Cotxe Intel·ligent de la Freescale Cup

Aitor Herrera Corredera

Resum— La Freescale Cup és una competició de cotxes autònoms en la qual els estudiants de diferents universitats, tots amb el mateix cotxe que proporciona Freescale, programen un algorisme de comportament que fa que aquest cotxe sigui capaç de recórrer un circuit marcat per una línia negra sense cap tipus de comandament humà directe. En aquest article veurem com vam desenvolupar el treball per tal de programar el nostre propi algorisme de funcionament autònom per al cotxe que s'utilitza en aquesta competició i com vam crear un programa per a comprovar que tots els components físics del vehicle funcionen correctament així com una *API* per a facilitar una possible programació d'altres algorismes en un futur. Per a realitzar aquestes tasques, vam recórrer a fonts externes per a obtenir les dades dels components físics que disposàvem així com fragments de codi per a fer funcionar cadascun d'aquests individualment i els vam sintetitzar en un codi únic que era capaç de fer funcionar tots aquests a la vegada. Fet això, es va programar l'*API* per a facilitar l'última part del projecte, la programació de l'algorisme final de comportament autònom. Finalment, vam aconseguir que el cotxe seguís el circuit marcat amb la pròpia *API,* havent superat diferents obstacles sorgits durant el projecte, amb un resultat final satisfactori.

Paraules clau—*API,* Freescale Cup, components físics, búffer, servomotors, càmera lineal, comportament autònom, Freescale KL25Z, freqüència de mostreig, processament de la imatge.

Abstract— The Freescale Cup is an autonomous car competition where students from many universities must programme a behaviour algorithm to steer a car provided by Freescale. This algorithm makes the car navigate through a track by following a black line and without any direct human control. In this article, we will examine how we carried out this project to program our own autonomous operating algorithm for the car used in this competition and how we created the software to check that all the physical components of the vehicle were working properly, as well as an API to facilitate other algorithm programming in the future. To achieve these tasks, we made use of external sources to obtain information about the physical components we had and code snippets to operate each of them individually. Then we synthesize these code snippets in a unique code that made all the components work at the same time. Having done so, we programmed the API to facilitate the last part of the project: the programming of the autonomous behaviour final algorithm. Finally, we made the car follow the marked circuit with its own API. So, having overcome several obstacles encountered throughout the project, the result was satisfactory

Index Terms— API, Freescale Cup, physical components, buffer, servomotors, line-scan camera, autonomous behavior, Freescale KL25Z, sampling frequency, image processing.

1 Introducció

Des de l'any 2003 existeix en l'àmbit educatiu a nivell d'universitats una competició en la qual cadascuna d'aquestes ha de programar un cotxe intel·ligent i realitzar una cursa per a demostrar les seves habilitats i per a incentivar l'aprenentatge als alumnes d'una manera més pràctica i entretinguda. Aquesta cursa consta d'un circuit marcat normalment per unes línies negres les quals els cotxes han de seguir sense cap mena de comandament directe, és a dir, només amb els mètodes que tinguin per a captar la realitat via components electrònics i amb el codi que hagin programat els alumnes per a aquesta funció, que està integrat a un circuit electrònic amb un processador que porten els cotxes integrats, els participants no han d'afegir res.

Durant el transcurs d'aquesta competició, les dues grans dificultats que han de superar els alumnes que participin en aquesta són, per una banda, el desconeixement previ que tenen sobre els components del cotxe i per l'altra, la realització d'un algorisme que permeti el reconeixement del circuit i doni les ordres adients al vehicle per tal que segueixi el recorregut correctament i arribi a la meta sense desviar-se.

El problema que se'ns planteja actualment en el nostre treball és el de programar un d'aquests cotxes per tal que tingui el comportament autònom que es requereix a la competició de la Freescale Cup[1] així com elaborar un programa per tal de comprovar que tots els components físics del cotxe funcionen correctament. A més, aquests dos problemes els haurem de solucionar una *API [2]* que també haurem de crear nosaltres per a facilitar la programació de codi relatiu al comportament físic i motriu del cotxe.

En aquest article s'exposarà tot el procés seguit per a la realització i acompliment de tots els reptes plantejats anteri-

- *E-mail de contacte: aitor.herrera@e-campus.uab.cat*

- *Menció realitzada: Enginyeria de Computadors.*
- *Treball tutoritzat per: Miquel Àngel Senar Rosell (Departament d'arquitectura de Computadors i Sistemes Operatius)*
- *Curs 2016/17*

orment. Així, veurem els objectius que ens hem marcat inicialment, el context en el qual es troba el projecte, la metodologia utilitzada per a aconseguir-ho i els resultats finals obtinguts. Per últim, elaborarem unes conclusions relatives als objectius marcats inicialment i als resultats finals obtinguts.

2 Objectius

Per a aquest treball, ens hem plantejat un seguit d'objectius per tal de poder elaborar-ho de manera ordenada de manera que ens ajudés a arribar al punt final correctament. Aquests objectius que volem assolir en aquest el projecte són:

- Crear una *API* per tal de facilitar la programació del cotxe de la Freescale Cup, tenint en compte els components específics que conté aquest i, utilitzant aquesta *API*, crear un programa que permeti a l'usuari comprovar el correcte funcionament de tots els components del cotxe sense necessitat de tenir un coneixement profund del dispositiu i de la programació d'aquest.

- Programar el cotxe de la Freescale Cup per a recórrer un circuit marcat per una línia negra de manera autònoma utilitzant els recursos físics que aquest ens proporciona, que serà principalment reconeixent la línia negra amb la càmera que porta incorporada, dirigint-lo amb el servo davanter i movent-lo utilitzant els dos motors de darrere.

3 Estat de l'art

Actualment, la normativa de la competició de la Freescale Cup obliga a tots els participants a utilitzar un model concret de cotxe, que només produeix Freescale. Aquest té uns components electrònics fixes que no es poden modificar, ja que suposaria un avantatge respecte a la resta de competidors. Els components dels quals disposa aquest són una càmera, un parell de motors per a la capacitat motriu del vehicle, un servomotor per a la direcció d'aquest i una placa base, la Freescale KL25Z[3], que és la que s'encarrega de tot el funcionament del cotxe.

Quant al comportament autònom, actualment encara està en fase de desenvolupament a nivell mundial, fet que es pot observar a les grans empreses automobilístiques que encara no tenen cap model en circulació que sigui capaç de circular de manera autònoma. El cas que s'apropa més a un comportament autònom que està en circulació actualment és el de l'*Autopilot* [4] de Tesla[5]. Aquest reconeix els carrils per on circula el cotxe i, el que és més avançat, és capaç de dirigir el cotxe, fent girar les rodes si és necessari, i fer-lo seguir el carril. A més, és capaç de detectar els cotxes que l'envolten i realitzar possibles maniobres d'emergència en cas que algun d'aquests fes un moviment que posés en perill la integritat del Tesla.
Per al comportament autònom en l'àmbit quotidià, com que la quantitat de situacions que es pot trobar un cotxe és infinita, no té sentit intentar programar el comportament d'aquest davant de totes les possibles situacions.

En el cas que engloba aquest treball sí que ens plantegem programar totes les possibles situacions pel fet que aquest vehicle recorrerà un circuit controlat, creat expressament per a aquest escenari, en el qual només hem de tenir en compte una variable per a la presa de decisions, que és la línia negra del terra. En canvi, en el cas comentat del cotxe real existeix l'aprenentatge autònom[6] que permet que no s'hagi de programar cadascun dels casos reals sinó que es programen un seguit de pautes/característiques comunes a diferents tipus de situacions reals. Així, no es programa el comportament que ha de tenir el vehicle per a cada possible situació sinó que es programa el comportament per a cada tipus de possible situació i, després d'haver-ne experimentat a la realitat, es van emmagatzemant les diferents experiències i es van creant els comportaments adients per a cadascun.

D'altra banda, en el camp del tipus de programació per a comportament autònom de vehicles hi ha diferents opcions. El més comú és la utilització del llenguatge de programació C (també C++)[7] per a interactuar amb els components físics, ja que aquest és un dels llenguatges d'alt nivell[8] de més baix nivell.[9], fet que facilita aquesta tasca. No obstant això, per a programar en llenguatge C pel que fa als components s'ha de tenir uns coneixements profunds tant del llenguatge com de les comunicacions entre components. A causa d'això, existeix una altra opció per a poder programar el funcionament del vehicle per a aquells que no tinguin aquests coneixements tan profunds o bé que vulguin enfocar el problema del funcionament autònom de manera més matemàtica. Per a aquests, està l'opció de programar el cotxe amb Matlab[10].

Matlab proporciona opcions per a fer dissenys basats en model, perfectes per a sistemes encastats com el que tenim nosaltres al nostre projecte, a més de la capacitat d'interactuar amb codi escrit en C. Tot això fa que aquesta opció tingui molta força en l'actualitat a l'hora d'interaccionar amb la realitat gràcies a la potència que té aquest quant a les funcions matemàtiques i de matrius perfecte, per exemple, per a tractament d'imatges.

A més, aquesta opció permet enfocar-se molt més en la part del processament de les dades que es capten de la realitat i en com s'ha de comportar el vehicle respecte a aquestes i no ocupar-se tant del camp de com captem les dades de la realitat o com es comuniquen els diferents components. Això fa que la persona que s'encarrega de programar el funcionament autònom (ja sigui en el cas de la Freescale Cup o en qualsevol altre cas) tingui més temps per a poder treballar els aspectes que s'encarreguen del comportament del vehicle i no pas consumeixi tant de temps en els aspectes de la programació de les comunicacions dels diferents components o de la programació a baix nivell de tractament té imatges, molt més tediós que no pas programar-ho amb les diferents llibreries i opcions que proporciona Matlab.

4 METODOLOGIA

Per a elaborar les diferents parts de les quals consta el projecte, primerament vam haver de recaptar informació dels components físics del cotxe per tal de poder afrontar el problema correctament. Una vegada feta aquesta part, vam procedir a la programació dels diferents mòduls per a complir els objectius. Així, podem dividir la metodologia en dues parts, la que engloba la recerca dels components físics disponibles i la que engloba l'elaboració del software. Primerament enumerarem els components físics del cotxe i després aprofundirem en el procediment realitzat per a programar el software.

4.1 Components físics

- Cotxe: Englobem en aquest apartat els components del cotxe físics que no formen part del conjunt elèctric/electrònic. Són el xassís, quatre rodes i el sistema de torsió i amortització del cotxe per si el ferm és irregular no fer malbé els components.

Figura 1. Xassís del cotxe

- Freescale KL25Z: Placa base de la marca Freescale de cost molt reduït que compta amb un processador ARM Cortex M0+ que treballa a 48Mhz, té 128 KB de memòria flash, 16KB de SRAM, interfície OpenSDA i, el que ens interessa, conversor Analògic-Digital.

Figura 2. FRDM KL25Z

- Càmera lineal amb sensor TSL1401CL[11]:Càmera que capta una imatge lineal de 128 píxels amb un temps d'exposició des de 267 microsegons fins a 68 mil·lisegons.

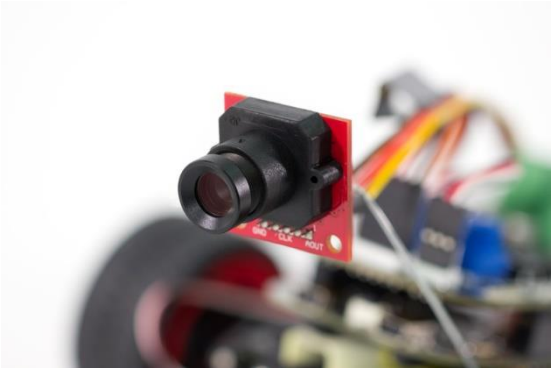

Figura 3. Càmera

- Servomotor: Motor que funciona amb corrent continu modulat per amplada de pols [12].

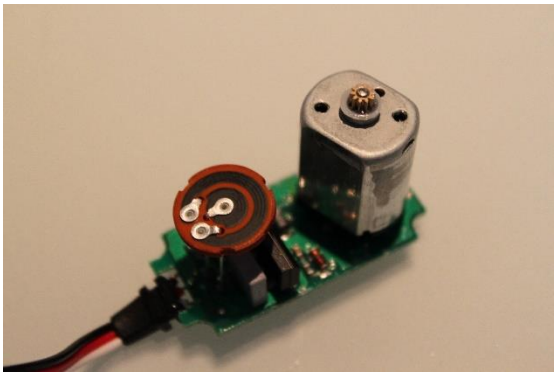

Figura 4. Servomotor

- Placa de control dels motors: Placa superposada i connectada a la KL25Z que s'encarrega del control dels motors del cotxe. Permet fer la modulació per amplada de pols i la regulació del voltatge per al control dels motors de les rodes.

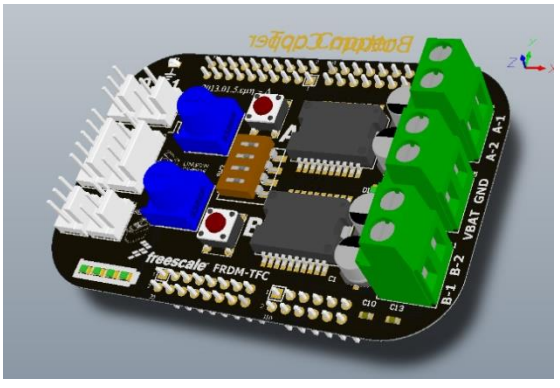

Figura 5. Placa de control dels motors

- Motors: Motors de corrent continu que s'encarreguen de fer girar les rodes motrius. Treballen a 7,2 volts podent girar fins a 16000 revolucions per minut (sense càrrega).

Figura 6. Motor DC

4.2 Elaboració del software

Per a l'elaboració del software, una vegada ja teníem clars els components físics que estan presents al cotxe i que necessitaríem controlar, vam procedir a buscar a fonts externes (internet) codis de control de cadascun d'aquests components individualment per a poder elaborar un únic codi que els englobi tots.

Vam poder obtenir els codis de tots els components, vam elaborar el codi per a poder controlar tots els components del cotxe sencer i, a partir d'aquí, l'*API* per a facilitar la programació dels diferents algorismes de comportament. Així, la disposició dels diferents mòduls del codi finalment va quedar de la següent manera:

- Mòdul orientat a possibilitar el funcionament dels components de la placa i la interacció entre ells: Els diferents codis que formen aquest mòdul els hem extret de la pàgina oficial del cotxe i s'encarreguen d'inicialitzar els components als quals el seu nom es refereix i de permetre les comunicacions entre aquests.
  El nom d'aquests són *Clock* per al rellotge, *sysTick* per a les freqüències de funcionament de la placa, *ADC* per a tots els conversors de senyals analògiques a digitals, *Terminal* per a poder utilitzar una terminal senzilla que interactués amb la placa en temps real, *UART* per a la interacció entre perifèrics, *ARM* per a la configuració de la placa i *Queue* per a la implementació dels buffers de dades que farem servir per a transmetre les dades de la imatge.

- Mòdul orientat als components primaris del moviment del cotxe, com son *Motor* per als motors principals que fan de força motriu, compost per les funcions *initMotorPWM()* que inicialitza els canals de comunicació i els registres necessaris per a fer funcionar els motors i *setMotorPWM(motorA, motorB)*, que s'encarrega d'aplicar els valors passats per paràmetre als dos motors del cotxe; *Servo* per al servomotor que permet dirigir el cotxe movent les rodes davanteres, que inclou les funcions *initServos()* per a inicialitzar les comunicacions amb els servomotors, *setServoDutyCycle (dutyCycle)* per a establir el temps entre polsos dels servos; per últim, *Camera* per a poder captar la imatge real i processar-la, que inclou les funcions *initLineScanCamera()* que inicialitza la càmera amb els seus *buffers* i components i setLineScanExposureTime(time) que establirà el temps que trigarà la càmera a fer una presa de dades.

- Mòdul anomenat *Movements*, en el qual hi ha les funcions encarregades de facilitar la tasca per al programador amb funcions que, treballant amb els altres dos mòduls, són molt intuïtives quant a nomenclatura-funcionalitat per tal de fer-ho tot senzill.
  Aquestes funcions són *goForward([speed])* que s'encarrega de fer que el cotxe vagi cap endavant, a la qual se li pot passar per paràmetre o no un valor que serà el que s'establirà com a velocitat, *turnRight([degrees])* i *turnLeft([degrees])* que s'encarreguen de fer girar el cotxe a dreta o esquerra respectivament i pot passar-se per paràmetre els graus que es vol que giri el cotxe i la funció *goBackward([speed])* que fa el mateix que *goForward()* però a la inversa, és a dir, que el cotxe vagi cap al darrere. Totes aquestes funcions comentades, si no se'ls hi passa per paràmetre un valor, estabiran un valor fixat per nosaltres.

Estant assemblat tot el codi que ens permet controlar tots els components del cotxe i l'*API*, vam procedir a l'elaboració del programa de testeig dels components i de l'algorisme de funcionament del cotxe.

El programa de testeig de components va ser el primer que vam elaborar, perquè així també comprovàvem tot el funcionament dels components del cotxe, no fos cas que algun d'aquests no funcionés correctament.

Per a fer-ho vam utilitzar el mòdul *Terminal*, amb el qual vam programar un petit menú al qual accedirem connectant-nos via port sèrie des de qualsevol ordinador amb un programa client de connexions, en el nostre cas el *Putty*. Aquest menú disposarà per pantalla a l'usuari unes opcions de moviment del cotxe bàsiques, que seran girar les rodes a la dreta o a l'esquerra, fer girar les rodes endavant o cap a darrere i captar imatge amb la càmera. L'usuari introduirà amb el teclat un dels números de les diferents opcions i el cotxe executarà el moviment que aquest indicava un període de temps curt i tornarà a mostrar el menú.

Un cop acabat el programa de testeig de components vam procedir a realitzar la funcionalitat final del cotxe: l'algorisme de funcionament autònom.

Primerament s'havia de captar la realitat mitjançant l'ús de la càmera col·locada al frontal del cotxe. Aquesta càmera capta una línia de 128 píxels en horitzontal que depenent del color de cada píxel el transforma en un voltatge o un altre. A causa d'aquest comportament, el primer que vam haver de fer és transformar aquesta diferència de voltatge en senyals digitals per tal de poder treballar amb les dades.

Per a fer-ho vam utilitzar el conversor d'analògic a digital que disposa la placa base. Les dades, un cop transformades a format digital, podem observar que (en l'àmbit del circuit que ha de seguir el nostre cotxe) gairebé tot el vector que conté els valors obtinguts és igual excepte una quantitat fixa de posicions que té un valor diferent. Aquesta és la línia negra amb la qual vam haver de treballar.

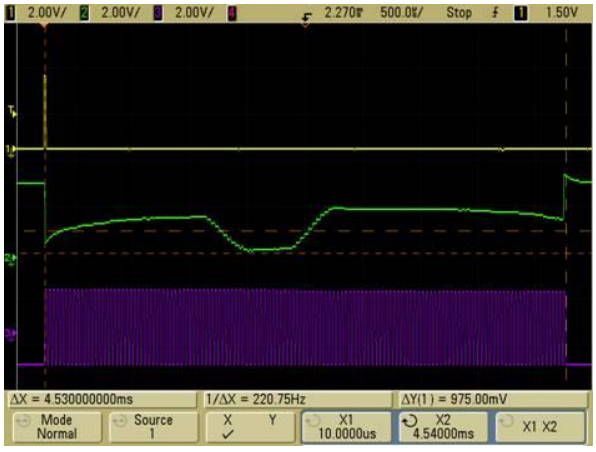

Figura 7. Dades que obtenim de la càmera

Cada vegada que la càmera capta una línia de píxels, aquesta imatge es processa i es guarda a un vector per tal de fer els càlculs adients per a reconèixer la posició de la línia i prendre les decisions corresponents respecte al comportament del cotxe.

La freqüència de mostreig de la càmera la variem manualment al codi. Inicialment vam indicar un temps de mostreig de cinc mil·lisegons però ens vam adonar que encara que puguem tenir aquesta velocitat de captura, no ens és del tot útil perquè la freqüència de treball del servomotor és menor i encara que el programa sigui capaç de veure que hi ha un canvi en la trajectòria hi haurà moments en els quals el servo encara no estarà disponible per a actuar. Per això, vam decidir descartar les imatges obtingudes mentre el servo no està disponible per tal de no carregar innecessàriament el processador.

Per al processament de la imatge vam prendre la decisió d'utilitzar la derivada d'aquesta per a trobar la posició de la línia. Els càlculs que realitzem són els següents:

1. Primerament, a causa de la qualitat del senyal que capta la càmera filtrem la imatge. Per a fer-ho calculem el valor mitjà de tots els píxels captats i analitzem píxel a píxel si aquest té un valor superior a aquesta mitjana o no. Si el valor és inferior a aquest es manté. En canvi si el valor és superior, assignem el valor mitjà a aquest punt.
2. Derivem píxel a píxel la imatge captada. Com que els píxels que capturem estan immediatament al costat dels altres, el càlcul es simplifica i és simplement la diferència entre el punt que volem calcular la derivada i el punt anterior (o el següent).
3. Tenint la imatge derivada, busquem la posició del píxel en el qual hi hagi una decrement gran del valor de la derivada (això es degut a que, si observem el gràfic de la figura 6, quan passem de blanc a negre la gràfica comença a decréixer i, per tant, el pendent és negatiu.
4. Al trobar aquest punt, busquem el punt on la derivada és igual a zero.
5. Per últim, busquem el punt on la derivada passa de ser un valor elevat a tornar-se zero o un número proper a aquest, que serà el punt on la gràfica torna a posar-se plana (on acaba la línia negra).
6. Si hem trobat aquest últim punt, podem assegurar que el punt situat entre el primer i l'últim punt trobat és el centre de la línia.

Una vegada que tenim el punt on està situat el centre de la línia decidir la direcció que ha de prendre el cotxe és molt senzill: si el centre de la línia està situat a un píxel d'índex menor a seixanta-dos, girar a l'esquerra. Si està situat entre seixanta-dos i seixanta-quatre continuar recte. Si està situat a un píxel de posició major a seixanta-quatre girar a la dreta. En les següents imatges podem comprovar com funciona el cotxe:

- El cotxe circula recte perquè la línia negra està al centre d'aquest. Podem observar la imatge processada en format de gràfica.

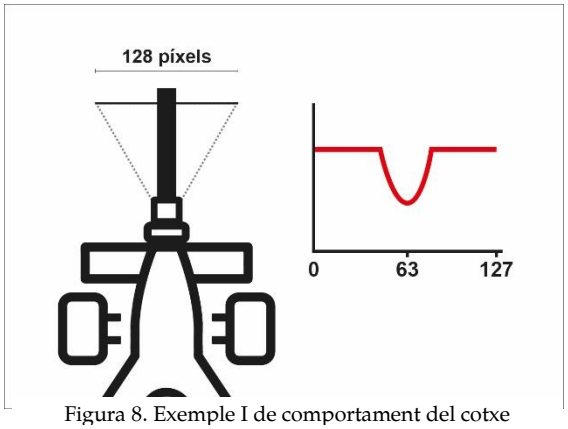

Figura 8. Exemple I de comportament del cotxe

- El cotxe detecta que la línia s'ha mogut. Com que el pic de la gràfica està situat en un píxel d'índex superior al seixanta-quatre l'algorisme fa que les rodes girin a la dreta per tal de seguir el camí marcat.

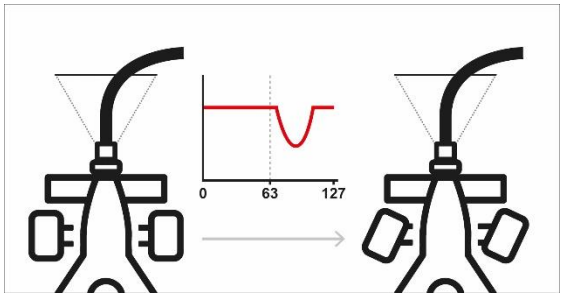

Figura 9. Exemple II de comportament del cotxe

Quant a la velocitat que utilitzem per a moure el cotxe és variable:

- Si el cotxe està actualment circulant per una línia recta, la velocitat augmenta fins al nivell màxim que indiquem que, depenent de si volem que la bateria ens duri més (per a realitzar llargues sessions de testeig, mode estalvi d'energia) o si volem que el cotxe recorri el circuit el més ràpid possible (mode màxim rendiment).

- Si el cotxe arriba a una corba, depenent de la posició de la línia negra respecte a l'eix central, reduïm la velocitat. Si la posició de la línia està entre cinc i vint píxels reduïm la velocitat una cinquena part de la velocitat actual en el cas d'estar en mode estalvi de bateria i dues cinquenes parts en el cas d'estar a màxim rendiment. Si la posició de la línia està situada més enllà de vint píxels de l'eix central, reduïm la velocitat a la meitat en el cas d'estalvi d'energia i tres cinquenes parts en el cas de màxim rendiment.

5 PLANIFICACIÓ I PROBLEMES SORGITS

Per a elaborar la planificació del treball es van tenir en compte els diferents objectius que es volien complir, si aquests tenien un ordre cronològic o bé era indiferent l'ordre de realització i el possible pes de cada part del treball. Les diferents tasques principals del projecte i el seu temps estimat d'elaboració van quedar de la següent manera:

Taula 1, assignació de mòduls i setmanes de temps

Tasca	Inici (Setmana)	Final (Setmana)
Programació de l'*API*	4	7
Programació programa proves de components	7	10
Proves motors	7	8
Proves servomotors	7	9
Proves càmera	7	10
Programació Proves funcionament autònom del cotxe	10	15

El color del fons marca el conjunt de les diferents tasques que vam plantejar.
Com podem observar vam dividir tot el projecte en tres blocs diferenciats.

1. El primer bloc va ser on vam realitzar la cerca dels programes dels diferents components del cotxe i la síntesi d'aquests en un de sol, elaborant l'*API* final. Durant l'elaboració d'aquest bloc tot va anar perfectament i vam complir els terminis establerts.

2. El segon bloc es va destinar a la prova dels diferents components del cotxe i a l'elaboració del programa de testeig d'aquests. Fins a aquest punt, la planificació es complia perfectament. A més, vam acabar aquesta part una setmana abans del previst.

3. L'últim bloc és el que vam assignar a l'elaboració de l'algorisme de funcionament autònom del cotxe. Durant aquesta fase vam experimentar diversos problemes que van ocasionar un retard de dues setmanes aproximadament en l'acompliment d'aquest bloc.

La fase final del projecte es va endarrerir respecte a la planificació realitzada originalment. No obstant això, el projecte es va acabar amb temps suficient perquè a l'hora de realitzar aquesta planificació vam tenir en compte la nostra inexperiència al camp i vam deixar un marge de dues-tres setmanes al final del projecte, de tal manera que si no es complia la planificació a temps (fet que era molt probable tenint en compte la nostra inexperiència) tindríem temps per a corregir-ho.

Com hem vist anteriorment, vam tenir un retard en la planificació a causa dels problemes sorgits durant el desenvolupament del projecte.

Aquest retard en la planificació ha estat degut a dos problemes concrets, tots dos relacionats amb la càmera del cotxe i el funcionament d'aquesta, que són:

- Inicialment, després de programar l'*API* bàsica, a l'hora de provar el funcionament de la càmera, no rebíem el conjunt de dades esperat. Després de comprovar que tot el codi implementat no tenia errors i d'investigar a les pàgines oficials de Freescale ens vam adonar que la font del problema era que la càmera estava mal connectada a la placa. Com no ens havíem plantejat la possibilitat d'un error de *hardware*, el temps que ens va ocupar la resolució d'aquest problema va ser excessiu.

- El següent problema que ens va sorgir, una vegada connectada la càmera, va ser que la lectura que ens proporcionava aquesta era completament plana en tot moment, no reaccionava a canvis de colors, com per exemple al passar la línia negra per davant. Aquest problema era ocasionat perquè la part de darrere de la càmera, és a dir, la placa d'aquesta, a l'estar a exposada a la llum,

contamina la lectura del sensor, ja que la llum que hi incideix per darrere és molt més directa que no pas la que capta per l'òptica. La solució que hem aplicat a estat col·locar al darrere de la placa un element que impedeix que la llum incideixi.

A causa aquests dos problemes ens vam endarrerir unes dues-tres setmanes aproximadament del temps que teníem establert per a la tasca, quedant finalment que la finalització del treball es va realitzar a la setmana divuit aproximadament, arribant a complir els terminis establerts per la universitat. Per tant, amb la modificació final realitzada, la tercera fase del projecte quedaria de la següent manera:

Taula 2, assignació final del mòdul de l'algorisme

Tasca	Inici (Setmana)	Final (Setmana)
Programació Proves funcionament autònom del cotxe	10	18

6 RESULTATS

Havent finalitzat el projecte, els resultats obtinguts d'aquest són molt positius, ja que: l'elaboració de l'*API* per a facilitar la programació del cotxe ha estat finalitzada amb èxit, inclou els tres mòduls principals enfocats a les diferents funcionalitats comentats a l'apartat de metodologia. El programa de testeig de components funciona correctament i ens permet comprovar que tots els components del cotxe que intervenen en el moviment d'aquest funcionen correctament de manera senzilla i intuïtiva i l'algorisme de funcionament autònom del cotxe s'ha elaborat amb èxit i funciona correctament. El cotxe és capaç de seguir la línia negra que marca el recorregut del circuit sense problemes.
No obstant això, quant al funcionament de l'algorisme de comportament autònom del cotxe existeixen limitacions perquè compleixi el seu objectiu:

- La velocitat de moviment del cotxe ha de ser reduïda (aproximadament d'un metre per segon) de no ser així el cotxe no és capaç de seguir la línia.
- Les corbes del circuit no poden ser tancades. Si ho són, el cotxe perd la línia i és probable que no pugui finalitzar el recorregut amb èxit.

Amb això observem relacions entre aquest i els nostres camps de coneixement que adquirim durant els estudis del grau.

La primera part del projecte, la programació de l'*API* està molt relacionada amb els coneixements estudiats de l'àmbit de la interconnexió de components/perifèrics amb microprocessadors, en la qual hem de tenir en compte el tipus de dades que podem obtenir dels diferents sensors del *hardware*, les freqüències amb què treballa cadascun d'aquests, adaptar-ho per tal que tot funcioni harmònicament i aconseguir establir les comunicacions, amb els busos disponibles, sense cap problema i, a partir d'aquí, utilitzar els recursos disponibles amb la màxima eficiència possible. Tots aquests aspectes d'interconnexió de mòduls els trobem semblants, encara que en menor escala, als casos amb què ens podem trobar en qualsevol àmbit de treball en el qual existeix un *software/ hardware* que té diversos mòduls en els quals estan treballant un equip d'enginyers, cadascun centrat en una part concreta però que després han de combinar i tot ha de funcionar perfectament sense conflictes originats en la programació.

Per una altra banda, a l'hora de programar el funcionament autònom del cotxe hem aplicat els coneixements adquirits al grau sobre la intel·ligència artificial, més concretament en l'apartat dels algorismes de comportament d'elements de funcionament autònom per al reconeixement del circuit i la presa de decisions a l'hora d'afrontar una recta o una corba, encara que els hem hagut de combinar amb una branca que no hem cursat, que és la de la visió per computador, fet que ens ha dificultat la tasca.

Aquest fet ens ha demostrat que, en la realitat, l'estimació dels temps que podem trigar a realitzar una tasca de la qual no tenim coneixements és fàcil que no sigui acurada i triguem més del temps que inicialment havíem previst i és per això que en l'àmbit professional es busquen persones amb molta experiència per als càrrecs que s'han d'encarregar de realitzar planificacions de projectes.

7 CONCLUSIONS

Amb el desenvolupament total del projecte, primerament extraurem conclusions sobre el resultat final respecte als objectius que ens vam marcar a l'inici del projecte i més endavant extrapolarem aquestes conclusions amb l'aprenentatge que guanyem pel projecte.
Les conclusions respecte als objectius inicials del nostre projecte són:

1. L'*API* realitzada compleix amb els objectius per la qual es va proposar i facilita la programació de codi senzill per tal de poder realitzar petites proves amb el cotxe, de tal manera que qualsevol persona sense coneixements en profunditat dels components d'aquest pot utilitzar les funcions incloses en aquesta *API* i fer funcionar els diferents components *hardware* del cotxe.

2. El programa de proves de funcionament de components físics compleix perfectament la seva funció i permet als usuaris comprovar de manera ràpida i senzilla si els components del cotxe funcionen correctament o no.
3. El funcionament autònom del cotxe s'ha aconseguit i funciona perfectament, de manera que el cotxe és capaç de reconèixer la línia negra que marca el circuit i de reaccionar als canvis en aquesta, girant i reduint/augmentant la velocitat quan el circuit ho requereix, sempre que la velocitat del cotxe sigui d'un metre per segon i que les corbes del circuit no siguin tancades.

Tot i que els objectius marcats inicialment s'han completat correctament, han sorgit una sèrie de problemes durant el transcurs d'aquest que han ocasionat retards en l'execució final.

En la primera part del projecte en la qual vam haver de recaptar informació i diferents mòduls de codi de diferents components per a després assemblar-los tots conjuntament vam observar que, encara que en el nostre cas no vam tenir excessiva dificultat perquè els components que havíem d'utilitzar eren molt quotidians, trobar fragments de codi ja elaborats pot ser una tasca molt difícil o gairebé impossible, a causa de la individualitat de cada sistema. No obstant això, el guany que ens va proporcionar poder adaptar fragments de codi ja existents va ser molt positiu, fet que inicialment no havíem tingut en compte, ja que la nostra idea inicial era programar cada component manualment. Aquesta conclusió la tindrem en compte per a futurs projectes i serà una de les alternatives principals a l'hora d'afrontar problemes de l'àmbit de sistemes multicomponents o sistemes encastats.

Quant al programa de testeig de components, una vegada acabada l'*API*, la dificultat per a elaborar-ho va ser reduïda. Gràcies a aquest fet, hem après que quan s'elaboren projectes d'aquest tipus, sovint és més productiu consumir temps en facilitar tasques futures, encara que potser no és indispensable fer-ho, per a reduir el cost que tindran algunes seccions més avançades del projecte i així agilitzar l'execució final d'aquest. A més, a causa d'un dels problemes que vam tenir durant el desenvolupament del nostre projecte ens hem adonat que tenir qualsevol tipus de mòdul (encara que sigui molt bàsic) per a poder comprovar el correcte funcionament dels components físics pot estalviar-te molt temps i molts problemes deguts a fallades en els components que en una primera instància no t'adones que són d'aquest tipus i perds molt de temps buscant el problema i la solució a nivell software.

En un futur, podríem avançar aquesta part del projecte desenvolupant un programa de testeig extern en l'entorn de desenvolupament, fent una interfície molt més amigable per a l'usuari en la qual aquest pogués simplement inicial el programa de testeig i aquest, tot automatitzat, anés comprovant un a un els components del cotxe i mostrés per pantalla, de manera molt visual, els resultats obtinguts sense la necessitat de tenir ni instal·lat ni obert l'entorn de desenvolupament.

Amb l'execució de la part final del projecte en la que havíem de programar l'algorisme de comportament del cotxe hem après que el procés de captació de la realitat i el posterior processament de les dades és un camp molt extens i complicat, més encara degut a la nostra inexperiència. Així, la gran majoria dels problemes que ens van sorgir en tot el procés van ser a aquesta última part.

Actualment, la metodologia utilitzada per a processar la imatge captada de la realitat i l'algorisme de presa de decisions són molt senzills, podríem dir que bàsics. Una possible millora en un futur del projecte seria canviar la manera de processar la imatge per tal de reconèixer més ràpidament els canvis de direcció i també utilitzar més factors per al càlcul de la velocitat a la qual pot anar el cotxe en cada moment, tenint en compte l'angle de la corba i afegint una diferència de velocitat respecte a les rodes motrius per tal de fer una funció semblant a la que fa el diferencial[13] en els cotxes que utilitzem quotidianament. A més, el fet que el cotxe no pugui seguir la línia si va ràpid o si les corbes són tancades ens va fer arribar a la conclusió que és degut al posicionament de la càmera. Aquesta està situada a la part davantera del cotxe i, el més important, a una altura molt baixa, pràcticament al nivell del terra. Aquest fet conjuntament amb el fet que la càmera no té un angle de visió excessivament gran, fa que el camp de visió del cotxe sigui molt reduït i no sigui capaç de veure les corbes amb el suficient temps per a frenar ni de poder captar un gir tancat.
Una solució molt senzilla i eficaç que aplicaríem seria elevar la càmera amb un masteler de tal manera que amb això podríem ampliar molt el camp de visió (més rang) i també guanyaríem amplada, ja que a l'estar més elevada la càmera la línia negra ocuparia menys espai (a nivell de píxels) de tal manera que podríem anar més ràpid i detectar girs més tancats.

Tot i aquestes limitacions, l'acompliment dels objectius el donem per satisfactori, ja que en cap cas volíem crear un cotxe capaç de competir sinó que volíem crear un prototip capaç de seguir una línia negra però utilitzant la nostra *API* per tal de facilitar la feina de programar el cotxe a futurs estudiants que sí volguessin competir.

Per últim, la presència dels diferents problemes que ens han sorgit durant tot el projecte i el conseqüent allargament del temps d'execució real respecte al temps planificat, conjuntament amb la presència de dates finals d'entrega (en el nostre cas les dates que imposa la universitat) ens han fet veure la importància de realitzar una bona planificació inicial i de deixar un cert marge d'error. Això ens fa arribar a la conclusió que és per aquest tipus de possibles problemes de mala planificació pels quals en els equips reals de treball es busquen persones amb molta experiència en el camp per a elaborar les planificacions, ja que encara que tinguis uns coneixements teòrics grans sobre un tema l'execució a la realitat és molt diferent i la capacitat d'estimar un temps d'execució i que aquest temps sigui acurat només es guanya amb l'experiència, a força de cometre errors.

AGRAÏMENTS

Voldríem agrair la realització d'aquest treball de final de grau al nostre tutor Miquel Àngel, sempre disposat a ajudar-nos en els moments en què no sabíem cap a on anar i indicant-nos els nostres errors per tal de perfeccionar al màxim el treball.

Agrair també a la família que ens ha ajudat en tot moment durant les llargues tardes de treball barallant-nos amb el cotxe i aguantant el soroll d'aquest i la invasió del territori de la llar per a muntar el circuit necessari per a les proves.

Per últim, agrair a l'Ana Ortega la seva ajuda amb tota la part visual del projecte sense la qual no hagués estat possible poder explicar amb claredat les parts d'aquest treball.

8 BIBLIOGRAFIA

[1] "The NXP Cup",[Online] Disponible a https://community.nxp.com/docs/DOC-1284

[2] "API", [Online] Disponible a https://en.wikipedia.org/wiki/Application_programming_interface

[3] "Freescle KL25z", [Online] Disponible a http://www.nxp.com/products/software-and-tools/hardware-development-tools/freedom-development-boards/freedom-development-platform-for-kinetis-kl14-kl15-kl24-kl25-mcus:FRDM-KL25Z

[4] "AutoPilot", [online] Disponible a https://www.tesla.com/es_ES/autopilot

[5] "Tesla", [Online] Dsiponible a https://www.tesla.com/es_ES/

[6] C. Chen *et al.* , "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving" , The IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2722-2730.

[7] Dennis Ritchie, Brian Kernighan, "The C programming language", AT&T Bell Laboratories, Murray Hill, New Jersey.

[8] "High level programming Language" [Online] Disponible a https://en.wikipedia.org/wiki/High-level_programming_language

[9] "Low level programming Language" [Online] Disponible a https://en.wikipedia.org/wiki/Low-level_programming_language

[10] "Matlab, el lenguaje del cálculo técnico" [Online] Disponible a https://es.mathworks.com/products/matlab.html

[11] "Line Scan Camera Board" [Online] Disponible a https://community.nxp.com/docs/DOC-1058

[12] "MCU101: Pulse Width Modulation" [Online] Disponible a https://community.nxp.com/docs/DOC-1082

[13] "Mecanismo diferencial" [Online] Disponible a https://es.wikipedia.org/wiki/Mecanismo_diferencial

APÈNDIX

A.1 ENTORN DE DESENVOLUPAMENT:

Per a la realització de tot el projecte hem utilitzat l'eina CodeWarrior for MCU v10.7 amb suport natiu per a Windows 8 i Windows 10, la versió especial d'estudiant que limita la mida del codi, que ja inclou el suport necessari per a treballar amb la placa Freescale KL25z, no s'han d'instal·lar llibreries externes. Per a treballar amb el llenguatge de programació C utilitza CDT 8.1

A.2 LLENGUATGE UTILITZAT

Per a programar tots els components del treball s'ha utilitzat el llenguatge de programació C, versió ANSI C90.

A.3 CODI

A continuació adjuntarem el codi de les funcions més pròpies del nostre projecte, que seran les del mòdul de moviments del cotxe i del funcionament dels principals components físics del cotxe de la nostra *API*, el programa de testeig i l'algorisme principal. No adjuntarem el mòdul orientat al funcionament de la placa de l'*API*, ja que és pràcticament genèric per a qualsevol programa que s'encarregui de treballar amb la Freescale KL25Z.

Codi principal de l'algorisme de funcionament autònom:

```
if(LineScanImageReady==1)
{
     LineScanImageReady=0;
     //Calculem el valor mitja que capta la camera per a filtrar la imatge
     for(i=0;i<128;i++)
     {
         acumulativeNumber += LineScanImage0[i];
     }
     medianNumber = acumulativeNumber/128;

     /*Procedim a filtrar la imatge:
     for(i=0;i<128;i++)
     {
         if( i <= 20 || i>=113)
         {
             filteredImage[i] = medianNumber;
         }
         else if (LineScanImage0[i] > medianNumber)
         {
             filteredImage[i] = medianNumber;
         }
         else
         {
             filteredImage[i] = LineScanImage0[i];
         }

         if(i>=15 && i<=127)
         {
             derivativeImage[i] = fabs( filteredImage[i]-filteredImage[i-1]);
         }
     }
```

Figura 10. Part I codi algorisme

```
/*Tractament de la imatge amb localitzacio del centre de la linia:
for(i=15;i<128;i++)
{
    if(derivativeImage[i] < 25)
    {
        if(edgeLeft != 0)
        {
            if(edgeCounter == 1 && centreCounter == 0)
                centreCounter++;
            else
            {
                if(edgeRight == 0)
                {
                    edgeRight = i;
                }
            }
        }
    }
    else
    {
       if(edgeLeft == 0)
       {
           edgeLeft = i;
           edgeCounter++;
       }
       else
       {
           if (centreCounter > 0 && edgeCounter == 1)
               edgeCounter++;
       }
    }

 }
```

Figura 11. Part II codi algorisme

```
//Una vegada tractada la imatge distingim si s'ha perdut la linia (+ o - de 2 pics)
//o no(2 pics) Si s'ha perdut es mante l'ultima ordre si no, s'actualitza
diferenciaCentres = fabs(oldEdgeLeft-edgeLeft);
if(edgeCounter != 2)
{
    //Aquesta comprobacio es per evitar falses lectures
    if( diferenciaCentres > 60)
    {
        lineCenter = oldEdgeLeft + 12;
    }
    else
    {
        lineCenter = edgeLeft + 12;
        oldEdgeLeft = edgeLeft;
    }
}
else
{
    if(diferenciaCentres >= 60)
    {
        lineCenter = oldLineCenter;
    }
    else
    {
        oldLineCenter = lineCenter;
        oldEdgeLeft = edgeLeft;
        lineCenter = ceil((edgeRight + edgeLeft)/2);
    }

}
```

Figura 12. Part III codi algorisme

```
                //Trobat el centre de la linia calculem els graus que han de girar les
                //rodes segons la distancia del centre de la linia al centre de la imatge
                degrees = fabs (64 - lineCenter);
                if(lineCenter > 70 )
                {
                    goForward(curveSpeed);
                    turnRight(degrees);
                }
                else if (lineCenter < 58 )
                {
                    goForward(curveSpeed);
                    turnLeft(degrees);
                }
                else
                {
                    goForward(straightSpeed);
                    goStraight();
                }

                edgeLeft = 0;
                edgeRight = 0;
                centreCounter = 0;
                edgeCounter = 0;
                medianNumber = 0;
                acumulativeNumber = 0;

            }
            break;
        }
    }
    return 0;
}
```

Figura 13. Part IV codi algorisme