

EcoRobot

Robotul care poate colecta selectiv

Mihnea Bucovan, Matei Crainiceanu
coord. prof. Ovidiu Rușet,
InfoEducație 2023

Ştim că oamenii, în general nu se preocupă de colectarea selectivă a deşeurilor. Acesta este punctul de plecare pentru acest proiect: cum ajutăm oamenii să colecteze selectiv?

1. **Educație** - Cu toții putem să ne formăm obiceiuri noi dacă suntem învățați și motivați.

2. **Build a robot**

Decât să învățăm sau să pedepsim populația, mai bine construim un robot care va conduce prin exemplu, dar va putea colecta el deşeurile.

Așa s-a născut conceptul de EcoRobot - robotul care adună și sortează deșeuri. Dar oare cum putem să construim acest robot?

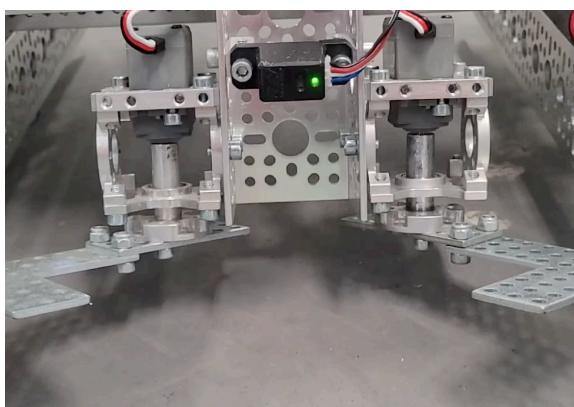
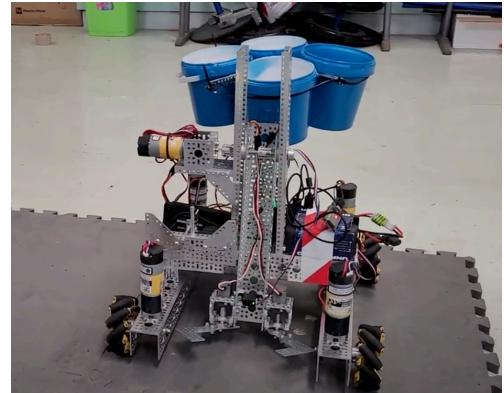
Mecanică

Construcția robotului

Robotul este construit în mare parte de piese goBilda, aceleași piese folosite în concursul First Tech Challenge.

Şasiul este cel standard cu roți la unghi de 45°, care permit robotului să meargă în toate direcțiile, spre deosebire de roți normale care se pot deplasa doar înainte și înapoi.

Brațul robotului este atașat pe șasiu și este ridicat de un motor cu cuplu mare (133.2 kg / cm).



“Gheara” robotului este mecanismul de prindere al deşeurilor. Acesta este format din două servo-motoare cu cuplu mare, fiecare având un grad de mișcare de 270°, din care folosim doar un unghi de 50° pe fiecare servo. De ele sunt atașate piese în L pentru a putea fixa deșeurile.

Mecanismul de depozitare selectivă prezintă 4 compartimente, făcute din cutii de iaurt, vopsite apoi, pentru a avea un aspect plăcut. Aceste cutii sunt fixate în jurul unui X din metal. Acest X este rotit de un servomotor (cu grad de libertate 270°) pentru a selecta și pregăti compartimentul respectiv pentru tipul de deșeu.



Pe șasiu sunt fixate și contollerile, despre care vorbim în secțiunea dedicată, alături de bateria (standard pentru FTC) de 12V, 3000 mAh.

REZUMAT: Pe robot se regăsesc 5 motoare, 4 folosite pentru deplasare și unul folosit pentru ridicarea și coborârea brațului. De asemenea folosim 2 servomotoare pentru gheara și unul pentru mecanismul de depozitare selectivă.

Electronica prezentă pe robot

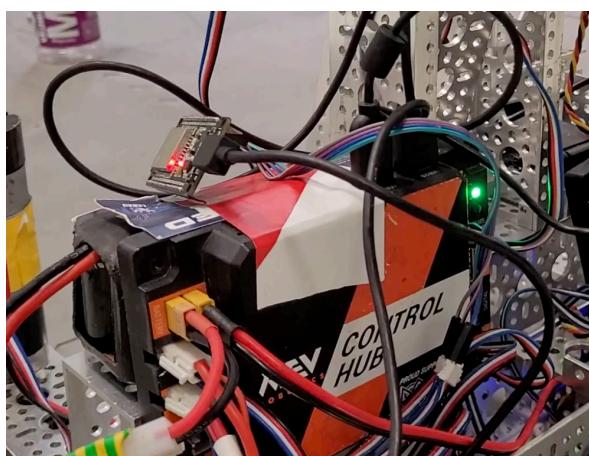
Robotul nostru conține 3 sisteme importante: Arduino-ul, REV-ul (format din două părți: Control Hub, Expansion Hub) și iPhone-ul.

Folosim REV-ul pentru mișcările robotului și citirea datelor din senzorul de distanță, deoarece erau controlerle pe care le aveam la începutul acestui proiect, alături de Kit-ul de piese pentru FTC. În REV sunt conectate toate motoarele:

- 4 Motoare pe curent continuu (312 go:bilda), fiecare cu câte un Encoder - pentru a vedea exact cât se rotește fiecare motor (ne folosește la deplasare).
- 1 motor + Encoder pentru braț, ca să îl putem ridica cât mai precis.
- 1 Servomotor al compartimentelor de colectare.
- 2 Servomotor ale brațului, pentru prinderea deșeurilor
- 4 Pini Analog pentru comunicarea dintre Arduino și REV.

Folosim un ESP-8266 ca și web server ca interfață de comunicare între REV și iPhone. iPhone-ul detectează tipul deșeurii și face un request către un server web care rulează pe Arduino.

Comunicarea dintre REV și iPhone este facilitată de către Arduino, prin 4 pini (Analog pe REV Control HUB, AnalogWrite pe Arduino).



Am conectat pinii din Arduino, direct în REV, fără a mai fi nevoie de masă, deoarece Arduino-ul se alimentează prin USB la REV și există o masă comună. În ceea ce privește senzorii, avem senzorul de distanță care este poziționat în gheara robotului pentru a putea vedea distanța necesară pentru ca robotul să aplice

deșeul. Acesta este un senzor de distanță REV 2m Distance Sensor, care comunică cu REV-ul pe interfața I2C.

Software

Mențiuni pentru cod și git

1. Pe REV am încărcat tot SDK-ul FTC, alături de codul de pe roboții trecuți. Codul de pe EcoRobot se găsește aici: [TeamCode/src/main/java/org/firstinspires/ftc/teamcode/drive/ConcecpEcoRobo4.java](https://github.com/firstinspires/ftc/teamcode/drive/ConcecpEcoRobo4.java)
2. Limbaje de programare & descriere generală

Pentru aplicația de detecție de pe iPhone folosim Swift - limbajul de programare dezvoltat de Apple pentru aplicațiile de pe iOS și macOS. Am antrenat un model CoreML cu imagini cu sticle, doze și sticle de plastic. Aceste imagini le-am luat în prima etapă de pe internet, iar după aceea am mai antrenat un model, la care am adăugat imagini făcute de noi.

Considerăm că al doilea model funcționează bine cu sticlele și dozele pe care am testat.

Pentru Arduino, folosim ArduinolDE, ca să încărcăm codul, pentru a crea serverul pe rețea.

REV Control Hub este un controler care are multe module incorporate: Driver de motoare, de servomotoare input-uri /output-uri digitale și analogice pentru server. Acest controler este folosit în competiția First Tech Challenge, și este ușor de lucrat cu el. Pentru a scrie programele folosim Android Studio, iar pentru a le încărca pe REV folosim ADB (Android Debug Bridge).

Programele sunt scrise în Java, folosind în principal SDK-ul oficial FTC, care ne oferă documentația necesară și ne permite să conectăm o gamă largă de piese la un singur Controller. Tot din comunitatea FTC, folosim Road Runner, un tool care permite mișcarea precisă a robotului din controler în modul în care este condus.

Algoritm pentru colectarea deșeurilor

Robotul se deplasează spre dreapta până când senzorul de distanță detectează un obiect aflat în apropiere. Apoi robotul începe să înainteze pentru a lua obiectul. Aceasta merge în funcție de distanță indicată de senzorul de distanță. Apoi robotul prinde deșeul și îl colectează.

Pentru mișcarea înapoi (revenirea după colectarea deșeului) folosim o metodă creată de noi. Care preia o distanță în cm și o transformă în tick-uri (unitatea de măsură care măsoară rotațiile motorului).

```
public void EncoderPower(double dist, double power, char heading)
{
    // 1 thick = 1/537.5 * 1 Motor Rotation
    // Wheel Diameter = 9.6 cm
    // Pi = 3.1415
    int tick1 = (int) (dist*537.7/9.6/3.1415);
```

Apoi această metodă calculează și transmite poziția la care motoarele trebuie să ajungă.

Pentru mișcarea laterală, pentru a găsi un deșeu, și pentru mișcarea înainte pentru a colecta acel deșeu, folosim o altă metodă, tot creată de noi: Power_Heading.

Iar pentru a apuca deșeul folosim metoda Deșeu(), care aduce robotul la o distanță de 4 cm de deșeu, iar apoi se deplasează încet către acesta.

Algoritm pentru sortare selectivă

În acest timp, telefonul procesează fiecare cadru și cauță obiectele din model. În momentul când observă ceva, acesta face un request către Arduino, în care trimitе ce a recunoscut în imagine. Apoi arduino-ul scrie pe pinul corespunzător tipului de deșeu detectat.

Pe iOS, după ce s-a încărcat View-ul (metoda viewDidLoad() a clasei UIViewController) inițializăm o sesiune de captură a unui feed, alegem camera, apoi configurăm Output-ul.

La fiecare Output, încercăm să detectăm pe imagine un obiect. din modelul de CoreML.

Am creat un struct pentru a-l folosi cu recunoașterile, pentru a putea afișa log-uri mai concise, deoarece informațiile se aflau la două nivele ierarhice diferite în structura obiectului (ne referim la Structura Recognition).

Am mai creat o structură (ObjectHandler) care să formeze request-ul către Arduino. Aici transformăm textul în cifre, care urmează apoi să fie scrise pe pinii Analog ai Arduino-ului. Folosim un Pod, SwiftHTTP pentru a face request-ul. (Screenshot dintr-o versiune mai veche).

În ultima versiune am optimizat aplicația, astfel încât

```
struct ObjectHandler {
    func handle(_ obj: Recognition){
        var midXIsOk: Int {
            get {
                if obj.box.midX < 0.35 && obj.box.midX > 0.25 {
                    return 255
                } else {
                    return 0
                }
            }
        }

        var doza = 0
        var sticla = 0
        var plastic = 0

        switch obj.name {
        case "doza":
            doza = 255
        case "sticla":
            sticla = 255
        case "plastic":
            plastic = 255
        default:
            print("no")
        }

        let ip = "192.168.0.140"
        let ip = "192.168.0.176"

        let link = "http://\((ip))/recognized?d=\(doza)&s=\(sticla)&p=\(plastic)&det=\(midXIsOk)"

        print(link)
        HTTP.GET(link)
    }
}
```

aceasta să transmită doar când recunoaște un alt tip de obiect.

În momentul în care, senzorul de distanță detectează un obiect, robotul citește toți pinii analog iar în funcție de semnalul transmis acesta

setează poziția coșului. Datorită unghiului camerei, telefonul vede prima dată obiectul, apoi senzorul, deci în momentul în care senzorul a detectat deșeful, pe pinii arduino-ului, este deja scris tipul acestuia.

Pe partea de Arduino, se efectuează o conexiune la Wifi, apoi se initializează un server Web. Pentru fiecare request, se transformă parametrii transmiși în Numere, pentru a putea fi apoi scriși pe pini.

```
httpServer.on("/recognized", []() {
    float doza = httpServer.arg(0).toFloat();
    float sticla = httpServer.arg(1).toFloat();
    float plastic = httpServer.arg(2).toFloat();
    float detected = httpServer.arg(3).toFloat();
    //float pickup = httpServer.arg(4).toFloat();

    digitalWrite(DOZA, doza);
    digitalWrite(STICLA, sticla);
    digitalWrite(PLASTIC, plastic);
    digitalWrite(DETECTED, detected);
```



Modelul în CoreML

CreateML este un Tool gratis oferit de către Apple, pentru dezvoltatorii de aplicații. Acesta poate genera un model (noi folosim object detection) cu imaginile date. Dar în acest model, Apple aduce și toate informațiile colectate de ei, deci este o fuziune dintre inteligența Apple (aceeași inteligență care te ajută să cauți prin galerie) și imaginile noastre.

Pentru object detection, datele de intrare în CoreML sunt imaginile, și un fișier annotations.json, care conține un vector de obiecte cu acest format (numele imaginii, adnotarea, cu eticheta (numele obiectului recunoscut) și coordonatele pentru localizarea acestuia în imagine).

Pentru a genera acel fișier json, am folosit un tool în python, care ne oferă o interfață vizuală pentru a selecta obiectul în imagine. Acest tool este open-source, modificat de noi, pentru a optimiza partea de labeling.

Pe pagina următoare se pot vedea datele din training, date care sunt normale pentru antrenarea unui model. Un Loss care scade, sub 0, și ajunge în final, la iterația 5000 la 0,73.

Am avut mai multe etape de dezvoltare ale acestui model, în paralel cu dezvoltarea hardware a robotului. În prima etapă am antrenat modelul pe un set de 90 de imagini (30 din fiecare categorie).

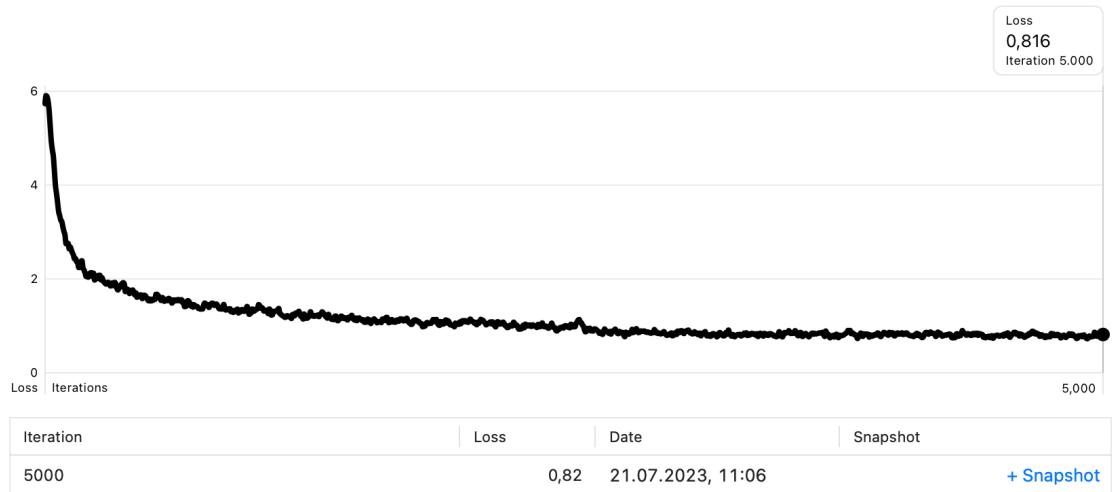
Pentru partea de testare, am decis să nu folosim tool-ul de testing din CreateML, deoarece, nu se aplică pe robotul nostru, care preia mai multe cadre (din mai multe unghiuri în funcție de deplasare), recunoașterea unui obiect fiind necesară doar pe unul din acele cadre.

Ca rezultate a primului model, am obținut o acuratețe foarte mare pe doză: Robotul a detectat-o corect în toate cazurile, dar o lipsă a recunoașterii sticlelor (atât de sticlă, cât și de plastic). Deși, foarte rar când recunoștea, era detecția era corectă.

În al doilea model am crescut numărul imaginilor la 50 (doze), 60-70 (sticlă etică și sticlă plastic). În teste izolate (numai cu log-urile de pe telefon, am observat că telefonul detectează și clasifică corect toate elementele).

Următorul pas este să creștem capacitatea modelului prin imagini cu doze noi, alte firme. Sticlele de plastic și sticlele de sticlă sunt în principiu recunoscute, indiferent de firmă.

În următoarele 3 modele (3, 3.1, 3.2) am adăugat alte imagini. Am decis să le versionăm în acest mod, deoarece în fiecare dintre subversiuni am adăugat doar un tip de sticlă respectiv plastic. Deja la finalul acestei etape am folosit aproximativ 100 de imagini pentru fiecare categorie. Pentru antrenare am mers pâna la iterația sugerată de Apple (5.000), iar la modelele anterioare 4000 respectiv la primul model 3000.



Utilitate

Proiectul nostru este optimizat pentru a colecta deșeuri pe terenul oficial FTC, dar poate să colecteze deșeuri pe orice suprafață plană.

Scopul robotului nostru este să colecteze selectiv, ceea ce poate fi aplicat în multe moduri în spații cu suprafete plane (în școli, pe stradă, în zone pietonale).

Totodată un asemenea robot poate să aibă ca scop educarea oamenilor în sprijinul stângelor deșeurilor și ajutarea lor în a conștientiza problema globală a poluării.

Proiectul acesta reprezintă o oportunitate de învățare, prin care am experimentat și combinat tehnologii noi. Noi credem că unul din scopurile acestui proiect poate fi chiar recreerea lui, cu scop didactic, prin care elevii pot experimenta, și la rândul lor să-și dezvolte propriile proiecte.

Posibilități de avansare a proiectului

Noi am creat acest proiect care combină mai multe tehnologii, spre acest produs final. Dar robotul are 4 compartimente, ceea ce înseamnă că poate fi adăugat un tip de deșeu.

Totodată, cu o schimbare din punct de vedere hardware, robotul poate fi transformat într-un coș de gunoi cu colectare selectivă, în care oamenii să

aduce deșeuri, iar coșul să le sorteze. Totuși noi am ales această formă pentru a fi mai interactiv cu cei care se uită la proiectul nostru.

Acest proiect este 100% open-source, pentru a facilita accesul persoanelor interesate, spre a ne continua munca, deoarece noi suntem clasa a 12-a iar la anul, focusul nostru se va schimba. Dar EcoRobot este un proiect la care abia așteptăm să revenim după examenul de Bacalaureat, spre a-l dezvolta în continuare.