Alumne	NIU	Tutor
Ricard López Olivares	1571136	Carlos García Calvo

Informe progrés II – 17 de Maig de 2023

Nota: **Els canvis afegits respecte al document "Informe progrés I" es mostraran del mateix format que aquest text, a excepció dels canvis en la planificació.**

Robot Industrial Stäubli TX60 com a classificador de residus

Resum

El projecte es centra en desenvolupar un robot de reciclatge que utilitza tecnologia de xarxes neuronals i està integrat en el marc de la Indústria 4.0. El robot està dissenyat per identificar i classificar de manera autònoma materials reciclables de corrents de deixalles, com ara plàstics, vidres, papers i matèria orgànica.

La tecnologia de xarxes neuronals permet al robot aprendre i adaptar-se a noves corrents de deixalles, així com millorar la seva precisió i eficiència amb el temps. La integració de la Indústria 4.0 proporciona monitoratge i anàlisi de dades en temps real, el que permet l'optimització del procés de reciclatge i la identificació de possibles problemes.

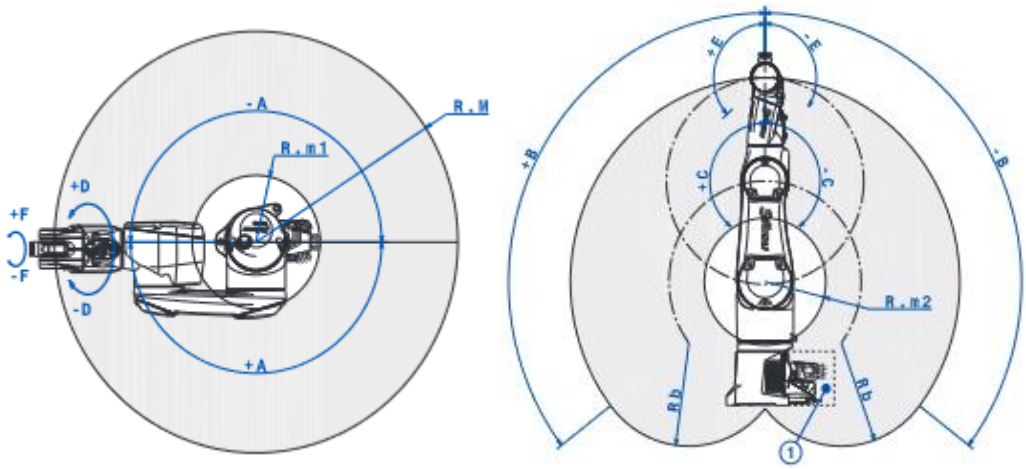
El projecte té com a objectiu millorar l'eficiència i sostenibilitat de les operacions de reciclatge, alhora que redueix la necessitat de mà d'obra humana en tasques perilloses o repetitives. També s'alinea amb els principis d'una economia circular, en la qual es minimitza el malbaratament i es reutilitzen els recursos.

En general, el robot de reciclatge amb tecnologia de xarxes neuronals integrada en la Indústria 4.0 representa una solució innovadora als desafiaments de la gestió de residus i la sostenibilitat, com a inspiració podem trobar el robot de l'empresa Glasier[1].

Especificacions i dataset

Els elements utilitzats per aquest projecte són:

- Robot Stäubli model TX-60[2]. **A l'hora d'escollir un robot s'ha de tenir en compte dos punts importants, aquest son els següents:**
    - **Àrea de treball: l'àrea de treball d'un robot són tots aquells punts on el treball pot arribar sense cap mena de problemes. En aquest cas l'àrea de treball del TX-60 és una esfera amb un forat en el centre per no permetre la col·lisió amb si mateix.**



- **Característiques del robot[16]: Les característiques són importants per veure el pes màxim i nominal que pot agafar, nombre d'eixos, informacions dels eixos, etc. Com bé he especificat abans he utilitzat el robot Stäubli TX60, aquest robot s'utilitza per aplicacions remotes (ja que te connexions per xarxa), per fer soldadures (és capaç de fer línies rectes sense cap mena de problema), dispensador (és un robot amb una repetibilitat bastant baixa, el qual el fa bo per realitzar accions repetitives). A continuació, es pot veure la taula amb totes les característiques d'aquest robot.**

Marca	Säubli
Model	TX60
Tipus	Braç robot (braç antropomòrfic)
Axes (joins)	6
Carrega	3.5 kg
Eixos	600 mm
Repetibilitat	0.02 mm
Pes	51 kg

- Controlador CS8C[3], **des d'aquest controlador podem controlar el robot manualment amb un comandament.**

- **Simulador, he utilitzat el simulador Stäubli robotics suite 2022[5]. Aquest software ens permet treballar únicament amb els robots de l'empresa Stäubli, però d'una manera fàcil, amigable i el més real possible, ja que es poden simular tots els components del robot i entrades del controlador per fer una simulació el més pròxim a la realitat possible, també ens permet programar des de la posada en marxa del robot fins a l'aturada, optimització del rendiment, detecció i previsió de col·lisions, calibració, entre molts altres avantatges.**

Aquest simulador utilitza el mateix llenguatge que el robot real anomenat VAL 3 [12], d'aquesta manera es pot fer un port directe sense tenir que transcriure el codi.

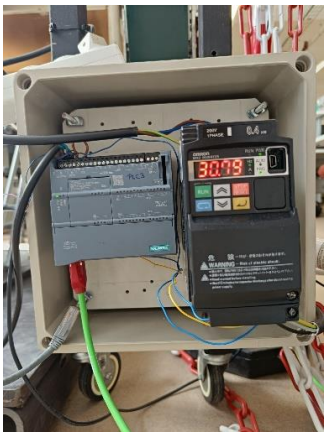
- Cinta transportadora amb encoder, així es pot fer un tracking (seguiments d'objectes) per agafar els objectes sense aturar la cinta.

- PLC 1200 **AC/DC/R4** Siemens amb un variador de velocitat de la marca Omron **MX2-A2002**, d'aquesta manera es pot controlar la parada i la velocitat de la cinta.

  Aquest dos dispositius es poden definir de la següent forma:

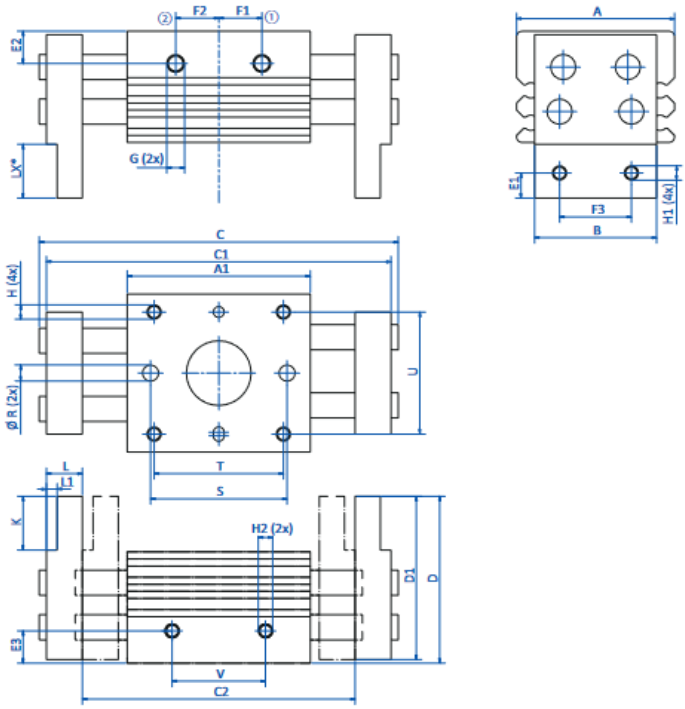
  Un PLC és un dispositiu electrònic utilitzat en l'automatització industrial per a controlar i supervisar altres sistemes i processos a temps real. Aquest tenen uns seguits d'entrades i sortides, poden ser digitals o analògiques, que es poden configurar per rebre informació de sensors o dispositius externs. Hi ha diferents estils de llenguatges per a un PLC, en aquest cas he fet servir un llenguatge de programació en blocs funcionals amb el programa TIA PORTAL[17].

  Per altra banda, tenim el variador de velocitat Omron, aquest dispositiu ens permet modificar la velocitat d'un motor trifàsic que dona un rang de potència d'entre 0.4 KW a 15 KW.



- **Telèfon mòbil o càmera amb tecnologia wifi**, per poder implementar la visió per computador.

- **L'eina utilitzada és un gripper[14] model K87703869, d'aquesta manera es pot agafar els objectes. Un gripper és una eina mecànica que es posa en l'extrem del robot i d'aquesta manera poder agafar i soltar objectes, aquest depenent de la tasca a realitzar poden tenir uns dits, pinces o ventoses, en aquest projecte s'ha dissenyat unes estructures que s'acoblen al gripper i obtenir una pinça. D'igual manera per obrir i tancar la pinça funciona amb aire, per tant, en un mecanisme pneumàtic.**
  A continuació, es pot veure les dimensions del gripper.



A [mm]	A1 [mm]	B [mm]	C [mm]	C1 [mm]	C2 [mm]	D [mm]	D1 [mm]	E1 [mm]	E2 [mm]	E3 [mm]	F1 [mm]	F2 [mm]
44	67	34	142	138	98	46	45.5	7	9	9	19.5	19.5

F3 [mm]	G [mm]	H [mm]	H1 [mm]	H2 [mm]	K [mm]	L [mm]	L1 [mm]	R [mm]	S [mm]	T [mm]	U [mm]	V [mm]
20	M5	M4	M4	M4	15	10	3	4.5	54	52	34	42

- La xarxa neuronal que he utilitzat és DenseNet[8], he decidit aquesta opció, pel fet que Yolo[7] i SSD[9] són xarxes neurals ja entrenades que diuen de quin objecte es tracta, quan en aquest cas interessa classificar l'objecte no dir quin és.

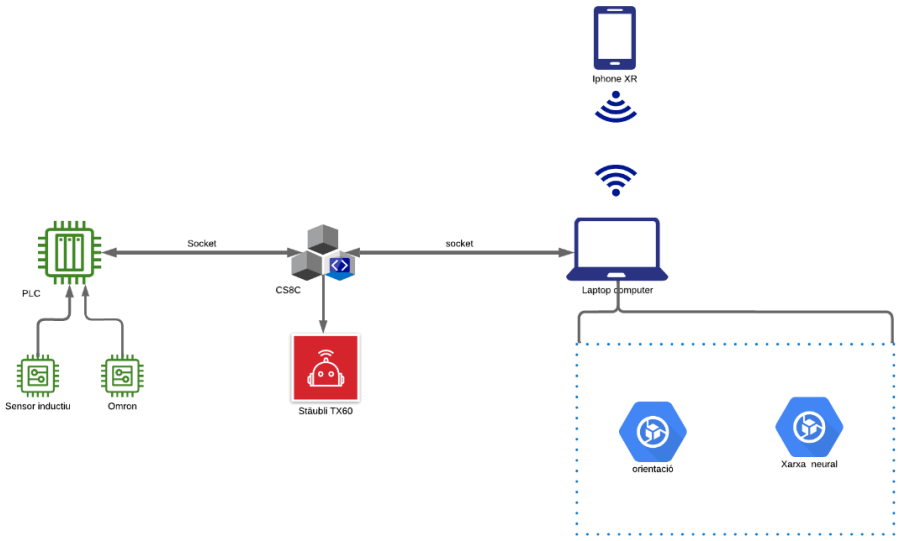
- Utilització d'un dataset, aquest dataset és una combinació d'imatges pròpies i d'un dataset ja creat amb productes de reciclatge, anomenat trashnet[6]. Aquest dataset consta de 4 classes amb el nom dels colors dels contenidors i dintre conté imatges dels objectes que podem contenir cada contenidor.

  Exemples de les imatges empleades en el dataset, part de train.

- **Una part important del projecte es la connexió entre els dispositius, aquestes connexions son important per enviar dades entre ells i fer accions depenen d'aquestes, en aquesta projecte he utilitzat 2 tipus de connexions, aquestes son:**
    - **Sockets: Aquestes ens permet comunicar processos o programes a través de la xarxa, d'aquesta forma obrir un flux bidireccional entre programes. Per implantar aquesta tecnologia ha d'haver-hi un servidor (és qui respondrà a les peticions) i un client (envia les peticions al servidor): Per afinar una mica més utilitzem un socket TCP, d'aquesta manera es pot dir que les dades són l'esperada en cada moment.**

      La tecnologia la podem veure en dos llocs del projecte, la primera és de l'ordinador a la controladora del robot (CS8C) i la segona es troba en el PLC a la controladora (CS8C).

    - **WiFi IvCAM: És una tecnologia sense fils que s'utilitza per a la transferència de dades entre una càmera i un dispositiu. Aquesta connexió es troba entre L'ordinador i el telèfon mòbil, amb aquesta connexió podem enviar senyals al telèfon per realitzar la fotografia, canviar la configuració de la càmera, etc.**



Objectius

Codi	Definició	Completat	Referencies
O1	Obtenir informació important sobre el robot TX-60 i simulador.	**Si**	[2, 4, 6]
O2	Realitzar la identificació d'objectes.	**Si**	[6,7,8,9]
O3	Buscar informació sobre la programació del robot i el simulador.	**Si**	
O4	Crear una simulació realista del projecte.	**Si**	
O5	Realitzar que la simulació funcioni en el món real i faci el circuit (agafar de la cinta, traslladar-ho al lloc).	**Si**	
O6	Manipulació d'un objecte estàtic, en simulació com en espai real.	**Si**	
O7	Realització d'agafar un objecte no estàtic.	**En procés**	
O8	Comunicar un ordinador amb el robot, per poder enviar-li accions a realitzar.	**Si**	
O9	**Utilització d'una pinça pneumàtica**	**En procés**	**[14]**
O10	**Creació de les extensions de la pinça per poder agafar objectes.**	**Si**	

Després d'un temps treballant en el projecte no s'han modificat els objectius, sinó tot el contrari, s'han augmentat, aquest objectius s'han afegir perquè no aporten un gran canvi en el projecte, ja que nomes es canviar l'eina en el simulador i en el robot físic . L'objectiu final segueix sent el mateix.

Metodologia

En aquesta secció s'explica quina és la manera de treball que s'utilitza en el projecte.
Hi haurà dos tipus de metodologies, una es fa amb el tutor i l'altre la fa l'alumne independentment per aconseguir una millor organització. La primera es fa amb el tutor, tracta de realitzar una sèrie de reunions, normalment cada setmana es fa mínim una, aquestes reunions serveixen per fer un seguiment, explicar conceptes que són útils, intercanviar punts de vista, etc. No només es faran aquestes reunions per fer un seguiment del treball, sinó que també són per poder fer servir el robot i fer les proves pertinents amb una persona que vigili el que es fa, això pel fet que pot ser perillós un robot. Per la segona, la qual només és responsable l'estudiant, és l'ús d'una metodologia àgil, anomenada kanban[10], aquest consta d'un taulell amb uns pòsits que es van canviant de posicions.

Aquesta secció, no és modifica, les reunions setmanals es mantenen per tenir un control del projecte més acurat, aclarir idees i fer proves en els robots reals si es cal. D'igual manera l'alumne manté la metodologia Kanban.

Planificació

En aquesta secció es realitza un esquema de seguiment estimat per a la realització del projecte. Cal destacar que l'esquema és orientatiu i ajustat al temps atorgat per dur a terme el projecte, per tant, alguna fase pot durar més temps del marcat o es faci abans del previst.

Per facilitar la tasca de seguiment, es posaran colors a les fases en funció al progres y desenvolupament actual del projecte.

Llegenda

- Verd : Tasca completada.
- Groc: Tasca en curs.
- Vermell: Tasca aturada.
- Blanc: Tasca pendent.

Planificació			
Fase	**Descripció**	**Objectius**	**Temps aprox.**
Reunió Inicial (19/02/2023)			
1	Definició detallada del projecte, objectius, metodologia i planificació. Redacta informe inicial.	-	1 setmana
2	Començar planificació del projecte.	O1	1 setmana
3	Començar a familiaritzar-se amb el simulador i la programació del robot.	O3	1 setmana
Lliurament informe Inicial (12/03/2023)			
4	Crear l'escena en el simulador, amb peces de colors i posar-ho en marxa.	O4	1 setmana
5	Instal·lar un sistema de visió en el robot i que es comuniqui amb l'ordinador.	O6	1,5 setmanes
6	Entrenar la xarxa neural per poder identificar els objectes a reciclar i classificar-los.	O2	2 setmanes
7	Implementar la comunicació ordinador – robot Stäubli	O8	1 setmana
8	Ampliar l'informe amb tots els avanços realitzats.	-	1 setmana
Lliurament informe de progrés I (23/04/2023)			

9	Manipular objectes estàtics en la simulació i classificar-los.	O6	2 setmanes
10	Passar la simulació a un àmbit real, utilitzant el robot Stäubli (agafar peça de la cinta, classificar-la i traslladar-la).	O5	2 setmanes
11	Obtenir la orientació de la peça	O7	1 setmana
12	**Crear els models 3D de les extensions per la pinça e imprimir-los**	**O10**	**1 setmana**
Lliurament informe de progrés II (28/05/2023)			
13	Unificació de tot el codi, per crear una única aplicació	-	1 setmana
14	Implementar el poder agafar una peça sense parar la cinta, mantenint la classificació d'objectes	O7	1,5 setmanes
15	Redactar el document final	-	4 dies
Lliurament informe final (18/06/2023)			
16	Crear la presentació i defensa.	-	5 dies
17	Acabar de polir el projecte corregint errors o millorant la intel·ligència artificial.	-	1 setmana
Proposta de presentació (30/06/2023)			
18	Creació del pòster	-	3 dies
Lliurament dossier (06/07/2023) **Lliurament pòster (06/07/2023)**			

FASE 1:

DESENVOLUPAMENT

La definició del projecte és la següent: Es realitzarà la visió per computador d'un robot Stäubli model TX-60, la qual ajuda al robot a ser capaç de classificar residus. A part d'implementar-ho de manera física en el robot real, s'implementarà una simulació el més semblant a la realitat.

Amb aquest afegit, el robot hauria de ser capaç d'identificar un, objecte classificar-ho i posar-ho en el lloc pertinent.

FASE 2:

DESENVOLUPAMENT

La planificació està detallada en el punt anterior, aquest es va actualitzant en cada entrega: per mantenir aquest punt sempre present i no perdre els avanços realitzats s'utilitza l'eina github, en un repositori propi anomenat recycling-robot[13].

FASE 3:

DESENVOLUPAMENT

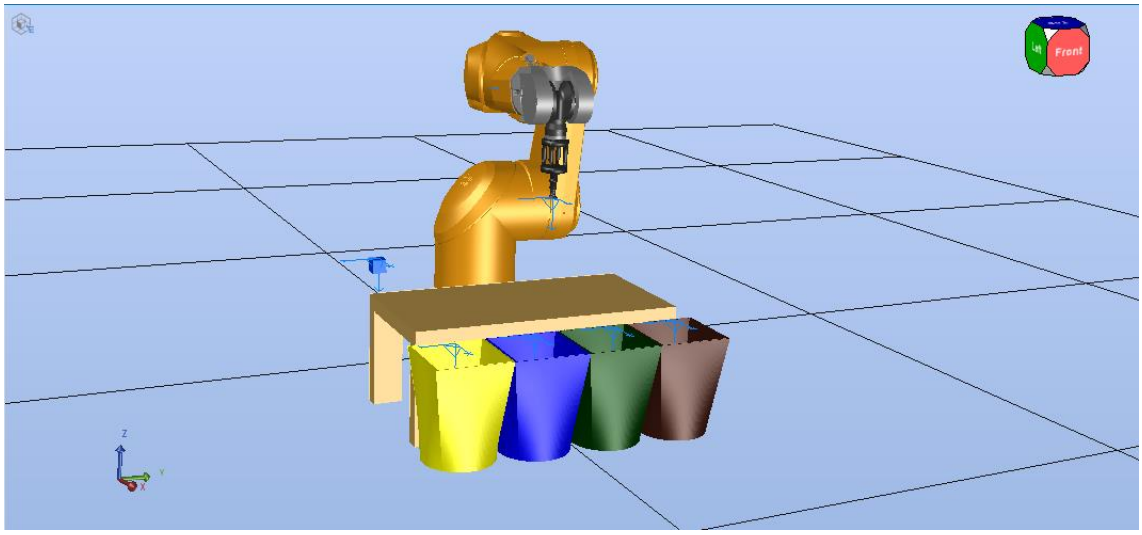
Creació de diversos programes de val3 en Stäubli Robotic Suite 2016, aquest software va ser proporcionat pel professor mentre ens proveïen la versió que s'utilitza avui dia (Stäubli Robotic Suite 2022). **Els programes realitzats en aquesta fase són programes senzills per acostumar-se a moure objectes, moure el robot, etc.**

FASE 4:

DESENVOLUPAMENT

Donat amb el que s'ha practicat s'ha creat una escena, la qual conté un objecte estàtic i posar-ho en una paperera. **Com que en el simulador no hi ha gravetat en els objectes, les peces es quedaran flotant sobre la paperera. Això no és un problema, ja que encara que no hi hagi gravetat, es pot eliminar la col·lisió entre objectes i les peces es quedaran superposades.**

RESULTAT



CONCLUSIONS

D'aquesta fase podem concloure que a pesar que no estigui finalitzada, pel fet que falta una cinta, no ens portarà un retard en el projecte, ja que es pot treballar sense problemes.

FASE 5:

DESENVOLUPAMENT

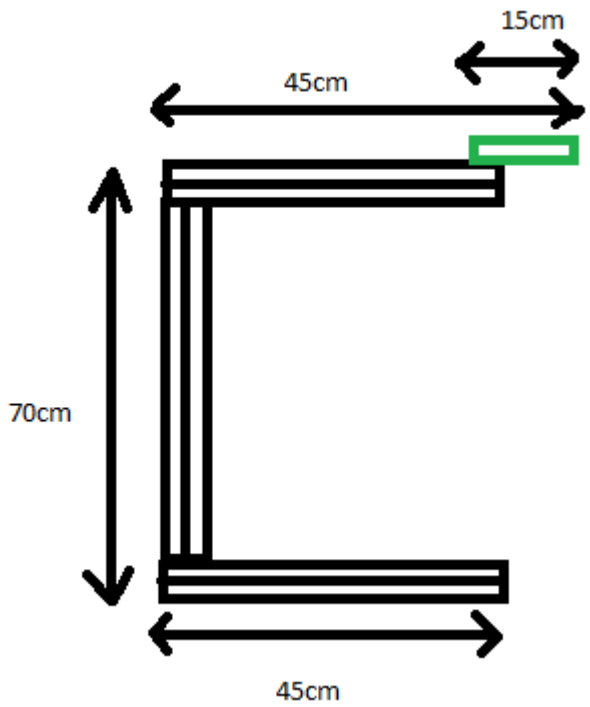
El sistema de visió implementat, consta de dues parts, una part que li anomenaré estructura i una altra anomenada software:

Estructura:

Tal com diu el nom, aquest apartat ve originat per la part física, en la qual també hi ha dues parts:

- **La primera és una estructura metàl·lica unida a la cinta, amb un suport per posar el telèfon mòbil. La unió no es fixa, per tant, l'estructura es pot acoblar i desacoblar a la mateixa cinta.**
- **La segona part constaria del telèfon mòbil, aquest telèfon és un Iphone XR, propi de l'alumne. S'ha optat per aquesta, ja que no teníem una càmera wifi, per tant, vam pensar a utilitzar, el telèfon mòbil, pel fet que no era gaire complex connectar-ho a l'ordinador i sempre està disponible. Si mirem les característiques, de la càmera tresarà del telèfon[17], són: càmera gran angular de 12 MPx, obertura de f/1.8 i un angle de visió de 120º i una profunditat de camp de f=16, aquestes dades venen per defecte en l'apartat de fotografia de la càmera i no es poden modificar a no ser que canviem de mode, però per aquest projecte no caldria canviar-ho.**

S'ha optat per la següent forma i mides d'aquesta manera el robot no pot col·lisionar amb l'estructura i no serà un impediment a l'hora de moure.



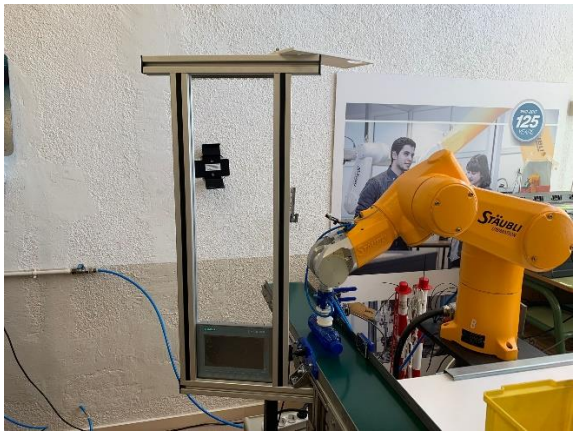
Software

En el apartat de software s'ha utilitzat el programa IvCAM, el qual ens permet la connexió de càmeres externes al nostre dispositiu mitjançant la xarxa wifi. Aquest software es molt simple d'utilitzar només s'ha de descarregar en els dos dispositius i unir-los, sempre han d'estar en la mateixa xarxa els dos dispositius. Un cop tenim "emulada" la càmera, des de python podem dir que realitzi la fotografia i desar-la en el ordinador.

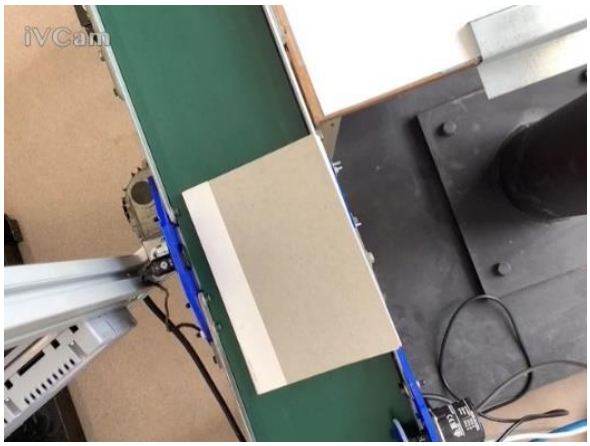
Per a que la fotografia sigui el mes nítida possible i evitar problemes en un futur, he creat un ambient "ideal", aquest ambient tracta de difuminar la llum la llum que entra per la finestra posant-hi un teixit blanc, d'aquesta manera la llum que hi entra és una llum més suau i no crear reflexos en les superfícies llices i s'evita les possibles ombres que es puguin generar.

<u>RESULTAT</u>

Estructura



Software



CONCLUSIONS

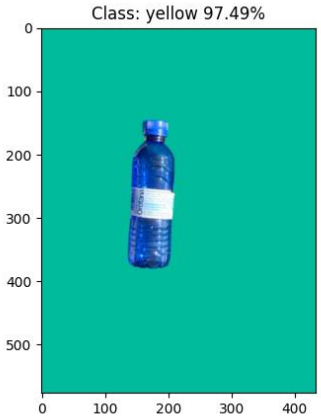
Aquesta fase ha sigut tot un èxit, ja que l'estructura és bastant resistent, però també és bastant versàtil a l'hora de moure-la o posicionar-la de diferent manera, la càmera al ser d'un dispositiu mòbil podem ajustar-ho en cas que sigui necessari com fer un zoom, etc.

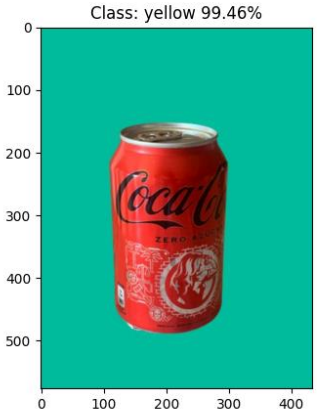
FASE 6:

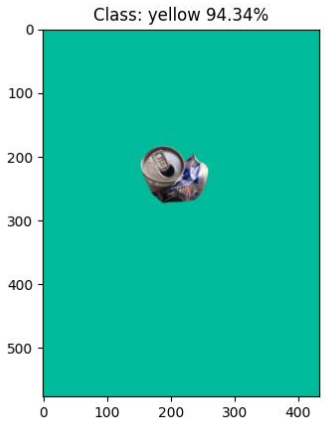
DESENVOLUPAMENT

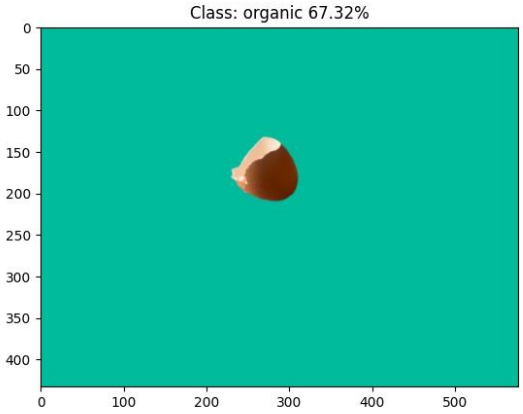
S'ha realitzat una xarxa neural DenseNet121, aquest tracta d'una xara convolucional CNN de 121 capes densament connectades, aquest model té com a entrades les imatges amb l'objecte a classificar i retorna el color del contenidor que es classifica amb un percentatge de seguretat, com es pot veure a continuació s'han fet diverses proves donants bons resultats, aquestes proves s'han fet en diferents àmbits i objectes.

RESULTATS









CONCLUSIONS

Podem veure que aquesta fase és un èxit, la xarxa neuronal funciona sense cap mena de problemes, per arribar a aquesta conclusió, he fet dos tipus de proves:

La primera són proves bastant controlades, les quals consten en agafar imatges d'objectes que estan fora del dataset amb un fons uniforme, és a dir, només hi ha l'objecte en la imatge que ens interessa, els resultats de les proves ho podem veure en l'apartat de resultat d'aquesta fase. Obviament aquestes proves tenen un accuracy, molt més alt, comparat amb les proves realitzades posteriorment, això com que només hi ha un objecte en la pantalla i el fons al ser tot uniforme és menys probable que doni problemes.

La segona prova és en l'àmbit real, aquest àmbit real són els objectes en la cinta, capturats amb la càmera del telèfon, aquestes donen un accuracy més baix, però està sobre el 70-85% normalment, és una classificació bastant bona.

En resum podem dir que el dataset està bé triat, amb unes dimensions adequades per aquest projecte.

(Consultí l'annex A per veure el codi que s'ha utilitzat).

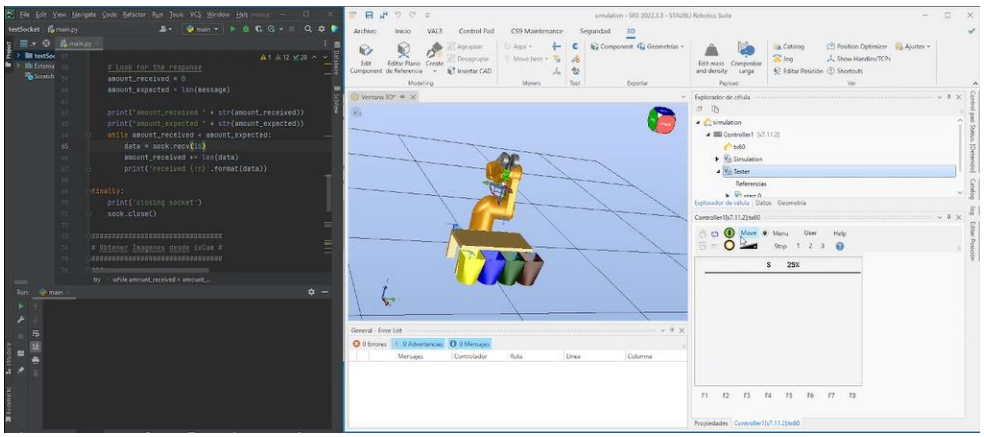
FASE 7:

DESENVOLUPAENT

Aquesta comunicació es fa amb socket, tant en la simulació com en el robot físic, aquesta consta amb una màquina client (ordinador) i un servidor (robot), aquesta informació s'envia/rep en bytes amb una codificació ASCII. No cal enviar un únic byte, es pot enviar un missatge sencer sense problemes, ja que es disposa d'un buffer, que es va omplint mitjançant arriben dades. Això s'utilitzarà en un futur per enviar a quin contenidor va l'objecte. Encara que per comprovar el funcionament s'han realitzat proves amb arxius de testatge creats per l'alumne.

(Consultí l'annex B per veure el codi que s'ha utilitzat).

RESULTAT



CONCLUSIONS

Aquesta fase es dona com a exitosa, ja que la comunicació funciona, en totes dues direccions tant per enviar i rebre dades.

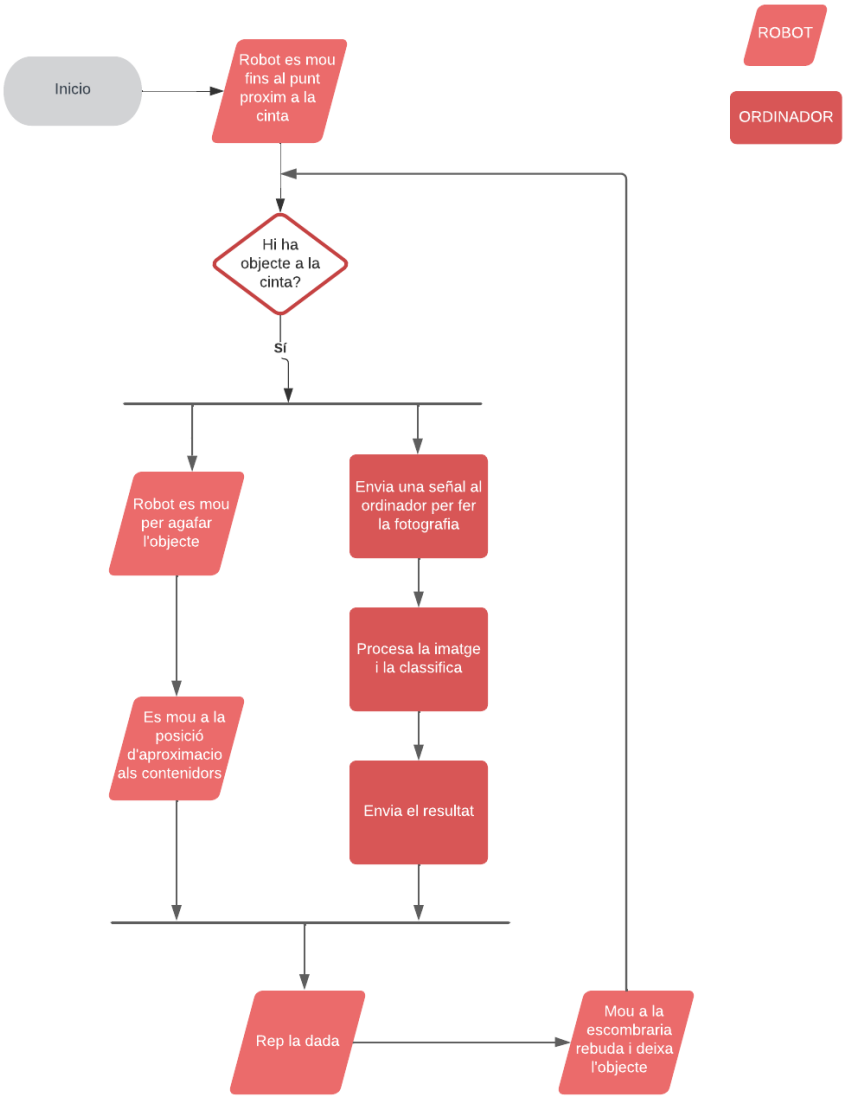
FASE 9:

DESENVOLUPAENT

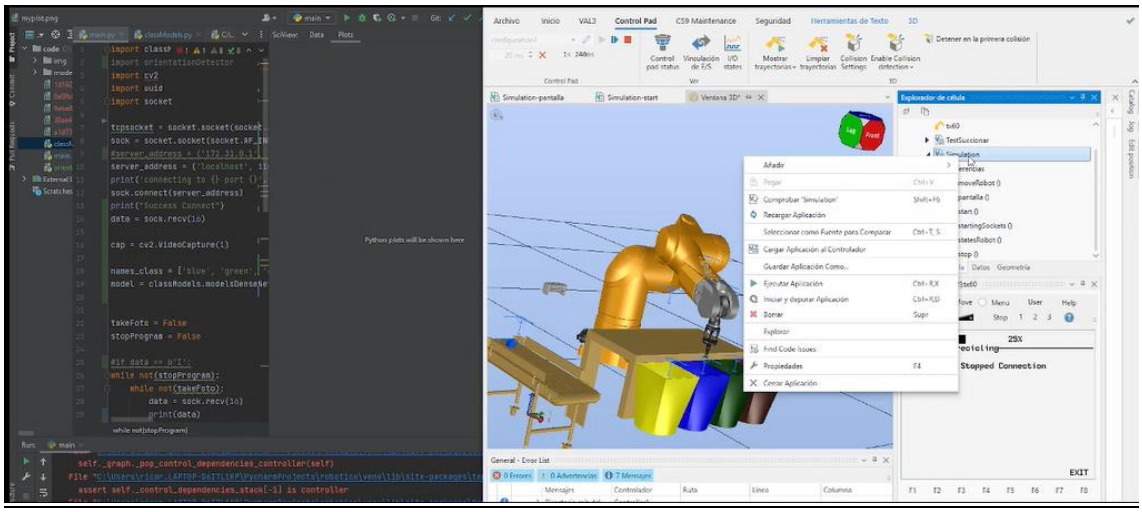
S'ha realitzat una simulació el més fiable a la realitat, és a dir, fa totes o pràcticament totes les fases que realitzarà el programa final. Com l'objectiu és que sigui el més real possible, no se simula cap dada, sinó que les dades que envia o rep són dades reals, és a dir, dades que enviaria el robot real si estigués en ús.

També cal mencionar que s'ha utilitzat les mateixes tecnologies que s'utilitzaran en el robot real com sockets per la comunicació entre robot i ordinador, IvCAM, càmera, etc.

El funcionament és el següent:



RESULTAT



CONCLUSIONS

La simulació ha estat exitosa, encara que no he pogut emular les dades que envia el PLC de la vida real, això només seria una connexió socket que rebi uns valors predefinits.

(Consultí l'annex C per veure el codi que s'ha utilitzat).

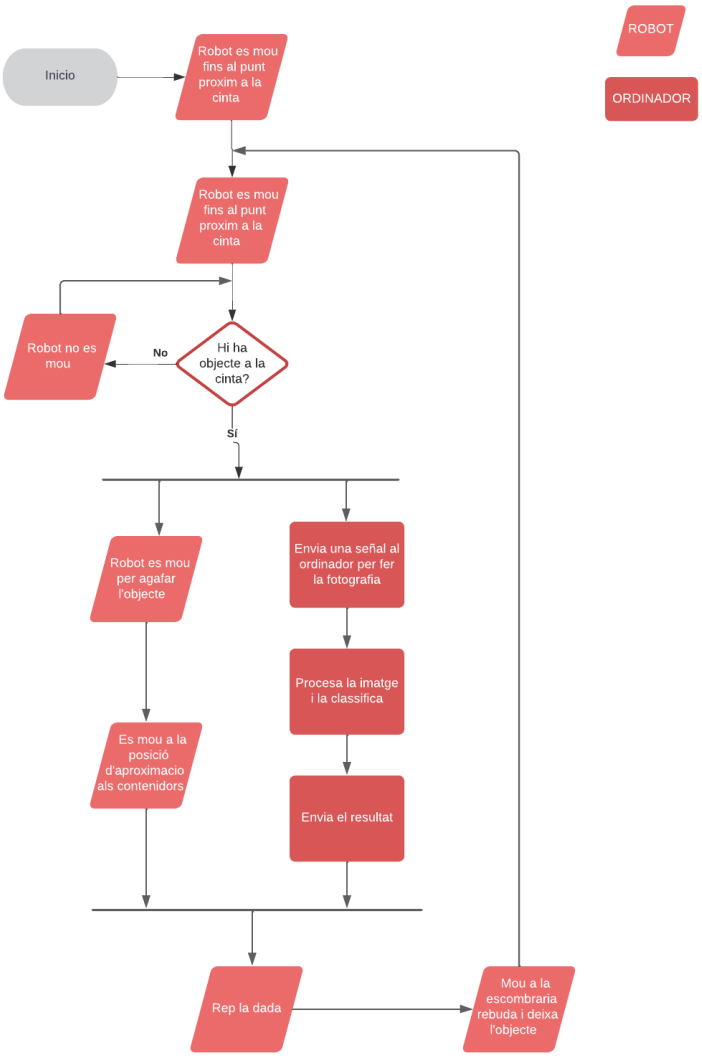
FASE 10:

<u>DESENVOLUPAENT</u>

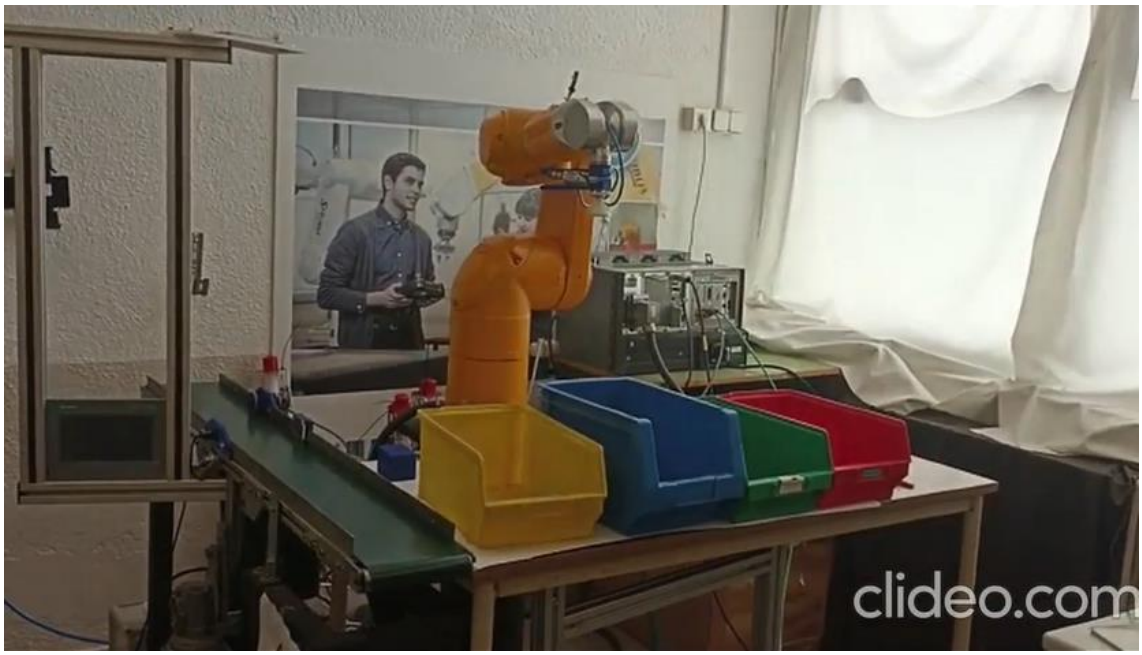
Un cop acabada la simulació s'ha procedit en passar-ho al món real, per aconseguir-ho el que s'ha fet ha sigut, agafar els fitxers de la simulació, posar-los en un USB i importar-ho en el CS8C del robot real.

El procediment és el mateix que la simulació, només que afegint l'obtenció de dades del PLC per dir si hi ha un objecte o no.

El flux seria de la següent manera:



RESULTAT



CONCLUSIONS

El port al robot real, ha sigut un gran èxit, no cal ni canviar els punts d'on para el robot, ja que es van agafar les distàncies, i alguns punts importants a l'hora de fer la simulació.

Una cosa que cal mencionar, és que en el simulador per agafar objectes està a la inversa les comandes.

(Consultí l'annex C per veure el codi que s'ha utilitzat).

FASE 11:

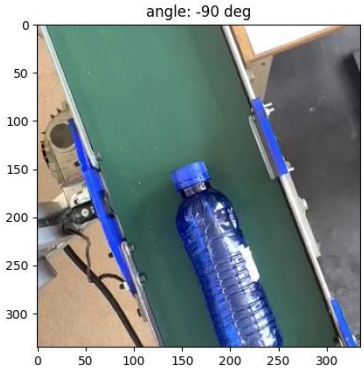
DESENVOLUPAENT

Al decidir un canvi d'eina, utilitzar una pinça en comptes d'una vàlvula de succió, he decidit crear un programari que em doni l'orientació de l'objecte que ha d'agafar el robot.

Aquest programari funciona amb la imatge, per tant, forma part del còmput i s'ha de fer en l'ordinador, també el podem fer en aquest dispositiu sense cap mena de canvi, només s'ha de cridar a les funcions pertinents, ja que es disposa de la imatge.

La idea és, intentar obtenir la direcció i després enviar-la per socket per orientar la pinça en la posició correcta.

RESULTAT



CONCLUSIONS

No podem dir si és viable tot això, sembla que el codi és bastant fiable, però no abans de dir res s'hauria de provar amb la pinça i veure el seu funcionament, d'aquesta manera es veurà si és fiable i es pot utilitzar o no.

En el test que he fet sembla que funciona correctament, però m'agradaria fer les proves pertinents abans de donar-lo per confluït.

(Consultí l'annex D per veure el codi que s'ha utilitzat).

FASE 12:

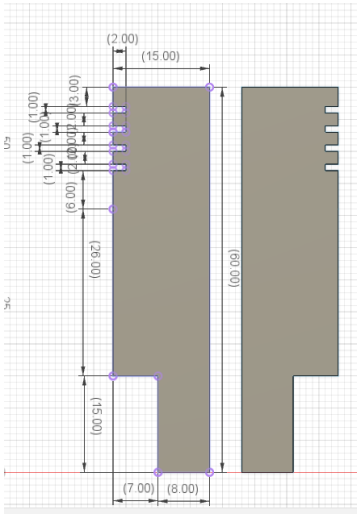
DESENVOLUPAENT

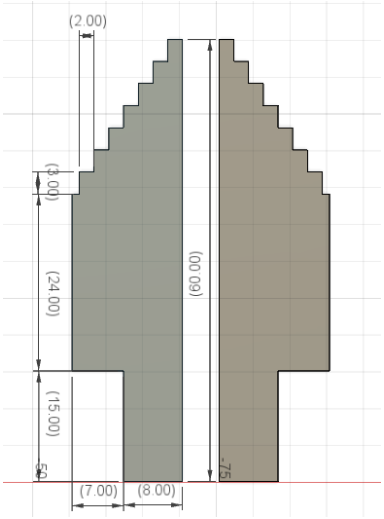
Aquesta fase s'ha d'afegir en aquest últim informe, això a causa del fet que es vol utilitzar una pinça, aquesta no té extincions per poder agafar elements, és només una "estructura". En veure aquesta carència he creat dues versions per l'extensió:

- **La primera és la més simple de les dues, és una estructura rectangular amb uns furats per poder posar algun element que impedeixi que rellisqui l'objecte, en cas de ser necessari.**
- **La segona és una mena d'escala, aquesta forma és intencional, ja que està pensada per agafar objectes de diferents dimensions amb una sola pinça.**

La dedició de fer servir la primera versió o la segona dependrà de la pinça, que encara està per arribar.

RESULTAT





Els dos models tenen una profunditat de 34 mm, també mencionar que les acotacions estoven en mil·límetres (mm)

CONCLUSIONS

En aquesta fase la podem donar per finalitzada, ja que la tasca en si és imprimir i planificar les extensions encara que no s'hagin testejat amb la pinça. Quan arribi la pinça veurem si funcionen com està previst i quina se selecciona.

FASE 13:

DESENVOLUPAENT

S'ha volgut unificar tot el codi en dues aplicacions, pel fet que vull un codi estructurat, entenedor i el més fàcil de llegir possible, per tant, el que he fet ha sigut unificar codi, eliminar fitxers, afegir arxius, eliminar funcions que siguin semblants, etc.

En el final s'han quedat en dues aplicacions, una és la que va al robot i l'altre va a l'ordinador.

La primera tracta de tot el moviment del robot, mostreig d'informació que va a la pantalla, creació de dades que necessita. El que s'ha fet ha sigut unificar diferents aplicacions independents en una, aquesta utilitza la paral·lelització de fitxers que té el robot, d'aquesta forma les aplicacions no han sofrat gaires canvis.

En la segona sí que hi ha hagut un canvi més dràstic, s'ha hagut de crear fitxers perquè estigui tot més organitzant, deixant l'arxiu base només per cridar a les funcions que ens calen

RESUM: El desenvolupament actual del projecte està en un bon punt, l'endarreriment que tenia s'ha compensat i sembla que es va avançat, però encara que sembli que va avançat s'ha de tenir en compte que encara no hi es la pinça per agafar objectes, si aquesta no arriba o s'atraça el projecte es veurà afectat i es tindrà que prendre la decisió si de seguir amb la vàlvula de succió o s'espera a la pinça.

Tampoc seria un gran impacte en el projecte, ja que la simulació està pensada amb les dues eines i es pot canviar fàcilment. En la vida real s'hauria de modificar els punts, però no és gaire treball fer-ho.

Estructurà i tecnologies empleades

En aquest apartat s'explica les tecnologies emprades en el projecte, separades per apartats.

Per al projecte utilitzaré python, amb l'IDE PyCharm, aquest em permetrà programa, executar codi i fer les proves de la xarxa neural utilitzada. En aquest àmbit també faré servir un llenguatge de Stäubli, anomenat VAL 3[12], per poder fer que el robot es mogui.

Per a la simulació, faré servir el programa "Stäubli robotics suite" proporcionat per la mateixa empresa.

A l'hora d'agafar la imatge via wifi, he utilitzat IvCAM[15], aquest és un programari que permet compartir càmeres de diferents dispositius amb l'ordinador.

Per acabar, cal mencionar que per realitzar la metodologia Kanban, faré servir l'eina online Trello[11].

Referències

[1] "Glacier." [Online]. Available: https://www.roboticsandinnovation.co.uk/news/funding/glacier-secures-us4-5m-for-ai-powered-recycling-robot.html

[2] "TS-60." [Online]. Available: https://robotsdoneright.com/Staubli/staubli-TX60.html

[3] "Controladora CS80." [Offiline]. Available: ./information/CS80_controller.pdf

[4] "TS2-60." [Online]. Available: https://www.staubli.com/content/dam/robotics/brochures/6-axis/datasheet/TX2-60-6-axis-product-data-sheet-EN.pdf

[5] "Staübli robotic suite." [Online]. Available: https://www.staubli.com/hk/en/robotics/products/robot-software/staeubli-robotics-suite.html

[6] "Trashnet." [Online]. Available: https://github.com/garythung/trashnet

[7] "YOLO v5." [Online]. Available: https://github.com/ultralytics/yolov5

[8] "DenseNet Keras." [Online]. Available: https://keras.io/api/applications/densenet/

[9] "SSD." [Online]. Available: https://github.com/balancap/SSD-Tensorflow

[10] "Kanban." [Online]. Available: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

[11] "Trello." [Online]. Available: https://trello.com/es

[12] "Val 3 language." [Online]. Available: https://usermanual.wiki/Document/val3referencemanual.275627616/view

[13] "Github propi" [Online]. Available: https://github.com/Ricardlol/Recycling-robot

[14] "Gripper." [Offiline]. Available: ./information/eoat-grippers-parallel-grippers-for-large-strokes-piston-diameter-10-20mm-staubli-en-FP01600301A.pdf

[15] "IvCAM" [Online]. Available: https://www.e2esoft.com/ivcam/

[16] "Caracteristiques TX-60" [Online]. Available: https://robodk.com/robot/es/Staubli/TX60

[17] "Tia portal software" [Online]. Available: https://new.siemens.com/mx/es/productos/automatizacion/industry-software/automation-software/tia-portal/software.html

[17] "Camera frontal Iphone XR" [Online]. Available: https://support.apple.com/kb/SP781

Annex A: Xarxa neural

La xarxa neural es una classe, la qual necessita unes dades prèvies com el numero de classes i el nom d'elles, el codi següent son les dades que s'han de passar al inicialitzar la classe.

```
imageSize = (224, 224)

batch_size = 32

names_class = ['blue', 'green', 'organic', 'yellow']

train_path = './dataset/train'

test_path = './dataset/test'

validation_path = './dataset/valid'


# preparació de traint, test i validació normalitzant les dades amb "preprocess_input"

train_batches = tf.keras.preprocessing.image.ImageDataGenerator(
        rotation_range=5,
        rescale=1./255,
        horizontal_flip=True,
        zoom_range=0.2,
).flow_from_directory(
        directory=train_path,
        target_size=imageSize,
        classes=names_class,
        batch_size=batch_size,
        class_mode='categorical'
)

valid_batches = tf.keras.preprocessing.image.ImageDataGenerator(
        preprocessing_function=preprocess_input
).flow_from_directory(
        directory=validation_path,
        target_size=imageSize,
        classes=names_class,
        batch_size=batch_size,
        class_mode='categorical'
)
```

```
test_batches = tf.keras.preprocessing.image.ImageDataGenerator(
        preprocessing_function=preprocess_input,
).flow_from_directory(
        directory=test_path,
        target_size=imageSize,
        classes=names_class,
        batch_size=batch_size,
        shuffle=False,
        class_mode='categorical'
)
# test no modificat per afegir-ho a la part de test
test_batches_nopre = tf.keras.preprocessing.image.ImageDataGenerator(
        dtype='uint8'
).flow_from_directory(
        directory=test_path,
        target_size=imageSize,
        classes=names_class,
        batch_size=batch_size,
        shuffle=False,
        class_mode='categorical'
)
#numero d'epoques i steps
epoch = 20
steps = int(train_batches.samples / 35)
val_steps = int(valid_batches.samples / 35)

# Creació de la xarxa amb les dades prèvies creades
secondModel = classModels.modelsDenseNet121(epoch, steps, val_steps,
train_batches, valid_batches, test_batches, test_batches_nopre, True)
```

Un cop es te les dades, es pot anar fent crides a les funcions de la classe per que faci el procés una mica mes automàtic, les crides son les següents:

Inicialitzador de la classe

```
def __init__(self, epoch, steps, val_steps, train_batches,
valid_batches, test, test_prepro, optimization=False):
        self.base_model = ''
        self.model = ''
        self.base_predic = ''
        self.predic = ''
        self.confusion = ""
        self.historic = ''
        self.callbackList = ''
        self.model_prepro = ""
        self.epoch = epoch
        self.steps = steps
        self.val_steps = val_steps
        self.train_batches = train_batches
        self.valid_batches = valid_batches
        self.testData = test
        self.optimization = optimization
        self.test_nopre = test_prepro
```

```
def baseModelDenseNet121(self):
“””
Creació de la xarxa neural DenseNet121, carregant els pesos
prentrenats de imagnet, no es creen capes totalment connectades, y
s’utilitza un mitjana en l’operació de pooling, així nomes tindrem 1
sortia.
En relació al model s’esclafa la sortida en una dimensió, s’afegeix
una capa densa amb 256 entrades y una RLU com a funció d’activació per
ajuda a la regularització s’utilitza L1 y L2 amb un valor de 0.01,
s’afegeix una capa de sortida amb una tassa de 0.5. Per últim
s’afegeix una capa de 4 neurones amb la funció softmax.
“””
    self.base_model = tf.keras.applications.DenseNet121(
                                    weights='imagenet',
                                    include_top=False,
                                    pooling='avg')
    x = self.base_model.output
    x = Flatten()(x)
    x = Dense(256, kernel_regularizer=regularizers.l1_l2(0.01),
activity_regularizer=regularizers.l2(0.01), activation='relu')(x)
    x = Dropout(0.5)(x)
    self.base_predic = Dense(4, activation='softmax')(x)

    self.model = Model(inputs=self.base_model.input,
outputs=self.base_predic)
```

```
def compileModel(self):
“””
Funció per compilar el model, hi ha dos opcions, per si es vol
optimitzar el model o no,
“””


    if self.optimization:
        self.model.compile(
                        optimizer=SGD(learning_rate=0.01,
                        momentum=0.9),
                        loss='categorical_crossentropy',
                        metrics=['accuracy', 'mse']
                        )
        stop = EarlyStopping(
                        monitor='val_loss',
                        patience=8,
                        verbose=1,
                        min_delta=1e-4
                        )
        reduceLr = ReduceLROnPlateau(
                              monitor='val_loss',
                              factor=0.1,
                              patience=2,
                              verbose=1,
                              min_delta=1e-6
                              )
        self.callbackList = [stop, reduceLr]
    else:
        self.model.compile(
                        optimizer='adam',
                        loss='categorical_crossentropy',
                        metrics=['accuracy', 'mse']
                        )
```

```
        stop = EarlyStopping(
                  monitor='val_loss',
                  patience=8,
                  verbose=1,
                  min_delta=1e-4
                  )
        reduceLr = ReduceLROnPlateau(
                        monitor='val_loss',
                        factor=0.1,
                        patience=3,
                        verbose=1,
                        min_delta=1e-4
                        )
        self.callbackList = [stop, reduceLr]


def fit(self):
“””
Aquesta funció és la funció fit, es indispensable per iniciar
l’entrenament.
“””
     self.historic=self.model.fit(
            self.train_batches,
             steps_per_epoch=self.steps,
             validation_steps=self.val_steps,
             validation_data=self.valid_batches,
             epochs=self.epoch,
             callbacks=self.callbackList,
             verbose=1
         )
```

```
def predictions(self, preproc=False):
“””
Funció per crear les prediccions amb les dades de test.
Arg
      preproc (bool): serveix per indicar si el test que utilitzarem esta preprocesat o no, da questa manera s’utilitzarà un dels dos text passats a la inicialització
“””


    if preproc:
        self.predic = self.model.predict(
                                    x=self.test_nopre,
                                    verbose=1,
                                    steps=self.val_steps
                                    )



    else:
        self.predic = self.model.predict(
                                    x=self.testData,
                                    verbose=1,
                                    steps=self.val_steps
                                    )
```

```
def evaluateModel(self, preproc):
“””
Funció per evaluar el model.
Arg
      preproc (bool): serveix per indicar si el test que utilitzarem esta preprocesat o no, da questa manera s’utilitzarà un dels dos text passats a la inicialització
“””
    if preproc:
        self.model.evaluate(
                  x=self.test_nopre,
                  verbose=1,
                  steps=self.val_steps
                  )
    else:
        self.model.evaluate(
                  x=self.testData,
                  verbose=1,
                  steps=self.val_steps
                  )


def save(self, name):
“””
Funció per guardar el model generat
Arg
      name(String): nom amb el que es vol guardar el model
“””
    self.model.save("./models/" + name + ".h5")
```

```
def loadModel(self, name):
“””
Funció per carregar un model, aquest model es posarà tant en model,
preprocessat com en model no processat.
Arg
      name(String): nom del model que volem carregar
“””


    self.model = load_model("./models/" + name + ".h5")
    self.model_prepro = load_model("./models/" + name + ".h5")


def layerFalse(self, num=0):
“””
Funció per modificar un números de capes, d’aquesta manera es permet
el aprenentatge.
Arg
      num(int): numero de la capa inicial que volem que aprengui
“””
    if num == 0:
        for layer in self.base_model.layers:
            layer.trainable = False
    else:
        for layer in self.model.layers[:num]:
            layer.trainable = False
        for layer in self.model.layers[num:]:
            layer.trainable = True
```

```
def testing(self, classes):
“””
Funció per mirar una imatge predefinida, aquesta es una imatge de
l’apartat de test i la classe orgànic. Al final de la funció mostrara
quin es el resultat amb la imatge que s’ha utilitzat
Arg
      classes(llista): llista amb el nom de les classes
“””
    img_test = "./dataset/test/organic/organic (31).jpg"
    img = tf.image.decode_jpeg(tf.io.read_file(img_test), channels=3)
    dimension = np.expand_dims(img, axis=0)
    imageResize = tf.image.resize_with_pad(
                                    dimension,
                                    224,224,
                                    method="bilinear",
                                    antialias=True
                                    )
    imageProces = preprocess_input(imageResize)

    results = self.model.predict(imageProces)
    print('Class: ' + classes[np.argmax(results)] + ' ' +
str(round(results[0, np.argmax(results)] * 100, 2)) + '%')

    img = cv2.imread(img_test, cv2.IMREAD_COLOR)
    img = cv2.resize(img, (224, 224))
    plt.imshow(img)
    plt.title("Class: " + classes[np.argmax(results)] + ' ' +
str(round(results[0, np.argmax(results)] * 100, 2)) + '%')
    plt.show()
```

Annex B: Codi connexió socket Stäubli – ordinador

Codi python que està situat en l'ordinador

```
# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect the socket to the port where the server is listening
server_address = ('192.168.0.12', 11001)
print('connecting to {} port {}'.format(*server_address))
sock.connect(server_address)

message = ''

try:

    # Send data
    message = b’0’
    print('sending {!r}'.format(message))
    sock.sendall(message)


    # Look for the response
    amount_received = 0
    amount_expected = len(message)

    print("amount_received " + str(amount_received))
    print("amount_expected " + str(amount_expected))
    while amount_received < amount_expected:
        data  = sock.recv(16)
        amount_received += len(data)
        print('received {!r}'.format(data))

finally:
    print('closing socket')
    sock.close()
```

Codi val3 situat en el controlador del robot

```
//Clean screen
cls()
// move robot
movej(JHome,tTool, mNomSpeed)
waitEndMove()
movej(JApproach1,tTool, mNomSpeed)
waitEndMove()
do
    // read character
    nTest = sioGet(sSioOrdenador, l_nNumero)
    delay(5)
    put("Test :")
    put(nTest)
    put(" L_nNumero :")
    // write character obtain
    put(l_nNumero)
    putln(" ")


  until l_nNumero == 13
```

Annex C: Codi connexió socket PLC- Stäubli – ordinador

Codi python que està situat en l'ordinador

```
import classModels
import orientationDetector
import cv2
import uuid
import socket

# Connect the socket to the port where the server is listening
tcpsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_address = ('172.31.0.1', 11001)
print('connecting to {} port {}'.format(*server_address))
sock.connect(server_address)

# recieve value to CS8C
print("Success Connect")
data = sock.recv(16)

# Connect to camera
cap = cv2.VideoCapture(0)

names_class = ['blue', 'green', 'organic', 'yellow']
# charge model
model = classModels.modelsDenseNet121("modelUse")

takeFoto = False
stopProgram = False

if data == b'I':
    while not(stopProgram):
        while not(takeFoto):
            data = sock.recv(16)
            print(data)
            if data == b'R':
                takeFoto = True
        leido, frame = cap.read()

        if leido:
            nameFoto = './img/' +str(uuid.uuid4()) + ".png" # uuid4
regresa un objeto, no una cadena. Por eso lo convertimos
            print("Foto tomada correctamente con el nombre
{}".format(nameFoto))
            cv2.imwrite(nameFoto, frame)
```

```
            #Obtain orientation
            #angle = orientationDetector.getOrientation(nameFoto)

            #Classify Image
            resultClass = model.classifyImage(names_class, nameFoto)

            #send result
            sock.sendall(resultClass)

        else:
            print("Error al acceder a la cámara")
            stopProgram = True
        takeFoto = False

cap.release()
sock.close()
```

Codi val3 situat en el controlador del robot

Codi fitxer start (codi que s'executa al posar en marxa el robot)

```
begin
  call startingSockets()

  // Creació de les tasques en paral·lel

  taskCreate "stateRobot", 1, statesRobot()
  taskCreate "moveRobot", 1, moveRobot()
  taskCreate "screen", 1, pantalla()
end
```

Codi fitxer moveRobot (codi que es dedica a moure el robot)

```
begin
  movej(JHome,tToolSuccion, mNomSpeed)
  waitEndMove()
  movej(JApproach1,tToolSuccion, mNomSpeed)
  waitEndMove()
  do
    movel(pTable,tToolSuccion, mNomSpeed)
    waitEndMove()
    sioGet(sPlc, nRecvPLC)
    if nRecvPLC == 100
      sioSet(sSocket, 82)
      movej(jAproachCinta, tToolSuccion, mNomSpeed)
      waitEndMove()
      movej(jGetCinta, tToolSuccion, mNomSpeed)
      waitEndMove()
      close(tToolSuccion)
      waitEndMove()
      movel(pApprBins, tToolSuccion, mNomSpeed)
      waitEndMove()
      nConnection = sioGet(sSocket, nRecvData)
    switch nRecvData
      case 48
        movel(pBinYellow, tToolSuccion, mNomSpeed)
        waitEndMove()
```

```
      break
      case 49
        movel(pBinBlue, tToolSuccion, mNomSpeed)
        waitEndMove()
      break
      case 50
        movel(pBinGreen, tToolSuccion, mNomSpeed)
        waitEndMove()
      break
      case 51
        movel(pBinBrown, tToolSuccion, mNomSpeed)
        waitEndMove()
      break
      default
        putln("Bin selected no correct")
      break
    endSwitch
      open(tToolSuccion)
      waitEndMove()
      nRecvPLC=0
      clearBuffer(sPlc)
    endIf
  until 1==2
end
```

Codi fitxer pantalla (codi que es dedica a mostrar la informació en la pantalla)

```
begin
  userPage()
  cls()
  title("Start to recicling")
  gotoxy(35,13)
  put("EXIT")
  do
    gotoxy(0,1)
    put("Port 11001:")
    gotoxy(12,1)
    switch nConnection
      case -1
        put("Waiting connection")
      break
      case 1
        putln("Connected succesful")
        put("Go to bin: ")
        switch nRecvData
          case 48
            putln("Yellow")
          break
          case 49
            putln("Blue")
          break
          case 50
            putln("Green")
          break
          case 51
            putln("Brown")
          break
          default
            putln("Bin selected no correct")
          break
        endSwitch
      break
      case 0
        putln("Stopped Connection")
      break
      default
        put("Connection lose")
      break
    endSwitch
  until 1==2
end
```

Codi fitxer StateRobot (codi que es dedica a parar el robot des de el comandament)

```
begin
  bStop = false
  do
    nkey = getKey()
    if(nkey == 278)
      bStop = true
      userPage()
      cls()
      gotoxy(12,7)
      put("FINAL PROGRAM")
      taskKill("moveRobot")
      taskKill("screen")
      taskKill("obatinValues")
      wait(taskStatus("moveRobot") == -1)
      wait(taskStatus("screen") == -1)
      wait(taskStatus("obatinValues") == -1)
    endIf
  until bStop == true
end
```

Codi fitxer startingSockets (codi que es dedica a inicialitzar tots els sockets)

```
begin
  clearBuffer(sSocket)
  clearBuffer(sPlc)
  wait(sioSet(sSocket, 73)==1)
  wait(sioGet(sPlc, nRecvPLC)==1)
end
```

Annex D: Codi orientació

Codi python que està situat en l'ordinador

```
def getOrientation(img):
    image = cv2.imread(img)
    im_size = image.shape
    img2 = image[int((im_size[0] / 2) - 60):int((im_size[0] / 2) +
185), int((im_size[1] / 2)) - 105:int((im_size[1] / 2) + 130), :]

    img2Colors = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    gray = cv2.cvtColor(img2Colors, cv2.COLOR_BGR2GRAY)

    # threshold the grayscale image
    _, bw = cv2.threshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)

    contours = contours[0] if len(contours) == 2 else contours[1]

    # get rotated rectangle from outer contour
    rotrect = cv2.minAreaRect(contours[0])
    angle = rotrect[-1]

    # from https://www.pyimagesearch.com/2017/02/20/text-skew-
correction-opencv-python/
    # the `cv2.minAreaRect` function returns values in the
    # range [-90, 0); as the rectangle rotates clockwise the
    # returned angle trends to 0 -- in this special case we
    # need to add 90 degrees to the angle
    if angle < -45:
        angle = -(90 + angle)
    else:
        angle = angle
    return angle
```