UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU
FACULTATEA DE INGINERIE
DEPARTAMENTUL DE CALCULATOARE ȘI INGINERIE ELECTRICĂ

PROIECT DE DIPLOMĂ

Conducător științific: Prof. dr. Ing. Florea Adrian

Absolvent: Corlaciu Andrei-Razvan

Specializare: Calculatoare

UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU
FACULTATEA DE INGINERIE
DEPARTAMENTUL DE CALCULATOARE ȘI INGINERIE ELECTRICĂ

MODELAREA ȘI SIMULAREA UNUI SISTEM AUTOMATIZAT ÎN CADRUL UNUI DEPOZIT AGRICOL

Conducător științific: Prof. dr. Ing. Florea Adrian

Absolvent: Corlaciu Andrei-Răzvan

Specializare: Calculatoare

CUPRINS

1.Introducere

1.1 Scopul lucrării

Scopul acestei lucrări este de automatiza un depozit agricol. Ansamblul este format dintr-un mBot care are atașat in laterala lui o cameră video conectată la modulul corespunzător al plăcuței Raspberry Pi 4.

1.2 Obiective

Obiectivele principale ale lucrării sunt:

- Crearea modelului care să rezolve problema
- Utilizarea tool-ului Bee-Up pentru realizarea scenariului dorit
- Scrierea codului pentru a executa anumite operații specifice (ex: script scanare QR)

1.3 Descrierea problemei

Problema pe care o rezolvă această temă este reprezentată de scenariul în care un robot se plimbă pe un traseu prestabilit în depozit oprindu-se în dreptul fiecărui raft și scanând QR code-ul aferent raftului, datele generate scanării QR code-ului reprezentând informații despre fructele/legumele aflate pe raftul respectiv (id, tip, cantitate, preț, țara de proveniență). Informațiile vor fi transmise într-o bază de date, urmând apoi a fi exportate într-o aplicație vizuală realizată în C#.

În ceea ce privește dezvoltarea mBotului, se va folosi tool-ul Bee-Up, creat cu ajutorul platformei ADOxx. Prin acest tool se transmit date sub formă de cod Adoscript către platformele web aferente dispozitivului, iar datele sunt transmise mai departe prin intermediul unei plăcuțe Arduino.

1.4 Necesitatea

Având in vedere dezvoltarea tehnologiei in ultimii ani precum inteligența artificială, revoluția Internet of Things (IoT) care au rol în crearea înterprinderii digitale care sunt capabile să ia decizii singure consider că este necesară automatizarea unui depozit agricol prin preluarea și stocarea automată de informații a fructelor și a legumelor din cadrul depozitului.

Se urmărește îmbunătățirea productivității atât în cadrul fabricilor industriale, cât și în depozitele agricole, de construcții etc, deoarece în viitor vor exista tot mai mulți roboți inteligenți care pot interacționa cu oamenii și le pot ușura munca.

Această automatizare o consider necesară deoarece scutește munca fizică a omului de a se plimba în tot depozitul pentru a prelua manual informațiile despre anumite alimente.

1.5 Motivația

Motivul principal pentru care am ales această temă este dorința de a combina Hardware-ul cu Software-ul, pentru a-mi pune în evidență atât cunoștințele de hardware cât și cele legate de software dobândite in decursul celor 4 ani de facultate.

În anul 2021 am participat la școala de vară NEMO Summer School [1][2][3][6][7] organizată online unde am fost inițiat în bazele modelării.

2. Noțiuni Teoretice

2.1 OMiLAB – Introducere

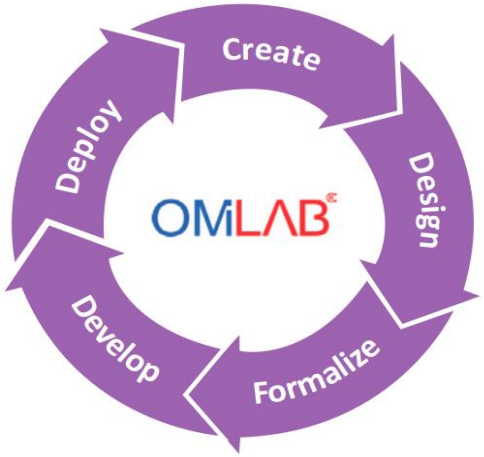


Fig. 1 OMiLAB

OMiLAB vine de la **Open Models Initiative Laboratory** – Laborator de inițiativă pentru modele deschise și a fost creată pentru a fi aplicată pe obiecte fizice, cum ar fi roboți. OMiLAB este un ecosistem digital care ajută la conceptualizarea, dezvoltarea și implementarea metodelor de modelare; este un spațiu dedicat pentru cercetare și experimentare în ceea ce privește metodele de modelare. Datorită faptului că este un loc atât fizic, cât și virtual, acesta este echipat cu instrumente pentru a explora crearea și proiectarea metodelor, și instrumente software pentru implementarea metodei de modelare. Ideea care stă la baza acestui laborator este de a facilita dezvoltarea și aplicarea metodelor științifice pentru orice comunitate care valorifică modelele și metodele de modelare.

În scop demonstrativ, a fost elaborată o dovadă academică a conceptului de interes didactic și de experimantare în cadrul OMiLAB pe platforma de meta-modelare ADOxx. Această platformă se folosește în etapele de proiectare, formalizare și dezvoltare. De asemenea, această platformă oferă posibilitatea de a folosi mai multe tool-uri de tip open source ca de exemplu: Scene2Model, Bee-Up care vor fi prezentate ulterior.

2.2 ADOxx

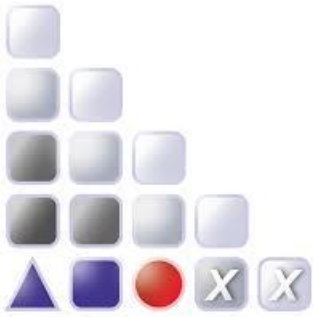


Fig. 2 ADOxx

ADOxx este o platformă de meta-modelare, de dezvoltare și configurare pentru implementarea metodelor de modelare. Platforma ADOxx a fost implementată pentru a putea extinde experimentele individuale care necesită noi instrumente de modelare.

ADOxx vine cu un meta-model predefinit care conține un set de clase abstracte. Acest set de clase abstracte oferă o structură de bază pentru dezvoltarea noilor meta-modele și oferă funcționalități care pot fi moștenite pentru a le folosi în setul de instrumente creat pentru modelare.

Figura 3 reprezintă structura meta-modelului predefinit dinamic, care asigură un concept pentru limbajele de modelare ce descriu procesele. Dreptunghiurile albe reprezintă clasele abstracte, iar cele gri reprezintă clasele de relație. Meta-modelul static prezintă conceptele pentru limbajele de modelare care descriu structuri.

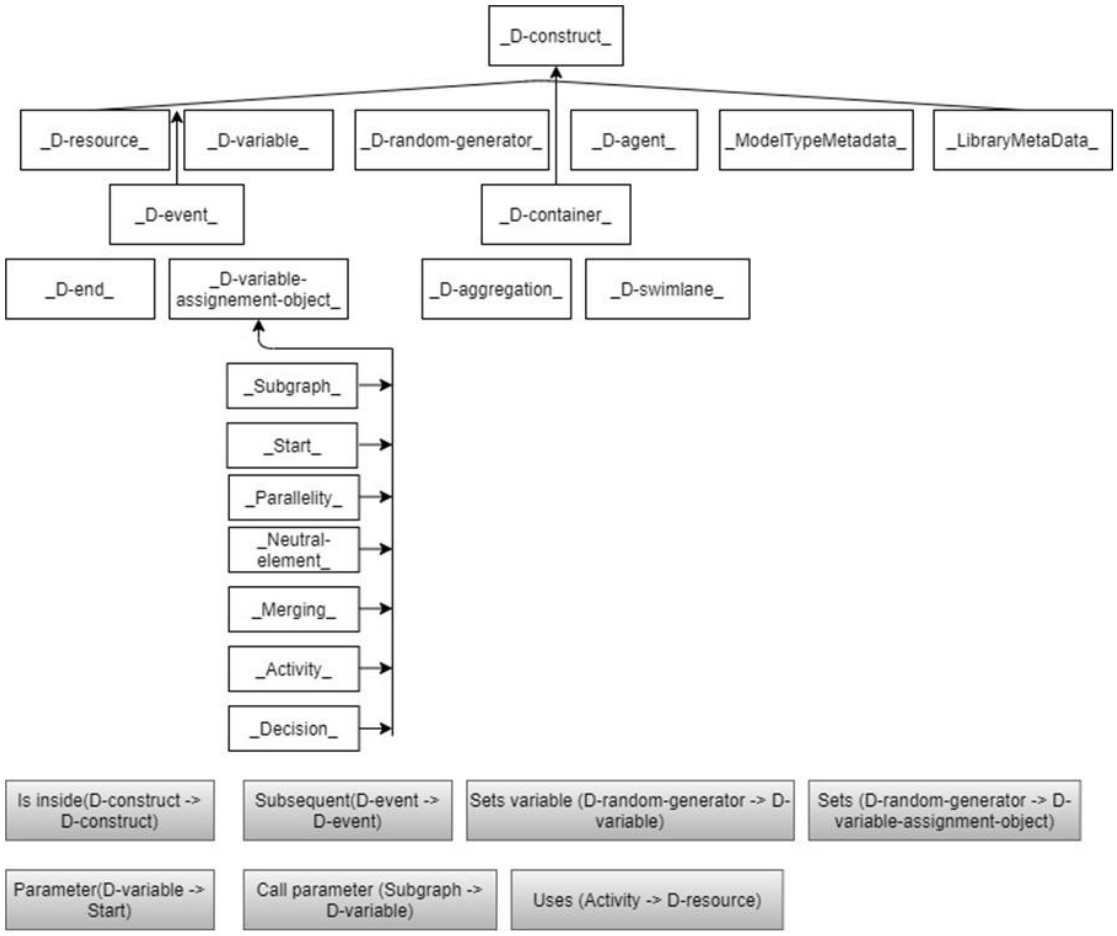


Fig. 3 ADOxx Meta-model

Platforma ADOxx a fost dezvoltată folosind limbajul de programare C++. Pentru dezvoltarea unei metode specific de meta-modelare, unele funcționalități pot fi moștenite din meta-modelul ADOxx, deci modelul va devein o instanță a meta-modelului specific metodei și este descries în limbajul de dezvoltare ADOxx.

Pentru dezvoltarea unui meta-model specific, ADOxx propune două metodalități de dezvoltare și anume abordarea de configurare și abordarea de implementare. Abordarea implementării este pur și simplu scrierea codului în limbajul bibliotecii ADOxx (ALL) folosind unul dintre instrumentele de dezvoltare bazate pe text.

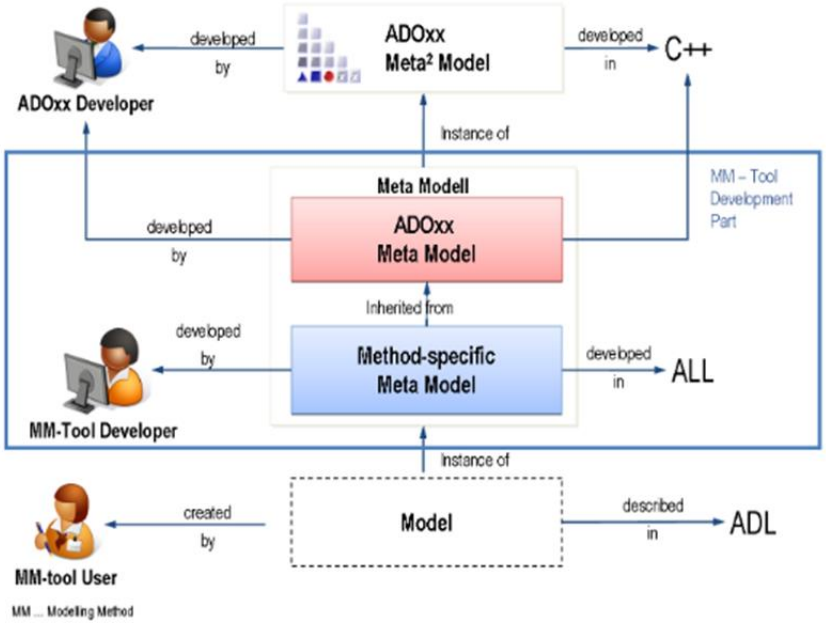


Fig. 4 Ierarhia platformei de meta-modelare in ADOxx

Pentru abordarea configurării este folosit mediul de dezvoltare ADOxx, care oferă un UI (user interface) pentru a susține configurarea unui meta-model. Următoarea figură ne oferă o imagine a celor două abordări de dezvoltare. După ce codul este scris (adică deținem deja ALL) acesta poate fi convertit în ABL, care reprezintă librăria executabilă, care este de fapt importată in ADOxx pentru a configura programul de modelare.

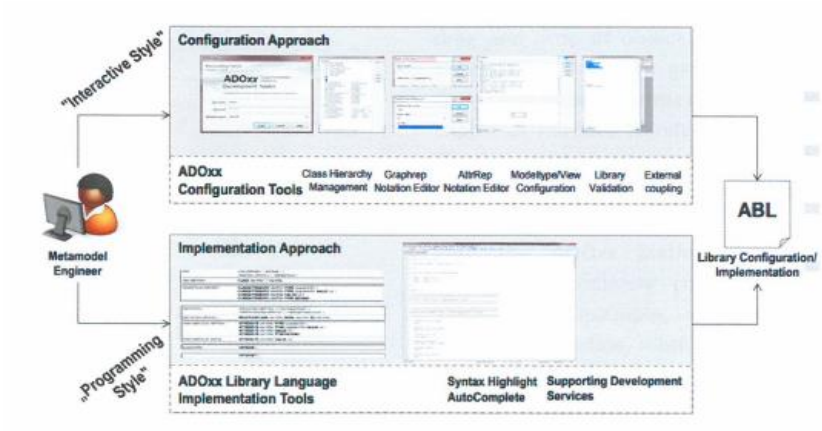


Fig. 5 Abordarea dezvoltării în ADOxx

Pentru ambele abordări, rezultatul final este un library application file (ABL), care conține toate informațiile meta-modelului. El conține definiția conceptelor de bază pentru crearea unui meta-model folosind ADOxx. Conceptele principale la definirea unui meta-model în ADOxx sunt tipurile de model, atributele, clasele si relațiile. Un tip de model este o sub-colecție de clase și clase de relații bine definită, a unui meta-model. O **clasă** de relație este o construcție care este folosită ca model pentru a crea obiecte din clasa respectivă. Toate obiectele unei clase se numesc instanțe ale acelei clase. Un **atribut** este o proprietate a unei construcții de modelare, de exemplu un model, un obiect sau o relație. O relație este definită între două clase. O **relație** este întotdeauna o conexiune directă intre obiecte. Figura 6 reprezintă relațiile conceptelor.

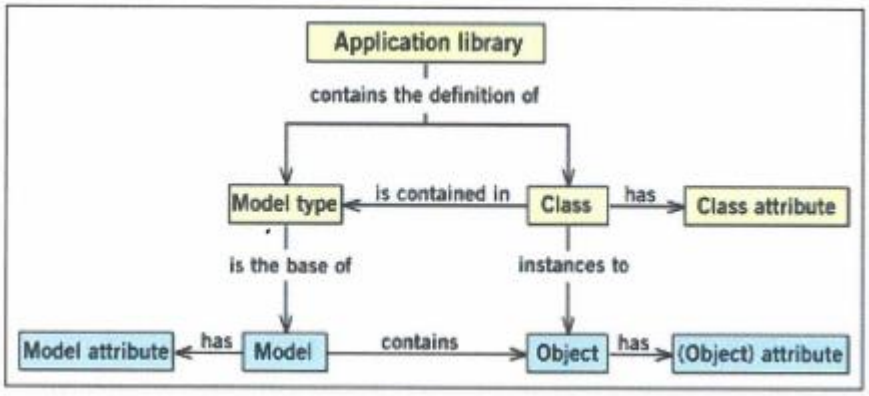


Fig. 6 Principalele concepte în dezvoltarea limbajelor de modelare

ADOxx vine cu două tool-uri diferite. Unul este **ADOxx Development Toolkit**, care este în principal folosit pentru dezvoltarea propriu-zisă de meta-modele, și pentru administrarea userilor, librăriilor care conțin meta-modelele, model managementului, managementul fișierelor etc. Al doilea tool din componența ADOxx este **ADOxx Modelling Toolkit** care servește la testarea mediilor de dezvoltare.

Folosind ADOxx nu este permisă doar crearea unui limbaj de modelare, dar permite adăugarea de mecanisme și algoritmi pentru a putea construi un instrument de modelare complet. Figura 7 reprezintă modul in care ADOxx susține acești pași.

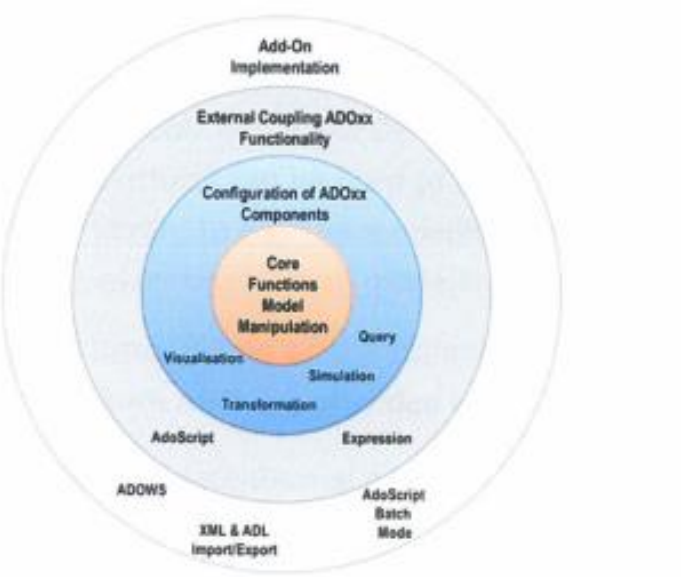


Fig. 7 ADOxx Onion

Miezul acestui model se referă la **functionalități** care vin cu ADOxx. Acesta se referă la functionalități precum drag and drop de obiecte în zona de modelare, generarea de fișiere grafice, vizualizarea cu grid, reprezentarea sub forma unui tabel, importarea si exportarea modelelor etc.

Cel de-al doilea inel al „cepei" denumit „**configurarea componentelor ADOxx**" reprezintă functionalitățile care vin cu ADOxx, dar pot fi schimbate sau reconfigurate in funcție de preferințele modelatorului. De exemplu, cu ADOxx este posibilă definirea interogărilor specifice metodei utilizând opțiunea „interogări de analiză predefinite" și sintaxa limbajului de interogare ADOxx Query Language.

Al treilea inel din reprezentare, numit "**External Coupling Functionality**" se referă la funcționalitățile care nu sunt integrate în platforma ADOxx, dar care pot fi foarte ușor adăugate folosind o platformă dependentă de limbaje și expresii de script. Această platformă se numește AdoScript și poate fi văzută ca un macro limbaj al platformei ADOxx. Acesta permite adăugarea de funcționalități unei metode de modelare într-un mod ușor și convenabil.

Cel mai exterior inel al "cepei" se referă simplu la addon-uri. Serviciile web de exemplu, pot fi folosite în ambele direcții. Pe de o parte, este posibil să utilizăm un serviciu web extern care

să adune datele necesare, dar pe de altă parte este de asemenea posibil să apelăm la ADOxx Web Service și să utilizăm funcționalități ale platformei din exterior. În plus, putem integra sau utiliza în moduri diferite functionalități sau algoritmi dezvoltați în diferite limbaje de programare. O modalitate de integrare, precum addon-urile, este de a dezvolta o interfață folosind AdoScript. Un alt mod ar putea fi reprezentat de utilizarea functionalității de import/export XML pentru a genera un fișier XML, care poate fi procesat ulterior. [2]

2.3 Bee-Up

Fig. 8 Bee-Up

Bee-up este unul dintre tool-urile create pe baza ADOxx, folosit pentru a reprezenta lumea reală (stratul 0). Cu ajutorul acestuia sunt realizate modelele (stratul 1), și pot fi simulate și analizate rezultatele obținute în urma simulării.

Bee-Up este o implementare a unei metode de modelare hibridă care încorporează și extinde mai multe limbaje de modelare și anume:

- **B**PMN – Business Process Model and Notation: BPMN este de cele mai multe ori cea mai bună notație pentru procesele de afaceri. Oferă mai multă rigoare și perspectivă decât diagrame simple. Este mai inteligibilă decât diagramele de activitate UML și este mai potrivit pentru analiza și proiectarea proceselor.
- **E**PC – Event Driven Proces Chains: EPC este un tip de diagramă pentru modelarea proceselor de afaceri. EPC poate fi utilizat pentru a configura execuția planificării resurselor întreprinderii și pentru îmbunătățirea procesului de afaceri. Poate fi utilizat pentru a controla o instanță de flux de lucru autonomă în schimbul de lucru.
- **E**R – Entity-Relationship: În ingineria software, un model ER este format în mod obișnuit pentru a reprezenta lucrurile pe care o afacere trebuie să le amintească pentru a realiza procesele de afaceri. În consecință, modelul ER devine un model de date abstracte, care definește o structură de date sau informații care poate fi implementată într-o bază de date, de obicei o bază de date relațională.
- **U**ML – Unified Modelling Language: UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase și instanțele acestora. Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru business Process Design etc.
- **P**etri Nets: Rețele Petri este un instrument grafic și matematic utilizat în multe domenii științifice diferite. Caracteristicile lor sunt limbajul intuitiv de modelare grafică și metoda de analiză formală avansată. Întregul spectru de aplicații Petri Nets este format din aplicații clasice precum sisteme informatice și de control, prin diagnosticare de erori, fabricație, sisteme de alimentare, sisteme de trafic, transport și aplicații Web.

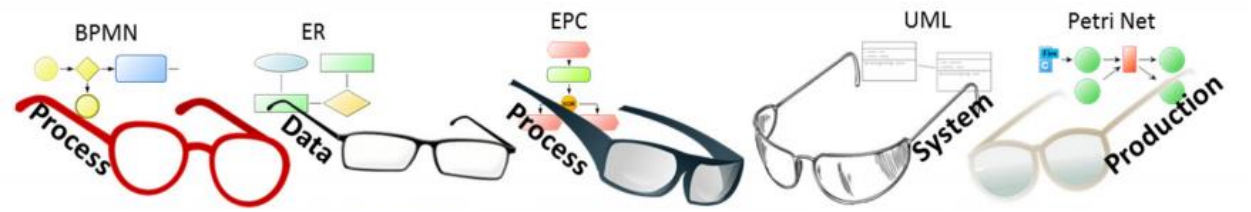


Fig. 9 Limbajele de modelare folosite în Bee-Up

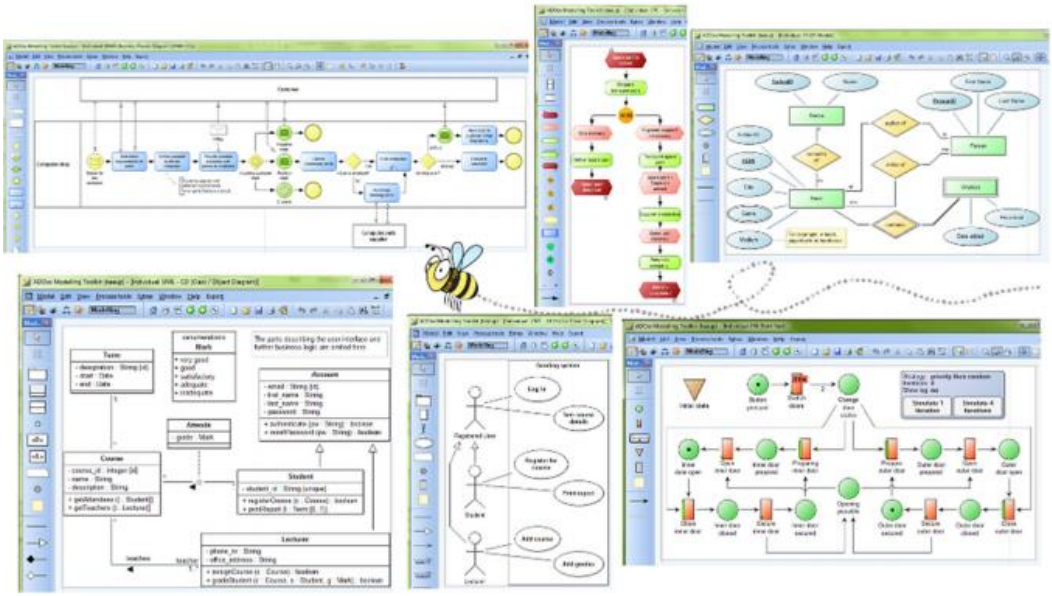


Fig. 10 Exemple de modele realizate în Bee-Up

Împreună cu implementarea limbajelor de modelare, Bee-Up folosește suplimentar unele funcționalități de procesare furnizate de platforma ADOxx. De asemenea, extinde capabilitatea procesării cu mecanisme și algoritmi suplimentari disponibili prin intermediul modelatorului, programat ca o extensie peste ADOxx. Câteva exemple pentru acestea sunt:

- O simulare uniformă a modelelor de proces(BPMN, UML). Simularea poate produce diferite tipuri de rezultate și permite diferite abordări pentru configurarea sa (de exemplu decizii bazate pe probabilități).
- Analiza rețelelor Petri prin intermediul executării manuale sau automate a tranzițiilor. Tranzițiile care sunt ready pot fi executate individual sau în grupuri printr-o simulare automată.
- Generarea de instrucțiuni SQL – create dintr-un model ER și testate cu MySQL.
- Exportul de modele în diferite formate (XML, ADL) și generarea de grafice care prezintă modelul (JPG). Unele formate pot fi utilizate pentru a prelucra în continuare conținutul modelului în afara instrumentului Bee-Up.

Toate aceste funcționalități permit instrumentului Bee-Up să fie folosit într-o gamă largă de domenii: de la utilizarea ca tool academic până la descrierea cerințelor sistemului IT.

Aria de aplicabilitate se poate extinde prin integrarea limbajelor de modelare implementate cu alte tipuri de modele și concepte comune.

2.4 Scene2Model

Fig. 11 Scene2Model

Scene2Model este un tool special creat pentru “Design Thinking” care trece de limitele abordărilor tradiționale de design thinking. Pentru vizualizare, se recomandă mijloace tangibile pentru crearea modelelor. Aceste mijloace (de ex. figurine din hârtie) sunt intuitive și flexibile pentru a favoriza creativitatea. Așadar, ca și rezultat, acest lucru duce către un obiect de design care are o semantică foarte mică incorporată și nu conțin informații specifice domeniului. Dar aceste informații de domeniu și contextul semantic sunt importante pentru implementarea și dezvoltarea unui nou model de produs / serviciu / afacere într-un context organizațional. Pentru rezolvarea acestei probleme, s-a creat tool-ul Scene2Model, care suportă modelarea diagramelor în combinație cu ontologii pentru captarea și îmbunătățirea rezultatea metodelor de design thinking. Figura 12 reprezintă modelul de funcționare:

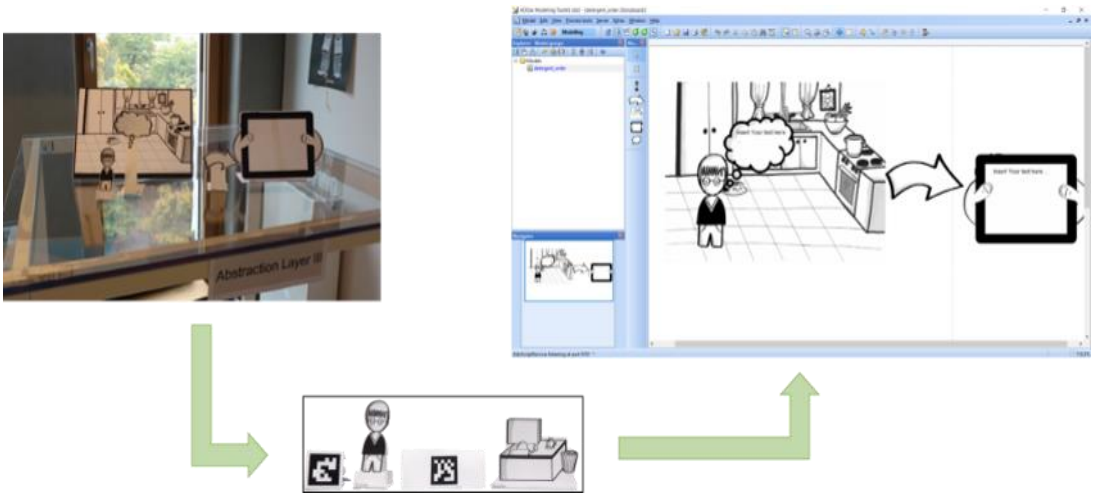


Fig. 12 Model de funcționare Scene2Model

2.5 Mediul Smart Innovation OMiLAB

Odată cu creșterea popularității conceptului de “Digital Twin” și al IoT, a apărut necesitatea unor abilități de digitalizare noi. Termenul de “Digital Engineer” a fost introdus încă de la sfârșitul anilor 1990 în scop didactic, însă acesta trebuie actualizat pentru tehnologia din zilele noastre deoarece interoperabilitatea și integrarea evoluează foarte repede în scopuri și posiblități. Una dintre provocările care apar odată cu evoluția este construirea unui mediu adecvat pentru dezvoltare și consolidare a abilităților de inginer digital. Practic, OMiLAB ne oferă un mediu proiectat și testat care are ca scop principal procesele de conceptualizare. Termenul de SMART INNOVATION [6] se referă la resursele hardware/software și configurarea spațiului de lucru experimental care ajută fie în scopuri didactice fie transformarea unui laborator în laborator de cercetare OMiLAB. Mediul propus incorporează spațiul de lucru pentru „Digital Design Thinking” și “Digital Twin-Based engineering” susținute de platforme de integrare și beneficierea de interoperabilite utilizând metode de modelare conceptuală pentru a acoperi decalajele semantice dintre “Design Thinking” și dovezile experimentelor fezabile cu ajutorul “Digital Twins”. Problemele de proiectare ale modelării conceptuale, au inspirat această idee spre scopul de a ridica valoarea modelării conceptuale, de la cazuri de utilizare tradițională (proiectarea software-ului, managementul proceselor de afaceri) la o propunere de valoare eralizată, unde limbajele de modelare sunt „scheme de cunoaștere” care pot fi adaptate oricărui domeniu. Această idee împreună cu mediile de modelare interoperabile, conduc la o formă de “Digital Twin engineering”

care este condus semantic în centru și intens din punct de vedere al calculului la margine. În continuare se va prezenta o listă de cerințe ale mediului pentru care această idee a fost proiectată:

- Cerința de agilitate a metodei de modelare. Este nevoie ca metodele de modelare concepuală să poată fi adaptate pentru a capta mai multe straturi de abstractizare, specificitate și granularitate- de la nivel înalt până la constrângeri de execuție și proprietăți.
- Cerința de tehnologie. Este nevoie de "prototyping enablers" – atât pentru metodele de modelare agile prezentate anterior cât și pentru omologii cyber-physical ale "Digital Twins".
- Cerința de deschidere. Sunt preferate platformele cu interfață în locul tehnologiei pentru a putea facilita reutilizarea blocurilor folosite.
- Cerința de integrare digitală. Mediile de modelare trebuie să ofere mediere semantică ușor de citit de la cel mai înalt nivel până la nivelul cyber-physical.
- Cerința privind ecosistemul cunoașterii. Co-crearea necesită cunoștiințe comune active pentru a sprijini transferul de cunoștiințe.

Ținând cont de cerințele prezentate mai sus, mediul propus pentru inovație a fost proiectat ca un laborator modular care poate fi ușor proiectat și rulat pe o rețea existentă și necesită spațiu fizic limitat în orice cameră de laborator obișnuită.

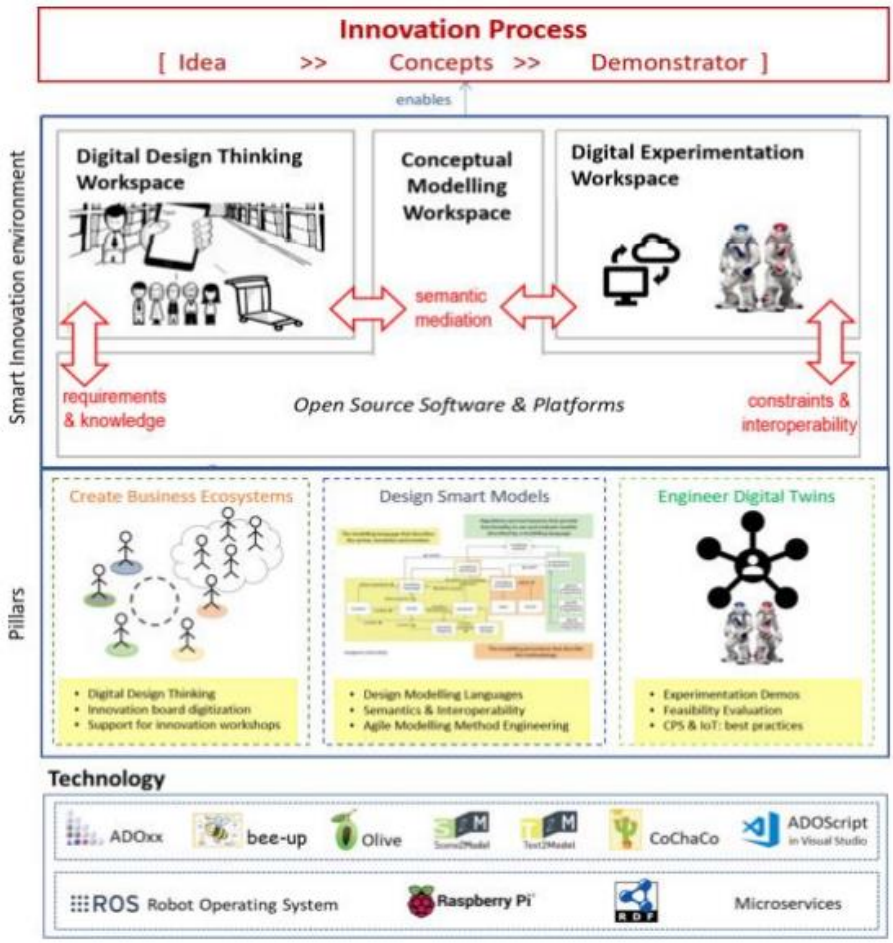


Fig. 13 Tehnologii Smart Innovation Enviroment

Așa cum se poate observa în figura de mai sus, acest mediu este construit pe trei baze, fiecare instanțiată într-un spațiu de lucru operational și reflectând una din cele trei categorii de abilitați:

- Crearea ecosistemului – instanțiat în spațiul de “Digital Design Thinking” cu un tool care suportă digitalizarea inovațiilor care apar de obicei în timpul atelierelor de inovare.
- “Smart modelling” – instanțiat în spațiul modelării conceptuale având baza în tool-ul BEE-UP (prezentat în paginile anterioare) care integrează și extinde anumite limbaje de modelare cum ar fi: UML, BPMN, EPC, ER, Petri Nets, DMN, flowcharts. În același timp, pentru a susține “smart model-ul”, acesta oferă interoperabilitatea modelului prin cereri de tip HTTP sau conversii model-grafic semantic de tip RDF. Mai mult, ADOxx-ul ne permite să extindem instrumentele de modelare pentru un domeniu sau motiv specific.

- “Digital Twin Engineering” – instanțiat în spațiul de experimentare digitală care permite dezvoltarea nivelului de cyber-physic (folosind Raspberry-Pi), sau servicii virtuale (de ex. simulări de rapoarte).

2.6 Firebase

Fig. 14 Firebase Logo

Firebase este o platformă Google care faciliteaza crearea, gestionarea și scalarea proiectelor pentru dezvoltatori. Aceasta le permite dezvoltatorilor să creeze aplicații mai rapid si mai sigur. Pe partea de Firebase, nu este necesară nicio programare, ceea ce face simplu să profitați de caracteristicile sale. Firebase funcționează Android, iOS, Web si Unity și oferă acces la stocarea în cloud. Baza de date este NoSQL (stochează datele într-un mod diferit față de tabelele relaționale). Bazele de date NoSQL sunt clasificate în funcție de modelul lor de date.

Principalele beneficii pe care le are Firebase:

- **Google Analytics**
  - Cu Google Analytics puteți crea rapoarte personalizate. Firebase, conform Google, oferă rapoarte gratuite nelimitate pentru până la 500 de evenimente diferite. Google Analytics pentru Firebase, la fel ca Analytics obișnuit, urmărește evenimentele cheie și parametrii utilizatorului imediat din cutie și permite specificarea de evenimente noi care sunt esențiale pentru aplicații. Firebase lucrează cu o serie de parteneri de analiză precum: Google Ads, Admob, TradeMob etc.
- **Realtime Database & Firestore**
  - Firebase a fost dezvoltat inițial pentru a fi o bază de date în timp real, iar astăzi rămâne una dintre caracteristicile importante ale acesteia. Baza de date în timp real este în esență o bază de date NoSQL găzduită în cloud care stochează date ca JSON în timp real. Unul dintre avantajele pentru o bază

de date în timp real este că funcționează offline folosind memoria cache locală pe dispozitiv pentru a stoca orice modificări făcute. Când conexiunea aplicației este reluată, datele sunt sincronizate.

- **Autentificarea**
  - Autentificarea este o caracteristică pe care majoritatea aplicațiilor o includ pentru a identifica utilizatorul. Întregul proces de autentificare poate fi configurat în doar câteva minute datorită SDK-ului simplu Firebase, modulelor UI gata de utilizat și serviciilor backend. Utilizatorii se pot conecta la Firebase folosind numărul de telefon, e-mailul, conturile Google sau Facebook (există mai multe posibilități).
- **Utilizarea gratuită a link-urilor dinamice**
  - Firebase Dynamic Links sunt adrese URL inteligente care permit agenților de marketing să facă publicitate aplicației printr-o varietate de canale externe, inclusiv social media, e-mail, web și multe altele. Faptul că legăturile dinamice pot fi folosite atât de utilizatorii de aplicații, cât și de persoanele care nu au aplicația instalată este unul dintre cele mai bune beneficii ale acestora.

2.7 Python language

Fig. 15 Python Logo

Python este un limbaj de programare de nivel înalt, orientat pe obiecte, cu semantică dinamică [8][9]. Structurile sale de date încorporate la nivel înalt, împreună cu tastarea dinamică îl fac ideal pentru dezvoltarea rapidă a aplicațiilor și ca limbaj de scripting. Sintaxa concisă și ușor de învățat a Python-ului promovează lizibilitatea, ceea ce reduce costurile de întreținere a software-

ului. Python acceptă module și pachete, ceea ce încurajează modularitatea programului și reutilizarea codului.

Principalele beneficii pe care le are Python sunt:

- Extensabilitatea
  - Python este extrem de extensibil, ceea ce înseamnă că se poate extinde în alte limbaje precum: C#, C++, Java și alte limbaje.
- Librăriile limbajului
  - Acest limbaj deține o cantitate mare de biblioteci pe care le poți include in proiectul tău. O bibliotecă este practic o bucată de cod reutilizabil care poate fi folosită pentru funcții comune precum procesarea imaginilor, efectuarea de ecuații, administrarea bazei de date etc.
- Comunitatea Python
  - Unul dintre cele mai importante beneficii ale limbajului este comunitatea sa vastă de dezvoltatori și ingineri software. Aceștia contribuie în mod regulat la resursele de învățare precum tutoriale, manuale, cărți, discuții pe forum și conținut video.

2.8 Limbajul de programare C#

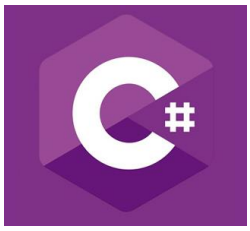


Fig. 16 Logo C#

C# este un limbaj de programare de nivel înalt orientat pe obiecte care încearcă să combine capacitatea de calcul a C++ cu ușurința de programare a Visual Basic. C# se bazează pe C++ și conține caracteristici similare cu cele ale Java. Acest limbaj este proiectat să funcționeze cu platforma .NET a Microsoft.

Limbajul este ușor de învățat și de folosit atât pentru cei începători cât și pentru cei avansați și experimentați. Practic este o variantă îmbunătățită a limbajului C++. În plus, C# oferă o varietate de biblioteci și tipuri de date.

Totodată, limbajul este unul modern din simplu fapt că are o gestionare a memoriei automată, tratarea excepțiilor care apar pot fi tratate în mai multe limbi. Acest limbaj acceptă calculele monetare și totodata un model robust.

Cele 3 domenii principale în care se folosește C#-ul sunt:

- Aplicații vizuale de tip Windows Form
- Domeniul WEB
- Realizarea jocurilor

Avantajele limbajului C# :

- Este pur orientat pe obiect, față de C++ care este o combinație de orientat pe obiect și orientat pe procedură
- Programatorul nu trebuie să se concentreze mult pe pierderea memoriei
- Conceptul de asamblare rezolvă bine problema controlului versiunilor
- Este ușor de dezvoltat, având o bibliotecă bogată de clase care face multe funcții ușor de implementat

2.9 Partea Hardware

2.9.1 mBot

Acest robot face parte dintr-o clasă de roboți mici și mobili. Posibilitatea de a recunoaște modificări pe sol și obstacole este inclusă în versiunea standard, în plus, este posibilă extinderea funcționalităților acestui robot prin adăugarea de echipament adițional. Acest robot face parte din grupul roboților mobili și online.

Acest mBot este o mașină mică, cu 2 roți mari mai in spate și încă o roată mică sub el, în față. Versiunea standard a robotului are un senzor ultrasonic si un senzor de tipul line follower.

Primul senzor măsoară distanța până la un anumit obstacol în fața acesteia, iar al doilea senzor este capabil să detecteze dacă mașina este deasupra unei zone întunecate sau a unei zone iluminate.

Sunt disponibile 2 tipuri de surse de alimentare pentru acest mBot. Putem folosi 4 baterii tip AA sau putem folosi o baterie de tip LiPo.

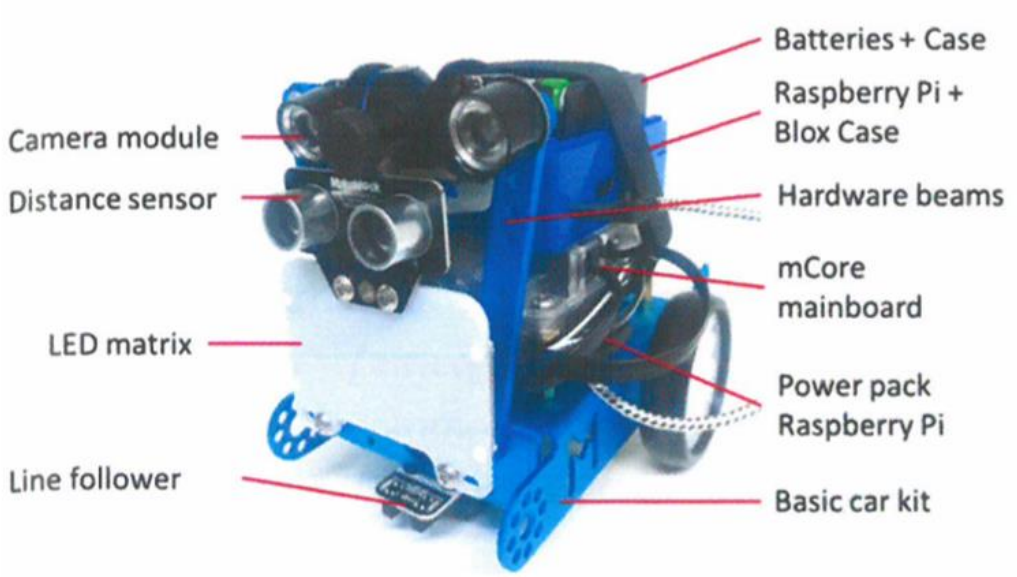


Fig. 17 mBot cu hardware adițional

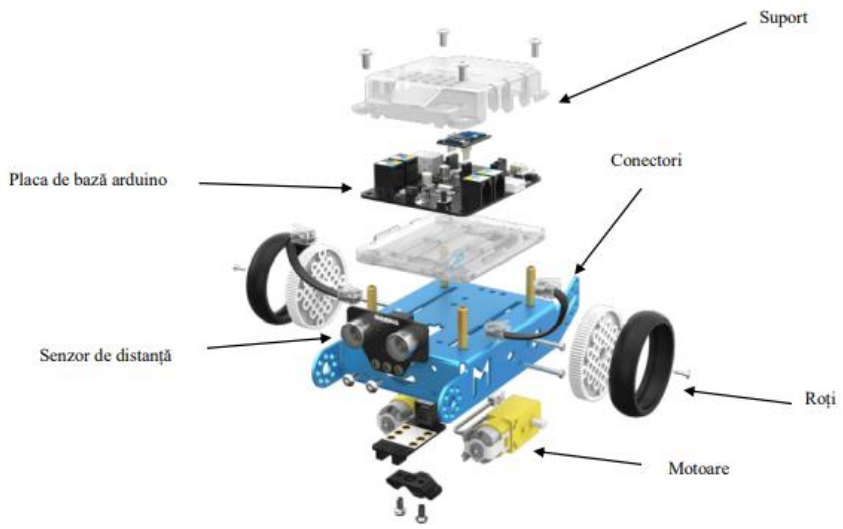


Fig. 18 Schelet mBot

Aceste mașini pot fi controlate folosind un device extern (laptop, tabletă, telefon mobil), care poate fi conectat la sistemul CPS (CPS – Cyber Physical System) prin intermediul unui cablu USB, bluetooth sau WLAN. În cadrul laboratorului OMiLAB, există interfața REST de pe placa Raspberry-Pi care este folosită sa ne conectăm la mașină și îi permite utilizatorului să apeleze diferite funcții ale acesteia. Aceste funcții ne permit sa controlăm motoarele, să generăm tonuri, să activăm matricea de LED-uri, să controlăm un motor de tip step-by-step. Funcțiile pentru senzori ne permit să citim valorile de pe senzorul de proximitate sau de pe senzorul de line following. Versiunea standard a robotului ne oferă două motoare, două roți mari, un senzor de distanță și un senzor de line follow, dar pot fi adăugate și alte echipamente hardware (Fig. 15).

În cadrul laboratorului OMiLAB din ULBS, am folosit pentru realizarea proiectului de diploma mBot-ul care are următorul ip: 10.14.10.241:8080/mBot [1][5] fiind conectat la rețeaua OMiLAB-ULBS.

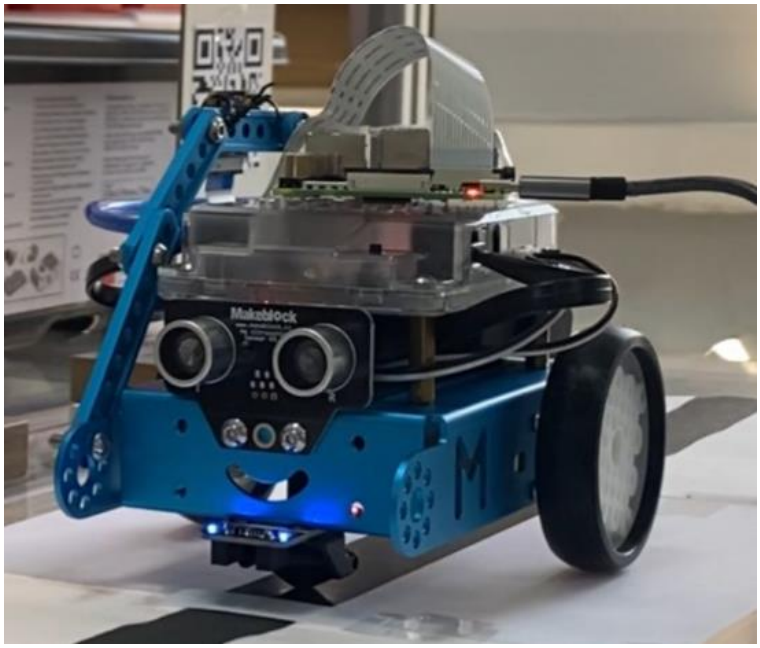


Fig. 19 mBot folosit pentru realizarea proiectului de diplomă

2.9.2 Raspberry Pi

Fig. 20 Raspberry Pi logo

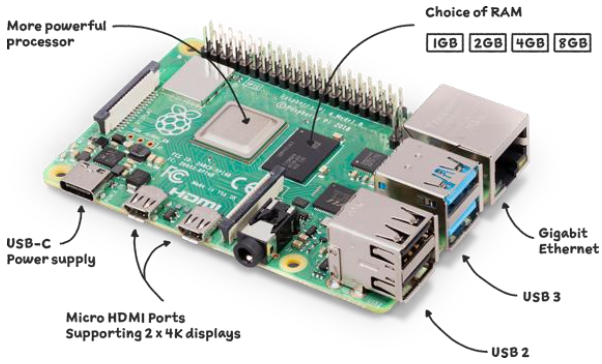


Fig. 21 Raspberry Pi 4

Raspberry Pi este un ‘computer’ mic, cu costuri reduse, cu o mărime destul de mică (Fig. 21) care se conectează la un monitor, putând conectându-se și un mouse si o tastatură. ‘Rpi’ poate să facă exact ce face un computer desktop, inclusiv accesarea internetului și vizionarea de videoclipuri HD, precum și gaming, foi de calcul.

Principalele **avantaje** ale acestei plăcuțe sunt:

- Acest microcomputer poate fi folosit de companiile mici cu un buget redus pentru a-și folosi produsul sau pentru a construi o nouă tehnologie care este încorporată în produs. Proprietarii de companii mici pot utiliza Raspberry Pi pentru a automatiza orice operațiune mică, cum ar fi rularea unui site web sau folosirea acestuia ca bază de date mică.
- Produsul oferă spațiu mare de a experimenta. Cardurile SD de pe placă pot fi schimbate cu ușurință, ceea ce permite schimbarea funcțiilor dispozitivului fără a pierde mult timp reinstalând software-ul.

Principalele **dezavantaje** ale plăcuței sunt:

- Nu este compatibil cu alte sisteme de operare precum Windows
- Procesorul nu este la fel de rapid ca un procesor al unui computer
- Incapabil de a face multitasking

2.9.3 Camera de filmat Raspberry Pi

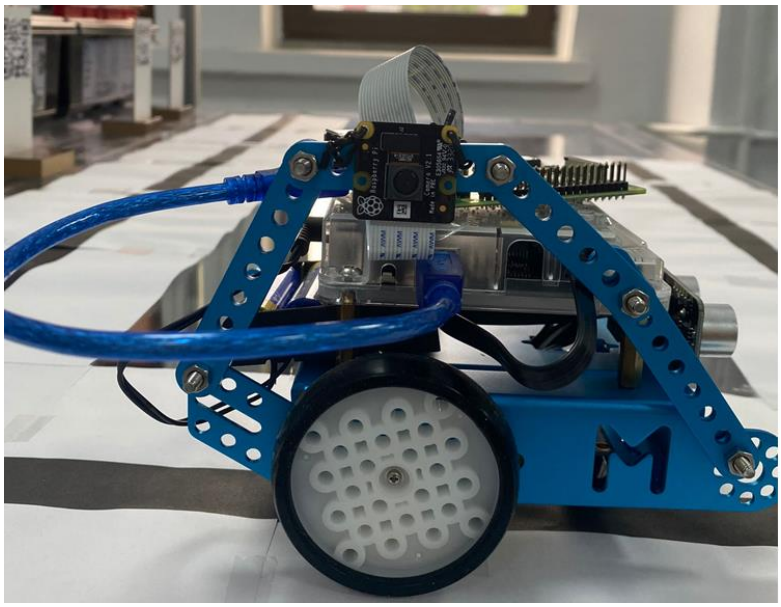


Fig. 22 Camera de filmat

Pentru scenariul propus am ales să atașez camera de filmat în laterala mBotului. Aceasta este o cameră de filmat Raspberry Pi NoIR Camera v2.

Această cameră de filmat are 8 megapixeli, model Sony IMX219, proiectată la comandă pentru Raspberry Pi cu o lentilă de focalizare fixă fiind capabilă să proceseze imagini pe 3280 x 2464 pixeli, și de asemenea acceptă resoluții de 1080 cu 30 frame-uri pe secundă, dar și 720, 640, 480 cu 60 frame-uri pe secundă.

Camera se atașează la Pi prin intermediul modulului special de pe suprafața superioară a plăcii și folosește interfața CSi dedicate, concepută special pentru interfața cu camerele foto.

Pentru a putea folosi camera de filmat trebuiesc făcute setările aferente în configurația Raspberry Pi -ului, mai exact trebuie activată opțiunea pentru cameră. În figura de mai jos sunt prezentați pașii:

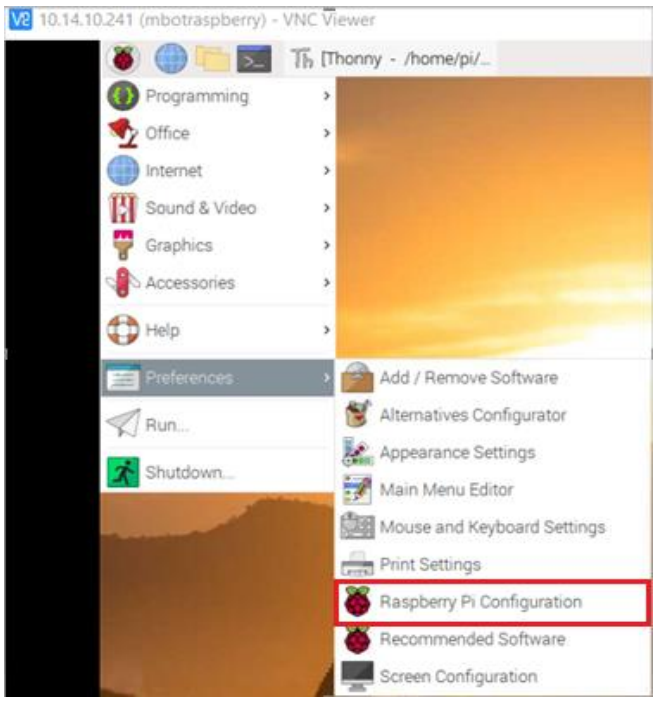


Fig. 23 Raspberry Pi Config path

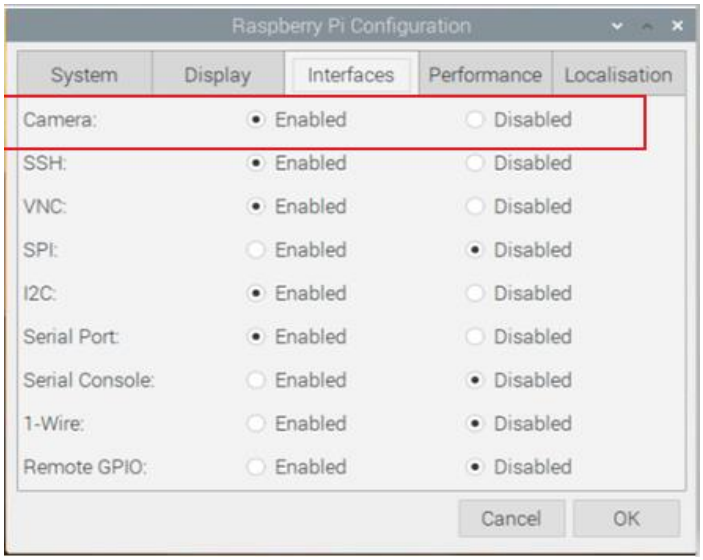


Fig. 24 Enable Camera

3. Elaborarea temei de proiect

Pentru a rezolva problema automatizării unui depozit agricol am avut nevoie de o serie de tool-uri, medii de dezvoltare și de echipament hardware precum: Bee-Up, Scene2Model, Visual Studio, Thonny Python IDE. Partea hardware a fost reprezentată de mBot și camera de filmat atașată in laterala acestuia.

3.1 Abordarea temei

În vederea rezolvării temei de proiect am folosit conceptul celor de la OMIiLAB și anume împărțirea problemei pe 3 niveluri de abstractizare:

- Design Thinking
- Conceptual Modelling
- Cyber Physical Systems

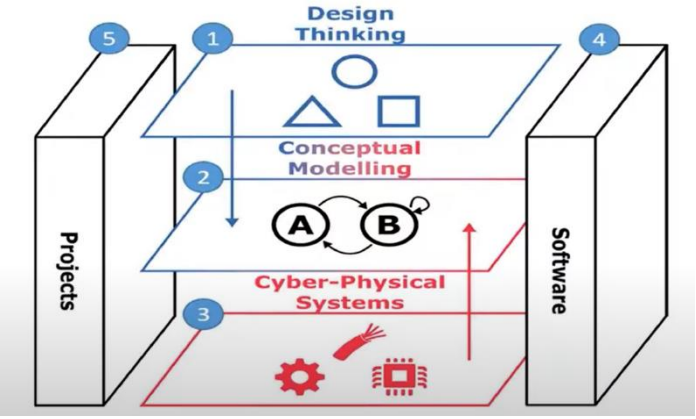


Fig. 25 Nivelurile de abstractizare OMiLAB

3.2 Design Thinking

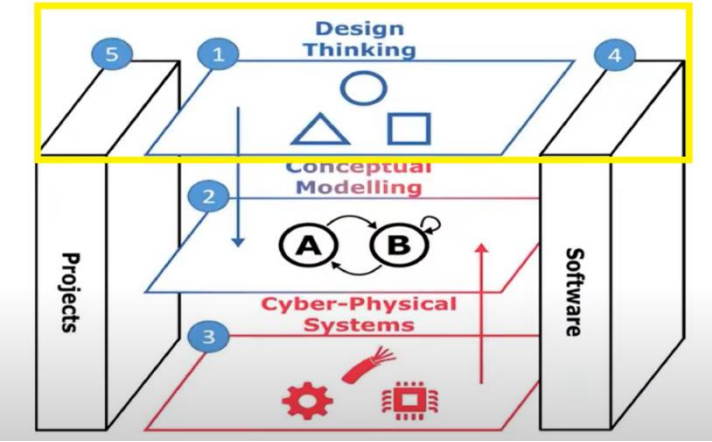


Fig. 26 Evidențierea conceptului Design Thinking

Acest concept a fost inițiat pe plan mondial la Universitatea Standford din Palo Alto USA în anul 2005, iar apoi, la nivel european a fost implementat de către Institutul Hasso Plattner Potsdam din Germania în anul 2007.

Conceptul presupune gândirea unui produs nou pornind de la faza de proiect atât din perspectivă tehnică (fezabilitate tehnologică), a necesității şi utilizării de către societate cât și a viabilității economice. Se bazează pe puterea muncii în echipă, colaborare, inovare. Implementarea acestui concept presupune un proces iterativ bazat pe următoarele stări: înţelegere, observare, definirea unui punct de vedere, concepere, dezvoltarea unui prototip şi testare.

Partea de Design Thinking este acea parte creativă si vizuală din proiect. Pentru a rezolva această parte m-am folosit de tool-ul Scene2Model. Acest tool ne permite să descriem în poze scena pe care vrem sa o implementăm.

Scena implementată de mine este următoarea:

mBotul pleacă dintr-un punct fix de start al traseului. mBotul se oprește în fața fiecărui raft fiind ajutat de câte un gap alb situat pe traseu. Fiecare raft este reprezentat de câte un cod QR care reprezintă câte un aliment împreună cu informațiile necesare despre acesta, urmând a fi scanat de

către camera de filmat atașată mBot-ului. După scanare, mBotul pornește către al 2 lea raft făcând acest ritual până ajunge la finalul traseului.

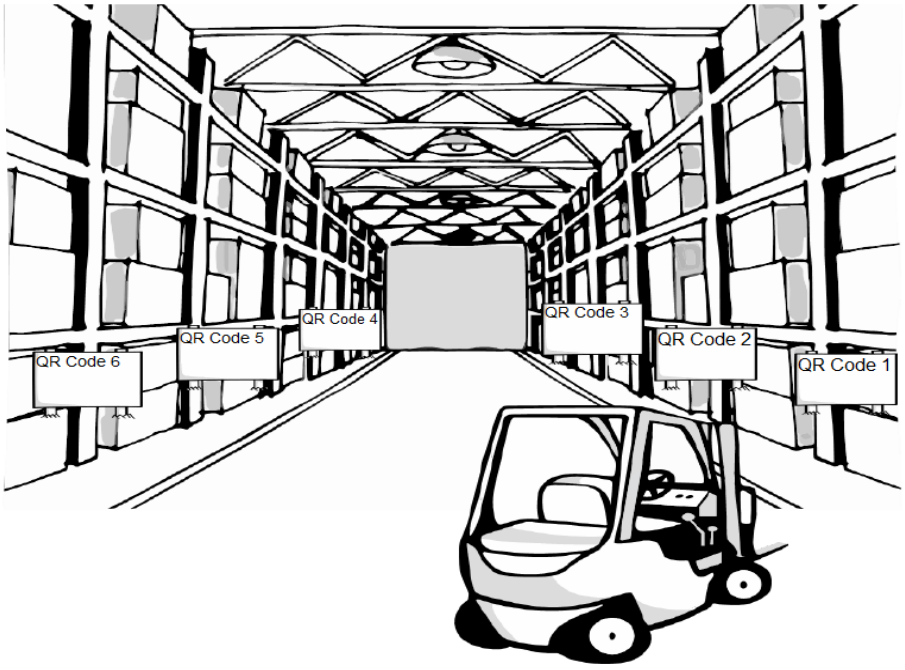


Fig. 27 Scenariu vizual

3.3 Concept Modelling

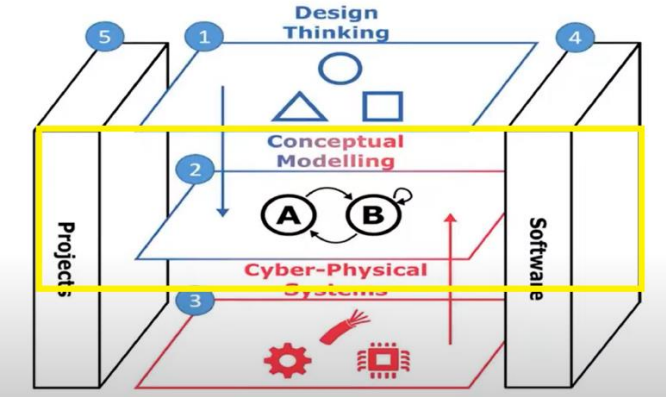


Fig. 28 Evidențierea conceptului Concept Modelling

Cel de-al doilea nivel este reprezentat de Concept Modelling care este realizat tot cu ajutorul tool-ului Scene2Model. Acest nivel reprezintă detalierea fiecărei scene pentru a avea o imagine mai clară asupra a ceea ce se întâmplă în scena propusă.

Acest concept este creat cu ajutorul modelelor de tip BPMN. Un model de tipul BPMN trebuie să înceapă cu un punct de START și să se termine cu un punct de STOP.

În figura de mai jos este prezentat acest model, cercurile galbene reprezentând Start si Stop, dreptunghiurile albastre reprezentând elemente de tip “Task” și romburile galbene reprezentând elemente de tip “Exclusive Gateway” care ajută la verificări.

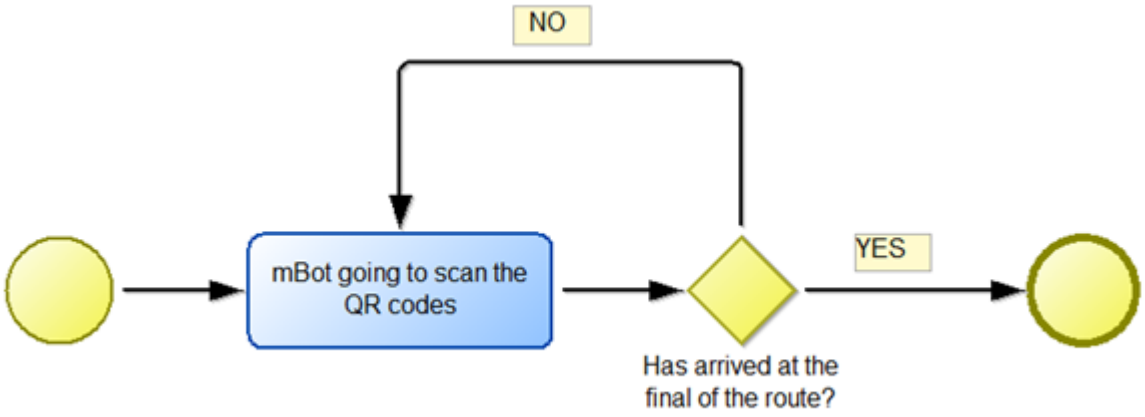


Fig. 29 BPMN Scenariu propus

3.4 Cyber Psihical System

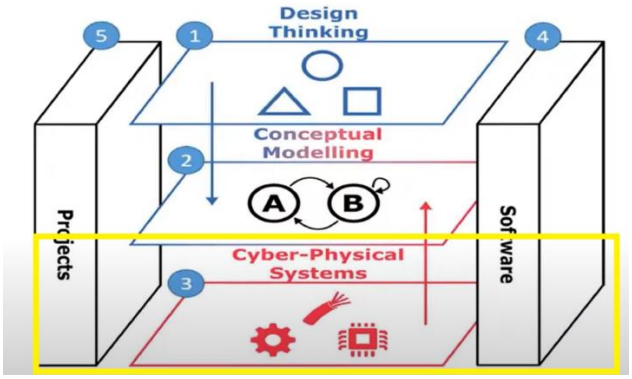
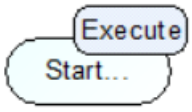


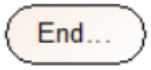
Fig. 30 Evidențierea conceptului Cyber Physical Systems

Cel de-al treilea nivel îl reprezintă CPS (Cyber Physical Systems) și este partea în care întreg scenariul este rezolvat în urma rulării programului. Acest nivel este realizat cu ajutorul tool-ului Bee-Up, program care are capacitatea de a simula modelele create. Scenariul a fost realizat cu ajutorul funcțiilor deja implementate care au venit împreună cu instalarea API-ului. Utilizând și adaptând aceste funcții am reușit să modelez o parte din automatizarea unui depozit agricol. Am reușit astfel să creez un scenariu în care este implicat un mBot care parcurge un traseu definit scanând pe rând coduri QR.

Ca și metodă de rezolvare, am folosit flowchart-ul din cadrul programului Bee-Up pentru scenariul gândit. Flowchart-ul execută toate instrucțiunile secvențial și odată cu apăsarea butonului de START aplicația va rula până la END fără intervenția utilizatorului.

În cadrul flowchart-ului realizat găsim următoarele elemente de tip “Start Terminal”, ”End Terminal”, ”Operation” și “Subsequent”.

- Start Terminal: 

- End Terminal: 

- Subsequent: 
- Operation: 

Flowchart-urile trebuie să înceapă cu un element de tip Start și să se încheie cu un element de tip Stop. Elementele de tip Subsequent ajută la legarea elementelor de orice tip între ele, iar elementele de tip Operation conțin cod care apelează anumite funcții din interfață, funcții ce vor fi descrise în detaliu.

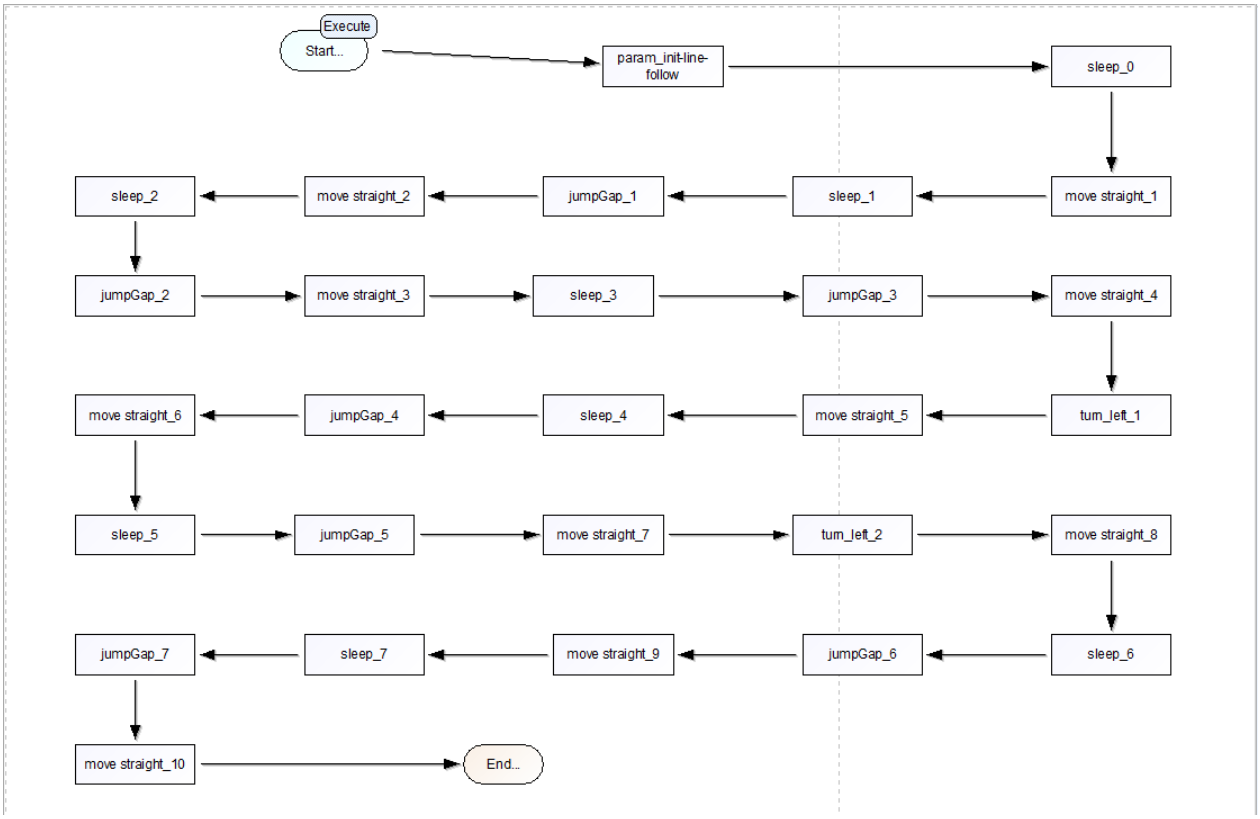


Fig. 31 Flowchart aplicație

3.4.1 Funcțiile mBot

Rolul mBotului este de a parcurge traseul oprindu-se câte 2 secunde în lateralul fiecărui QR code semnalizat de câte un gap alb. În figura de mai jos este prezentat traseul:

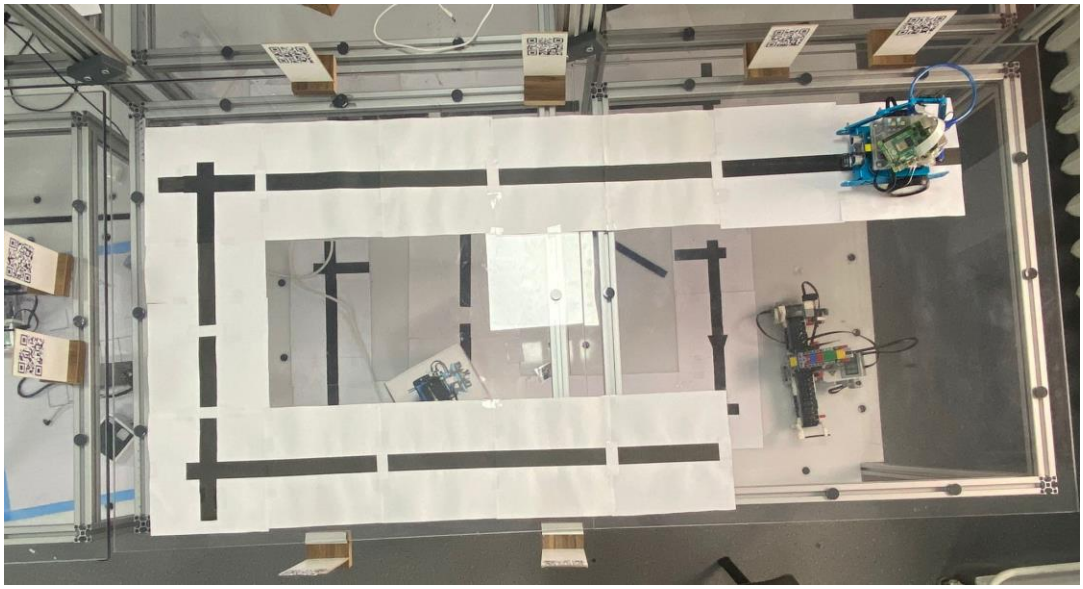


Fig. 32 Traseul mBotului

Funcțiile pentru mBot sunt implementate în elementele de tip Operation din cadrul tool-ului Bee-Up, aceste comenzi fiind de tip Adoscript. Acestea vor fi transmise prin Raspberry-Pi la platforma WEB. Dupa finalizarea comenzilor, codul este transferat la mBot tot prin Raspberry-Pi către Arduino. În figura următoare este prezentat flow-ul apelului de funcție:

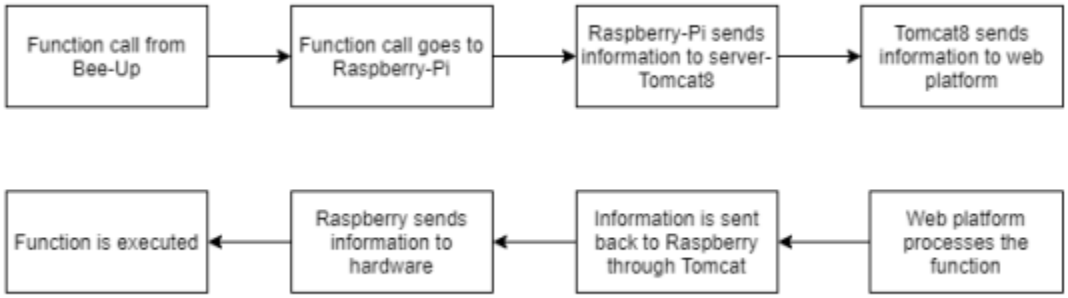


Fig. 33 Flow-ul apelului de funcție

Comenzile disponibile pe platforma Web in momentul de față sunt următoarele:

Comenzi pentru configurare:

- /getAvailablePorts: returnează toate porturile disponibile
- /getPort: returnează toate porturile USB folosite pe placa Raspberry
- /getMbotConfiguration: returnează configurația curentă de pe mBot sub formă de string
- /status: returnează un mesaj de status sub formă de text

Comenzi pentru line-following:

- /moveStraight: această comandă oferă posibilitatea de a rula mBotul pe direcția înainte cât timp ambii senzori de detecție vor fi pe suprafață întunecată (negru). Când cei doi senzori vor fi pe suprafata luminată, mBotul se va opri. Parametrul de intrare al acestei comenzi este doar viteza de deplasare. Acest parametru de input poate să fie și o valoare negativă care va face ca mBotul să meargă cu spatele.

- /jumpGap: această comandă face ca mBotul sa meargă înainte până când ambii senzori de detecție ajung să citească o suprafață întunecată. Ca și parametrul de intrare este necesară doar viteza de deplasare.
- /turnRight: această comandă oferă posibilitatea de a roti mBotul către dreapta până când măcar un senzor de detecție citește o suprafață întunecată. Pentru folosirea acestei comenzi este necesar ca ambii senzori de detecție să fie pe o suprafață iluminată. Ca și parametrul de intrare este necesară doar viteza de deplasare.
- /turnLeft: această comandă oferă posibilitatea de a roti mBotul către stânga până când măcar un senzor de detecție citește o suprafață întunecată. Pentru folosirea acestei comenzi este necesar ca ambii senzori de detecție să fie pe o suprafață iluminată. Ca și parametrul de intrare este necesară doar viteza de deplasare.

Comenzi pentru mișcare:

- /moveForward: această comandă permite robotului de a se deplasa înainte. Parametrii de intrare vor fi viteza și timpul de deplasare
- /moveBackward: această comandă permite robotului de a se deplasa înapoi. Parametrii de intrare vor fi viteza și timpul de deplasare
- /turnRight: această comandă permite robotului de a roti mașina spre dreapta. Parametrii de intrare vor fi viteza și timpul de rotire
- /turnLeft: această comandă permite robotului de a roti mașina spre stânga. Parametrii de intrare vor fi viteza și timpul de rotire

Comenzi pentru mișcare și detecție a obiectelor:

- /moveForwardObstacle: această comandă permite robotului să meargă înainte cât timp în fața lui nu se află niciun obiect la distanță mai mică decat distanța minimă. Parametrii de intrare vor fi viteza de deplasare si distanța minimă
- /turnRightObstacle: această comandă permite robotului să se rotească spre dreapta cât timp nu detectează vreun obiect mai apropiat decât distanța minimă. Parametrii de intrare vor fi viteza de rotire și distanța minimă

- /turnLeftObstacle: această comandă permite robotului să se rotească spre stânga cât timp nu detectează vreun obiect mai apropiat decât distanța minimă. Parametrii de intrare vor fi viteza de rotire și distanța minimă

Comenzi pentru citirea senzorilor:

- /readUSSensor: această comandă returnează valoarea citită de senzorul de distanță. Acesta citește valori între 3 si 400 de centimetrii. Dacă valoarea este mai mare de 400, se va returna 400
- /readLFSensor: această comandă returnează starea curentă a senzorilor de line-following. Valorile ce pot fi returnate sunt 0, 1, 2 sau 3. Dacă ambii senzori se află pe suprafața întunecată va fi returnata valoarea 0, iar daca ambii senzori se află pe suprafața luminată va fi returnata valoarea 3. Dacă un senzor se află pe suprafața luminată, iar celălalt se află pe suprafața întunecată, se va returna 1 sau 2
- /readSensor: această comandă returnează starea unui anumit senzor. Parametrul de intrare va fi un string cu numele de identificare al senzorului respectiv
- /readAllSensors: această comandă returnează starea tuturor senzorilor

Comenzi pentru sunet:

- /honk: această comandă permite robotului să scoată sunete asemănătoare cu un claxon
- /playTone: această comandă permite robotului să scoată un sunet customizat. Paremetrii de intrare vor fi frecvența semnalului si durata sa in milisecunde

3.5 Detalierea funcțiilor mBot folosite

Pentru realizarea flowchart-urilor pentru mBot, am folosit o parte din funcțiile amintite mai sus. Acest flowchart începe cu un bloc de START, acesta neavând cod scris, fiind doar un punct de intrare în flowchart. De asemenea, flowchart-ul se termină cu un bloc de END, la fel, care nu conține cod, cu rol în oprirea execuției programului.

Funcția de inițializare a parametrilor

Pentru a putea folosi mBotului, trebuie întâi să inițializăm anumite variabile. Astfel se setează ip-ul robotului astfel:

```
SETL str_roboturl1:("http://10.14.10.241:8080/mBot/api/linefollowing_operation/")
```

În același bloc de inițializare va trebui declarată încă o variabilă numită ”map_headers” necesar pentru apelul funcției „HTTP_SEND_REQUEST”.

```
SETL map_headers:({"Content-Type":"application/json"})
```

Funcția de sleep

Această funcție am folosit-o pentru a avea timp să pornesc toate procesele întregii aplicații(rulare Bee-Up, rularea scriptului de scanare QR, rulare aplicația vizuală a bazei de date).

De asemenea am folosit funcția de sleep de fiecare dată când mBotul se oprește în lateralul unui QR pentru a-i da timp să scaneze codul QR.

Codul acestei operații arata în felul următor:

```
Operation code:
CC "AdoScript" SLEEP
ms:5000
```

Funcția /moveStraight

Funcția /moveStraight este implementată într-un bloc de tip Operation.

```
Operation code:
HTTP_SEND_REQUEST (str_roboturl1+ "moveStraight?speed=100") str_method:("GET") map_reqheaders:
(map_headers) str_reqbody:("do") val_respcode:val_httpcode map_respheaders:map_respheaders
str_respbody:str_respbody
```

În acest caz, va fi apelată funcția /moveStraight din cadrul operațiilor de line-following de la adresa dată de variabila roboturl1. Parametrul de intrare se poate observa în variabila speed.

Funcția /jumpGap

Pentru funcția aceasta, inițializarea variabilei ce conține ip-ul și tipul operației este asemănătoare cu cea a funcției /moveStraight.

```
Operation code:
HTTP_SEND_REQUEST (str_roboturl1 + "jumpGap?speed=100") str_method:("GET") map_reqheaders:
(map_headers) str_reqbody:("do") val_respcode:val_httpcode map_respheaders:map_respheaders
str_respbody:str_respbody
```

Această funcție este folosită pentru a face robotul să mearga pe direcția înainte pe o suprafață luminată (alb în cazul meu), până când ambii senzori de detecție ai luminii vor detecta o suprafață întunecată. Ca și la funcția de /moveStraight, parametrul de intrare pentru viteză poate fi negativ, acest lucru va face ca mBotul să meargă cu spatele.

Funcția /turnLeft (line-following)

Pentru a folosi această funcție de turn left cu un singur parametru, inițializarea variabilei este asemănătoare cu cea de la /jumpGap si /moveStraight. Codul pentru această operație de /turnLeft este următorul:

```
HTTP_SEND_REQUEST (str_roboturl1 + "turnLeft?speed=120") str_method:("GET") map_reqheaders:
(map_headers) str_reqbody:("do") val_respcode:val_httpcode map_respheaders:map_respheaders
str_respbody:str_respbody
```

Parametrul de intrare al funcției este viteza de rotire și se poate observa in variabila speed.

3.6 Scanarea codurilor QR

Având în vedere evoluția tehnologiei în ultimii ani, consider că este necesară o îmbunătățire în cadrul depozitelor agricole & industriale, a fabricilor și nu numai.

Codurile QR se regăsesc peste tot în lumea noastră modernă și din motive întemeiate. Acestea au multe asemănări cu codurile de bare, dar în loc de laser, este folosită o cameră pentru a identifica spațiile dintre pătratele aflate în colțurile codului QR. Capacitatea de a codifica aceste date reprezentate prin pătrate alb-negru este incredibil de utilă, învățarea automată făcând posibila descifrarea codului, fiind indescifrabilă pentru ochiul uman. În funcție de software-ul folosit, se pot afișa rezultatele în direct, se pot trimite acele informații pentru a fi procesate, puteți permite să redirecteze către un site web sau să faceți orice doriți.

Aceste QR code-uri pot codifica o varietate de tipuri de date inclusiv caractere, numere și binare, ceea ce poate permite numeroase utilizări creative. De exemplu, agenții de publicitate codifică adresele URL cu ei pentru a redirecționa utilizatorul către site-ul lor web. Companiile plasează informații importante despre produs într-un cod QR, cum ar fi un număr de serie. Există o mulțime de standarde și tipuri de coduri QR, dar acest sistem va funcționa cu toate tipurile comune.

Utilitatea implementării acestui script este de a reduce timpul si munca fizică petrecută de ființa umană. Spre exemplu, fără un robot inteligent, omul trebuie sa parcurgă singur traseul scanând manual fiecare cod QR fiind nevoit să bage manual informațiile într-o bază de date pentru a ține o evidență.

Astfel, cu ajutorul mediului de programare Thonny Python IDE, am reușit să implementez un script în limbajul de programare Python care reușește să detecteze coduri QR și să le scaneze.

Mai mult de atât, datele scanate vor fi transmise către baza de date configurată la începutul scriptului care va fi prezentată ulterior.

Placa Raspberry-Pi poate fi conectată la monitor în două moduri:

- Printr-un cablu HDMI conectat din plăcuță direct la monitor.
- Prin programul special “VNC Viewer”

Conectarea direct prin **cablul HDMI** este mult mai avantajoasă din punct de vedere al calității imaginii și al vitezei de reacție, dar în același timp are un dezavantaj major, și anume portabilitatea.

Conectarea **remote prin VNC Viewer** are avantajul portabilității, dar în același timp viteza de reacție și calitatea imaginii sunt scăzute.

Așadar, având nevoie de o portabilitate mare am ales conectarea prin **VNC Viewer**. Procesul este unul simplu. Se scrie adresa ip a plăcii Raspberry Pi, urmând a fi necesare datele de login compuse din username si password (user-ul si parola plăcuței). După logare, în fereastra principală a programului va apărea o imagine a sistemului de operare Raspbian care reprezintă calea către conectarea remote.

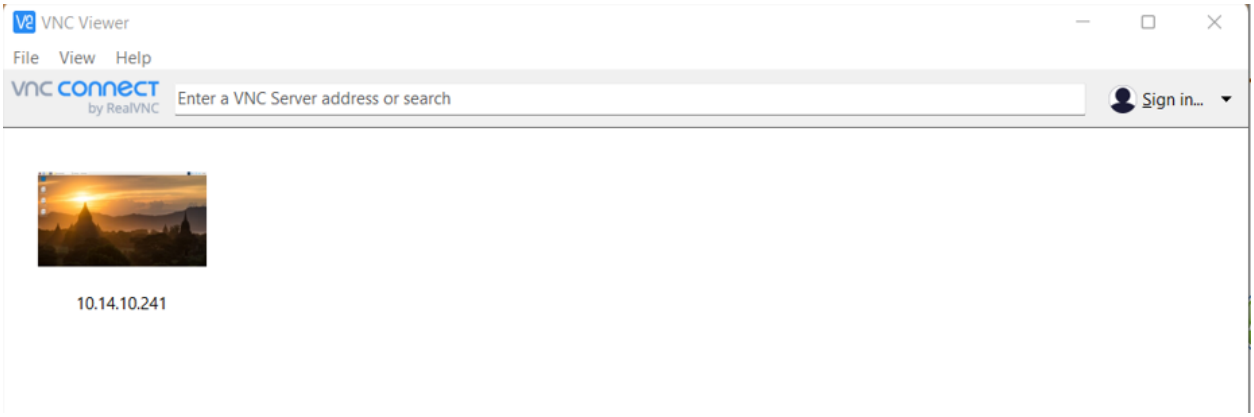


Fig. 34 VNV Viewer

3.6.1 Sotfware set-up

Pentru început, pentru rezolvarea problemei a fost nevoie de importarea a câtorva biblioteci precum:

```
1 import cv2
2 import re
3 import csv
4 import pyrebase
5 import time
6 from datetime import date, datetime
```

Fig. 35 Biblioteci importate

- cv2 (Open-CV): pentru procesarea imaginilor
- re (Regular expression operation) : oferă operații de înlocuire/potrivire a expresiilor regulate
- pyrebase: librăria bazei de date Firebase (ajută la conectarea acesteia cu aplicația)
- time: librăria pentru ora exactă în timp real
- date, datetime: librăria pentru data în timp real

Open CV este o resură mare care ajută la rezolvarea în timp real a problemelor de viziune computerizată și de procesare a imaginilor. Pentru a-l instala, am introdus urmatoarele comenzi in terminalul Raspberry-Pi:

```
sudo apt-get update

sudo apt-get install python3-opencv

sudo apt-get install libqt4-test python3-sip python3-pyqt5 libqtgui4 libjasper-dev libatlas-base-dev -y

pip3 install opencv-contrib-python==4.1.0.25

sudo modprobe bcm2835-v4l2
```

Fig. 36 comenzi de instalare set-up

3.6.2 Conectarea bazei de date cu aplicația de scanare

Pentru stocarea datelor, am ales baza de date de tip NoSQL Firebase. Pentru a putea transmite datele am făcut următoarea configurare:

```
firebaseConfig = {
  "apiKey": "AIzaSyD6xLtIq25e1we1jyvUtGZF7V6bDJeeVmo",
  "authDomain": "depozit-agricol.firebaseapp.com",
  "projectId": "depozit-agricol",
  "databaseURL":"https://depozit-agricol-default-rtdb.europe-west1.firebasedatabase.app/",
  "storageBucket": "depozit-agricol.appspot.com",
  "messagingSenderId": "709977840888",
  "appId":"1:709977840888:web:fb731e4fbd05fe5a8a2f0d",
  "measurementId": "G-JHVJTJVSBP"
};
```

Fig. 37 Configurare Firebase

```
firebase=pyrebase.initialize_app(firebaseConfig)
db=firebase.database()
```

Fig. 38 Instanțiere obiect

3.6.3 Prezentarea aplicației

Pentru început vom configura obiectul numit Cap (prescurtat de la cuvântul Capture) de tip VideoCapture. Parametrul acestei proprietăți a Open-CV-ului fiind 0 deoarece acesta indică indexul dispozitivului de filmat, în cazul nostru fiind prima cameră. Apoi, se va inițializa o variabilă cu detecția QR code-urilor.

```
cap = cv2.VideoCapture(0)
detector = cv2.QRCodeDetector()
```

Fig. 39 configurare VideoCapture + inițializare detecție QR

```
while True:
        _, img = cap.read()
        data, bbox, _ = detector.detectAndDecode(img)
        now = datetime.now()
        timeRN = now.strftime("%H:%M:%S")
        if bbox is not None:
            for i in range(len(bbox)):
                cv2.line(img, tuple(bbox[i][0]), tuple(bbox[(i+1) % len(bbox)][0]), color=(255,
                         0, 0), thickness=2)

            cv2.putText(img, data, (int(bbox[0][0][0]), int(bbox[0][0][1]) - 10), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (255, 250, 120), 2)
```

Fig. 40 Colorarea conturului si al textului QR-ului

În figura de mai sus este prezentată o buclă infinită în care, în primă faza este apelată metoda de preluare a imaginii QR code-ului, apoi una dintre cele mai importante metode, și anume cea de citire a QR-ului prin detectarea căsuței de delimitare și decodificare a datelor QR ascunse.

În continuare este prezentat modul prin care se obține conturul albastru din jurul QR-ului, urmând a fi implementat și textul de deasupra codului QR care reprezintă datele ce vor fi transmise în baza de date. Culorile pot fi schimbate alterând valorile variabilei color.

Fig. 41 Contur & text cod QR

Pentru o bună vizualizare în timp real a datelor scanate am decis să afișez informațiile fiecăui cod QR și în consola mediului de programare Thonny Python IDE.

Fig. 42 Consola în timpul scanării

Acest lucru este posibil datorită splituirii datelor dupa caracterul ‘’,’’, deoarece informațiile codului QR au urmatorul aspect: id, tip, cantitate, pret, tara (13, Mandarine, 116, 5, Italia).

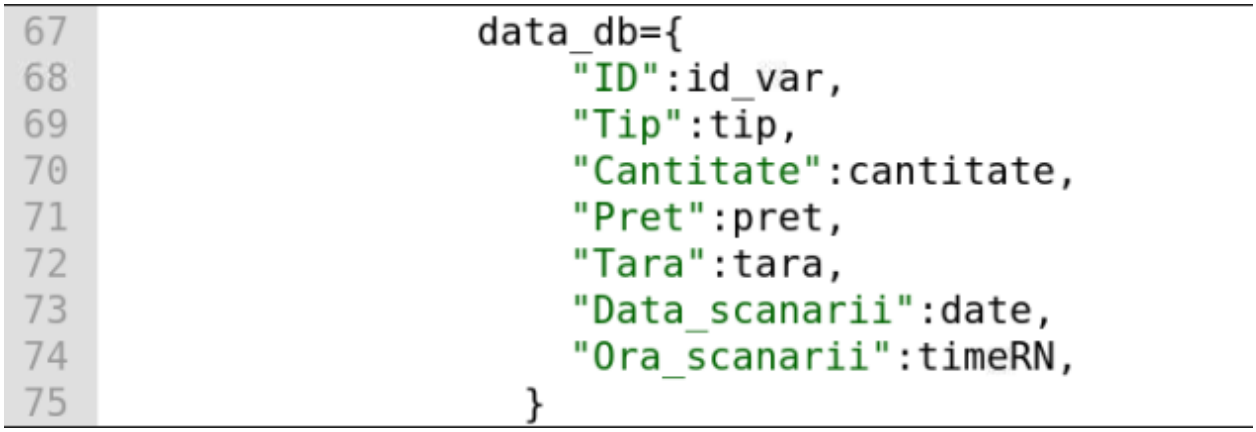
```
47          if data:
48
49              data = data.split(",")
50              print("ID: " + data[0])
51              print("Tip: " + data[1])
52              print("Cantitate: " + data[2])
53              print("Pret/Kg: " + data[3])
54              print("Tara provenienta: " + data[4])
55              print("Data scanarii: " + date)
56              print("Ora scanarii: "+ timeRN)
57              print()
```

Fig. 43 Splituirea si afișarea datelor în consolă

Așadar, datele vor fi splituite, formându-se un array de valori. Pentru configurarea intrărilor în Firebase am inițializat și atribuit variabilelor aferente cu datele splituite.

```
60              id_var= data[0]
61              tip=data[1]
62              cantitate=data[2]
63              pret=data[3]
64              tara=data[4]
```

Fig. 44 Inițializarea variabilelor



```
67                  data_db={
68                      "ID":id_var,
69                      "Tip":tip,
70                      "Cantitate":cantitate,
71                      "Pret":pret,
72                      "Tara":tara,
73                      "Data_scanarii":date,
74                      "Ora_scanarii":timeRN,
75                    }
```

Fig. 45 Structura bazei de date

În figura de mai sus am creat o structură în care fiecare intrare în tabel este atribuită string-ului respective (exemplu: informația despre preț scanată va fi atribuită prețului în baza de date).

După ce am structurat informațiile, datele scanate sunt pregătite pentru a fi adăugate în baza de date. Așadar am folosit proprietatea “child” a obiectului db pentru a crea o nouă intrare de tip child in baza de date.

```
db.child("ultima_scanare").set(data_db)
db.child("inventar_depozit").push(data_db)
```

Fig. 46 Adăugarea datelor în Firebase

Proprietatea SET a obiectului reprezintă ultima scanare, adică la fiecare scanare, în baza de date vor fi adăugate doar ultimele date scanate în secțiunea „ultima_scanare”

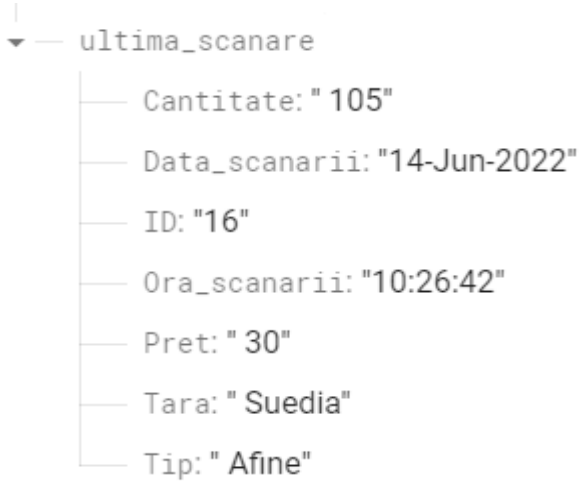


Fig. 47 Firebase, ultima scanare

Proprietatea PUSH a obiectului reprezintă adăugarea fiecărei scanări în baza de date, în secțiuni diferite. În figura de mai jos este prezentată imaginea secțiunii “inventar depozit”, fiecare string (reprezentat din cifre+litere) fiind o scanare unică.

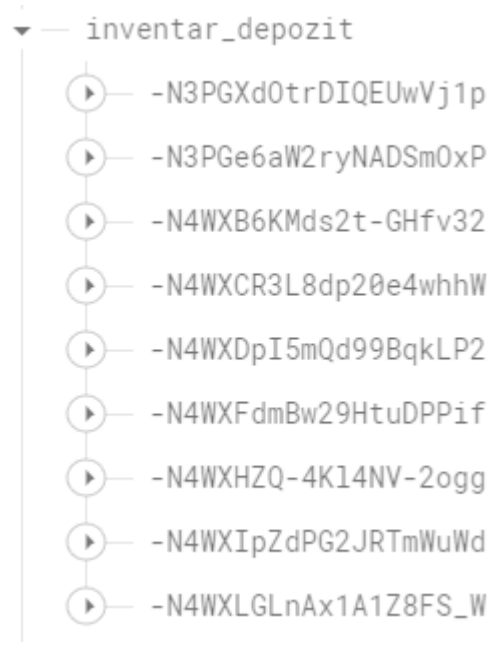


Fig. 48 Firebase, “inventar depozit”

În următoarea figură sunt prezentate intrăriile in baza de date a două dintre scanări.

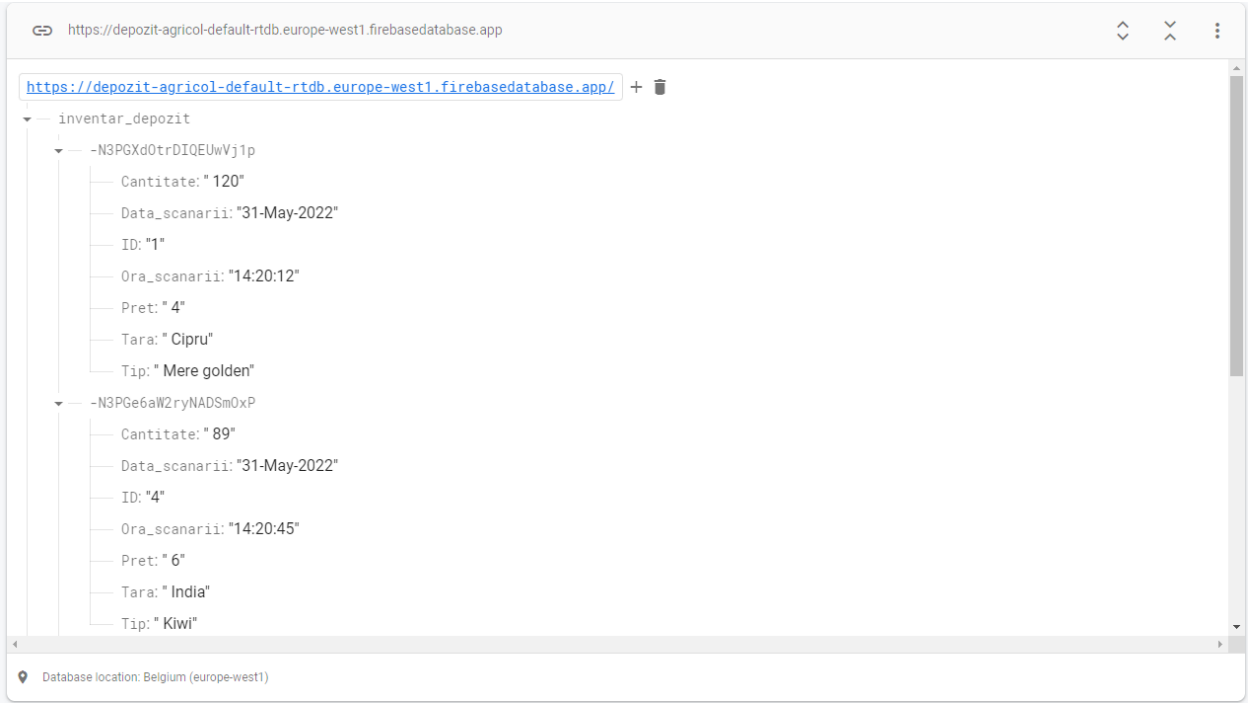


Fig. 49 Exemplu intrări în Firebase

Pentru afișarea în timp real a imaginilor furnizate de camera de filmat am folosit următoarea proprietate a obiectului din Open-CV.

```
cv2.imshow("code detector", img)
```

Fig. 50 Afișarea imaginilor în timp real

În final, am adăugat o condiție reprezentată de o funcție a librariei OpenCV care permite afișarea display-ului pentru un anumit număr de milisecunde sau până când tasta “q” va fi apăsată de utilizator.

După ce s-a terminat de rulat programul, se vor închide toate aplicațiile/ferestrele pe care le-a creat.

```
        if cv2.waitKey(500) == ord("q"):
            break

cap.release()
cv2.destroyAllWindows()
```

Fig. 51 Închiderea ferestrelor

3.7 Aplicație pentru afișarea stocului depozitului

Am creat această aplicație cu scopul de a vedea în timp real stocul de fructe / legume deținute în depozitul agricol. După scanarea fiecărui cod QR de către mBot, datele sunt transmise într-o bază de date (Firebase), apoi exportate în timp real în interfața vizuală C#.

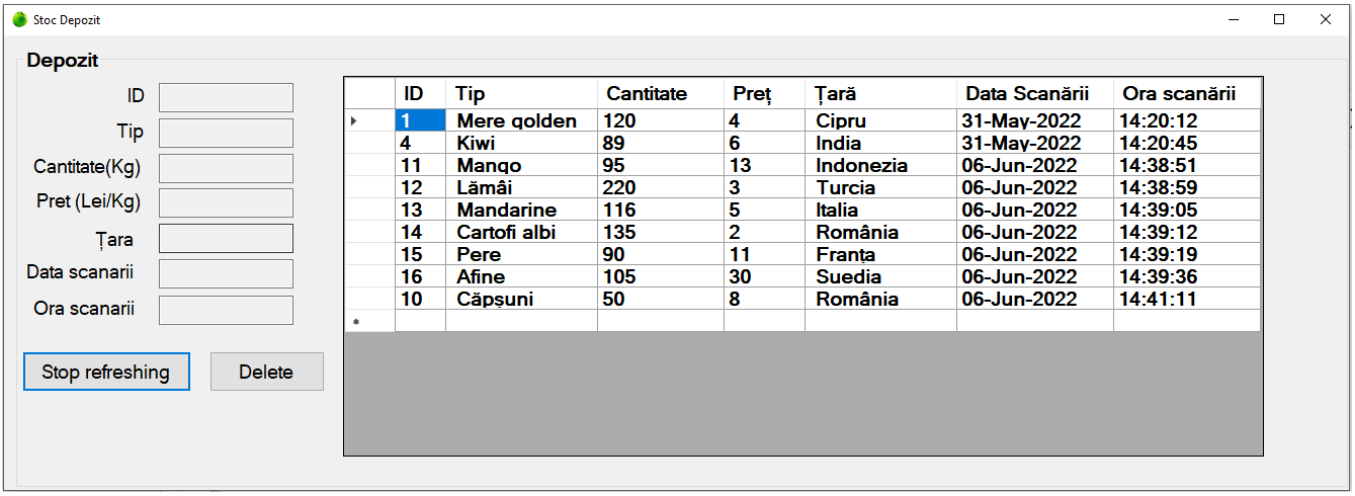


Fig. 52 Aplicație stoc depozit

3.7.1 Conectarea bazei de date cu aplicația C#

Înainte de a mă conecta la baza de date a fost nevoie de instalarea NuGet Package-ului FireSharp pentru achiziționarea bibliotecilor necesare. Pentru instalare, am folosit Package Manager Console din Visual Studio.

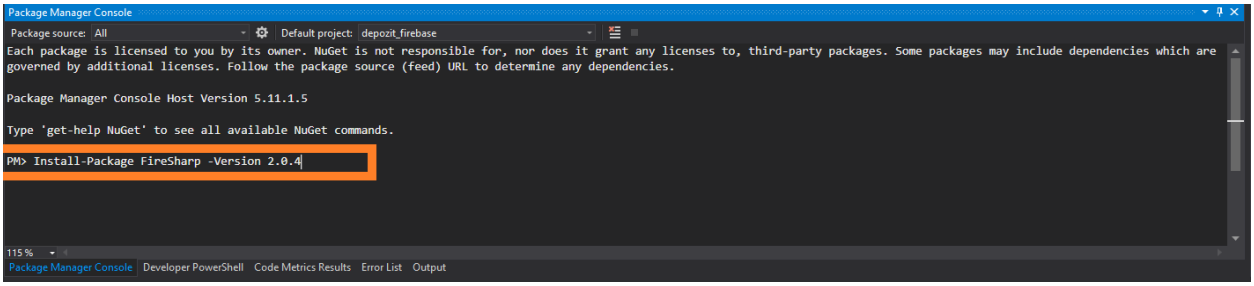


Fig. 53 Instalare FireSharp NuGet Package

Fig. 54 FireSharp Package

Figura de mai jos reprezintă configurarea conectării bazei de date Real Time Database (Firebase) cu aplicația vizuală.

```
IFirebaseConfig config = new FirebaseConfig()
{
    AuthSecret = "uZAiylU60C5P3R1G3MUOdPFOCrYSk1DLyKyNYz8p",
    BasePath = "https://depozit-agricol-default-rtdb.europe-west1.firebasedatabase.app/"
};
```

Fig. 55 Configurare Firebase

În configurare se află calea către baza de date a depozitului și codul de autentificare unic care se găsește în setările acesteia.

Se declară o variabilă cu tipul configurării, apoi la deschiderea aplicației se instanțează obiectul declarat anterior cu tipul FirebaseClient având ca parametru configurarea bazei de date.

```
IFirebaseClient client;
private void Form1_Load(object sender, EventArgs e)
{

    client = new FirebaseClient(config);

}
```

Fig. 56 Conectarea bazei de date

3.7.2 Butonul de refresh

Apăsând acest buton utilizatorul poate vedea informațiile despre alimentele din baza de date, actualizându-se la fiecare 5 secunde.

În cazul în care computerul de pe care se accesează aplicația nu este conectat la internet, la apăsarea butonului „Start refreshing” va apărea următorul pop-up:

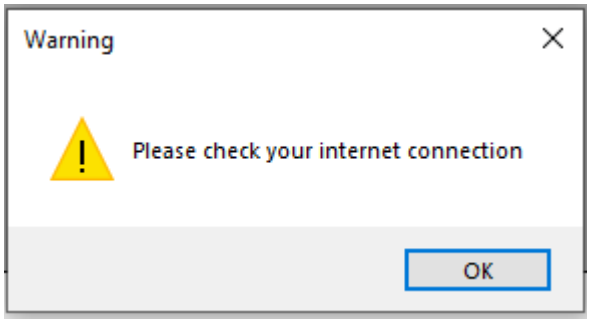


Fig.57 Verificare conexiune internet

În acest caz, utilizatorul va trebui să se conecteze la o rețea pentru a putea folosi aplicația. Pentru această verificare a conexiunii am folosit o funcție de tip boolean pe care am apelat-o într-o altă funcție și anume cea de adăugare a intrărilor în tabelă.

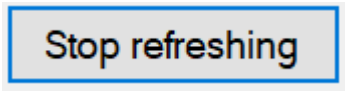
```
public static bool IsCheckInternet()
{
    try
    {
        using (var net = new WebClient())
        {
            using (net.OpenRead("http://www.google.com"))
            {
                return true;
            }

        }
    }
    catch (Exception e)
    {
        return false;
    }
}
```

Fig. 58 Funcție verificare conexiune internet

```
private async void load_entries()
{
    if (network.IsCheckInternet())
    {
        var data = await client.GetAsync("inventar_depozit");

        Dictionary<string, Data> depozit = data.ResultAs<Dictionary<string, Data>>();

        foreach (var aliment in depozit)
        {
            dataGridView1.Rows.Add(aliment.Value.Id, aliment.Value.Tip, aliment.Value.Cantitate, aliment.Value.Pret,
                aliment.Value.Tara, aliment.Value.Data_scanarii, aliment.Value.Ora_scanarii, aliment.Key);
        }

    }
    else
    {
        MessageBox.Show("Please check your internet connection", @"Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
```

Fig. 59 Funcția de exportare a datelor din baza de date în tabelă

Apăsând din nou acest buton, refresh-ul se oprește, dându-i posibilitatea utilizatorului să vadă clar tot stocul și de asemenea să șteargă din baza de date o anumita intrare in tabel. (ex: un anumit fruct / legumă)



După cum am precizat mai sus, utilizatorul poate să vizualizeze un anumit aliment in detaliu. Procesul constă în a da dublu-click în celula primei coloane pe rândul care indică alimentul dorit. Figura următoare reprezintă un exemplu pentru o mai bună vizualizare a cartofilor albi.

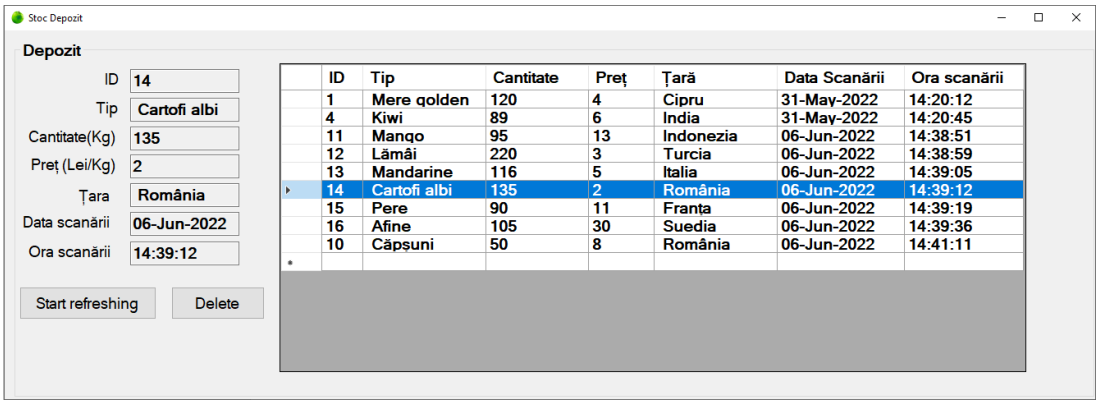
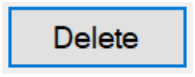


Fig. 60 Vizualizare aliment specific

3.7.3 Butonul Delete

Apăsând butonul Delete, utilizatorul poate să șteargă un anumit aliment selectându-l cu dublu-click din prima coloană a tabelului.



După apăsarea butonului va apărea un pop-up de asigurare a acțiunii ce va urma a fi făcută.

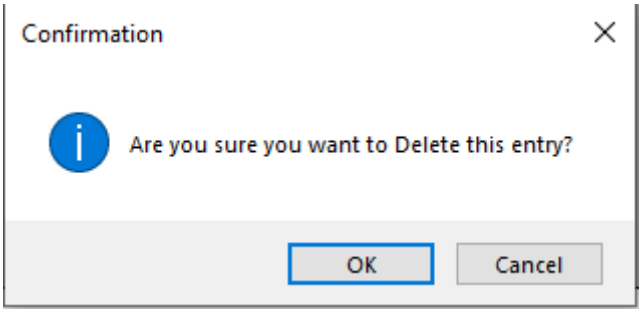


Fig. 61 Pop up de confirmare

Această acțiune de ștergere a unui aliment am realizat-o implementând o funcție specifică:

```
private async void delete_entry()
{
    DialogResult res = MessageBox.Show("Are you sure you want to Delete this entry?", "Confirmation", MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
    if (res == DialogResult.OK)
    {
        string id = labelKey.Text;
        var data = await client.DeleteAsync("inventar_depozit/" + id);
        if (data.Body != null)
        {
            dataGridView1.DataSource = null;
            dataGridView1.Rows.Clear();
            load_entries();
        }
    }
}
```

Fig. 62 Funcția acțiunii de Delete

3.7.4 Sortarea mărfii

Un lucru necesar în realizarea inventarului unui depozit agricol este de a putea sorta alimentele in funcție de informația dorită de utilizator. Este foarte important acest lucru în industria agricolă deoarece ușurează munca manuală.

Spre exemplu, dacă utilizatorul dorește să vadă stocul descrescător în funcție de cantitate pentru a știi ce alimente urmează să fie comandate, o poate face prin apăsarea coloanei “Cantitate” din cadrul tabelei. De asemenea se pot trage concluzii în privința vânzării acestora.

ID	Tip	Cantitate	Preț	Țară	Data Scanării	Ora scanării
12	Lămâi	220	3	Turcia	06-Jun-2022	14:38:59
14	Cartofi albi	135	2	România	06-Jun-2022	14:39:12
1	Mere golden	120	4	Cipru	31-May-2022	14:20:12
13	Mandarine	116	5	Italia	06-Jun-2022	14:39:05
16	Afine	105	30	Suedia	06-Jun-2022	14:39:36
11	Mango	95	13	Indonezia	06-Jun-2022	14:38:51
15	Pere	90	11	Franța	06-Jun-2022	14:39:19
4	Kiwi	89	6	India	31-May-2022	14:20:45
10	Căpșuni	50	8	România	06-Jun-2022	14:41:11

Fig. 63 Sortare dupa cantitate

De asemenea, utilizatorul poate sorta marfa în funcție de prețul acesteia. Este important ca într-un depozit să existe o evidență a mărfii din punct de vedere al prețului. Sortarea are loc apăsând pe coloana “Preț” din cadrul tabelei.

ID	Tip	Cantitate	Preț	Țară	Data Scanării	Ora scanării
14	Cartofi albi	135	2	România	06-Jun-2022	14:39:12
12	Lămâi	220	3	Turcia	06-Jun-2022	14:38:59
1	Mere golden	120	4	Cipru	31-May-2022	14:20:12
13	Mandarine	116	5	Italia	06-Jun-2022	14:39:05
4	Kiwi	89	6	India	31-May-2022	14:20:45
10	Căpșuni	50	8	România	06-Jun-2022	14:41:11
15	Pere	90	11	Franța	06-Jun-2022	14:39:19
11	Mango	95	13	Indonezia	06-Jun-2022	14:38:51
16	Afine	105	30	Suedia	06-Jun-2022	14:39:36

Fig. 64 Sortare după preț

O altă sortare importantă este cea după data scanării, deoarece se poate ține o evidență a vechimii alimentelor în depozit și nu numai. Sortarea se face apăsând pe coloana ”Data Scanării” din cadrul tabelei.

	ID	Tip	Cantitate	Preț	Țară	Data Scanării	Ora scanării
	10	Căpșuni	50	8	România	06-Jun-2022	14:41:11
	12	Lămâi	220	3	Turcia	06-Jun-2022	14:38:59
	11	Mango	95	13	Indonezia	06-Jun-2022	14:38:51
	14	Cartofi albi	135	2	România	06-Jun-2022	14:39:12
	16	Afine	105	30	Suedia	06-Jun-2022	14:39:36
	13	Mandarine	116	5	Italia	06-Jun-2022	14:39:05
	15	Pere	90	11	Franța	06-Jun-2022	14:39:19
▸	1	Mere golden	120	4	Cipru	31-May-2022	14:20:12
	4	Kiwi	89	6	India	31-May-2022	14:20:45
*							

Fig. 65 Sortare după data scanării

4. Concluzii și dezvoltări ulterioare

O parte din acest proiect de diplomă a fost realizat cu ajutorul Universității din Viena care au ajutat la dotarea laboratorului cu echipamentele atât hardware cât și software. Aceste tipuri de lucrări sunt foarte noi, inovative, cu o plajă largă de posibilități și de extindere, iar folosirea limbajelor de modelare este simplă și ușor de folosit. Se pot crea foarte multe tipuri de aplicații de către orice tip de utilizator, nefiind necesare cunoștiințe avansate de programare. Mă bucur că am reușit combin atât partea hardware si software pusă la dispoziție de Universitatea din Viena, cât și partea software reprezentată de aplicațiile din C# și Python care au ajutat la realizarea acestui proiect.

În ceea ce privește dezvoltarea ulterioară a proiectului, îmi doresc, în primul rând, să îmbunătățesc acest proiect prin însumarea cantității unui produs cu același ID dacă acesta a mai fost scanat o data. În momentul de față, algoritmul verifică dacă ID-ul QR-ului a mai fost scanat, dacă da, acesta nu va mai fi bagat în baza de date. Ideea a fost gândită pentru un depozit agricol de dimensiuni mici, ideea reușind a fi observată chiar din valorile mici ale cantităților. De aceea codurile QR folosite sunt unice, iar intrările în baza de date vor fi unice.

5. Bibliografie

[1]: OMiLAB-Team – “The OMiLAB Documentation”

[2]: Dimitris Karagiannis, Heinrich C. Mayr, John Mylopoulos – “Domain-Specific Conceptual Modelling” Concepts, Methods and Tools.

[3]: https://nemo.omilab.org/2021/

[4]: https://www.raspberrypi.com/documentation/

[5]: http://10.14.10.241:8080/mbot

[6]: https://www.omilab.org/activities/bee-up.html

[7]: https://www.digifof.eu/results/iot-platform-enabler-new-product-service-omilab-use-case/

[8]: https://www.python.org/

[9]: https://www.thepythoncode.com/