



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Estudi d'algorismes de control de velocitat d'un cotxe de slot

Document:

Memòria

Autor/Autora:

Jose Maria García Calderó

Director/Directora - Codirector/Codirectora:

Ramon Sarrate Estruch

Titulació:

Grau en Enginyeria Electrònica Industrial i Automàtica

Convocatòria:

Tardor Pròrroga, 2022.

TREBALL DE FI D'ESTUDIS





Agraïments

Durant el transcurs d'aquest treball, he tingut ajuda de varies persones que m'han ajudat en la realització del treball.

En primer lloc, agrair l'ajut i esforç del meu tutor, Ramon Sarrate, per haver estat al damunt i haver-me ajudat en les etapes del treball mes complicades. Hem realitzat varies reunions al llarg dels últims mesos per valorar el progrés del treball i m'ha donat indicacions de com poder avançar quan m'he estancat o quan feia les coses d'una manera incorrecte.

També agrair els meus pares, per haver-me orientat en la redacció de la memòria i per haver-se llegit el treball.

Resum

Aquest treball de fi de grau ha consistit en l'estudi dels algorismes de control Proporcional i Proporcional-Integral els qual s'han dissenyat per assolir certes especificacions sense la interacció humana. Les especificacions han consistit en mantenir el cotxe de slot circulant per el circuit a una velocitat constant de 100 cm/s utilitzant la visió Artificial i un algorisme de control en llaç tancat.

El treball ha estat la continuació del TFG de l'alumna Roger Viaplana anomenat "Disseny i implementació d'un sistema de monitoratge i comandament d'un circuit slot" realitzat al 2017 on es dissenyava una aplicació amb la finalitat de controlar el cotxe de slot en llaç obert utilitzant l'eina Matlab. Per tal de millorar aquest treball es va dissenyar un programa mes robust utilitzant el llenguatge de programació Python per utilitzar la visió artificial i poder fer l'algorisme de control en llaç tancat.

El treball s'ha dut a terme en l'edifici TR11, a l'aula 1.06 del primer pis, on es disposava d'una taula preparada amb el circuit de slot. També s'ha disposat de tot el material que va utilitzar en Roger, des de la càmera per realitzar la visió artificial com l'Arduino per poder enviar les senyals al circuit.

Per tal de dissenyar l'algorisme de control s'ha dissenyat un programa per mesurar la velocitat del cotxe en tots el trams del circuit i després es va provar amb diferents valors de KP i KI fins trobar uns valors que complissin amb les especificacions establertes.

Abstract

This end-of-degree work has consisted of the study of Proportional and Proportional-Integral control algorithms which have been designed to achieve certain specifications without human interaction. The specifications consisted of keeping the slot car moving around the circuit at a constant speed of 100 cm/s using Artificial Vision and a closed loop control algorithm.

The work was the continuation of the TFG of the student Roger Viaplana called "Design and implementation of a monitoring and control system of a slot circuit" carried out in 2017 where an application was designed in order to control the slot car in open loop using the Matlab tool. In order to improve this work, a more robust program was designed using the Python programming language to use artificial vision and be able to perform the closed-loop control algorithm.

The work was carried out in the TR11 building, in classroom 1.06 on the first floor, where there was a table prepared with the slot circuit. All the material that Roger used has also been made available, from the camera to perform the artificial vision to the Arduino to be able to send the commands to the circuit.

In order to design the control algorithm, a program has been designed to measure the speed of the car in all sections of the circuit and then it was tested with different values of KP and KI until finding values that met the established specifications.

Índex

Contingut

RESUM.....	III
ABSTRACT	IV
ÍNDEX	V
ÍNDEX DE TAULES	VI
ÍNDEX DE FIGURES	VII
1 INTRODUCCIÓ	VIII
1.1 OBJECTE.....	VIII
1.2 ABAST	VIII
1.3 REQUERIMENTS	VIII
1.4 JUSTIFICACIÓ.....	IX
1.5 METODOLOGIA.....	IX
2 EINES I RECURSOS.....	XI
2.1 MATERIAL DE TREBALL.....	XI
2.1.1 <i>Taula de treball</i>	<i>xi</i>
2.1.2 <i>Cotxe de slot</i>	<i>xi</i>
2.1.3 <i>Circuit de slot</i>	<i>xii</i>
2.1.4 <i>Cable USB 2.0 tipus A-mascle i B-mascle</i>	<i>xiii</i>
2.1.5 <i>Arduino + circuit de comandament + cable jack to jack</i>	<i>xiii</i>
2.1.6 <i>Ordinador + perifèrics</i>	<i>xv</i>
2.1.7 <i>Font d'alimentació</i>	<i>xv</i>
2.1.8 <i>Càmera</i>	<i>xvi</i>
2.2 PYTHON.....	XVI
2.3 ARDUINO UNO	XVII
2.4 DOCUMENTACIÓ	XVIII
2.4.1 <i>Treball de recerca previ</i>	<i>xviii</i>
2.4.2 <i>Llibreries de Python</i>	<i>xviii</i>
2.4.3 <i>Llibreria d'Arduino</i>	<i>xviii</i>
2.4.4 <i>Controlador PID</i>	<i>xviii</i>
2.5 CODI	XXI
2.5.1 <i>Visió Artificial</i>	<i>xxi</i>
2.5.2 <i>Arduino</i>	<i>xxiv</i>
2.5.3 <i>Algorisme de control</i>	<i>xxv</i>
2.6 RESULTATS	XXVI
2.6.1 <i>Programa final</i>	<i>xxvi</i>
2.6.2 <i>Algorisme de control</i>	<i>xxxi</i>
3 RESUM DEL PRESSUPOST.....	XXXVIII
4 CONCLUSIONS	XXXVIII
5 BIBLIOGRAFÍA.....	XXXIX
6 ANNEX.....	XL



Índex de taules

TAULA 1: LLIBRERIES DE PYTHON UTILITZADES PER A LA REALITZACIÓ D'AQUEST TREBALL.	XVII
TAULA 2: VALOR DE ISE PER CADA KP	XXXII
TAULA 3: VALOR DE ISE PER CADA KI	XXXIII
TAULA 4: VALOR DE VELOCITAT MITJA PER CADA KP	XXXIII
TAULA 5: VALOR DE VELOCITAT MITJA PER CADA KI AMB UNA $K_P=0,007$	XXXIV

Índex de figures

FIGURA 1: VISTA PANORÀMICA DE LA TAULA DE TREBALL CONTENINT ALGUNS DELS ELEMENTS UTILITZATS PER A LA REALITZACIÓ DELS EXPERIMENTS.....	XI
FIGURA 2 I 3: IMATGE FRONTAL I LATERAL DEL COTXE DE SLOT UTILITZAT. LES DIMENSIONS REALS DEL COTXE SÓN 6,5 CM D'AMPLE PER 14 CM DE LLARG.	XII
FIGURA 3: CARACTERÍSTIQUES DEL CIRCUIT DE SLOT	XII
FIGURA 4: IMATGE DEL CIRCUIT DE SLOT.....	XIII
FIGURA 5: CABLE JACK TO JACK PER CONNECTAR ÀRDUINO I EL CIRCUIT DE SLOT	XIII
FIGURA 6: IMATGES PRESES DES DE DIFERENTS PERSPECTIVES CORRESPONENTS A L'ÀRDUINO I EL CIRCUIT DE COMANDAMENT.	XIV
FIGURA 7: IMATGES PRESES DES DE DIFERENTS PERSPECTIVES CORRESPONENTS A L'ÀRDUINO I EL CIRCUIT DE COMANDAMENT.	XIV
FIGURA 8: IMATGE DE L'ORDINADOR I ELS PERIFÈRICS UTILITZATS PER A LA REALITZACIÓ DEL TREBALL.....	XV
FIGURA 9: FONT D'ALIMENTACIÓ PEL CIRCUIT.....	XVI
FIGURA 10: ESQUEMA DE MUNTATGE DE LA CÀMERA FOTOGRÀFICA	XVI
FIGURA 11: IMATGE PER MOSTRAR LA SORTIDA PROMIG EN FUNCIÓ DEL CICLE DE TREBALL (LLAMAS, 2015)	XVII
FIGURA 12: DIBUX PER REPRESENTAR LA POSICIÓ DEL COTXE DINS DE LA CORBA	XIX
FIGURA 13: DIAGRAMA DE FLUX PER A LA DETERMINACIÓ DE LA VELOCITAT	XX
FIGURA 14: DIAGRAMA DE BLOCS D'UN CONTROLADOR PID (MARTÍN, 2022).....	XX
FIGURA 15: RANG DE COLORS I VALORS EN HSV (SOLANO, OMES, 2019).....	XXII
FIGURA 16: IMATGE AMB FILTRE.....	XXII
FIGURA 17: IMATGE QUE CAPTA LA CÀMERA AMB EL FILTRE APLICAT I LA DETECCIÓ DEL COTXE	XXIII
FIGURA 18: COORDENADES X I Y DURANT EL CIRCUIT	XXIII
FIGURA 19: DIAGRAMA DE BLOCS.....	XXV
FIGURA 20: DIAGRAMA DE FLUX DEL PROGRAMA COMPLERT.....	XXVII
FIGURA 21: GRÀFIC DE LA VELOCITAT (CM/S) EN FUNCIÓ DEL TEMPS (S) AMB UNA $K_P = 0,007$ I UNA $K_I = 0$	XXXIV
FIGURA 22: GRÀFIC DE LA VELOCITAT (CM/S) EN FUNCIÓ DEL TEMPS (S) AMB UNA $K_P = 0,007$ I UNA $K_I = 0,0006$	XXXV
FIGURA 23: GRÀFIC DE L'INCREMENT D'ESPAI (CM) EN FUNCIÓ DEL TEMPS (S).....	XXXV
FIGURA 24: GRÀFIC DE L'INCREMENT DE TEMPS (S) EN FUNCIÓ DEL TEMPS TOTAL (S).....	XXXVI
FIGURA 25: GRÀFIC DEL SENYAL DE SORTIDA EN FUNCIÓ DEL TEMPS (S) AMB $K_P = 0,007$ I UNA $K_I = 0,0006$	XXXVI
FIGURA 26: SORTIDA DEL CONTROLADOR SATURADA AMB UNA $K_P=0,1$	XXXVII
FIGURA 27: GRÀFIC DE L'ISE EN FUNCIÓ DEL TEMPS (S) AMB $K_P=0,007$ I UNA $K_I=0$	XXXVII
FIGURA 28: GRÀFIC DE L'ISE EN FUNCIÓ DEL TEMPS (S) AMB $K_P=0,007$ I UNA $K_I=0,0006$	XXXVIII

1 Introducció

1.1 Objecte

L'objectiu principal d'aquest treball de fi de grau és dissenyar un algorisme de control de velocitat d'un cotxe de slot per tal de que aquest assoleixi certes especificacions sense la interacció humana. Les especificacions seleccionades són una velocitat constant del cotxe de slot en cm/s al llarg de tot el circuit de slot, consistent en dos trams rectes i dues corbes. Per a assolir aquest objectiu principal, es plantegen dos objectius específics:

- Explorar l'ús de l'entorn de programació Python per al control del sistema a partir de l'adquisició i el tractament d'imatges del cotxe de slot en circulació
- Dissenyar un controlador de llaç tancat per a l'emissió i recepció de senyals

El projecte parteix d'un treball de fi de grau previ en el qual s'havia creat una aplicació amb Matlab per poder mostrar per la pantalla de l'ordinador les imatges que una càmera està rebent del cotxe en moviment i controlar a través d'un Arduino la seva velocitat amb un controlador de llaç obert. Aquest disseny presenta una sèrie de limitacions, com el fet que el controlador de llaç obert, si bé pot enviar senyals, no permet obtenir dades de retroalimentació de la velocitat en temps real del vehicle, la qual es veu necessàriament influenciada per diferents factors, com per exemple les característiques del traçat del circuit (tram recte o corba).

En el present treball es pretén crear un controlador de llaç tancat, que permeti tant l'emissió de les senyals com la retroalimentació de velocitat, per tal de modular amb més precisió la velocitat del cotxe al llarg de tot el circuit. A més, s'explorarà per primer cop el llenguatge de programació Python. Aquest entorn és més avantatjós que Matlab, ja que conté un gran nombre de llibreries i presenta una major facilitat de programació.

1.2 Abast

Per a assolir els objectius proposats, aquest projecte engloba la creació dels següents elements :

1. Un sistema d'adquisició d'imatge a través d'una càmera USB
2. Un tractament d'imatge per aïllar el cotxe de slot del seu entorn
3. Un controlador en llaç tancat per modular la velocitat del cotxe

1.3 Requeriments

En aquest treball de fi de grau s'han utilitzat diferents eines per tal d'assolir els objectius. Els diferents requeriments per a la realització del treball són:

- Un Arduino per poder establir la connexió entre l'ordinador i el circuit . L'arduino s'encarregarà de rebre les consignes que s'envien des del programa en Python i d'enviar els senyals elèctrics al circuit.
- Utilitzar Python per fer l'adquisició i el tractament d'imatge i per utilitzar-lo com a controlador per aplicar l'algorisme de control.

- Utilitzar la càmera per poder visualitzar el circuit de Scalextric en temps real.
- Un espai de treball ample on poder situar l'ordinador i on situar el circuit i el cotxe de slot.
- Un circuit que pugui enviar les senyals electròniques del circuit Arduino al circuit de slot

1.4 Justificació

Aquest treball de fi de grau continua el treball realitzat en altres treballs previs i proposa noves aportacions que resolen limitacions i milloren els dissenys obtinguts anteriorment. Els dos treballs previs, als quals dona continuïtat aquest treball són "Study of automatic control algoritmes for a slot car" i "Disseny i implementació d'un sistema de monitoratge i comandament d'un circuit de slot". Aquests treballs es van estudiar i crear un sistema de control bàsic en llaç obert del circuit de slot i per tant, aquest treball busca anar un pas més enllà i millorar el sistema d'adquisició i tractament i crear un programa en Python per millorar l'aplicació del treball anterior.

Assolir el compliment d'unes especificacions de forma automàtica sense la intervenció humana és de gran importància en molts camps d'aplicació. Això requereix obtenir i analitzar dades de forma precisa i aplicar les correccions necessàries en el moment oportú. El present projecte pretén controlar de forma precisa la velocitat d'un cotxe slot a partir de l'emissió i recepció constant de dades obtingudes i analitzades mitjançant un disseny basat en l'entorn de programació Python i la creació d'un controlador de llaç tancat. Aquest disseny presenta clars avantatges respecte a un disseny anterior (de l'any 2017), basat en Matlab, ja que Python es un entorn de programació més complet, amb millor rendiment i alhora més àgil i simple que Matlab. A més, es proposa la creació d'un controlador de llaç tancat, aquest tipus de controlador no ha estat estudiat amb anterioritat en aquest context. A diferència dels controladors de llaç obert, els controladors de llaç tancat permeten no només l'emissió de senyal sinó també una recepció, fent possible una retroalimentació i la consegüent millora d'eficàcia. És d'esperar que l'ús d'un controlador de llaç tancat contribueixi a una important millora en el control de les especificacions, ja que la recepció de senyal és essencial per a aplicar les correccions justes i necessàries en el senyal emès a continuació, modulant així de forma més precisa la velocitat del cotxe.

És d'esperar que els resultats d'aquest projecte es puguin traslladar al món real en molts aspectes. Per exemple, el control de velocitat i la supervisió amb la càmera poden ser d'utilitat en robots o qualsevol vehicle que es mogui en rails. El sistema de control PID es pot utilitzar per controlar no només vehicles sinó també temperatures o pressions i l'adquisició d'imatges a través de la càmera es pot utilitzar en qualsevol planta o vehicle que ho necessiti.

1.5 Metodologia

Aquest treball s'ha realitzat en diferents etapes, una etapa de preparació, una etapa d'estudi i una etapa de realització.

Etapa de preparació

L'etapa de preparació va consistir en l'obtenció i preparació del material necessari per aquest treball. Part del material utilitzat ha procedit dels treballs previs realitzats amb anterioritat per l'exalumne Roger Viaplana i que va servir com a punt de partida per aquest treball. D'entre els materials previs disponibles, cal destacar el circuit electrònic, el qual serveix per connectar el circuit de slot amb l'Arduino permetent utilitzar l'ordinador com a controlador. Es va muntar tot el circuit a una taula de l'aula 1.06 de l'edifici TR11 a l'UPC de Terrassa i es va preparar també l'ordinador amb el qual es va treballar durant tot el projecte.

Per posar a punt l'ordinador, va ser necessari instal·lar la versió més actual de Matlab degut a que les llibreries de la càmera no estaven disponibles en versions antigues. També va ser necessari instal·lar la llibreria d'Arduino a Matlab i descarregar tot el material del TFG del Roger Viaplana per poder estudiar-lo.

Per últim, va ser necessari instal·lar el programa Python per poder programar en ell i també va ser necessari instal·lar totes les llibreries necessàries.

Etapa d'estudi

Un cop es va finalitzar la preparació, va ser necessari llegir i entendre el TFG del Roger Viaplana per poder tenir un punt de partida i poder valorar quin material seria d'utilitat per realitzar aquest treball. El treball va consistir en la creació d'una aplicació amb la fi de controlar la velocitat d'un cotxe de slot a través d'un algorisme de control. El treball es va realitzar a l'any 2017 i per tant bona part del material estava antiquat. El material de software que va resultar més útil va ser el software per l'obtenció i el tractament d'imatge, no obstant, tot el codi es va tornar a fer completament ja que es va decidir canviar de Matlab a Python per motius de rendiment. També va ser necessari estudiar tant les noves llibreries per a Python com els mètodes d'adquisició i tractament d'imatge i la comunicació amb l'Arduino a través de Python.

L'algorisme de control utilitzat per en Roger Viaplana consistia en un algorisme en llaç obert, el qual dividia el circuit en 4 segments i permetia adjudicar una consigna de control a cada part del segment. Per aquest treball també es va decidir dissenyar un nou algorisme de control en llaç tancat el qual millora l'eficàcia del controlador. Per tal de dissenyar el nou controlador, es va optar per dissenyar un controlador Proporcional (P) i un Proporcional Integrador (PI).

Etapa de realització

En l'etapa de realització es pot separar en dos parts, la part d'adquisició i la part de comandament. Per realitzar l'adquisició, es va començar per programar un codi capaç de visualitzar les imatges de la càmera per pantalla i poder tractar-les. Va ser necessari adaptar les condicions de lluminositat per tal de que la càmera pogués mostrar de manera correcta les imatges i més tard les pogués tractar sense problemes. Un cop assolida la fita de que el programa pogués captar les imatges i detectar el cotxe a dins de les mateixes, es va procedir a ampliar el codi per tal de poder enviar senyals a través de l'Arduino. Per tal de fer-ho, es connectava l'Arduino tant a l'ordinador com al circuit i a la font d'alimentació.

El següent pas va ser realitzar un programa que fos capaç de mesurar la velocitat del cotxe de slot en tot el circuit i per poder realitzar les proves per el algorisme de control. Per dissenyar l'algorisme de control, es va començar implementant un controlador Proporcional i es van realitzar múltiples proves amb diferents valors de K_p i es van analitzar els resultats amb l'índex ISE i la velocitat mitjana. Un cop es va seleccionar una K_p , es va afegir l'acció integral amb la mateixa metodologia, aplicant múltiples proves amb diferents valors fins trobar un valor que dogues els millors resultats seguint l'índex i la velocitat mitja.

2 Eines i Recursos

2.1 Material de treball

Per poder realitzar aquest treball, s'ha necessitat disposar d'un material tant a nivell de software com a nivell de hardware. Tot el material necessari ha estat subministrat pel tutor del TFG i ha consistit en els següents elements:

2.1.1 Taula de treball

El cotxe i el circuit s'han muntat damunt una taula prèviament preparada a l'aula 1.06 de l'edifici TR11. Aquesta taula compta amb una petita manta de color negre per poder diferenciar el fons dels elements d'interès que ubicats damunt d'ella.



Figura 1: Vista panoràmica de la taula de treball contenint alguns dels elements utilitzats per a la realització dels experiments.

2.1.2 Cotxe de slot

El cotxe de slot utilitzat, és un cotxe de 6,5 cm d'ample per 14 cm de llarg. El cotxe compta amb una base on hi han situades rodes, el motor, i les escombretes per fer arribar el corrent cap al motor. La carcassa del cotxe, va ser pintada de color taronja per poder diferenciar-la clarament del entorn (principalment de color negre) i facilitar el tractament d'imatges. Per realitzar el contacte elèctric entre el circuit i el cotxe, s'utilitzen dos escombretes que venen incorporades en la base del cotxe. Aquestes escombretes fan contacte amb els dos carrils metàl·lics del circuit, una per cada carril, i permeten fer circular el corrent des de el circuit elèctric al motor que es situa a dins del cotxe. Al inici de cada prova, es necessari preparar les escombretes de tal manera en que quedin de la manera mes vertical possible i facin contacte amb els carrils. Això es degut a que quan el cotxe es posa en marxa, degut al

moviment horitzontal, les escombretes tendeixen a aplanar-se i es probable que en algú punt del circuit deixin de fer contacte les escombretes amb els carrils.



Figura 2 i 3: Imatge frontal i lateral del cotxe de slot utilitzat. Les dimensions reals del cotxe són 6,5 cm d'ample per 14 cm de llarg.

2.1.3 Circuit de slot

Es tracta d'un circuit de la marca NINCO i consta de dues rectes de 80 cm de llarg i dues corbes de 35 cm de diàmetre de color negre amb dos carrils, un interior i un altre exterior. Per aquest projecte s'ha utilitzat el carril exterior ja que en el treball de fi de grau realitzat al 2017 també es va utilitzar el carril exterior. Per indicar la marca de sortida, el carril compta amb una línia perpendicular al sentit de circulació en la recta 1. També, com a la imatge es pot veure, en el mateix punt d'inici del circuit, hi ha de color vermell l'endoll per connectar tant el cable de comandament i el cable d'alimentació de 12 volts en cas de voler utilitzar el circuit de manera lúdica amb el comandament que porta per defecte el circuit, no obstant, per aquest projecte només es connecta un cable jack to jack per connectar el circuit a l'Arduino.

Cada recta està composta per 2 peces de recta de 40 cm i cada corba està composta per un 4 peces. No totes les peces tenen el mateix fregament ni la mateixa conductivitat, per exemple, tal i com es mostra a la imatge, a l'inici del tram de la recta 2 hi ha un major fregament entre la pista i el circuit. També, en cas de notar un mal contacte en el circuit es necessari polir el carrils.



Figura 3: Característiques del circuit de slot

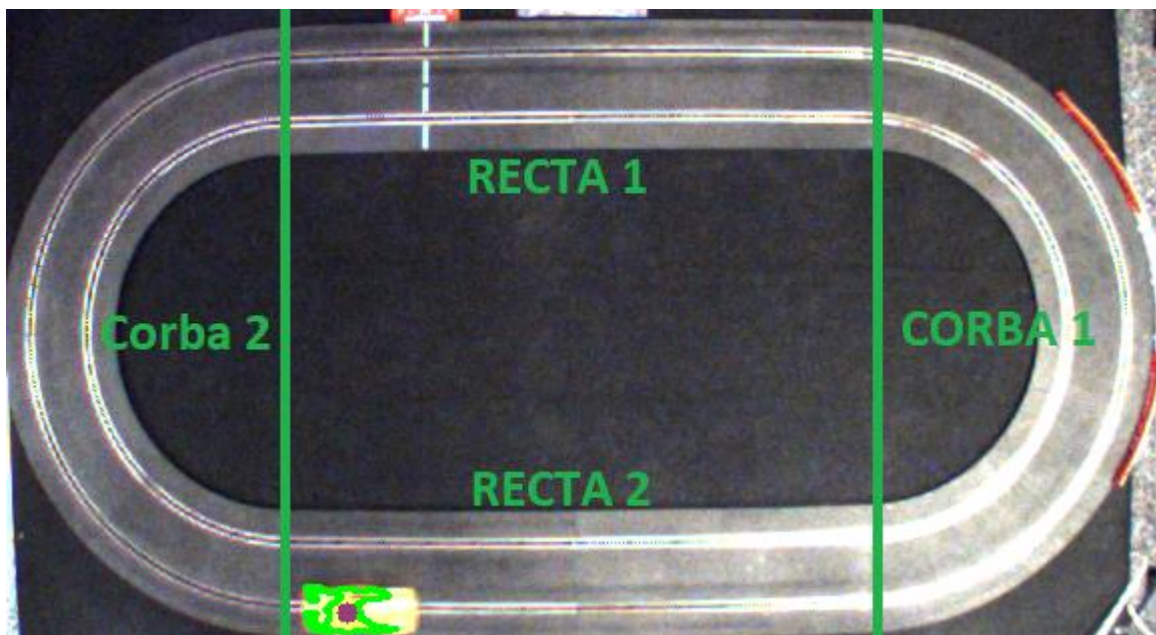


Figura 4: Imatge del circuit de slot.

2.1.4 Cable USB 2.0 tipus A-masclle i B-masclle

Aquest cable és l'encarregat de transmetre la informació entre l'ordinador i l'Arduino.

2.1.5 Arduino + circuit de comandament + cable jack to jack

Gràcies al treball d'en Roger Viaplana, es va disposar d'un circuit amplificador que constava d'un transistor Darlington amb el qual es va poder establir la connexió entre el circuit de slot i l'Arduino. Aquest circuit amplificador, permetia connectar la font d'alimentació del circuit de slot de 12 volts i el cable jack to jack al circuit amb el qual s'enviaven les senyals de corrent per moure el cotxe.

El circuit també comptava amb dos pins, un per connectar el circuit al pin 3 de l'Arduino, i un altre per connectar el pin negatiu. L'Arduino anava connectat a l'ordinador a través del cable USB 2.0 de tipus A-masclle a B-masclle esmentat en el punt anterior.



Figura 5: Cable jack to jack per connectar Arduino i el circuit de slot



Figura 6: Imatges preses des de diferents perspectives corresponents a l'Arduino i el circuit de comandament.



Figura 7: Imatges preses des de diferents perspectives corresponents a l'Arduino i el circuit de comandament.

2.1.6 Ordinador + perifèrics

L'ordinador utilitzat és un dispositiu de sobretaula de la marca Dell que s'utilitza habitualment a la universitat consistent en una unitat central, equipat amb una pantalla Samsung, un teclat i un ratolí. Està situat en una taula a part, al costat de la taula on està situat el circuit i compte amb el sistema operatiu de Windows. Ha estat necessari instal·lar l'eina Matlab 2021, el programa Python 3.10, el programa Arduino IDE 2.0.0 i totes les llibreries necessàries dels diferents programes.



Figura 8: Imatge de l'ordinador i els perifèrics utilitzats per a la realització del treball.

2.1.7 Font d'alimentació

Per alimentar el circuit, dins del material venia inclòs una font d'alimentació de 12 volts, marca NINCO, model HGS6120150. Aquesta font d'alimentació es l'encarregada de subministrar el corrent al circuit per permetre el moviment del cotxe.



Figura 9: Font d'alimentació pel circuit.

2.1.8 Càmera

Situada en el centre de la taula, es troba la càmera amb la qual s'han captat les imatges del circuit i amb la qual s'ha pogut realitzar el seguiment del cotxe de slot. La càmera és el model DBK 21AU04.AS de la marca alemanya TheimagingSource, i realitza fotografies a color a 60fps. Està subjecta del sostre a través d'una peça metàl·lica i uns cargols. (Astro shop, 2022)



Figura 10: Esquema de muntatge de la càmera fotogràfica .

2.2 Python

Python es un llenguatge d'alt nivell que es caracteritza per la seva facilitat alhora de ser llegit i per l'àmplia biblioteca de llibreries que existeixen per aquest llenguatge.

El projecte anterior s'havia dut a terme amb la plataforma de programació Matlab. No obstant, comptava amb algunes limitacions, tenia una mala base i mancava de robustesa. La llibreria de recollida d'imatges de Matlab no funcionava gaire be a l'hora d'executar la recollida de vídeo i el tractament d'imatges de forma simultània.

En contrast, Python compta amb una gran varietat de llibreries que faciliten molt la realització del treball. S'han utilitzat 5 llibreries per aquest treball i són les següents:

Nom	Abreviació	Descripció
OpenCV	Cv2	Llibreria de visió artificial
Numpy	Np	Llibreria per crear vectors, matrius i funcions matemàtiques
Pyplot	matplotlib.pyplot	Llibreria per generar gràfics
Time	time	Llibreria per utilitzar funcions de temps com cronometres, timers, etc.
Pyfirmata	pyfirmata	Llibreria per utilitzar Arduino

Taula 1: Llibreries de Python utilitzades per a la realització d'aquest treball.

2.3 Arduino Uno

Arduino és una placa que consta d'un microcontrolador re-programable i varies entrades femella que permeten fer connexions entre l'Arduino i diferents actuadors i sensors compatibles amb la placa. Entre la gran varietat de dispositius Arduino (Arduino Mega, Arduino Nano, Arduino Leonardo, etc), es va optar per utilitzar Arduino Uno degut a com s'adaptava al projecte.

Arduino Uno té 16 entrades i sortides digitals i 6 analògiques. Cada pin de l'Arduino pot subministrar 5V (exceptuant els pins de 3,3 V) com a màxim.

- Les entrades i sortides digitals només poden estar en dos estats, subministrant 0V o 5V (és similar a escriure 0 o 1 ja que un 0 equival a 0V i un 1 equival a 5V).
- Les entrades analògiques poden llegir qualsevol valor entre 0V i 5V.
- Les sortides PWM són sortides que poden donar qualsevol valor entre 0V i 5V utilitzant polsos digitals en funció del cycle de treball. El cycle de treball determina el temps que una senyal digital es troba a 1 i pot anar entre 0 i 255 on 0 equival a 0V i 255 a 5V.

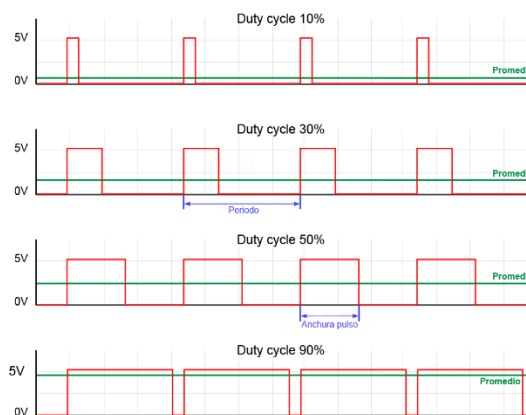


Figura 11: Imatge per mostrar la sortida promig en funció del cycle de treball (Llamas, 2015)

2.4 Documentació

2.4.1 Treball de recerca previ

Com s'ha mencionat amb anterioritat, es va agafar com a base per aquest TFG un antic treball realitzat al 2017 anomenat "Disseny i implementació d'un sistema de monitoratge i comandament d'un circuit slot" on l'objectiu era crear un sistema de control automàtic amb la finalitat de controlar el cotxe de slot sense l'actuació humana. Gràcies en aquest treball, s'ha pogut encaminar aquest TFG amb l'objectiu de millorar l'anterior.

Aquest TFG es va realitzar utilitzant Matlab i varies llibreries que van permetre la realització del treball i també ha aportat el estudi del funcionament del circuit de slot junts amb el circuit que unifica l'Arduino amb el comandament del circuit. El circuit utilitza un transistor per amplificar la senyal de l'Arduino i poder enviar la senyal al circuit. Aquest circuit està explicat amb més detall dins del TFG del 2017 i és necessari connectar en l'ordre i de manera correcta el circuit per evitar espallar-lo.

Ha estat important entendre el codi Matlab de l'antic TFG per poder aprofitar la base per poder realitzar aquest treball.

2.4.2 Llibreries de Python

Com ja s'ha mencionat en anteriors apartats, s'han utilitzat 5 llibreries per realitzar aquest treball. La més important ha estat OpenCV ja que ha permès la captura i tractament de les imatges en temps real i han ajudat altres llibreries com matplotlib.pyplot, pyfirmata, numpy i time.

Per instal·lar qualsevol llibreria de Python en la versió més actual, es necessari utilitzar pip, un programa per d'instal·lacions que ve per defecte a Python a partir de la versió 3.4. Primer es necessari anar a la carpeta

```
C:\Users\Usuari\AppData\Local\Programs\Python\Python310\Scripts
```

i copiar la direcció de la ruta. El següent pas es obrir la consola de comandes i escriure `cd C:\Users\Usuari\AppData\Local\Programs\Python\Python310\Scripts`.

Aquesta funció situarà la consola en la carpeta que hem seleccionat i ara s'introdueix el codi "`pip install NOM_DE_LA_LLIBRERIA`".

Quan la llibreria ja esta instal·lada, per utilitzar-la alhora de programar, a l'inici del programa s'ha de escriure `import NOM_DE_LA_LLIBRERIA`.

2.4.3 Llibreria d'Arduino

La llibreria Pyfirmata d'Arduino és una llibreria que ve per defecte quan s'instal·la l'aplicació. Aquesta llibreria utilitza el protocol Firmata, el qual permet la comunicació des de un software instal·lat en el mateix ordinador. La llibreria ja compte amb uns quants casos d'exemple per poder usar-la i per aquest treball s'ha utilitzat un d'aquests exemples. (Bruijn, 2017)

L'exemple utilitzat es coneix com a *StandardFirmata*. Per poder obrir aquest exemple, es necessari primer tenir l'aplicació d'Arduino oberta i anar al menú

2.4.4 Controlador PID

2.4.4.1 Velocitat

Per calcular la velocitat s'ha desglossat el circuit en quatre parts: les dues rectes i les dues corbes. Per calcular la velocitat del cotxe en les rectes s'han utilitzat les fórmules del Moviment Rectilini Uniforme i per calcular la velocitat del cotxe s'han utilitzat les formules del Moviment Circular Uniforme.

Rectes:

La fórmula per calcular la velocitat en les rectes es l'increment de l'espai dividit entre l'increment de temps:

$$v = \frac{\Delta x}{\Delta t} = \frac{x_{actual} - x_{anterior}}{t_{actual} - t_{inicial}}$$

La posició actual (x_{actual}) és el punt on es troba el cotxe actualment respecte al punt d'inici de la secció on es troba. La posició anterior ($x_{anterior}$) és la posició del cotxe en el instant de temps anterior. L'increment de temps es el temps en segons que transcorre entre que el cotxe es trobava a la posició anterior i quan es troba a la posició actual.

Corbes:

Per calcular la velocitat del cotxe en la corba, primer es va mesurar el radi de la corba i es van declarar les coordenades del centre de la corba. També, igual que a la recta, s'han determinat unes coordenades inicials. Per saber la posició del cotxe respecte al punt inicial han estat necessari dos vectors, un que representés la posició en l'eix X del cotxe respecte del centre i un altre que representes la posició en l'eix Y del cotxe respecte del centre.

La fórmula per saber la posició del cotxe es la següent:

$$L = \alpha * R$$

Donat que es tracte d'un triangle rectangle, per obtenir l'angle, es va utilitzar la fórmula trigonomètrica de la tangent:

$$\tan(\alpha) = \frac{CO}{CC}$$

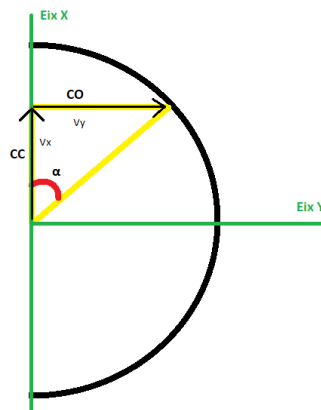


Figura 12: Dibuix per representar la posició del cotxe dins de la corba

Un cop es té l'angle i el radi, es calcula la velocitat dividint l'increment d'espai entre l'increment de temps com es fa amb la recta:

$$v = \frac{\Delta L}{\Delta t}$$

El diagrama de flux per determinar la velocitat va ser el següent:

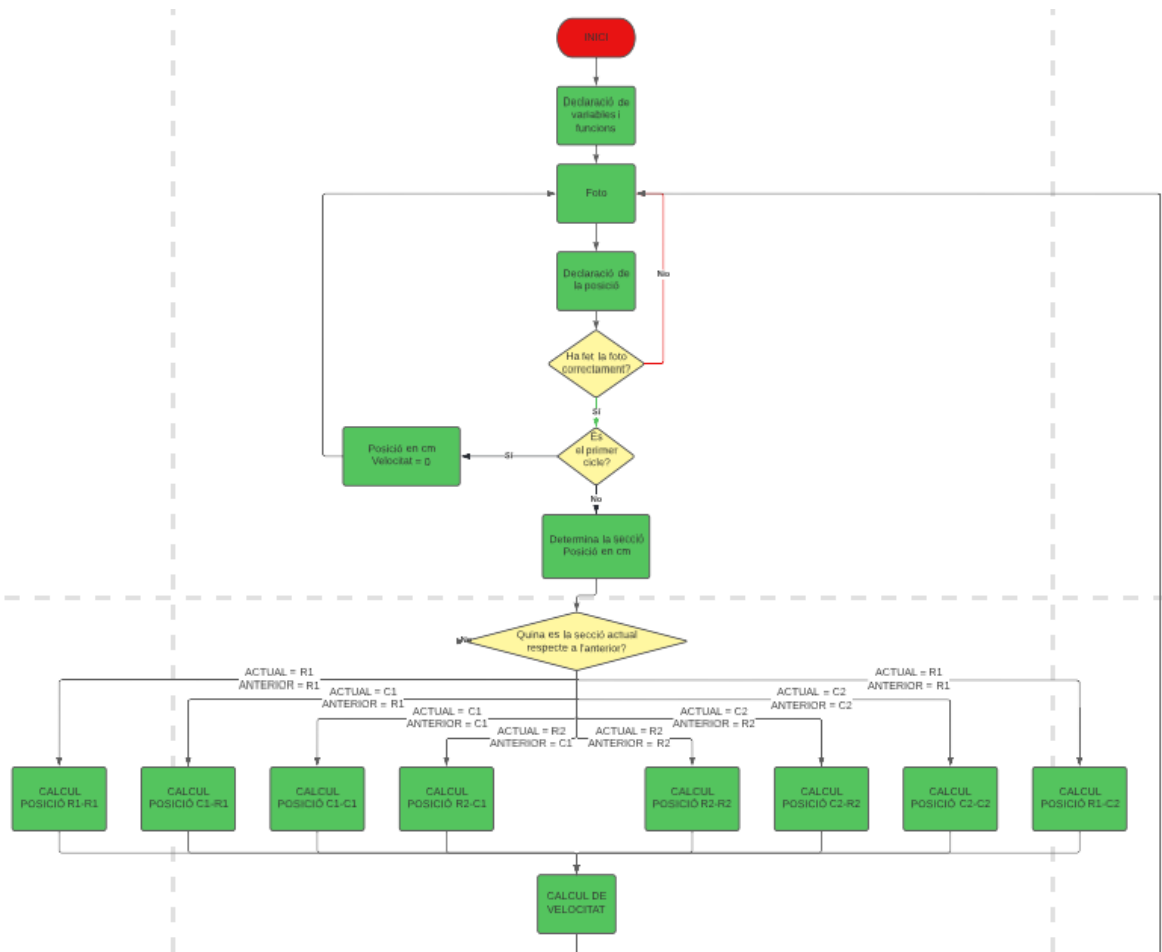


Figura 13: Diagrama de flux per a la determinació de la velocitat

2.4.4.2 Control PID

Un controlador PID, és un controlador en laç tancat que ajusta la sortida del mateix a una consigna desitjada.

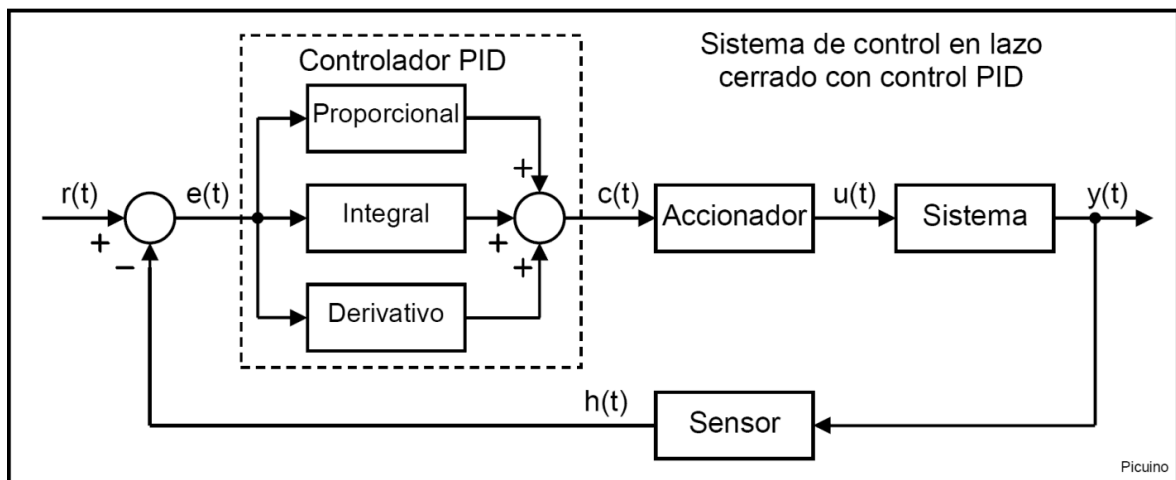


Figura 14: Diagrama de blocs d'un controlador PID (Martín, 2022)

En la figura, $r(t)$ és la consigna que se li envia al controlador per a que aquest tracti d'arribar a aquest valor. $H(t)$ és el valor de realimentació, en el nostre cas, es la posició actual del cotxe. $E(t)$ és l'error, la diferència entre la consigna que se li introdueix i el valor actual de la variable manipulada. A l'error, se li aplica el controlador i després amb un accionador s'aplica al sistema el valor de la variable manipulada.

Existeixen diferents tipus de controladors com controladors P, I, PI, PD, PID, etc. Per aquest projecte s'han fet un controlador P i un controlador PI i s'han comparat els seus resultats.

Un controlador Proporcional (P) és un controlador que multiplica la senyal d'error per una constant K_P . Es tracta d'un controlador molt simple i eficaç però poc exacte degut a que al multiplicar la senyal d'error per la constant, quan l'error es 0, el sistema tendeix a desestabilitzar-se. No obstant, tendeix a minimitzar l'error i presenta una resposta rapida.

$$y(t) = K_P \cdot e(t)$$

Un controlador Proporcional-Integral (PI) és un controlador proporcional que afegeix una integral de l'error. L'acció integral va acumulant les senyals d'error a mesura que avança el temps i per lo tant, cada cop es mes gran. De la mateixa manera en que al controlador Proporcional s'utilitzava la constant K_P , per a l'acció integral s'utilitza la constant K_I .

$$y(t) = K_P \cdot e(t) + K_I \cdot \int e(t)dt$$

2.5 Codi

2.5.1 Visió Artificial

La visió artificial ha estat un dels pilar d'aquest projecte. Ha estat necessari que la visió Artificial funcionés de la millor manera possible, amb la màxima capacitat de fotogrames per segon i sense cap retrac en l'obtenció d'imatges. Es va constatar que amb Matlab, es presentaven múltiples problemes a l'hora d'utilitzar la llibreria de Visió Artificial degut principalment a que en utilitzar la recol·lecció d'imatges i el tractament junt amb altres processos com els càlculs de l'algorisme de control o la simultaneïtat amb la llibreria d'Arduino, la visió Artificial agafava un retard de 30 segons entre que el cotxe es trobava a una posició i Matlab mostrejava la imatge. Aquest contratemps feia impossible el control a través de Matlab i per això es va optar per utilitzar Python.

La llibreria de Python de Visió Artificial es diu OpenCV i compta amb funcions tant per a la recol·lecció d'imatges com per el tractament de les mateixes. Per tal de poder utilitzar la visió Artificial correctament, es necessari declarar les condicions de lluminositat. L'aula compte amb quatre columnes de llums, es necessari encendre la segona i la quarta columna de llum començant a contar des de la columna mes pròxima a la porta d'entrada. Alhora, les persianes de l'habitació han d'estar baixades per evitar la contaminació lumínica de la llum solar en el sistema.

A l'inici del programa, es captura la imatge del circuit i es selecciona només l'àrea del circuit. Un cop es té la imatge amb l'àrea d'interès, es fa un filtre per poder aïllar el color taronja del cotxe amb el entorn. Per poder fer el filtre, es necessari primer passar la imatge de RGB a HSV. Es necessari convertir la imatge de RGB a HSV ja que a l'hora de crear un filtre amb un rang de colors, resulta més senzill utilitzar HSV ja que el color ve donat per només un valor mentre que per RGB es necessiten 3. Tot seguit, és necessari crear dos filtres, un per aïllar el color taronja i un altre per eliminar totes les impureses un cop aplicat el filtre, aquest procés es coneix com a "Erosió". Per aïllar el color, es crea una mascara que consta d'un rang entre dos tonalitats diferents. Per escollir quins valors s'utilitzen per el filtre, s'utilitza la següent imatge:

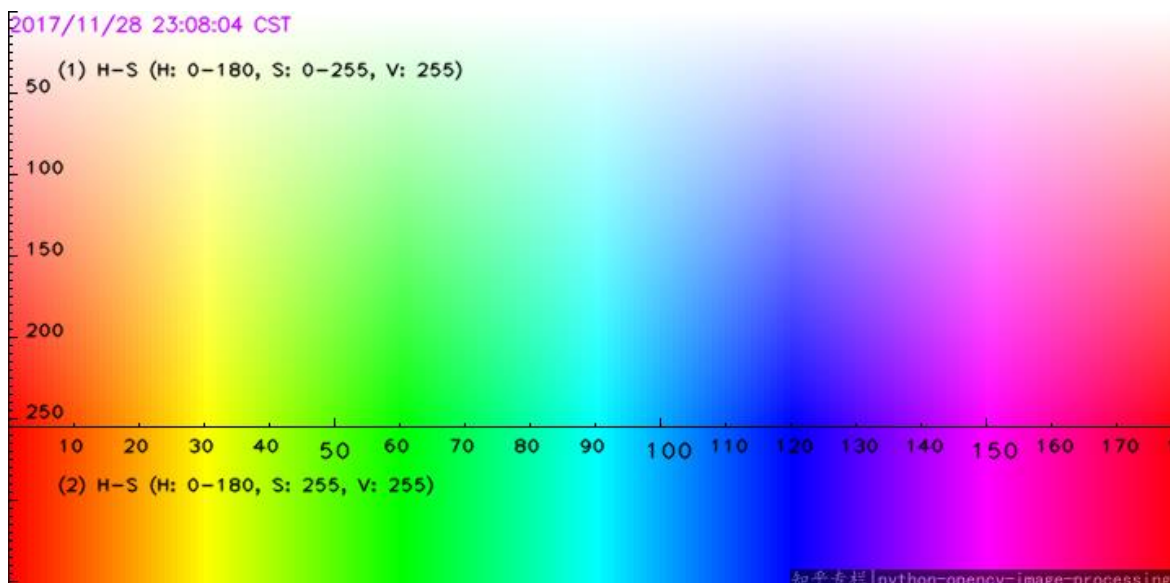


Figura 15: Rang de colors i valors en HSV (Solano, Omes, 2019)

HSV utilitza 3 valors per determinar un color. El primer valor determina la tonalitat, el segon la saturació i el tercer el valor. Per determinar un rang prou ample, el rang del filtre s'ha descrit entre (18, 100, 20) i (33, 255, 255) que si es mira a la imatge, es pot veure que correspon al color taronja. Un cop es té la mascara, s'aplica a la imatge HSV amb la funció *inRange* la qual s'encarrega de posar a 1 tots els bits dels quals el seu color entri dins del rang i posa a 0 tots els demés. La nova imatge amb el filtre de color, necessita un tractament d'erosió per eliminar les impureses. Aquest cop s'utilitza un filtre Kernel, el qual en aquest cas és una matriu 3x3 i qualsevol punt de la imatge que no tingui com a mínim un conjunt de bits 3x3 quedarà eliminat.

La resta del codi, s'encarrega de treure la posició del cotxe utilitzant la funció *findContours* de la mateixa llibreria.

La imatge per mostrar el filtre que s'aplica es el següent:

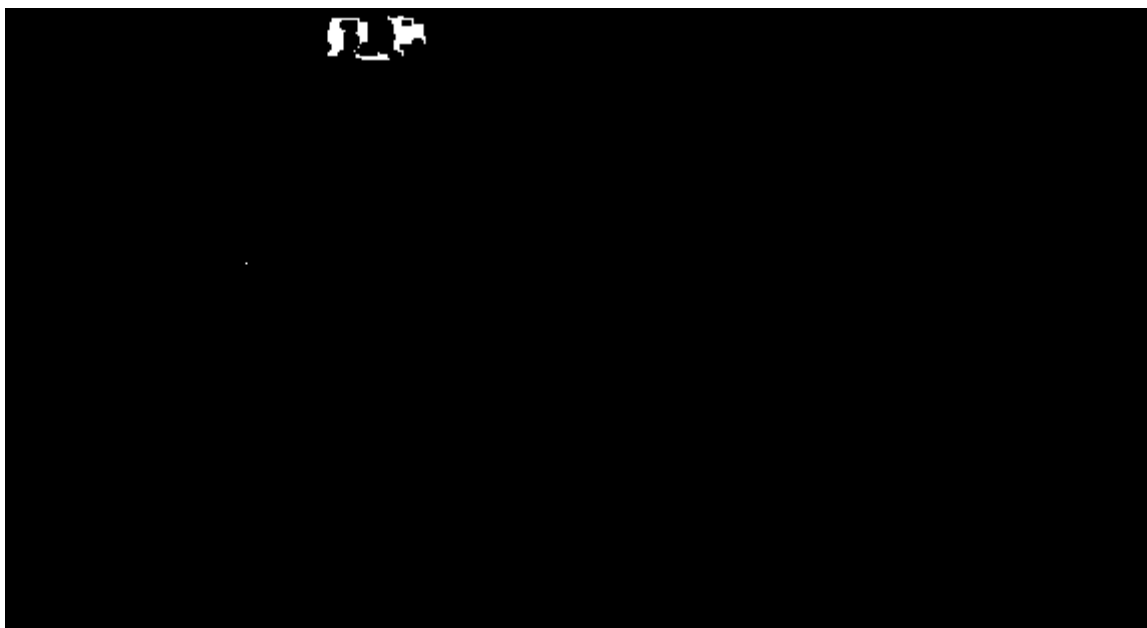


Figura 16: Imatge amb filtre

La imatge que es mostra, amb el filtre aplicat i el cotxe seleccionat es la següent:

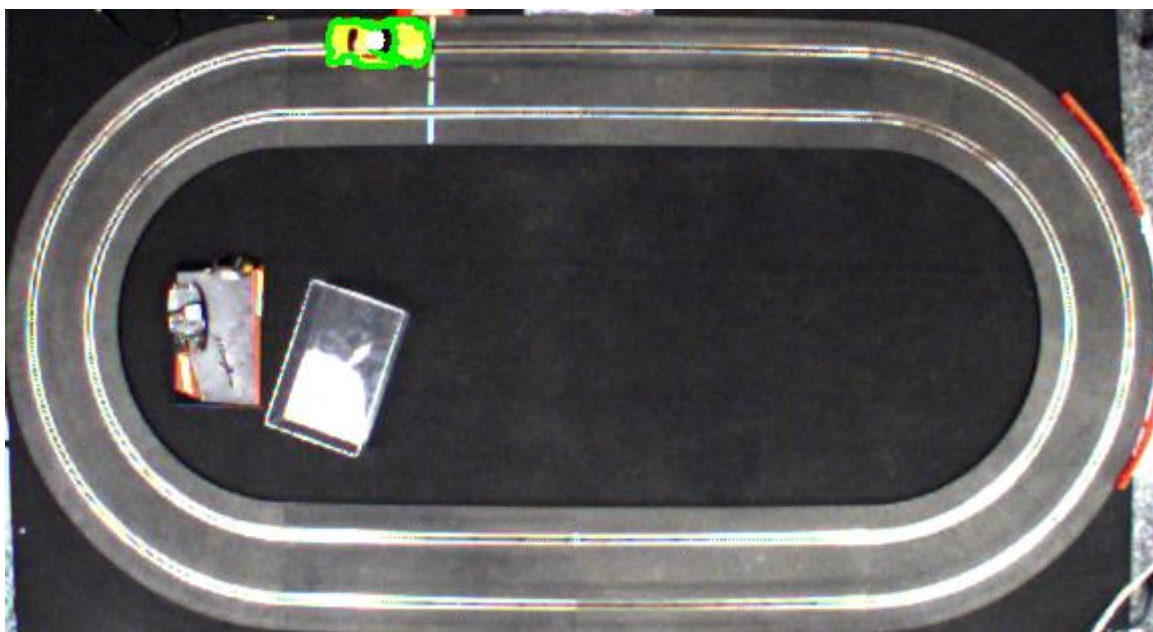


Figura 17: Imatge que capta la càmera amb el filtre aplicat i la detecció del cotxe

També es pot veure com l'adquisició de dades s'ha fet correctament ja que s'han mostregjat les coordenades X i Y en el següent gràfic

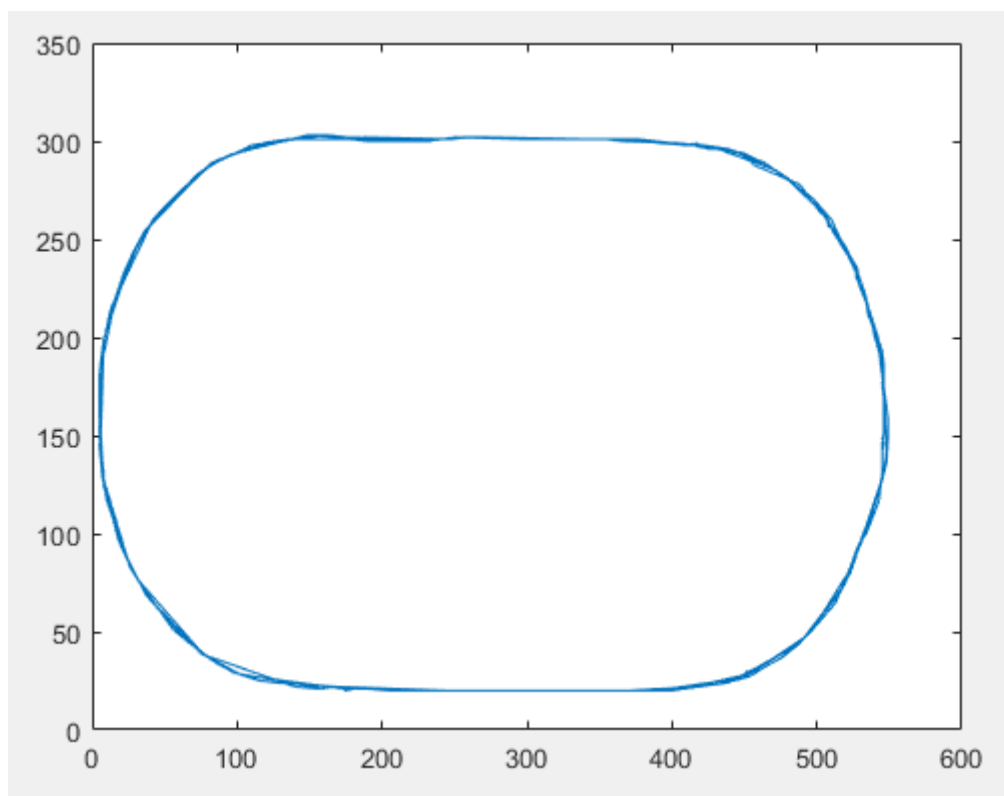


Figura 18: Coordenades X i Y durant el circuit

Les funcions utilitzades per a la recollida de dades són les següents:

```
import cv2
```

```
import numpy as np
```

```
from skimage.io import imshow, imread #Funcions per mostrar i recopilar imatges
```



```
from skimage.color import rgb2hsv, hsv2rgb #Funcions per conversió de la imatge
cap = cv2.VideoCapture(0)
object_detector = cv2.createBackgroundSubtractorMOG2()
currentFrame = 0
Inici = time.time() #Agafa el temps al inici del cicle
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read() #Captura la imatge
    frame = frame[115:430, 25:605] #Retalla la imatge
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) #Convertir imatge a HSV
    lower_green = np.array([18,100,20],np.uint8) #Limit inferior de color
    upper_green = np.array([33,255,255],np.uint8) #Limit superior de color
    mask = cv2.inRange(hsv, lower_green, upper_green) #Aplica la mascara
    kernel = np.ones((3, 3), np.uint8) #Filtre per eliminar impureses
    img_erosion = cv2.erode(mask, kernel, iterations=1) #Aplicar erosió per eliminar impureses a la mascara
    result = cv2.bitwise_and(hsv, hsv, mask=img_erosion) #Aplica la mascara a la imatge hsv
    res = cv2.cvtColor(result, cv2.COLOR_HSV2BGR) #Transforma el la imatge hsv a BGR
    contours_ = cv2.findContours(mask, cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE) #Utilitza la llibreria per trobar els
contorns del cotxe
    for cnt in contours:
        area = cv2.contourArea(cnt) #Extrau l'area de la zona seleccionada
        if area > 100: #En cas de que l'area blanca seleccionada sigui major a 100 es troba el centroide
            cv2.drawContours(frame, [cnt], -1, (0, 255, 0), 2) #Dibuixa els contorns de color verd
            M = cv2.moments(img_erosion) #Agafa el centroide del cotxe
            cX = int(M["m10"] / M["m00"]) #Calcula la posició del cotxe
            cY = int(M["m01"] / M["m00"])
            cv2.circle(frame, (cX, cY), 5, (125, 60, 152), -1) #Dibuixa un punt en el centroide
            print(cX, cY)
    ActualX = cX
    ActualY = cY
    cv2.imshow('Erosion', img_erosion)
    cv2.imshow('Result',res)
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    currentFrame += 1
cap.release()
cv2.destroyAllWindows()
```

2.5.2 Arduino

Per utilitzar l'Arduino, s'han emprat dues llibreries, una per Python i una altra per l'Arduino. La llibreria de Python es diu Pyfirmata i és necessari instal·lar-la tant a Python com a Arduino i aquesta permet establir una comunicació entre el programa Python i el microcontrolador Arduino.

En el codi de Python, és necessari instal·lar primer la llibreria com es menciona en el apartat 2.2.2. Un cop es té instal·lada la llibreria, és necessari indicar a través de la funció `pyfirmata.Arduino()` a quin port està connectada la placa i tot seguit es determina quin pin de la placa s'utilitzarà i amb quina finalitat amb la funció `board.get_pin()`.

La funció `pyfirmata.Arduino()` retorna un objecte amb el qual podrem utilitzar totes les operacions de les quals disposa l'Arduino i es necessari indicar el port el qual la placa utilitzarà a dins del dos parèntesis. Per aquest cas s'ha utilitzat de la següent forma: `board = pyfirmata.Arduino('COM4')`, on 'board' és el nom de l'objecte i COM4 el port que utilitzarem.

La funció `board.get_pin()` permet declarar un pin de la placa Arduino indicant el tipus de pin, el número i el mode en el que s'utilitzarà. En el nostre codi s'ha utilitzat de la següent manera: `outp = board.get_pin('d:3:p')`, on indiquem que s'utilitzarà el pin digital (d) número 3 i amb el mode PWM (p).

Un cop tenim l'objecte `outp` que fa referència al pin digital 3 en mode PWM, se li pot introduir a través de la funció `write()` un valor de velocitat que es trobi entre 0 i 1 i sigui del tipus float. Aquesta funció només cal executar-la un cop, no cal executar-la en bucle per enviar un valor de manera constant.

Respecte al Arduino, per instal·lar la llibreria és necessari obrir el programa, anar a la barra de menú a la part superior, picar a Arxiu → Exemples → Firmata → StandardFirmata. StandardFirmata és un programa que rebra la informació que enviem des de Python i l'executarà a la placa, es necessari descarregar el programa a la placa sense fer-li cap modificació. (Bruijn, 2017)

El codi utilitzat és:

```

import pyfirmata #Arduino
board = pyfirmata.Arduino('COM4') #Port on es connecta l'Arduino
outp = board.get_pin('d:3:p') #Pin digital 3 en mode PWM
outp.write(0.6) #Valor entre 0 i 1 per enviar com a sortida al pin 3
board.exit() #Tancar la comunicació amb l'Arduino
  
```

2.5.3 Algorisme de control

Es pot veure en el diagrama de blocs següent la implementació del controlador al sistema:

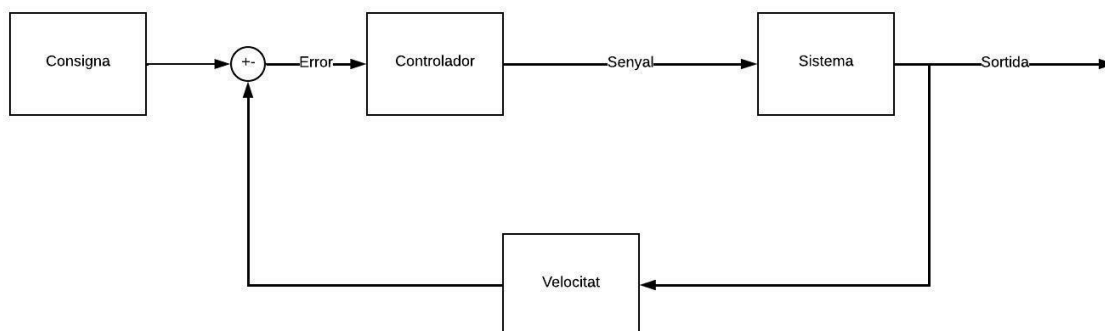


Figura 19: Diagrama de blocs

Per realitzar l'algorisme de control, es va començar amb el controlador proporcional i més endavant es va introduir una acció integral per fer el controlador PI.

Com s'ha mencionat prèviament, la sortida del controlador és el resultat de la multiplicació entre la constant de l'acció proporcional K_P i l'error. Com la funció `.write()` només ens permet un rang entre 0 i 1, es va limitar la sortida del controlador en un rang entre 0 i 1.

Quan la sortida del controlador era menor que 0, la sortida del controlador era 0 i quan la sortida del controlador era major que 1, la sortida del controlador era 1. Degut a la alta no linealitat del circuit, la rapidesa del sistema i a la freqüència de mostreig, la metodologia per poder determinar quina era la KP mes adient va ser a través de dues mesures, el ISE i la velocitat mitja.

El ISE (Integral square-error) és el índex que s'ha utilitzat per comparar els diferents resultats. L'índex és l'integral del error al quadrat, d'aquesta manera, s'ha comparat quin dels diferents resultats tenia l'ISE mes baix per determinar el millor resultat.

$$ISE = \int_0^T e(t)^2 dt$$

La velocitat mitjana va ser una mesura més imprecisa donades les altes variacions alhora d'obtenir les dades de la velocitat. A pesar de no ser una dada gaire precisa, també es va utilitzar per comparar la millora entre el controlador Proporcional i el Proporcional-Integral.

La secció de codi empleada per l'algorisme de control va ser la següent:

```
Error = setp - Velocitat
Output = KP*Error+KI*I
I=I+Error*(A-B)
if Output<0:
    Output = 0
elif Output>1:
    Output = 1
outp.write(Output)
ISE = ISE + Error*Error*(A-B)
```

2.6 Resultats

2.6.1 Programa final

A l'inici del programa, s'implementen totes les llibreries necessàries per el projecte. Tot seguit, es declaren totes les funcions i variables necessàries del programa.

Un cop s'han inicialitzat les variables i les funcions, s'entra en la fase d'adquisició i tractament on s'utilitza la llibreria OpenCV. Per obtenir les imatges, s'entra en un bucle on primer s'obté una imatge del circuit i després es retalla per poder treballar amb la secció de la imatge desitjada. Després es crea la mascara per poder aïllar el cotxe de slot del fons i després s'obtenen les coordenades del cotxe a través de la llibreria. Quan s'obté la posició del cotxe, també s'obté l'instant de temps en que s'han obtingut les coordenades del cotxe.

Després d'obtenir la posició del cotxe en píxels, s'entra en la fase de comandament. Depenent de la posició del cotxe, se li determina una secció dins del circuit, per exemple: Recta1, Corba1, Recta2 i Corba2. Un cop determinada la secció, també es passa la posició del cotxe en píxels a cm per comoditat. El següent pas, és calcular la diferència de posició respecte a la posició del cotxe en l'imatger anterior i després es calcula la velocitat del cotxe. Quan ja es te la velocitat del cotxe, s'entra en la part de l'algorisme de control. Primer es calcula l'error de velocitat restant la consigna de velocitat a la velocitat actual del cotxe. Un cop obtingut l'error, s'aplica el controlador PI i com s'ha mencionat anteriorment, la sortida es limita entre 0 i 1 i després s'envia la senyal a l'Arduino.

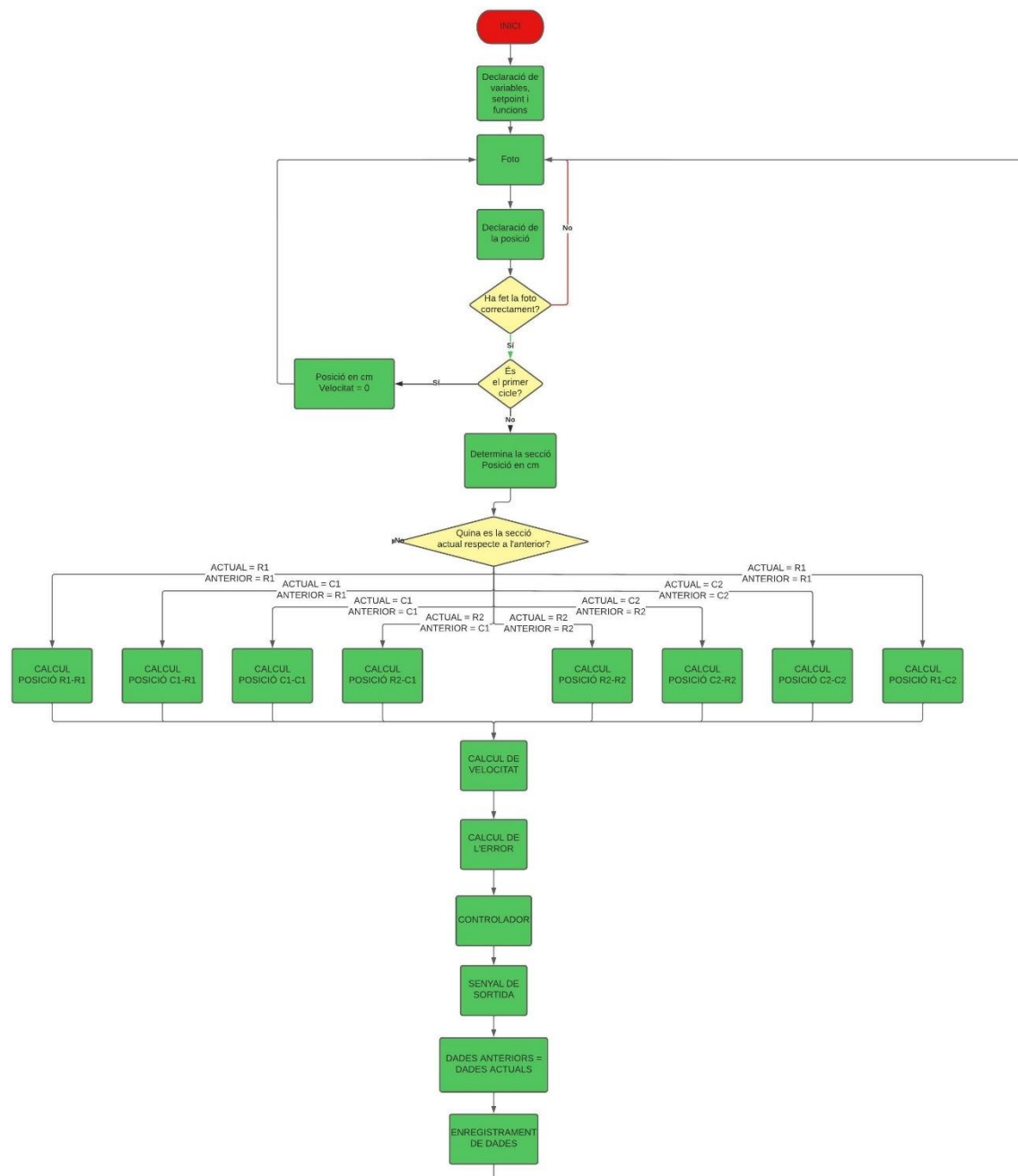


Figura 20: Diagrama de flux del programa complet

El codi final es el següent:

```

import cv2
import matplotlib.pyplot as plt
import pyfirmata #Arduino
import numpy as np #Llibreria per utilitzar la funcio arange
import time

cap = cv2.VideoCapture(0)
object_detector = cv2.createBackgroundSubtractorMOG2()
currentFrame = 0
    
```



```
setp = 100 #Declarar el setpoint en cm/s
KP=0.007 #Valor de KP
KI=0.0006 #Valor de KI
#Arduino
board = pyfirmata.Arduino('COM4') #Declarar Arduino
outp = board.get_pin('d:3:p') #Declarar el pin utilitzat i el mode
def Recta1(X): #Funció de la primera recta
    ActualX = (0.2614*(X-120)) #Conversió de pixels a cm, Entrada de la recta = 0 i sortida de la recta = 80
    return ActualX #Retorna la posició del cotxe en cm
def Corba1(X, Y): #Funció de la primera corba
    V2X = X-Cbx1 #Component X del vector entre la posició actual del cotxe i la posició d'entrada a la corba
    V2Y = Cby1-Y #Component Y del vector entre la posició actual del cotxe i la posició d'entrada a la corba
    Angle = np.arctan2(V2X,V2Y) #Arc tangent entre la component X i la component Y del vector.
    L = 37*Angle #Multiplica l'angle en radians per el radi en cm. El radi de les dues corbes es de 37cm. Entrada de la corba
    = 0 i sortida de la corba = 116,23
    return L #Retorna la posició del cotxe en cm
def Recta2(X): #Funció de la segona recta
    ActualX = (0.2614*(431-X)) #Conversió de pixels a cm, Entrada de la recta = 0 i sortida de la recta = 80
    return ActualX #Retorna la posició del cotxe en cm
def Corba2(X, Y): #Funció de la segona corba
    V2X = Cbx2-X
    V2Y = Y-Cby2
    Angle = np.arctan2(V2X,V2Y)
    L = 37*Angle
    return L
Cas = [[0, -1, -1, 7], [1, 2, -1, -1], [-1, 3, 4, -1], [-1,-1, 5, 6]] #Matriu dels diferents sectors
Iteracions = 0 #Variable que compte el nombre de cicles que s'han realitzat
Cbx1 = 431 #Variable que representa la coordenada X d'entrada de la primera corba
Cby1 = 157 #Variable que representa la coordenada Y d'entrada de la primera corba
Cbx2 = 120 #Variable que representa la coordenada X d'entrada de la segona corba
Cby2 = 157 #Variable que representa la coordenada Y d'entrada de la segona corba
Total = 0 #Variable que compte el temps total en segons
DTotal = 0 #Variable que compte la distància total en cm
Dades = open("DadesKP=0,007KI=0.0006.txt", "w") #Archiu en s'emmagatzemaran
cXant=0 #Posició X del cotxe
cYant=0 #Posició Y del cotxe
ISE = 0 #Variable ISE
I=0 #Variable acció integral
Voltes = 0;
while(True):
    while (True): #Bucle per evitar imatges repetides
        ret, frame = cap.read() #Captura l'imatge
        frame = frame[115:430, 25:605] #Retalla l'imatge
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) #Convertir imatge a HSV
        lower_green = np.array([18,100,20],np.uint8) #Limit inferior de color
        upper_green = np.array([33,255,255],np.uint8) #Limit superior de color
```

```

mask = cv2.inRange(hsv, lower_green, upper_green) #Aplica la mascara

kernel = np.ones((3, 3), np.uint8) #Filtre per eliminar impureses
img_erosion = cv2.erode(mask, kernel, iterations=1) #Aplciar erosió per eliminar impureses a la mascara
result = cv2.bitwise_and(hsv, hsv, mask=img_erosion) #Aplica la mascara a l'imatge hsv
res = cv2.cvtColor(result, cv2.COLOR_HSV2BGR) #Transforma el l'imatge hsv a BGR
contours,_ = cv2.findContours(mask, cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE) #Utilitza la llibreria per trobar
els contorns del cotxe

for cnt in contours:
    area = cv2.contourArea(cnt) #Extrau l'area de la zona seleccionada
    if area > 100: #En cas de que l'area blanca seleccionada sigui major a 100 es troba el centroide
        cv2.drawContours(frame, [cnt], -1, (0, 255, 0), 2) #Dibuixa els contorns de color verd
        M = cv2.moments(img_erosion) #Agafa el centroide del cotxe
        cX = int(M["m10"] / M["m00"]) #Calcula la posició del cotxe
        cY = int(M["m01"] / M["m00"])
        cv2.circle(frame, (cX, cY), 5, (125, 60, 152), -1) #Dibuixa un punt en el centroide
    if cX != cXant or cY != cYant:
        cXant = cX
        cYant = cY
    break

print(cX, cY) #Mostreja les coordenades del cotxe en pixels
A = time.time() #Instant de temps
cv2.imshow('Result',res) #Mostreja la mascara aplicada a la imatge
cv2.line(frame, (120,25), (431,25), (255,0,0), 2) #Mostreja una linea de color blau per poder calibrar el circuit amb la camara
cv2.line(frame, (120,300), (431,300), (255,0,0), 2)
cv2.imshow('frame',frame) #Mostreja la imatge

if Iteracions == 0: #En cas que sigui el primer cicle, declara les condicions inicials
    PosActual = Recta1(cX) #Rep la posició inicial del cotxe en cm
    PosAnterior = -1 #Degut a que es el primer cicle, es declara la posició anterior del cotxe com a -1
    SActual = 0 #El sector inicial es la recta 1
    SAnterior = -1
    Velocitat = 0 #La velocitat al inici es 0
    Distancia = 0 #La distància recorreguda al inici es 0
else: #En cas de que ja no sigui el primer cicle
    if cX in range(120,430) and cY in range(0,100): #En cas de que el cotxe es trobi en la recta 1
        print("Recta 1")
        SActual = 0 #Secció del cotxe es la recta 1
        PosActual = Recta1(cX)

    elif cX in range(431,565): #En cas de que el cotxe es trobi en la corba 1
        print("Corba 1")
        SActual = 1 #Secció del cotxe es la corba 1
        PosActual = Corba1(cX, cY)

    elif cX in range(120,430) and cY in range(250,350): #En cas de que el cotxe es trobi en la recta 2
        print("Recta 2")
        SActual = 2 #Secció del cotxe es la recta 2

```



```
PosActual = Recta2(cX)
elif cX in range(0,120): #En cas de que el cotxe es trobi en la corba 2
    print("Corba 2")
    SActual = 3 #Secció del cotxe es la corba 2
    PosActual = Corba2(cX, cY)
print("Sector ", SActual, " y ", SAnterior)
match Cas[SActual][SAnterior]: #Depenent del cas actual i l'anterior s'opera de formes diferents
    case -1:
        print("Fora de rang")
    case 0: #R1R1 #Posició actual = Recta 1, Posició anterior = Recta 1
        Distancia = PosActual - PosAnterior
        print("R1R1")
    case 1: #C1R1 #Posició actual = Corba 1, Posició anterior = Recta 1
        Distancia = PosActual + 80 - PosAnterior
        print("C1R1")
    case 2: #C1C1 #Posició actual = Corba 1, Posició anterior = Corba 1
        Distancia = PosActual - PosAnterior
        print("C1C1")
    case 3: #R2C1 #Posició actual = Recta 2, Posició anterior = Corba 1
        Distancia = PosActual + 116.18 - PosAnterior
        print("R2C1")
    case 4: #R2R2 #Posició actual = Recta 2, Posició anterior = Recta 2
        Distancia = PosActual - PosAnterior
        print("R2R2")
    case 5: #C2R2 #Posició actual = Corba 2, Posició anterior = Recta 2
        Distancia = PosActual + 80 - PosAnterior
        print("C2R2")
    case 6: #C2C2 #Posició actual = Corba 2, Posició anterior = Corba 2
        Distancia = PosActual - PosAnterior
        print("C2C2")
    case 7: #R1C2 #Posició actual = Recta 1, Posició anterior = Corba 2
        Distancia = PosActual + 116.18 - PosAnterior
        print("R1C2")
        Voltes = Voltes+1;
Velocitat = Distancia/(A-B) #Divisió entre l'increment de distància i l'increment de temps

print("PosActual ", PosActual, " i PosAnterior ", PosAnterior)
print("Distància ", Distancia)
print("Temps ", A-B)
PosAnterior = PosActual
Total = Total+(A-B)
DTotal = DTotal + Distancia

SAnterior = SActual
print(Velocitat)
print(" ")
```

```

Error = setp - Velocitat #Calcul de l'error
Output = KP*Error+KI*I #Controlador PI
I=I+Error*(A-B) #Acció Integral
if Output<0: #En cas de que la sortida sigui menor a 0, la sortida serà un 0
    Output = 0
elif Output>1: #En cas de que la sortida sigui major a un 1, la sortida serà un 1
    Output = 1
outp.write(Output) #Envia la senyal de sortida a l'Arduino
ISE = ISE + Error*Error*(A-B) #Calcul de ISE
Dades.write(str(Velocitat))
Dades.write('\t')
Dades.write(str(DTotal))
Dades.write('\t')
Dades.write(str(Total))
Dades.write('\t')
Dades.write(str(SActual))
Dades.write('\t')
Dades.write(str(cX))
Dades.write('\t')
Dades.write(str(cY))
Dades.write('\t')
Dades.write(str(Output))
Dades.write('\t')
Dades.write(str(Distancia))
Dades.write('\t')
Dades.write(str(A-B))
Dades.write('\t')
Dades.write(str(ISE))
Dades.write('\n')
B = A
Iteracions = Iteracions + 1;
if cv2.waitKey(1) & 0xFF == ord('q'):
    outp.write(0)
    break
board.exit()
cap.release()
cv2.destroyAllWindows()
Dades.close()

```

2.6.2 Algorisme de control

S'han dut a terme múltiples proves a fi de determinar els valors òptims de la KP i la KI.

Per determinar amb quina consigna treballar, es va crear un programa en Python on es mesurava la velocitat del cotxe en tot el circuit quan se li introduïa una senyal de 0.6 sobre 1 i el valor i es va determinar que la velocitat adequada era 100 cm/s. Es va començar amb l'acció proporcional únicament realitzant proves assajant diferents valors de KP i amb un setpoint.

Per tenir una orientació dels valors sobre els quals treballar, es va agafar la senyal de sortida més alta, es a dir, 1, i es va dividir entre l'error més alt possible, es a dir, la consigna de velocitat. Per lo tant, es va començar a tractar amb valor al voltant de 0,01:

$$Error = Consigna - Velocitat actual = 100 - 0$$

$$Senyal = KP * Error \rightarrow KP = \frac{Senyal}{Error} = \frac{1}{100} = 0,01$$

Amb aquest càlcul, es va començar a provar valors al voltant d'aquest valor. En cas de que tinguéssim un valor de KP molt elevat, el sistema es saturaria i en cas de que tinguéssim un valor de KP molt petit, el cotxe no tindria força per arrancar.

2.6.2.1 Millor rendiment ISE

2.6.2.1.1 KP

Es va començar amb un valor de KP=0.01 i es va veure un ISE de 66480. Com es mostra a la taula següent, es pot veure com el resultat amb un ISE més baix es el de la KP = 0,007. Als resultats es pot apreciar una no-linealitat que pot estar deguda a múltiples factor, per exemple, el fregament del circuit, el contacte de les escombretes amb els carrils, la baixa freqüència de mostreig i la rapidesa del sistema, etc. També, degut al fregament, el cotxe manca de força per desplaçar-se com més petit es el senyal.

KP	ISE
0.005	El cotxe no te força per moure's
0.006	63799
0.007	59725
0.008	60869
0.009	64452
0.01	66480
0.02	70090
0.05	79303
1	108024

Taula 2: Valor de ISE per cada KP

2.6.2.1.2 KP+KI

Utilitzant la constant escollida de KP de 0.007, aquests son els diferents valors de ISE provant diferents valors de KI.

KI	ISE
0.0002	61160

0.0004	54055
0.00055	54787
0.0006	51929
0.00065	56903
0.0008	53692
0.001	57460
0.005	74699

Taula 3: Valor de ISE per cada KI

Es pot veure com el valor de KI que millor ISE té es 0.0006. En l'apartat 2.6.2.3 es poden veure els gràfics del comportament del sistema amb aquests valors de KP i KI.

2.6.2.2 Millor rendiment mitjà

2.6.2.2.1 KP

En aquest cas, es pot veure que conforme van incrementant els valors de KP, els valors de la velocitat mitja també van incrementant, no obstant, degut a les grans oscil·lacions, el valor de velocitat mitjà no es un valor representatiu com es pot comprovar al veure el ISE ja que no tenen el mateix tipus de comportament.

KP	Velocitat mitja [cm/s]
0.006	30.54
0.007	45.17
0.008	49.47
0.009	72.79
0.01	82.13
0.02	108
0.05	116.7
1	120

Taula 4: Valor de velocitat mitja per cada KP

2.6.2.2.2 KP+KI

En aquesta taula es poden veure els resultats de la velocitat mitja provant diferents valors de KI mantenint una KP=0,007 constant. Es pot veure com afegint l'acció integral la velocitat mitja ha incrementat considerablement apropant-se a la consigna de velocitat, no obstant, a pesar de tot no pot arribar al valor de consigna desitjat.

KI	Velocitat mitja [cm/s]
0.0002	56.88
0.0004	65.67
0.00055	70.41
0.0006	72.38
0.00065	73.87
0.0008	77.41

0.001	79.94
0.005	95.75

Taula 5: Valor de velocitat mitja per cada KI amb una KP=0,007

2.6.2.3 Resultat escollit

El resultat amb el valor del ISE mes baix és el resultat amb una KP de 0.007. Per a analitzar el resultat, s'han utilitzat 5 gràfics que mostren el funcionament del cotxe al llarg de 4 voltes: Un gràfics de velocitat-temps, un de l'increment d'espai-temps, un de l'increment de temps-temps total, un de la senyal de sortida-temps total i un de l'ISE.

Degut a la no-linealitat del sistema, a la baixa freqüència de mostreig i a la rapidesa del sistema, el gràfic de velocitat apareix amb altes oscil·lacions. Tot i aquest fet, es pot veure com les oscil·lacions estan al voltant de la consigna de 100 cm/s. La línia taronja mostra la secció on es troba el cotxe en cada instant de temps on 0 es la recta 1, 100 es la corba 2, 200 es la recta 2 i 300 es la corba 3. Aquestes dades es mostrejant per veure la velocitat del cotxe en les diferents parts del circuit.

D'aquest gràfic també podem veure la velocitat mitjana i compara els resultats entre aplicar un algorisme de control amb acció integral i un algorisme de control sense acció integral. En el primer gràfic podem veure com varia la velocitat en funció del temps quan no s'aplica l'acció integral on podem veure que la velocitat mitja es de 45,17 cm/s i en el segon gràfic es pot veure la velocitat respecte el temps quan actua l'acció integral on la velocitat mitja es de 72.38 cm/s. Es pot veure una clara millora en la velocitat mitjana degut a l'acció integral i gràcies a les taules també podem veure una millora en el ISE.

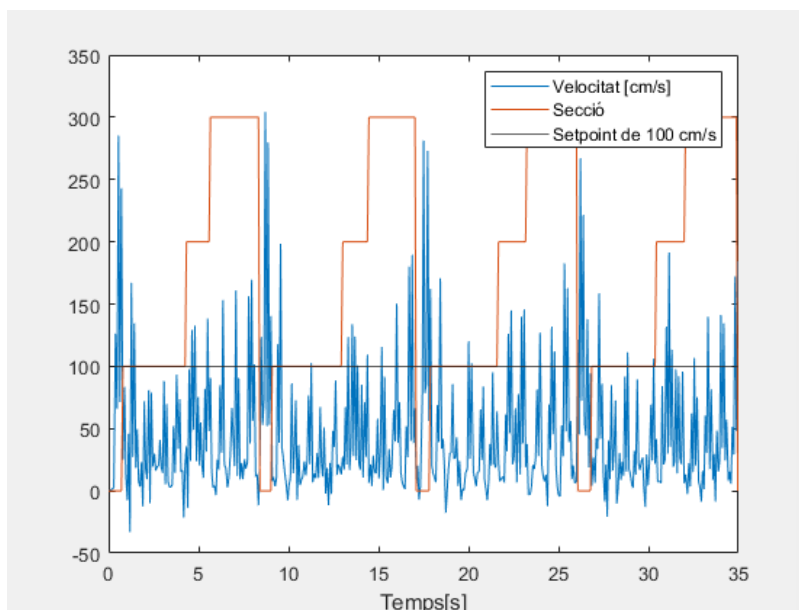


Figura 21: Gràfic de la velocitat (cm/s) en funció del temps (s) amb una KP = 0,007 i una KI = 0

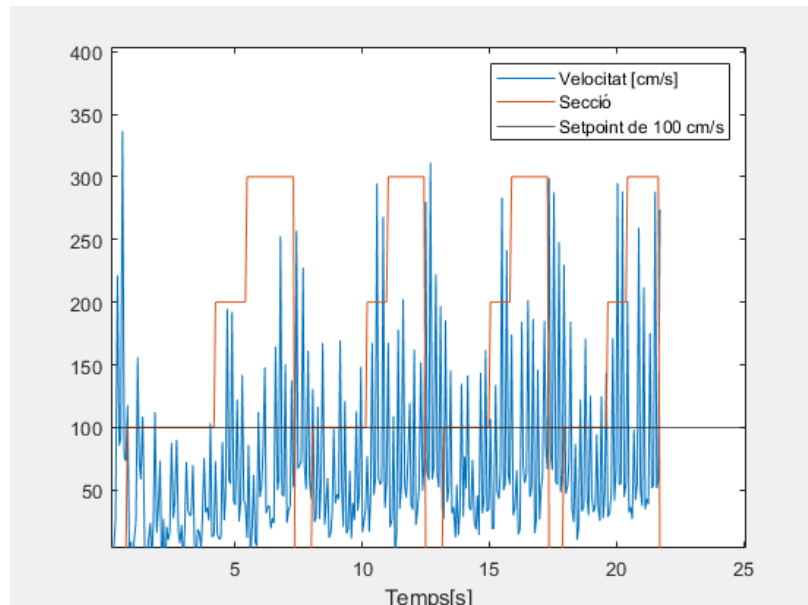


Figura 22: Gràfic de la velocitat (cm/s) en funció del temps (s) amb una $KP = 0,007$ i una $KI = 0,0006$

A pesar de no tenir un temps de mostreig constant, es pot veure com la diferència d'espai es força similar al gràfic de velocitat. També es pot veure com existeixen sobrepics deguts a la rapidesa del sistema.

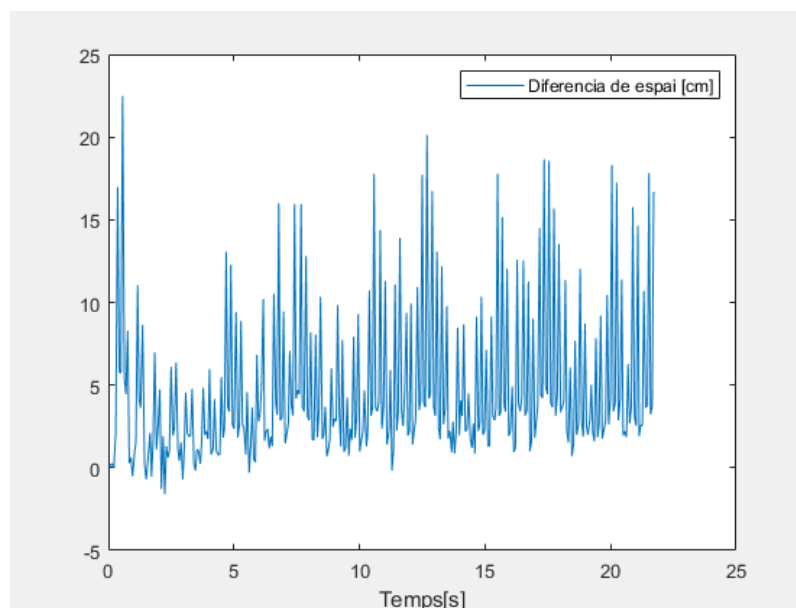


Figura 23: Gràfic de l'increment d'espai (cm) en funció del temps (s)

Degut a que es necessari la màxima rapides alhora d'obtenir noves imatges, en comptes d'establir un temps de mostreig fixe, s'ha fet el procés d'adquisició de forma cíclica. D'aquesta manera, realitzarà cada cicle de la manera mes rapida possible. En el gràfic, es pot veure com el temps de mostreig mitjà es de 0.06s.

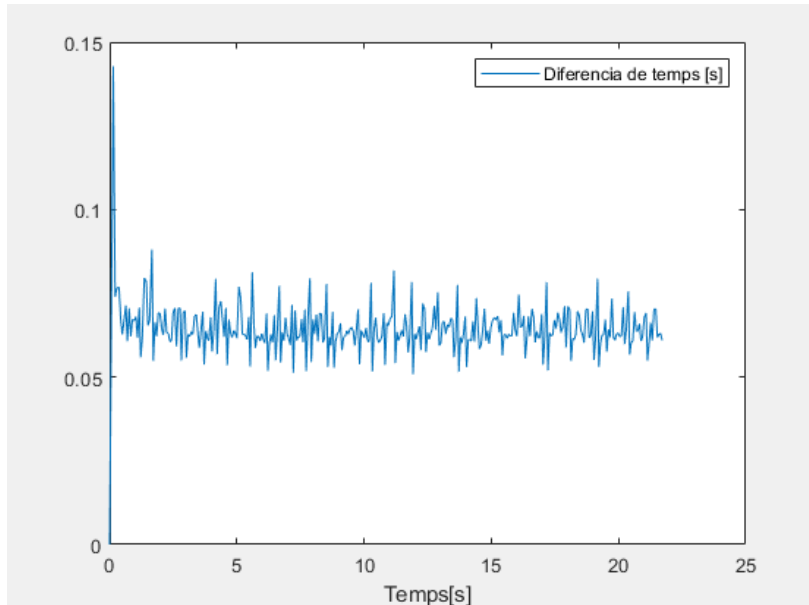


Figura 24: Gràfic de l'increment de temps (s) en funció del temps total (s)

En aquest gràfic es pot apreciar l'efecte de del controlador variant entre els valors de sortida de 0 i 1. Per comparar-ho amb una sortida saturada, també s'ha provat amb un valor alt de KP per mostrejar-ho.

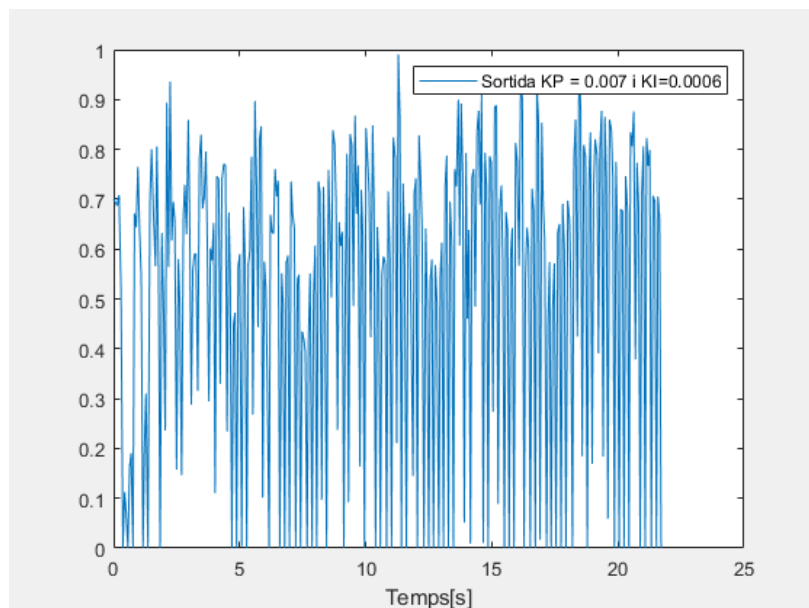


Figura 25: Gràfic del senyal de sortida en funció del temps (s) amb $KP = 0,007$ i una $KI = 0,0006$

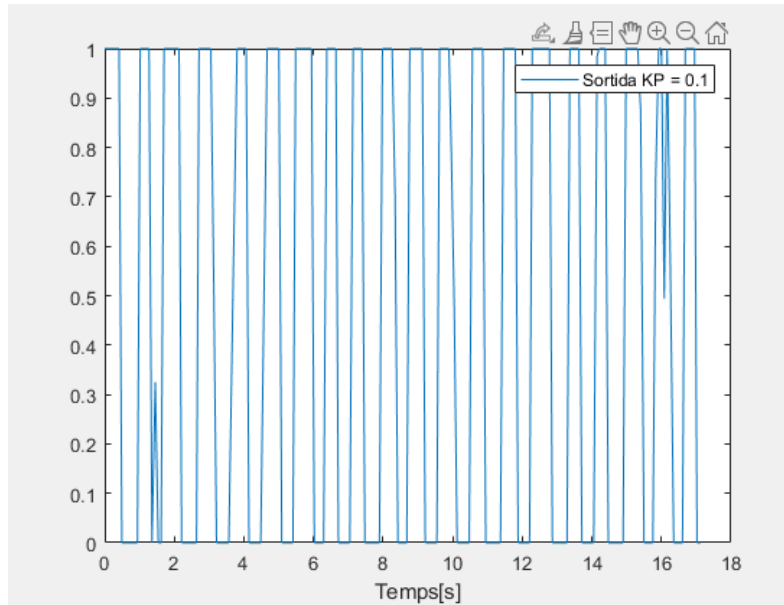


Figura 26: Sortida del controlador saturada amb una $K_P=0,1$

Per últim, es pot veure el gràfic del ISE per comprovar l'increment de l'error conforme va passant el temps. Aquest gràfic ha estat necessari per comparar amb altres resultats ja que analitzant la forma del gràfic podem saber si l'error es mes gran o mes petit.

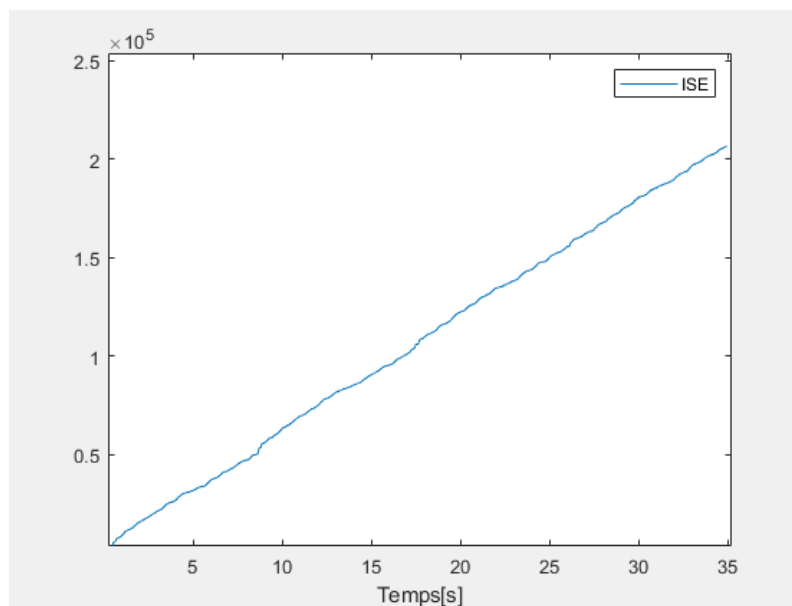


Figura 27: Gràfic de l'ISE en funció del temps (s) amb $K_P=0,007$ i una $K_I=0$

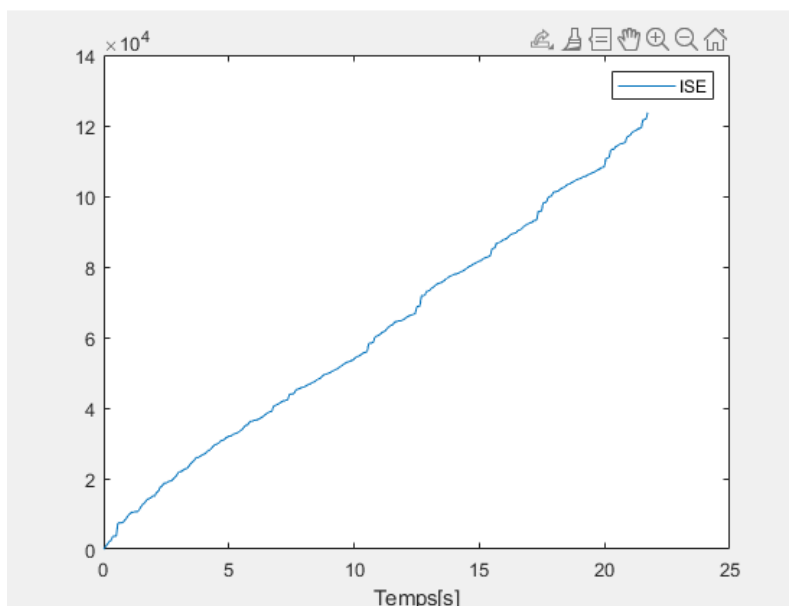


Figura 28: Gràfic de l'ISE en funció del temps (s) amb $KP=0,007$ i una $KI=0,0006$

3 Resum del pressupost

Com no ha estat necessari realitzar cap compra per aquest treball, s'han comptat les despeses com a les hores de treball que s'han dedicat per aquest treball i s'ha definit un temps de 8 euros per hora. La major part de les hores ha anat dedicada al disseny del controlador.

Capítol	Resum	Hores	Euros	%
1	Documentació.....	83	640,00	10,49
2	Preparació mate- rial.....	27	216,00	3,54
3	Visió.....	210	1640,00	26,90
4	Controlador.....	450	3600,00	59,05
TOTAL PRESSUPOST GENERAL		770	6096,00	

4 Conclusions

En aquest treball s'ha dissenyat un algorisme de control de velocitat d'un cotxe de slot per tal de que aquest assoleixi certes especificacions sense la interacció humana. Les especificacions seleccionades són una velocitat constant del cotxe de slot en cm/s al llarg de tot el circuit de slot, consistent en dos trams rectes i dues corbes. Inicialment es volia realitzar el treball amb l'eina Matlab, no obstant, es va comprovar que Matlab no era una bona eina alhora de processar simultàniament tant l'adquisició d'imatges i el tractament de les mateixes com l'algorisme de control degut a que quan es tractava d'executar tot alhora la visió artificial patia d'un retràs que feia impossible el control en llaç tancat.

Per resoldre els problemes amb la visió Artificial es va utilitzar la llibreria OpenCV amb la qual es va poder realitzar l'adquisició i tractament d'imatges evitant els inconvenients que presentava Matlab. És molt important que les condicions de lluminositat es mantinguin constants i sempre siguin les mateixes ja que si varien, el programa no pot identificar correctament el color del cotxe de slot.

Per realitzar el comandament va ser necessari aprofitar el circuit elèctric que es va dissenyar en l'anterior treball, no obstant, al no utilitzar el Matlab va ser necessari canviar el codi de l'Arduino i utilitzar la llibreria Pyfirmata.

També s'ha arribat a la conclusió que el circuit de slot, es un sistema massa ràpid per el hardware del que es disposa, per exemple, hagués estat necessari utilitzar una càmera amb una freqüència de mostreig mes alta amb la qual poder captar les imatges del circuit mes ràpid amb la finalitat de poder tenir una realimentació més precisa de la posició actual del cotxe. També, amb la finalitat de realitzar una adquisició i un tractament mes ràpid, hagués estat millor utilitzar C++ en comptes de Python degut a que C++ és un llenguatge de programació que s'executa mes ràpidament i comparteix varies de les llibreries que s'han utilitzat per aquest projecte.

Per últim, es va dissenyar un controlador PI, no obstant, com s'ha mencionat prèviament i degut a la rapidesa del sistema, no ha estat un controlador gaire precís. Hagués estat millor valorar altres algorismes de control que poguessis realitzar un millor control per aquest sistema o millorar el hardware per un mes ràpid.

5 Bibliografia

- Astro shop. (2022). *Astroshop*. Obtenido de <https://www.astroshop.es/camaras-para-astronomia/the-imaging-source-dbk-21au04-as-camara-de-colores-usb/p,11776>
- Bruijn, T. d. (07 de 12 de 2017). *pyFirmata Documentation*. Obtenido de <https://buildmedia.readthedocs.org/media/pdf/pyfirmata/latest/pyfirmata.pdf>
- Llamas, L. (26 de 08 de 2015). *Salidas analógicas PWM en Arduino*. Obtenido de <https://www.luisllamas.es/salidas-analogicas-pwm-en-arduino/>
- Martín, C. P. (09 de 2022). *Picuíno*. Obtenido de Tecno Recursos: <https://www.picuíno.com/es/control-pid.html>
- Solano, G. (14 de 09 de 2019). *Omes*. Obtenido de Detección de colores en OpenCV – Python (En 4 pasos): <https://omes-va.com/deteccion-de-colores/>
- Solano, G. (14 de 09 de 2019). *OMES*. Obtenido de Detección de colores en OpenCV – Python (En 4 pasos): <https://omes-va.com/deteccion-de-colores/>

6 Annex

Inicialització del sistema

Primer de tot, es necessari establir les condicions de lluminositat adequades. Com s'ha mencionat amb anterioritat, les persianes de l'aula han d'estar baixades del tot i la segona i quarta columna de llum contant des de el cantó més pròxim a la porta han d'estar enceses. Després, es necessari encendre l'ordinador i executar l'aplicació de "Càmera" que porta per defecte Windows. Això es degut a que hi ha cop en que la càmera, al iniciar-la, té un filtre de color groc que desapareix al obrir l'aplicació. Després, cal tancar l'aplicació i obrir el programa Python.

En següent pas consisteix en connectar l'Arduino al ordinador a través del cable USB 2.0 tipus A-masclé i B-masclé i també el cable jack to jack tant al ordinador com al circuit de slot. Per últim, cal connectar la font d'alimentació de 12V al circuit elèctric afegit al Arduino.

El següent pas es executar el codi de Python de la càmera per poder quadrar les línies blaves amb els carrils superior e inferior per calibrar la posició del circuit.

Ja per últim, cal agafar el cotxe de slot i col·locar les escombretes en una posició perpendicular a la base del cotxe, de tal manera en que al col·locar el cotxe en la posició d'inici del carril exterior, les escombretes facin contacte tota l'estona amb el carril. Un cop posicionat el cotxe amb la part davantera mirant cap al a la recta 1 i d'esquenes a la corba 2, es pot executar el codi Python. Per aturar el codi, cal prémer la lletra q.

Codi en Matlab per mostrejar dades

```
clear all
close all
load 'DadesKP=0,007KI=0.0006.txt'
a=DadesKP_0_007KI_0_0006;
figure(1);
plot(a(:,3),a(:,1));
xlabel('Temps[s]')
hold on
plot(a(:,3),a(:,4)*100);
yline(100);
legend('Velocitat [cm/s]', 'Secció', "Setpoint de 100 cm/s")
figure(2);
plot(a(:,3),a(:,8));
xlabel('Temps[s]')
legend('Diferencia de espai [cm]')
figure(3);
plot(a(:,3),a(:,9));
xlabel('Temps[s]')
legend('Diferencia de temps [s]')
figure(4);
plot(a(:,3),a(:,7));
xlabel('Temps[s]')
legend('Sortida KP = 0.1')
figure(5);
plot(a(:,3),a(:,10));
xlabel('Temps[s]')
legend('ISE')
disp(a(127,3))
disp(a(127,10))
```