2 Elementele electronice si protocoalele de comunicare din ansamblul robotului

Componentele electronice și de control folosite pentru construcția robotului sunt:

- Raspberry Pi 4b(x1) [14]
- ESP32(x2) [15]
- Convertor DC-DC lm2596s(x4) [16]
- Sursă DC-DC XL4015(x1) [17]
- Modul de comandă dual pentru motor HW-231 MC33886(x2) [18]
- Modul de comandă pentru motor Pololu DRV8801(x2) [19]
- Modul convertor logic cu 4 canale(x3)
- Motor Pololu 37D50L fără encoder(x4) [20]
- Motor Pololu 37D50L cu encoder(x2) [20]
- Servomotor MG996(x2)
- Senzor Sparkfun IMU bno086(x1) [21]

Schema electrică integrală se află la anexa 2.

### 4.2.1 Circuitul de alimentare a componentelor

Schema bloc din figură descrie ansamblul pentru alimentarea componentelor, acesta poate fi regăsit în schema electrică din anexă, dar aceasta poate fi dificil de urmărit, ținând cont că ansamblul de alimentare nu este trivial, astfel, este reprezentat într-o formă mai ușor de urmărit în figura 6..

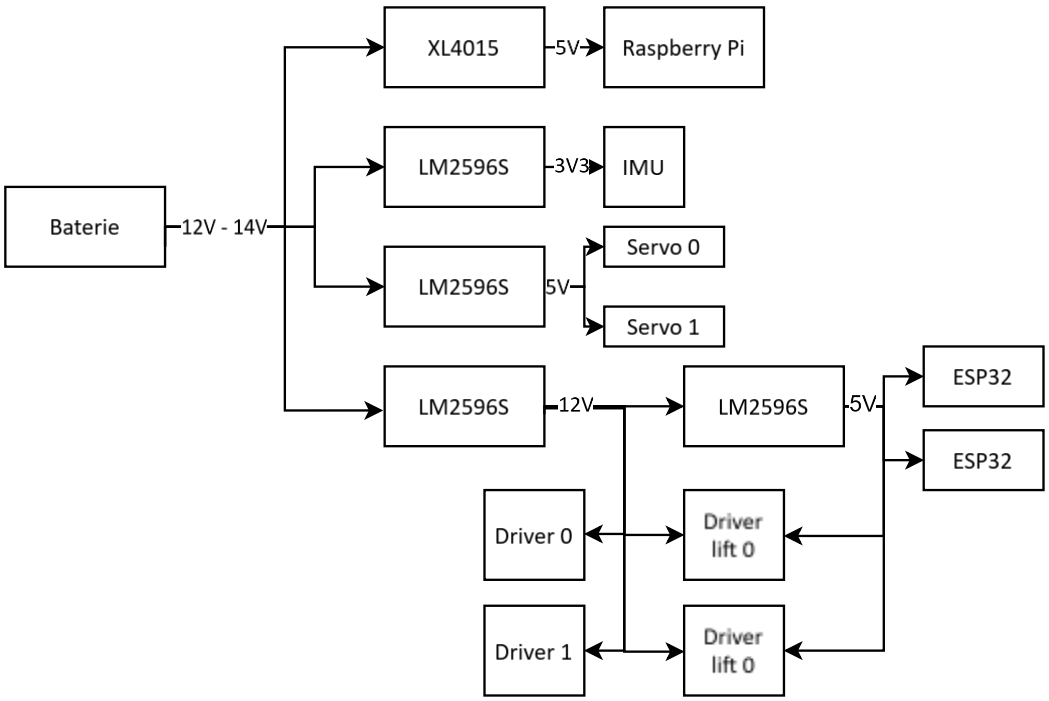



Figura 7 Schema electrică pentru alimentarea componentelor robotului

Robotul este alimentat de o baterie Tetrix, de tipul NiMH, de 12V și o capacitate de 3000mAh.

Bateria alimentează patru convertoare DC-DC, trei fiind de tipul lm2596s, iar unul XL4015, ce poate genera un curent mai mare, scopul acestuia este a alimenta raspberry-pi-ul. Am ales o sursă diferită pentru raspberry-pi, deoarece convertorul de tip lm2596s nu poate produce 3A, decât în condiții ideale și cu răcire dedicată, acesta fiind insuficient.

Un convertor este folosit pentru a stabiliza tensiunea din baterie la 12V, pentru a putea alimenta modulele de comandă ale motoarelor și pentru a alimenta cele două microcontrollere ESP32, printr-un alt convertor.

Servomotoarele sunt alimentate separat dintr-unul din convertoarele ce pleacă din baterie. Pentru a prinde un obiect, acestea stau în tensiune, crescând semnificativ puterea consumată, astfel afectând comportamentul oricărei alte componente conectate la același convertor.

În ultimul rând, senzorul IMU este conectat separat, prin ultimul convertor ce este alimentat de la baterie. Motivația pentru această decizie este că, vorbind de un senzor complex, cu un microprocesor dedicat, Arm Cortex M0, acesta, iar, la inițiere, consumul acestuia crește semnificativ. În urma experimentelor, curentul pe care-l putea genera un ESP32 pe linia de

3.3V era insuficient. O altă soluție ar fi fost legarea pe această linie, un condensator care să se descarce la inițiere, dar, am considerat că, un convertor dedicat este soluția cea mai elegantă..

### 4.2.2 Rolurile Raspberry Pi-ului și ESP32-urilor

Conform descrierii din capitolul 4 și a reprezentării din figura 3, robotul este construit pe baza unei arhitecturi de control distribuită, astfel, Raspberry Pi-ul este unitatea centrală de comandă, ce, pe baza datelor primite, trimite comenzi către ESP32-uri, ce sunt responsabile de controlul componentelor mobile ale robotului, motoare și servomotoare.

Unul dintre microcontrolerele ESP32 este conectat la modulele de comandă ce acționează motoarele șasiului. Modulele folosite pentru șasiu sunt HW-231 MC33886. Cum tensiunea minimă de intrare detectată garantat că unu logic este de 3.6V ($V_{IH}$ = 3.6V, conform fișei tehnice a piesei, iar un ESP32 poate genera semnal cu tensiunea de maxim 3.3V, ieșirile ESP32-ului au fost conectate la un convertor logic, câte unul pentru fiecare modul de control, ce amplifică semnalul logic la tensiunea de 5V. Pentru a alimenta convertoarele a fost folosită linia de 5V a modulelor de comandă. Cum, din construcția șasiului, fiecare roată trebuie acționată individual, controlul fiecărui motor este controlat separat, fiecare cu câte doi pini de PWM.

Celălalt ESP32 este responsabil de controlul motoarelor ce acționează liftul și de mișcarea servomotoarelor. Cum motoarele responsabile de acționarea liftului sunt acționate simultan, întotdeauna, sunt necesare doar două conexiuni de date, una de PWM și una pentru direcție, conexiune ce duce la intrările ambelor module de comandă. Cum motoarele trebuie să se rotească în direcții opuse, unul dintre acestea este conectat în sens invers. Comenzile servomotoarelor sunt individuale, pentru simplitate. Motivația este una destul de simplă, dacă la motoarele clasice mergeau pur și simplu inversate comenzile, pentru a obține direcții opuse, la servomotoare acest model de control nu este posibil.

De asemenea, tot la ultimul ESP32 este conectat și senzorul IMU. ESP32-ul reține ultima valoare și o trimite spre Raspberry Pi.

### 4.2.3 Senzorul Sparkfun IMU bno068

Acesta reprezintă o placă de dezvoltare creată în jurul senzorului bno086, proiectat de CEVA. Bno086 este un senzor IMU cu nouă grade de libertate.

Punctul său forte constă în capacitatea de a fuziona datele oferite de accelerometru, giroscop și magnetometru, direct pe placă cu ajutorul unui procesor ARM Cortex-M0+.

Senzorul se folosește de SH-2, o serie de algoritmi implementați de CEVA, pentru a procesa datele brute și a calcula parametrii precum vectori de rotație sau cuaternionii ce descriu rotația robotului în 3 dimensiuni.

Datele de care sistemul are nevoie sunt accelerația liniară de-a lungul axelor x și y, precum și de cuaternioni.

Cuaternionii [9] sunt folosiți o reprezentare, matematic elegantă, a rotației unui obiect în 3 dimensiuni, folosindu-se numere de forma:

$$q = w + x * i + y * j + z * k \quad (19)$$

Unde mulțimea {1, i, j, k} reprezintă baza canonică pentru spațiul vectorial al acestora, izomorf cu spațiul $\mathbb{R}^4$, aceștia reprezentând, practic, numere in 4 dimensiuni si respectă următoarea proprietate:

$$i^2 = j^2 = k^2 = ijk = -1 \quad (20)$$

Relația matematică între unghiurile de rotație în 3 dimensiuni și cuaternioni este dată de următoarele formule[11]:

$$\cos(\theta_2) * \cos(\theta_3) = w^2 + x^2 - y^2 - z^2 \quad (21)$$

$$\cos(\theta_2) * \sin(\theta_3) = 2 * (xy + wz) \quad (22)$$

Unde $\theta_2, \theta_3$ reprezintă rotațiile de-a lungul axei y, respectiv axei z.

Robotul aflându-se pe un teren plat, acesta se poate învârti doar de-a lungul axei z, iar cum senzorul este paralel cu solul, obținem relația:

$$\theta_2 = 0 \quad (23)$$

De unde rezultă:

$$\theta_3 = arctg((w^2 + x^2 - y^2 - z^2)/(2 * (xy + wz)) \quad (24)$$

Conform capitolului 3, cu modelul matematic al filtrului Kalman extins (EKF), este necesară calcularea rotației în jurul axei z.

Senzorul comunică cu ESP32 prin intermediul protocolului $I^2C$, cu precizarea că, conform fișei tehnice a senzorului, acesta se folosește și de pinii de RST și INT ai acestuia, astfel încât, în momentul în care senzorul are date noi pregătite, coboară pe 0 linia de INT, declanșând o întrerupere pe ESP32, în care citește datele obținute de la senzor.

ESP32-ul păstrează ultimele date de la senzori si la trimite la Raspberry Pi.

### 4.2.4 Protocoalele de comunicatie folosite intre Raspberry Pi si ESP32-uri

Comunicația între ESP32-ul responsabil cu controlul șasiului și Raspberry Pi se realizează prin protocolul UART. Pachetele se trimit inclusiv dintre Raspberry Pi spre ESP32, acesta reprezintă un stream de octeți, ce reprezintă o structură ce encapsulează un tuplu de forma $(v_x, v_y, \omega_z)$, iar calculele necesare pentru a controlul individual al fiecărui motor, conform formulei (1) ce

calculează vitezele de rotație ale fiecarui motor pentru a obține deplasarea după vectorii $v_x$ și $v_y$, compus cu o mișcare de rotație în jurul propriei axe cu viteza $\omega_z$.

Comunicația între ESP32-ul ce controlează șasiul și IMU și Raspberry Pi se realizează prin protocolul SPI. Aici comunicarea este bidirecțională, Raspberry Pi-ul trimițând comenzi despre controlul servomotoarelor și al liftului, iar ESP32-ul trimite informațiile generate de senzorul Sparkfun.

## 4.3 Server-ul extern

Sistemul extern are rolul de a extrage dintr-o filmare în timp real poziția robotului în sistemul de coordonate al terenului. Acesta are ca principale componente telefonul mobil, amplasat pe un suport la o înălțime de aproximativ 2 metri, și stația de procesare.

Telefonul rulează o aplicație ce pornește un server la care apoi stația de procesare se conectează și preia cadrele filmării în timp real, aplicația a fost descărcată din „Play Store” și se numește „Ip camera”.

Principiul de funcționare al stației de procesare este a lua noul cadru din filmare, realizarea proiecției omografică[22], estimarea poziționării robotului în spațiu și trimiterea acestei poziții spre Raspberry Pi, principiu descris și în figura 8.

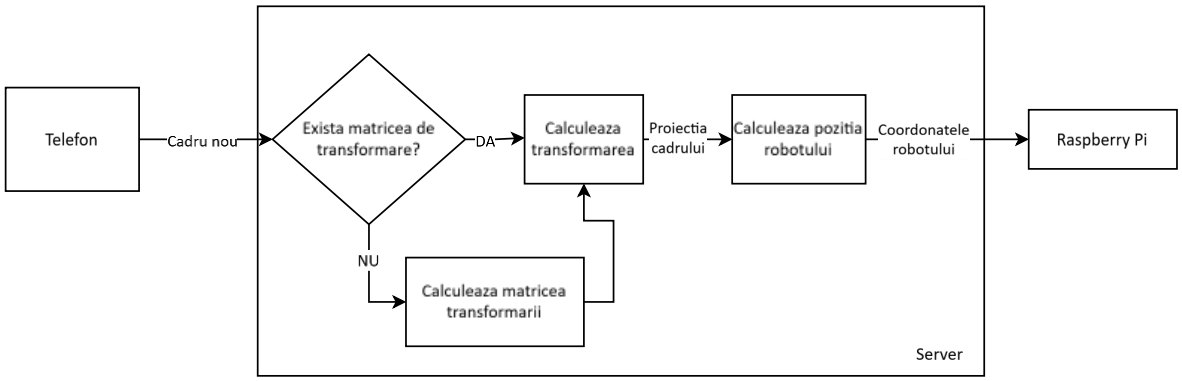



Figura 8 Principiul de funcționare al server-ului

### 4.3.1 Metoda folosita pentru procesarea filmarii

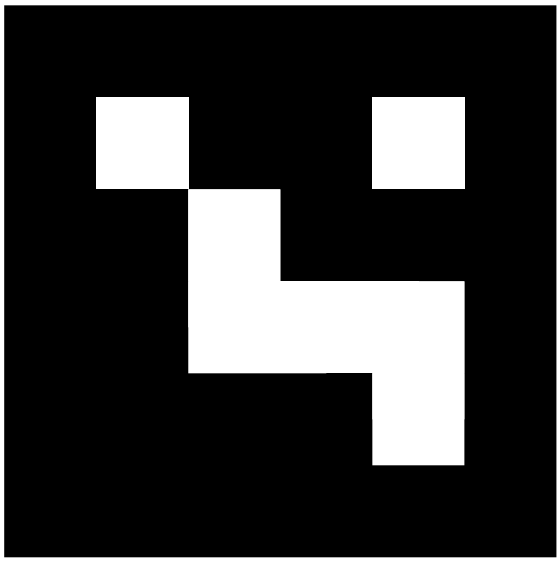
Întreaga procesare se realizează folosind modulul de OpenCV[23]. Acesta are capabilitățile de a recunoaște Markere ArUco[10] și a le poziționa în cadrului filmării.

Proiecția omografică[22] a terenului este realizată tot folosind funcțiile deja existente ale modulului OpenCV, astfel.

Pe teren, conform cu figura, se află poziționate 4 Markere ArUco[10], figura 8 prezintă un astfel de marker. Folosindu-le, OpenCV, este capabil de a genera o matrice de transformare corespunzătoare. Aceasta este reținută și în memorie și refolosită pentru fiecare cadru viitor.

Această soluție reduce zgomotul provocat de recunoașterile imperfecte ale markerelor la fiecare iterație, de asemenea, costul computațional este redus semnificativ, eliminând nevoia de recalculare a matricei de transformare pentru fiecare nou cadru ajuns la server.

De asemenea, deasupra robotului se află un alt Marker ArUco[10], cel prezentat în figura 9, iar, la fiecare iterație este identificată poziția robotului în sistemul de coordonate al filmării, și după este transformată în sistemul de coordonate al proiecției.



Figură 9 Marker-ul ArUco[10] amplasat pe robot

O dificultate o reprezintă faptul că marker-ul de pe robot se află deasupra robotului, la o înălțime de aproximativ 36 de centimetri, astfel, centrul său nu reprezintă, în realitate, centrul robotului. Pentru corecție s-a estimat poziția marker-ului în spațiul 3D, estimând astfel poziția reală a centrului robotului raportat la sistemul de coordonate. Problema este prezentată și vizual, în figura 10(b), ce prezintă proiecția terenului, comparat cu o captură ce nu a fost trecută prin transformare, prezentată în figura 10(a).

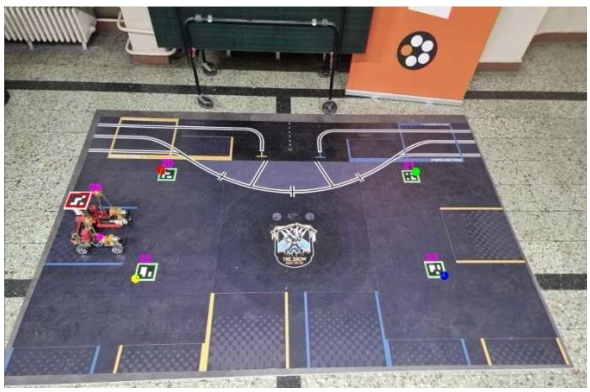


Figura 10(a) Imagine înaintea transformării

Figura 10(b) Imaginea rezultată

Figura 10 Rezultatul transformării olografe asupra unei imagini a terenului ce conține și robotul

### 4.3.2 Calibrarea și filtrarea datelor

Cum a fost menționat anterior OpenCV oferă posibilitatea identificării poziției în spațiu a unui marker ArUco[10], dar, pentru asta, are nevoie să cunoască parametrii intrinseci ai camerei: distanțele focale $(f_x, f_y)$, și centrele optice ale camerei $(c_x, c_y)$, măsurate în pixeli și reprezintă punctul pe care se proiectează centrul camerei în proiecția 2D, brută, realizată în timpul unei filmări sau poze. Pe lângă aceștia, există și parametrii de distorsiune ai camerei, radiali, practic aceștia creează un efect de butoi, ce creează iluzia că liniile drepte de pe margini să pară curbate, notați $k_1, k_2, k_3$ și parametrii tangențiali, ce distorsionează imaginea în cazul în care camera nu este perfect paralelă cu planul pozei, aceștia sunt notați cu $p_1, p_2$.

OpenCV, în documentația lor, oferă o metodă inteligentă de calibrare a acestor parametrii, folosind o poză ce conține pătrate negre și albe, similar cu o tablă de șah. În figura 11 este prezentată imaginea folosită pentru identificarea parametrilor camerei.

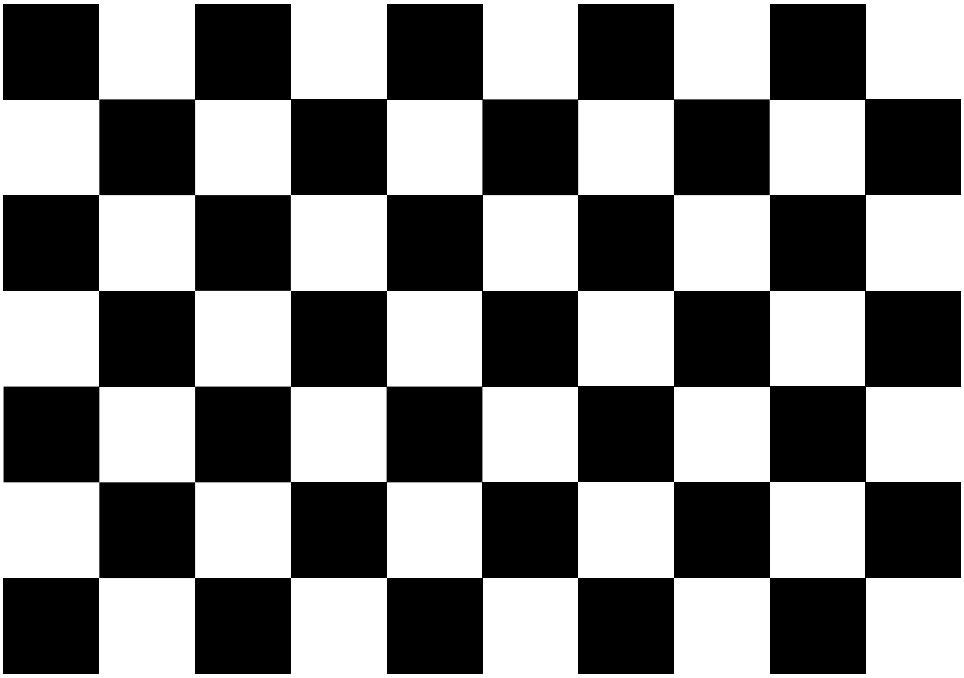


Figura 11 Imagine folosită pentru a calcula automat parametrii camerei

Realizând mai multe fotografii ale acestui model, din diferite unghiuri, poziționări și distanțe față de cameră, și folosind metode din librăria acestora, sunt calculate valorile menționate mai sus.

Totuși, în anumite situații, metoda de poziționare în spațiu oferită de OpenCV poate poziționa punctul fie în centrul actual al robotului în proiecție, dar, experimental, s-a observat că uneori punctul „sare” în jurul simetricului centrului real al robotului față de poziția efectivă a

markerului în filmare, figura 12 prezentând cele două cazuri, arătând deficiențele unei proiecții necalibrate. Astfel în figura 12(a) se poate observa o estimare, aflată în vecinătatea centrului robotului, iar în figura 12(b) se poate observa fenomenul descris mai sus.

Figura 12(a) Punctul apropiat de centru

Figura 12(b) Pozitia "sărită" a punctului

După calibrare totuși, în majoritatea cazurilor, punctul calculat ca fiind centrul real al robotului, se află în vecinătatea centrului efectiv al acestuia. Rezultatele amănunțite vor fi prezentate în capitolul următor.

Procesul de eliminare al acestor date eronate a fost făcut prin calcularea distanței de la ultima citire la noua citire, iar dacă aceasta depășea un anumit prag, era ignorată. Această soluție este posibilă mulțumită numărului mic de citiri eronate, rezultat obținut prin folosirea parametrilor obținuți în urma calibrării camerei.

Datele obținute de stația externă de calcul, sunt transmise către Raspberry Pi prin intermediul unei conexiuni UDP. Motivația din spate o reprezintă viteza crescută a protocolului, fiind mai rapid față de o conexiune de tipul TCP, și nu are mediatori ce ar putea încetini tranzacția, precum MQTT.

## 4.4 Implementarea filtrului Kalman extins

În capitolul anterior (3) a fost prezentat modelul matematic, teoretic al filtrului.

În implementarea propusă, filtrul primește poziționarea de la stația externă de calcul ce are rolul de a imita comportamentul unui GPS, iar accelerațiile și rotațiile din senzorul inerțial, aplicând calculele descrise mai sus pentru a obține unghiul de rotație de-a lungul axei z, din informațiile oferite de senzor.

Prima problemă practică întâmpinată a fost imposibilitatea poziționării senzorului inerțial în centrul de rotație al robotului, din mai multe considerente. În primul rând forma unică a robotului, unde interiorul acestuia este dedicat special pentru mecanismul de prindere. Era posibilă amplasarea totuși deasupra ansamblului liftului, dar această poziționare provoca două dezavantaje majore. Primul era reprezentat de înălțimea la care ar fi trebuit poziționat. Acest lucru reprezenta o problemă datorită vibrațiilor provocate de roțile mecanum, vibrații

ce se resimțeau mai puternic, ceea ce ar fi crescut zgomotul și reducea acuratețea măsurătorilor. Cea de-a doua problemă o reprezenta distanța față de ESP32. Senzorul comunicând prin $I^2C$, protocol ce este sensibil la distanțe lungi și ar fi putut cauza probleme în procesul de trimitere a datelor. Astfel a fost aleasă o poziționare pe unul din colțurile robotului, foarte aproape de ESP32.

Soluția găsită a fost de a folosi filtrul Kalman extins pentru a calcula poziția senzorului inerțial, translatând citirile făcute de stația externă din centrul robotului în colțul acestuia unde este poziționat senzorul, iar, apoi, valorile predicțiilor, translatate înapoi în centrul robotului.

O altă modificare o reprezintă prezența a două metode de actualizare, una pentru fiecare citire de la senzor, față de una singură ce avea loc doar în momentul în care veneau datele atât de la GPS, cât și de la IMU, astfel există două matrice R, $R_{loc}$, respectiv $R_{rot}$.

Figura 13 arată arhitectura folosită pentru filtrul Kalman extins folosit în implementarea soluției, diferențele față de modelul matematic de mai devreme fiind, așa cum a fost menționat mai sus, existența a două metode separate pentru actualizarea datelor și, așa cum evidențiază și diagrama, cele două procese de translație, primul aplicat asupra datelor de la sistemul extern de localizare, din centrul robotului la centrul IMU-ului, și, după, înapoi în centrul robotului.

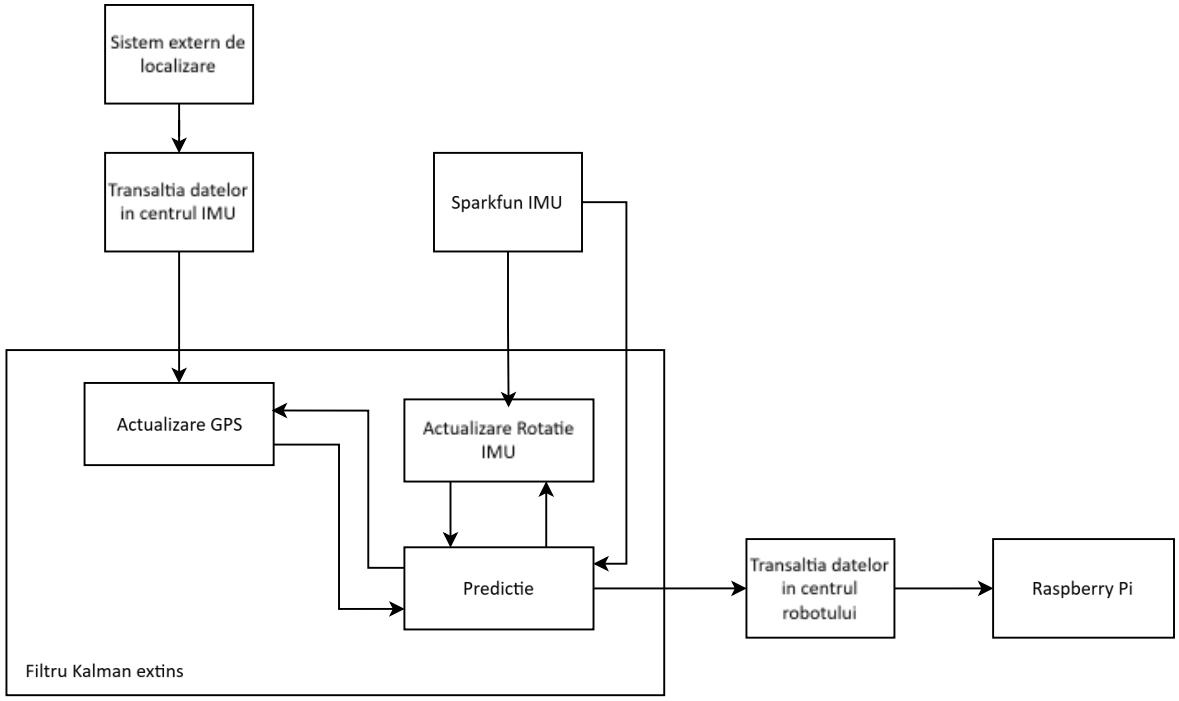



Figura 13 Arhitectura filtrului Kalman extins folosit in implementarea prezentata

### 4.4.1 Tunarea parametrilor R și Q

Spre deosebire de x și P, parametrii prezentați în capitolul trecut, ce sunt actualizați dinamic pe parcursul mișcării robotului, R și Q sunt constante. Acestea au un rol important în acuratețea filtrului și, de aceea trebuie găsite valori optime pentru aceștia.

Metoda folosită este inspirată din modul în care sunt tunați parametrii unei rețele neurale, folosind algoritmul de „backpropagation” [24] . Algoritmul constă în trei etape:

- „Forward”, unde este rulat sistemul cu o fidelitate cât mai mare raportat de evenimentele din realitate, și unde se obține calea prezisă
- Calcularea erorii, ce reprezintă compararea rezultatelor prezise cu cele ce sunt cunoscute ca fiind adevărate
- „Backward”, unde sunt actualizați parametrii în vederea minimizării costului.

Întâi, pentru a realiza un astfel de proces de tunare, avem nevoie de a recrea, cu o acuratețe cât mai bună, un sistem ce poate reproduce deciziile luate de filtrul Kalman, într-un mediu simulat. Astfel, pentru a obține datele, a fost creată o metodă de stocare eficientă a valorilor de intrare, la fiecare moment de timp, când apare o actualizare în sistem.

Datele sunt stocate sub forma unui fișier csv, ce reține datele sub următoarea formă, descrisă în tabelul 1.

| Timpul masurat | Eveniment | $Data_1$ | $Data_2$ | $Data_3$ |
|---|---|---|---|---|
| - | EKF | $x_r$ | $y_r$ | $\theta_{r,z}$ |
| - | GPS | $x_r$ | $y_r$ | $\theta_{r,z}$ |
| - | IMU | $a_{IMU,x}$ | $a_{IMU,y}$ | $\theta_{IMU,z}$ |

Tabel 1 Structura fișierului de date necesar reproducerii EKF

Unde $x_r$, $y_r$ reprezintă coordonatele centrului robotului prezise, pentru evenimentul de EKF, respectiv calculate din GPS. $\theta_{r,z}$, în mod similar, reprezintă rotația atât prezisă, cât și calculată din orientarea marker-ului ArUco[10]. $a_{IMU,x}$, $a_{IMU,y}$ reprezintă accelerațiile senzorului în sistemul de coordonate propriu, fiind aplicată transformarea de rotație înaintea înregistrării datelor. $\theta_{IMU,z}$ reprezintă unghiul de rotație a senzorului, extras din prelucrarea datelor de la cuaternioni și apoi înregistrat.

Motivul stocării datelor de la IMU, respectiv sistemul extern de localizare este evident, ele fiind esențiale în reproducerea deciziilor luate în timp real de către filtru. Totuși, și momentele de timp la care au fost făcute predicții sunt relevante, deoarece o predicție modifică vectorul de stare, vector folosit în cadrul funcțiilor de actualizare. Pe lângă acest aspect, prezicerile făcute de EKF pot fi folosite pentru a compara predicția inițială, unde au fost aleși parametrii doar intuitiv.

Acest fișier conține toate informațiile necesare recreării unei căi pe care a parcurs-o robotul.

Pentru antrenament au fost folosite 10 astfel de rulări, robotul parcurgând căi cât mai diferite, alcătuite din linii drepte pe diferite direcții, curbe ușoare, curbe abrupte, rotații în jurul propriei axe și combinații dintre acestea.

Pentru antrenamentul efectiv am folosit mediul de lucru oferit de Google Colab, oferind mai multă putere de calcul, față de ceea ce este disponibil local, iar pentru implementarea algoritmului au fost folosite metodele oferite de pytorch.

Pytorch reprezintă un modul de python ce introduce conceptul de tensor, ce reprezintă, de fapt, un vector multidimensional. Avantajul principal al acestui modul comparat cu diverse alternative ce oferă această capabilitate, precum numpy, este că, pytorch, analizează toate operațiile efectuate pe tensori și calculează automat gradienții, esențiali pentru algoritmul de antrenare.

Pentru pasul de optimizare al procesului de antrenament a fost folosită funcția Adam, ce este integrată în modulul pytorch.

# 5 EVALUAREA REZULTATELOR

Proiectul realizat este unul complex, alcătuit din mai multe subansamble, construcția și funcționalitatea corectă a robotului cu capabilități de mișcare holonomică, sistemul de localizare extern bazat pe viziune computerizată, prin folosirea de markere ArUco[10], și estimatorul poziției robotului, implementat cu ajutorul filtrului Kalman extins.

Ordinea prezentării rezultatelor nu este aceeași cu ordinea în care au fost descrise abordările și implementările, deoarece unele rezultate se bazează pe cele anterioare, nefiind posibilă altă metodă de evaluare.

## 5.1 Calitatea sistemului de localizare

Datele măsurate în acest sistem au fost făcute în milimetri, pentru ușurință și pentru evitarea erorilor de calcul, lungimea și lățimea terenului, în implementare, cât și în sistemul de colectare al datelor, fiind înregistrate în milimetri.

Pentru această evaluare, robotul a fost plasat în centrul markerului ArUco[10] aflat pe teren, cel mai apropiat de originea sistemului de coordonate al terenului, astfel încât centrul robotului se află la o distanță de 600 de milimetri de fiecare dintre laturile terenului, poziționându-l în realitate la coordonatele (600, 600) ale sistemului de referință.

Pentru comparație vor fi întâi prezentate datele obținute înaintea procesului de calibrare al camerei, mai apoi fiind prezentate cele rezultate după calibrare și prezentată o comparație între cele două situații, arătând efectul pozitiv, absolut necesar al calibrării.

### 5.1.1 Rezultate folosind camera necalibrată

Pozițiile percepute de sistemul de localizare cu datele necalibrate, pentru robotul poziționat conform descrierii de mai sus, sunt prezentate în figura 14.

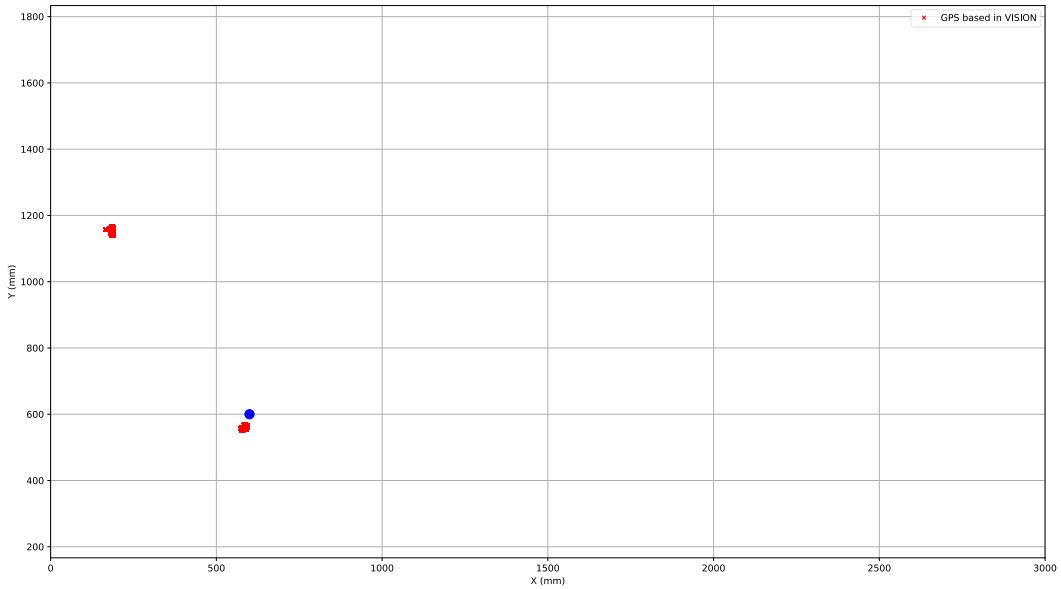



Figura 14 Datele extrase de la sistemul de localizare folosind camera fără informaţiile de calibrare

Conform graficului din figura 14, camera necalibrată generează două grupuri de puncte, grupul 1 ce se află în vecinătatea centrului robotului, marcat cu punctul albastru, și, altul, grupul 2, ce se află, la prima vedere, complet deplasat de realitate, dar după o evaluare mai atentă, acesta se află în vecinătatea simetricului punctului reprezentat de media punctelor din grupul 1, ceea ce indică o eroare că sistemul de localizare, folosind datele de la cameră, nu poate calcula corect rotația robotului în plan, încurcând între 90 de grade și -90 de grade.

Pentru a putea extrage informații exacte din datele colectate, a fost folosit algoritmul de grupare („clustering”), KMeans, pentru a putea separa cele două grupe de puncte..

În urma unei măsurători, au fost extrase următoarele rezultate, prezentate în tabelul 2.

| Numar grup | Numar de puncte | Media pe axa X | Media pe axa Y |
|---|---|---|---|
| 1 | 15434 | 583.44 | 186.67 |
| 2 | 7155 | 558.19 | 1150.00 |

Tabel 2 Datele extrase in urma măsurătorii poziției robotului aflat la coordonatele (600, 600) folosind camera necalibrată

Observăm că numărul punctelor din grupul 2 reprezintă aproximativ 31.67% din numărul total de puncte înregistrat, astfel, pur și simplu, încercând o filtrare, un număr prea mare de puncte este pierdut, iar, chiar și considerând doar punctele ce se află în apropierea centrului robotului, distanța medie între acestea și centrul efectiv al robotului este de aproximativ 42.3 mm.

### 5.1.2 Rezultate in urma calibrării

A fost efectuată calibrarea camerei, conform descrierii procedurii din capitolul cinci, rezultatele fiind descrise în figura 15.

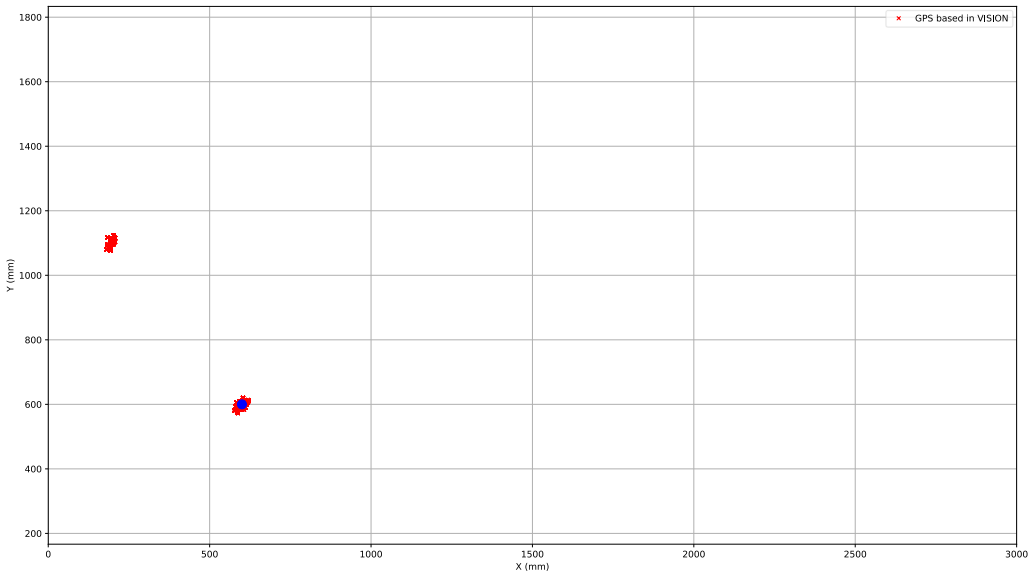



Figura 15 Grafic ce prezintă datele extrase in urma calibrării camerei

După cum se poate observa și cu ochiul liber, analizând graficul din figura 15, acuratețea localizării a crescut semnificativ, în urma procesului calibrării.

Analizând rezultatele numerice oferite colectate în urma acestei testări, rezultate prezentate în tabelul 3.

| Numar grup | Numar de puncte | Media pe axa X | Media pe axa Y |
|---|---|---|---|
| 1 | 23732 | 602.69 | 210.00 |
| 2 | 1156 | 597.76 | 1086.67 |

Tabel 3 Datele extrase in urma măsurătorii poziției robotului aflat la coordonatele (600, 600) folosind camera calibrată

Astfel, procentul de citiri complet eronate este de doar 4.7% din numărul total de citiri, valoare suficient de mică pentru a putea implementa un filtru, fără a pierde o cantitate semnificativă de date, iar eroarea medie între citirile corecte și centrul efectiv al robotului este de aproximativ 3.5mm.

Conform rezultatelor anterioare, numărul de citiri eronate, în urma calibrării, a scăzut de aproximativ 6.73 ori, reducând astfel ponderea acestora de la 31.67% la doar 4.7%, iar acuratețea citirilor a crescut, eroarea medie scăzând de aproximativ 12 ori, reducând-o de la 42.3mm la doar 3.5mm.

## 5.2 Analizarea alunecării sașiului de tip mecanum

Am observat în secțiunea anterioară, faptul că, în urma calibrării camerei, erorile de citire ale poziției sunt suficient de mici, astfel, putem folosi sistemul de localizare extern, pentru a calcula distanțele alunecărilor produse de roțile mecanum.

### 5.2.1 Mișcare liniară în direcția orientării robotului

Pentru a estima alunecarea ce are loc în timpul unei mișcări ideal rectilinii, în direcția orientării robotului, acesta este plasat într-un punct fix, în cazul acesta (600, 600) și este dată comanda de deplasare înainte, de-a lungul axei x, iar, după un timp, acesta se va opri. Măsurând diferența pe axa x, ne putem da seama de alunecarea ce are loc. În figura 16 este prezentat graficul citirilor sistemului de localizare de-a lungul acestei deplasări.

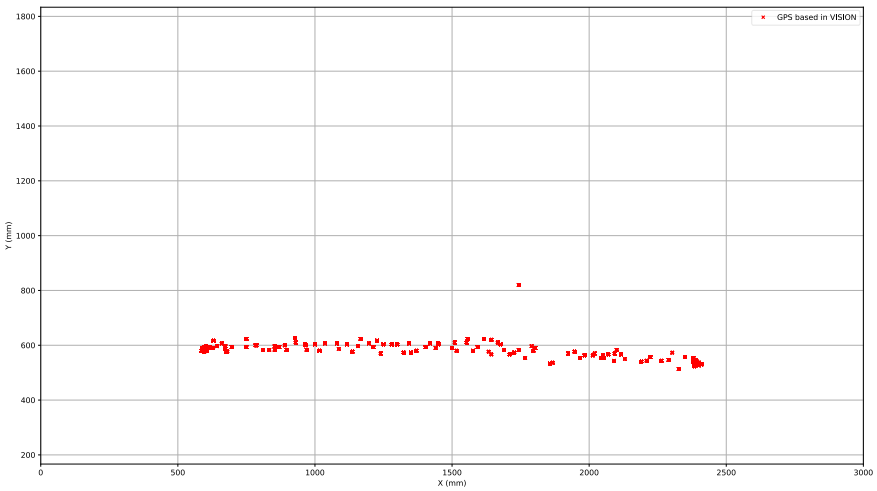



Figura 16 Citirile realizate de sistemul de localizare, folosite pentru a calcula alunecarea roţilor mecanum pe o mişcare de mers înainte.

Astfel, efectuând măsurătorile, coordonatele punctului în care ajunge sunt următoarele: 2390.19mm pe coordonata x și 536.74mm pe coordonata y. Obținând o alunecare de aproximativ 63 mm pentru o deplasare de 1790 mm, obținând o eroare a alunecării pe direcția perpendiculară de aproximativ 3% din lungimea drumului parcurs.

### 5.2.2 Mișcarea de deplasare liniară perpendiculară pe direcția dată de orientarea robotului

În mod similar, robotul este plasat la un punct de coordonate fix (600,600) și va fi deplasat perpendicular, folosind capabilitățile roților mecanum. Pentru o astfel de deplasare, mișcarea va avea loc de-a lungul axei y, iar alunecarea va fi realizată măsurând distanța parcursă de-a lungul axei x.

În figura 17 sunt prezentate coordonatele robotului de-a lungul deplasării.

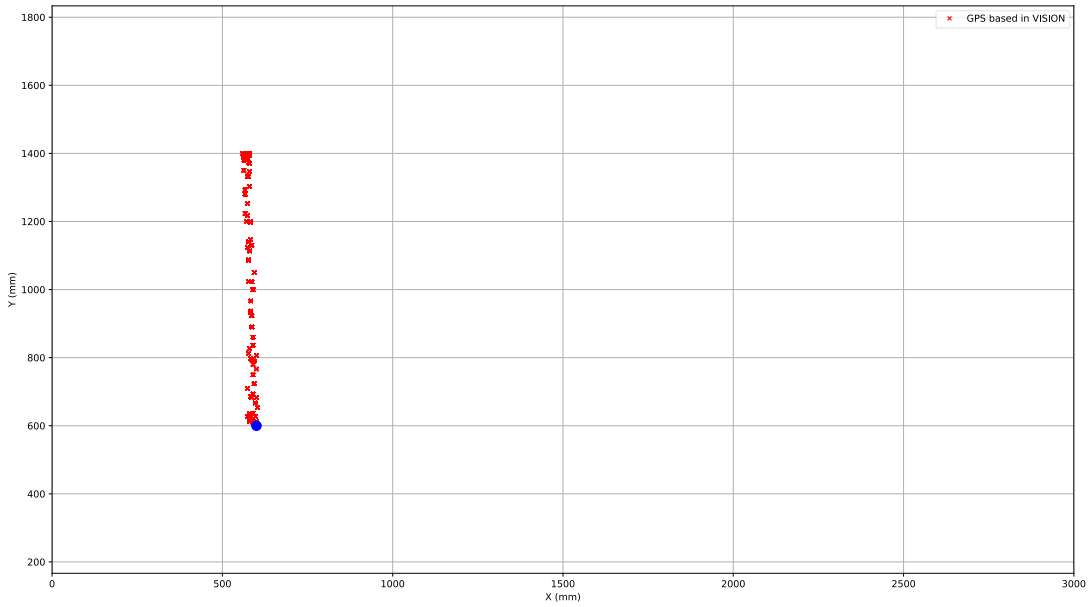



Figura 17 Traiectoria obţinută prin deplasare laterală, perpendiculară pe orientarea robotului

În urma măsurătorilor, robotul s-a deplasat din punctul de coordonate (600, 600) în punctul (573.79, 1391.27), măsurătoare obținută prin măsurarea mai multor puncte din momentul în care robotul a devenit staționar. Astfel pentru o deplasare de 791 de mm, robotul a alunecat pe axa x aproximativ 26 de mm, obținând o eroare a alunecării de aproximativ 3% din distanța parcursă pe direcția dorită.

## 5.3 Rezultatele obținute folosind filtrul Kalman extins

În continuare vor fi prezentate graficele realizate pe baza înregistrărilor estimărilor realizate de filtru și analiza momentelor în care acesta se abate de la traiectoria marcată de sistemul extern de localizare, pentru mai multe trasee..

### 5.3.1 Linie dreaptă, pe direcția de deplasare a robotului

În figura 18 sunt prezentate, prin puncte roșii, informațiile oferite de sistemul de localizare, iar prin linia punctată gri, predicția filtrului Kalman extins.

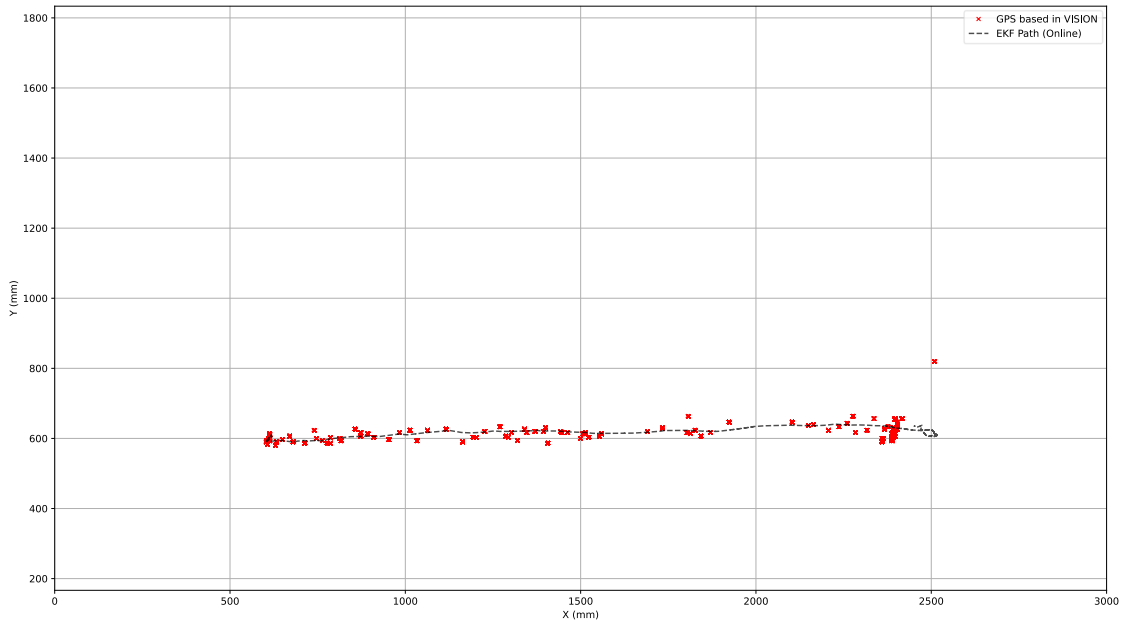



Figura 18 Predicţia efectuată de filtrul Kalman extins pe direcţia de deplasare paralelă cu orientarea robotului

Pe parcursul deplasării, filtrul a reușit să prezică corect pozițiile robotului, din nou, din cauza roților mecanum, și a lipsei unui compensator care să regleze viteza roților în timpul deplasării, această linie nu va putea fi dreaptă, nu pentru că estimatorul nu reușește să afle valorile corecte, acesta reușește, dar traiectoria robotului nu este perfect dreaptă.

O deficiență pe care o putem observa este aceea că, predicția filtrului nu s-a oprit în momentul în care s-a oprit robotul, deși a avut loc o corecție, filtrul estimând ulterior poziția finală a robotului, ultima estimare a filtrului fiind în punctul (2451.42, 635,10), iar ultima valoare a sistemului de localizare fiind (2420.33, 613.12), eroarea fiind de aproximativ 40 de mm.

### 5.3.2 Linie dreaptă pe direcția perpendiculară orientării robotului

În acest caz rezultatele ar trebui să fie similare celor din secțiunea anterioară, rezultate prezentate în graficul din figura 19.

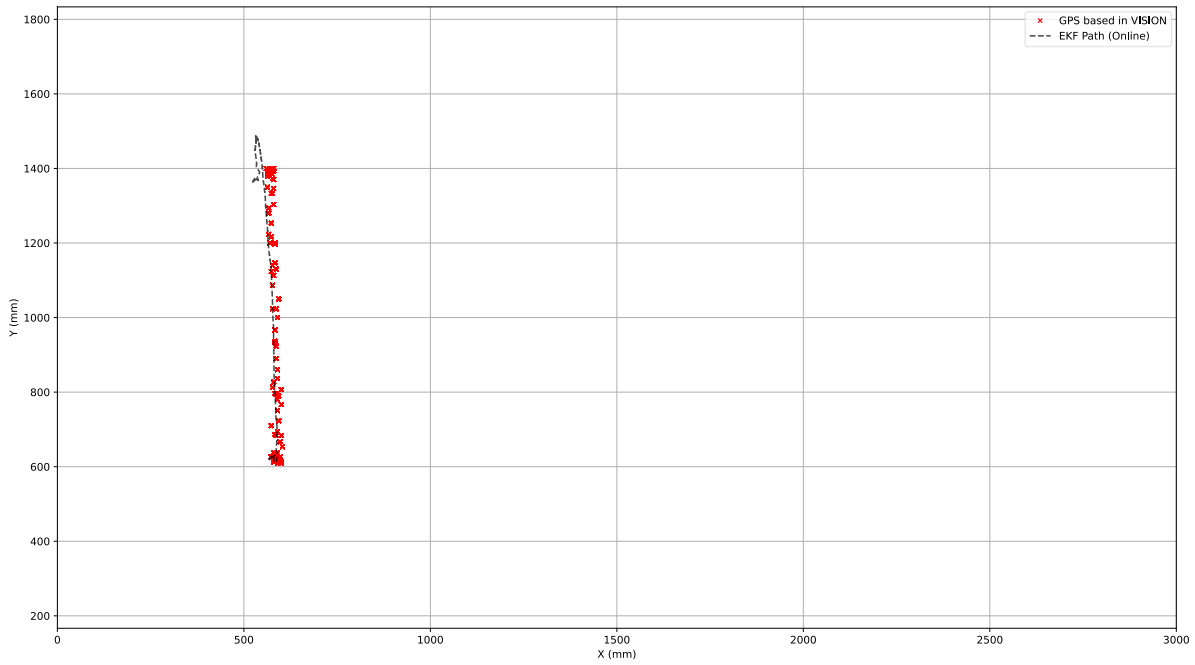



Figura 19 Predictia filtrului Kalman extins pentru deplasare pe directia perpendiculara cu orientarea robotului

Exact cum a fost menționat, rezultatele sunt similare, eroarea fiind similară, în cazul acesta aproximativ 43 de mm.

Au fost analizate doar erorile de la final, deoarece, pe parcursul mișcării nu cunoaștem traiectoria perfectă a robotului și, oricum, din grafic, acesta urmărește bine traiectoria dată de sistemul de localizare, ce este precis, dar produce poziționări noi la o frecvență redusă.

Un alt factor ce produce acele erori ale filtrului la terminarea mișcării o reprezintă decelerarea bruscă a robotului, ce vine în contradicție cu legile matematice ale acestui sistem. Practic, face sistemul mai greu de liniarizat, provocând dificultăți filtrului Kalman extins de a prezice cu acuratețe poziția.

### 5.3.3 O cale mai complexă ce include curbe, dar și schimbări de direcție

Prin acest test, dorim să ducem la maxim capabilitățile filtrului, observând performanța acestuia pe o traiectorie mai complexă. Astfel putem observa și cum se redresează în urma unei schimbări bruște de direcție.

Va fi prezentată o analiză a comportamentului filtrului pentru fiecare tip de schimbare de direcție, fie prin curbă, fie, direct, prin mișcare laterală.

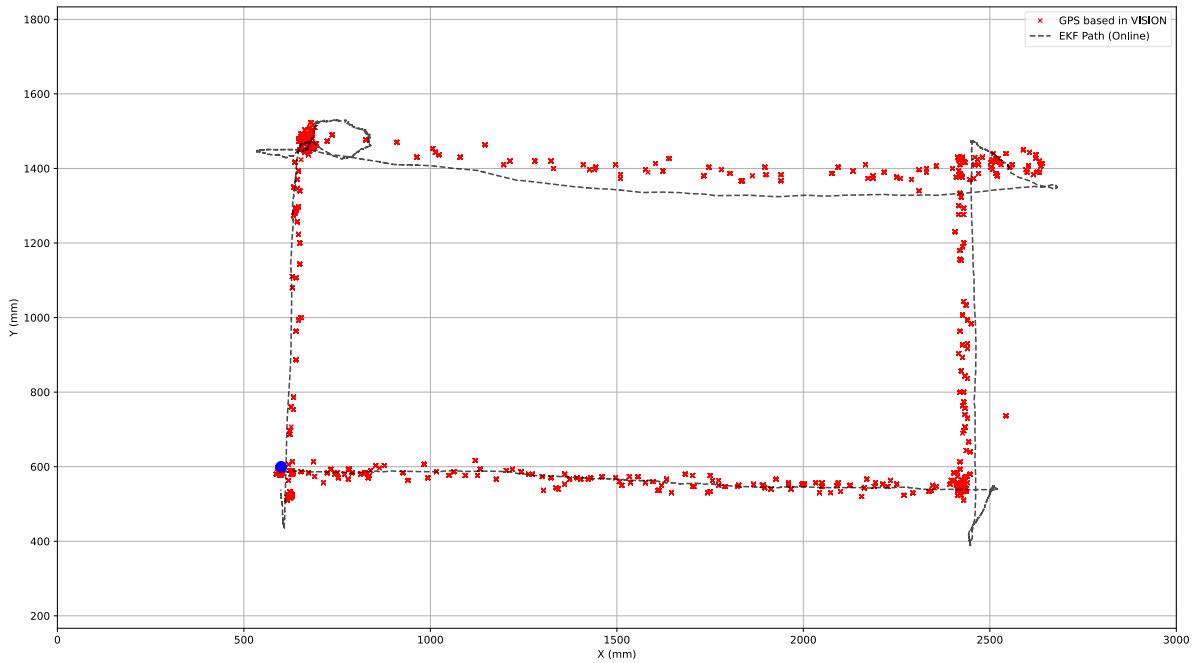



Figura 20 Rezultatele filtrului Kalman extins, pe o cale complexa ce combina atat rotatii, cat si deplasari perpendiculare pe directia de mers.

Conform figurii 20, robotul a pornit din punctul de coordonate (600, 600), în colțul din dreapta jos a efectuat o rotație în jurul propriei axe, apoi s-a deplasat drept, urmat de o deplasare pe direcția axei y, fără a se roti, apoi a efectuat câteva rotații în jurul propriei axe, rotație evidențiată în stânga sus, ca mai apoi să se întoarcă aproximativ în punctul de plecare.

Astfel se pot observa deficiențele filtrului în momentele schimbării de direcție, motivate în secțiunea anterioară, totuși, aspectul important de urmărit în cadrul acestui test a fost că, deși filtrul nu este perfect în colțurile chenarului, acesta reușește să se redreseze, fără a propaga eroare în sistem, a se vedea că ultima linie, cea unde robotul se întoarce spre start, nu prezintă eroare substanțială.

# 6 CONCLUZII

Obiectivul lucrării a fost de a construi un robot complet funcțional, bazat pe mișcare holonomică, capabil de a interacționa cu mediul înconjurător, cât și un sistem robust de localizare a acestuia în cadrul unui teren predefinit, esențial pentru competiții de robotică internaționale.

Pe parcursul lucrării au fost prezentate concepte fundamentale din domeniul roboticii, de la tipuri de șasiuri ce permit mișcare holonomică, la senzorii utilizați în mod uzual pentru a poziționa un astfel de robot în spațiu, algoritmi avansați de fuzionare ale acestor date, până la algoritmi de viziune computerizată și cum pot fi ei utilizați în domeniul roboticii.

Au fost abordate întâi fundamentele matematice ce stau la baza implementării propuse, modele de nivel înalt ale arhitecturii sistemului robotului, apoi fiind prezentată implementarea detaliată, până la nivelul circuitelor electrice.

Rezultatele sunt promițătoare, deși încă nu perfecte. Sistemul în continuare suferă de mici erori, existând loc de îmbunătățiri.

Ca obiective pentru viitor reprezintă încercarea mai multor algoritmi de fuzionare de senzori, precum UKF sau filtrul de particule, cât și îmbunătățirea platformei mecanice a robotului și includerea de circuite integrate.

# 7 BIBLIOGRAFIE


[1] Eurobot, "Official Rules 2024" [Online]

https://www.eurobot.org/wp-content/uploads/2024/10/Eurobot_General_Rules_EN.pdf

[2] Kalman, R. E. (March 1, 1960). "A New Approach to Linear Filtering and Prediction Problems." ASME. *J. Basic Eng*. March 1960; 82(1): 35–45

[3] Stanley F. Schmidt (1966) "Application of State-Space Methods to Navigation Problems" *Advances in Control Systems*, Vol. 3, pp. 293-340. (Editat de C. T. Leondes)

[4] Simon J. Julier și Jeffrey K. Uhlmann (1977) "A new extension of the Kalman filter to nonlinear systems" *Proceedings of SPIE AeroSense: Signal Processing, Sensor Fusion, and Target Recognition VI*, Vol. 3068.

[5] N.J. Gordon, D.J. Salmond, și A.F.M. Smith (1993) "Novel approach to nonlinear/non-Gaussian Bayesian state estimation" *IEE Proceedings F - Radar and Signal Processing*, Vol. 140, No. 2, pp. 107-113

[6] Sebastian Thrun, Wolfram Burgard, și Dieter Fox (2005) "Probabilistic Robotics" MIT Press

[7] Bengt Erland Ilon (1972) Patentul pentru rotile mecanum [Online] https://patents.google.com/patent/US3876255A/en

[8] Taheri, Hamid & Qiao, Bing & Ghaeminezhad, Nourallah. (2015). Kinematic Model of a Four Mecanum Wheeled Mobile Robot. International Journal of Computer Applications. 113. 6-9. 10.5120/19804-1586.

[9] NASA Mission Planning and Analysis Division (July 1977). "Euler Angles, Quaternions, and Transformation Matrices". NASA.

[10] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Medina-Carnicer, R. (2014). "Automatic generation and detection of highly reliable fiducial markers under occlusion". *Pattern Recognition*, 47(6), 2280-2292.

[11] Olson, E. (2011). "AprilTag: A robust and flexible visual fiducial system". In *2011 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3400-3407). IEEE.

[12] Taheri, H., Ghassemi, A., & Ghaffari, T. (2015). "A review on omnidirectional wheeled mobile robots". *International Journal of Advanced Robotic Systems*, 12(3), 29

[13] Soe, Ma & Htay, Soe & Phyu, Khaing & Win, Wut. (2024). Kinematics and Control A Three-wheeled Mobile Robot with Omni-directional Wheels.

[14] Raspberry Pi 4 datasheet [Online] https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

[15] Espressif ESP32 WROOM S3 [Online]

https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf

[16] Texas Instruments LM2596s [Online] https://www.ti.com/lit/ds/symlink/lm2596.pdf

[17] XLSEMI XL4015 [Online] https://www.xlsemi.com/datasheet/XL4015-EN.pdf

[18] NXP 33886 H-Bridge [Online] https://www.nxp.com/docs/en/data-sheet/MC33886.pdf

[19] Texas Instruments DRV880x [Online] https://www.ti.com/lit/ds/symlink/drv8801.pdf

[20] Pololu 37D Metal Gearmotor [Online] https://www.pololu.com/file/0J1736/pololu-37d-metal-gearmotors-rev-1-2.pdf

[21] Ceva BNO06X [Online]

https://docs.sparkfun.com/SparkFun_VR_IMU_Breakout_BNO086_QWIIC/assets/component_documentation/BNO080_085-Datasheet_v1.16.pdf

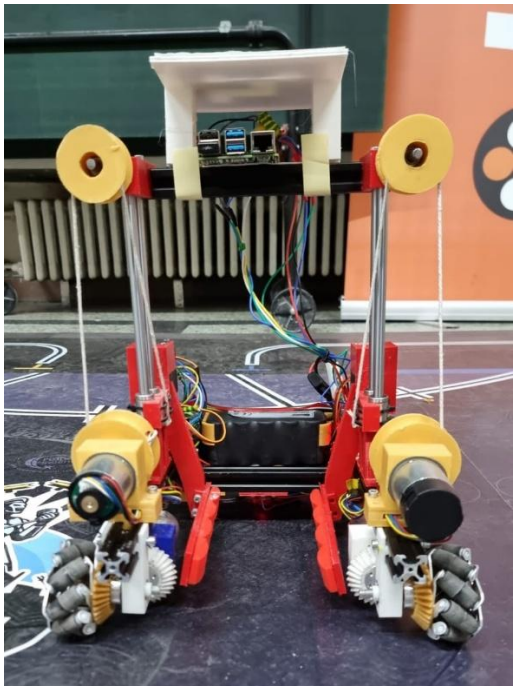
[22] DeTone, Daniel & Malisiewicz, Tomasz & Rabinovich, Andrew. (2016). Deep Image Homography Estimation. 10.48550/arXiv.1606.03798.

[23] OpenCV documentație oficiala [Online]

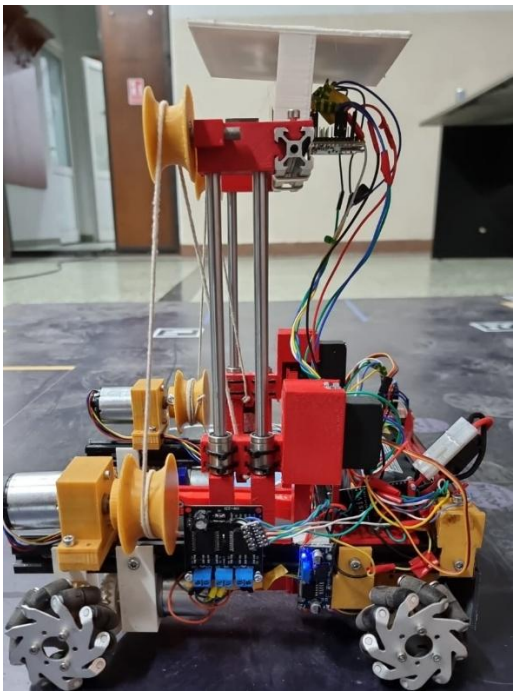
https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

[24] Rumelhart, David E., Geoffrey E. Hinton and Ronald J. Williams. “Learning representations by back-propagating errors.” *Nature* 323 (1986): 533-536.

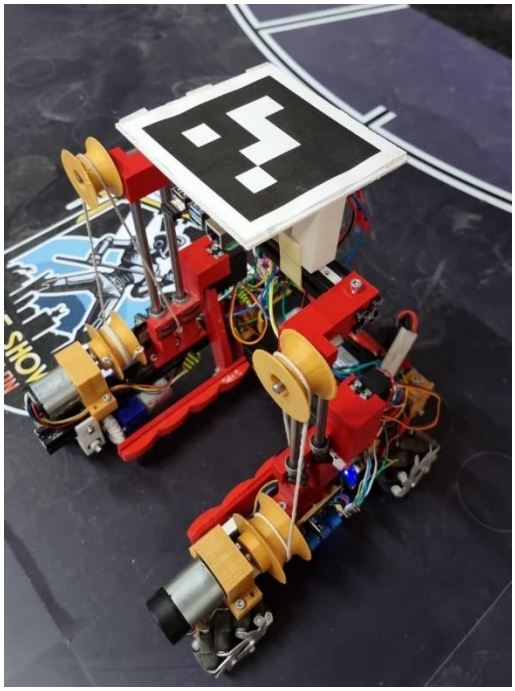
# 8 ANEXE



Anexa 1. (a) Vedere din față a robotului

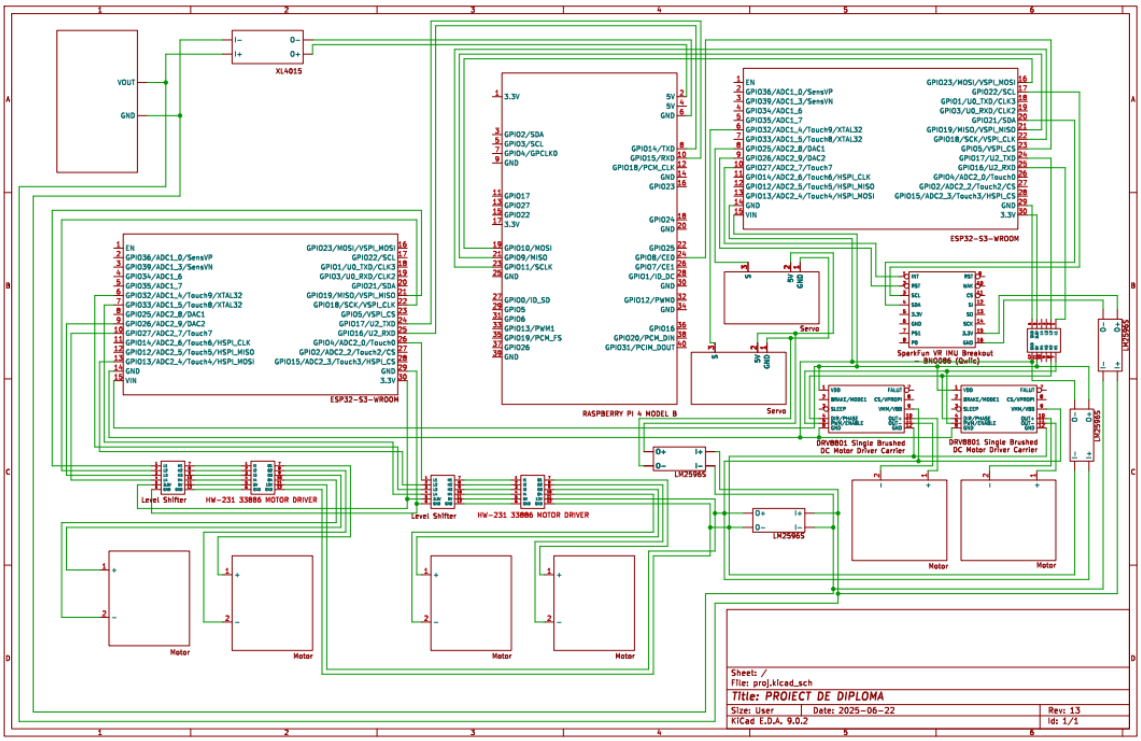


Anexa 1. (b) vedere laterală a robotului



Anexa 1. (c) vederea din perspectivă


Anexa 1. (d) vederea de sus a robotului




Anexa 2. Schema electrică