Universistat Politècnica de Catalunya
(UPC) - BarcelonaTech

Facultat Informàtica de Barcelona

Grau d'Enginyeria Informàtica: Especialitat en Enginyeria de Computadors

Offside Real Time System

Autor:
Daniel Pujazón Bonet

Director:
Juan Jose Costa Prats.
Arquitectura de Computadors

Abril, 23, 2019

Abstracte

"L'esport pot canviar el món"."S'ha tornat terriblement obvi que la nostre tecnologia ha superat la nostre humanitat". Aquestes dues frases són propietat de Nelson Mandela i Albert Einstein. L'enorme potencial de la tecnologia sobre l'esport i els seus resultats és la idea trasversal d'aquestes dues frases. Aquest Treball de Final de Grau intentarà, humilment, posar un gra de sorra en l'aplicació de la tecnologia en el món de l'esport. El fora de joc és l'acció més polèmica de l'esport més seguit arreu del món. Paradigmàticament, i sent una situació purament lògica, és l'única per a la qual no s'ha desenvolupat cap sistema de detecció específic. Intentarem doncs donar el primer pas en el desenvolupament d'un sistema autònom en temps real que sigui capaç de detectar situacions de fora de joc en un partit de futbol.

Abstarcto

"El deporte puede cambiar el mundo". "Se ha vuelto terriblemente obvio que nuestra tecnología ha superado nuestra humanidad". Estas dos frases son propiedad de Nelson Mandela y Albert Einstein. El enorme potencial de la tecnología sobre el deporte y sus resultados es la idea trasversal de estas dos frases. Este Trabajo de Fin de Grado intentará, humildemente, poner un grano de arena en la aplicación de la tecnología en el mundo del deporte. El fuera de juego es la acción más polémica del deporte más seguido en todo el mundo. Paradigmáticamente, y siendo una situación puramente lógica, es la única para la que no se ha desarrollado ningún sistema de detección específico. Intentaremos pues dar el primer paso en el desarrollo de un sistema autónomo en tiempo real que sea capaz de detectar situaciones de fuera de juego en un partido de fútbol.

Abstract

"Sport can change the world." "It has become terribly obvious that our technology has surpassed our humanity." These two sentences are property of Nelson Mandela and Albert Einstein. The enormous potential of technology on sport and its results is the cross-cutting idea of ??these two phrases. This Final Work of Degree will humbly try to put a grain of sand in the application of technology in the world of sport. Offside is the most controversial action of sport in the world. Paradigmatically, and being a purely logical situation, it is the only one for which no specific detection system has been developed. We will try to take the first step in the development of an autonomous system in real time that is capable of detecting Offside situations in a soccer game.

Agraïments

A la meva mare. Els estudis, la feina... és una càrrega de responsabilitat i de treball molt gran i díficil de portar en molts moments. Tu ets la persona que cuida incondicionalment de mi especialment en els moments més dificils. Aquest Treball de Final de Grau no ha sigut més que un altre exemple. Moltes gràcies.

A mi padre. Porque no és el qué sinó el como. I la dedicación, cuando me explicas, la preocupación de hacerlo lo mejor posible, buscando en internet o repitiéndomelo hasta la saciedad, en cosas tan simples como hacer un presupuesto o resolver una duda sobre el fuera de juego són para mi el gesto de amor más grande que me puedes dar. I gracias a ti sé lo que és el sacrificio, la responsabilidad i la organización. Muchas gracias.

I especialmente a mi hermano porque si he llegado a donde he llegado es gracias a ti. Para mi siempre has sido un referente, precisamente en los momentos más difíciles, de lo que es levantarse en los momentos más duros y buscar una solución a cualquier problema sin poner excusas ni lamentarse. Te lo debo todo. Muchas gracias

Contents

Llista de figures

Llista de taules

Llista d'acrònims

- Offside: Fora de joc.
- FIFA: Fédération Internationale de Football Association.
- FIB: Facultat d'Informàtica de Barcelona.
- GEP: Gestió en Projectes.
- AR:. Àrbitre Assistent.
- VAR: Videarbitratge.
- STR: Sistema en Temps Real.
- SBC: Sistem on Board Chip, PC en una placa.
- PoC: Proof of Concept, Prova de Concepte.
- LAMP: Servidor tipus Linux,Apache,MySQL,Perl (PHP, Python...)
- OS: Sistemes Operatius.
- RTOS: Sistemes Operatius per a Sistemes en Temps Real.
- RGB: Format de color Red, Green and Blue.
- HSV: Format de color Hue,Saturation and Value.
- SSH: Secure Shell

Chapter 1

Introducció

1.1 Contextualització

1 de cada 100 euros aproximadament que es generen a l'economia espanyola provenen del futbol professional. El que vol dir que sobre l'1% (0,75%) del PIB de l'Estat Espanyol prové de La Liga segons un estudi de la consultora KPMG l'any 2015. Això reafirma, amb el cas concret de La Liga, la transcendència econòmica, i també social, del Futbol arreu del món. [1]

Una de les accions més polèmiques en el futbol és el fora de joc també conegut com orsai. Més endavant ho veurem en profunditat però,a grans trets, el fora de joc és una infracció que es dóna quan un jugador atacant rep una passada d'un company estant més a prop de la seva porteria contrària que el defensor més a prop d'aquesta.

Fins ara la manera de detectar el fora de joc consisteix en dos àrbitres corrent sempre en paral·lel a cadascuna de les línies de banda, tenint assignat un dels dos equips respectivament, a l'altura del jugador de l'equip assignat més proper a la seva porteria. En el moment que es produeix una situació de fora de joc aixequen el braç amb una banderola indicant la infracció.

Aquest mètode té grans inconvenients com, per exemple, la poca precisió que té l'ull humà per diferenciar de distàncies relativament petites des de la línia de banda i el seu temps de resposta. Això ho va recollir Francisco Belda Maruenda, metge del Centro de Salud de Alguirías a Múrcia, en un estudi sobre el temps de reacció de l'ull humà en situacions de detecció de fores de joc l'any 2003. [2]

Una de les solucions ha sigut la implementació de la tecnologia en el futbol. Des de l'any 2016 es va implementar el "Video Assistant Referee" (VAR) que, a grans trets, permet veure per pantalla accions polèmiques que poden tenir una transcendència directa en el resultat com podria ser un fora de joc que procedeix un gol.

Aquesta solució, sent l'actual, té encara molt inconvenients com la incapacitat de detectar en temps real la infracció de manera que a l'aplicar-se només en certes situacions deixant

a moltes d'altres sense revisar i amb el possible error de no detectar-se correctament per les limitacions humanes comentades anteriorment. Aquest Treball de Final de grau desenvoluparà la implementació d'un sistema integrat de diferents subsistemes encastats que detecti situacions de fora de joc i determini si hi ha infracció de manera automàtica en temps real i amb una qualitat superior a la de l'ull humà en la detecció de distàncies mitjançant visió per computadors.

1.2 Actors implicats

Aquest Treball de Final de Grau té una finalitat purament docent. No obstant això, veurem a continuació que és un producte amb un clar valor afegit respecte als productes actuals que desenvolupen la mateixa tasca. Així doncs, podria tenir una sortida cap al mercat a diferents sectors. Tot això configura el conjunt d'actors implicats o potencialment implicats, stakeholders, del projecte.

Per una banda el treball està desenvolupat sense cap conveni amb empreses per efectuar el Treball de Final de Grau, per tant, és un projecte íntegrament dissenyat i implementat pel desenvolupador. Fent que el 50% dels possibles beneficis del projecte fossin pel desenvolupador i l'altre 50% per a la Facultat donat que la propietat intel·lectual del projecte, al ser purament acadèmic com recull la Normativa del Treball Final de Grau del Grau en Enginyeria Informàtica de la FIB recau en la pròpia Facultat. [3]

Els actors que hi intervenen de forma directa a part del desenvolupador són: el director del Projecte, Juan Jose Costa Prats, els professors de GEP, director del projecte Fernando Barrabes representats pels responsables Xavier LLinàs Audet i Ferran Sabaté Garriga i els membres del Tribunal 4 , Curs 2018 quadrimestre de primavera. Torn d'abril, Carlos Àlvarez Martínez com a president, Jose Antonio Gonzalez Alastrue, Antoni Grau Saldes com a vocals i Gemma Sese Castel com a vocal suplent.

Els potencials beneficiaris del projecte en cas que sortís al mercat seria la indústria de l'Esport en tot el seu conjunt: Des dels equips més modestos i amb menys ingressos de la categoria donat el preu econòmic del producte com els equips de primer nivell que compten amb les facilitats, tant en tecnologia com en infraestructures, per implementar el projecte desenvolupat.

Chapter 2

Estat de l'art

Aquest projecte planteja una implementació alternativa als sistemes actuals de la detecció de fores de joc en partits de futbol. Per tant en aquest apartat, primerament, analitzarem els diferents sistemes actuals que existeixen per detectar el fora de joc i veurem quins són els límits que ofereixen. A partir d'aquest punt presentarem la proposta de solució amb els avantatges que ofereix respecte a els seus predecessors i com ho implementarem.

2.1 Fora de joc. Concepte

Abans de posar-nos a analitzar els diferents sistemes que detecten la situació de fora de joc veurem el concepte del fora de joc. Segons la Fédération Internationale de Football Association (FIFA), des de l'origen del primer reglament oficial escrit per l'Associació Anglesa de futbol (FA) 1863, el fora de joc ja hi formava part. [4] La FIFA estableix actualment que

un jugador atacant està en fora de joc si es troba més a prop de la línia de porteria contrària que el penúltim defensor i que la pilota. Remarcar amb aquesta definició que si el jugador atacant es troba a la mateixa posició que el penúltim defensor o que la pilota, no és fora de joc. [5]

D'això se n'anomena la regla dels dos jugadors. Va ser establerta el 1990 per motivar el joc de passades entre companys. Si només fos infracció en el cas que l'atacant està més endarrerit que l'últim jugador (el porter) i no del penúltim els equips posarien com més jugadors a prop de la porteria rival i els rivals llençarien la pilota llarga cap a la porteria sense preocupar-se si els atacants estaven en fora de joc ja que el porter estaria sempre sota la porteria, el punt més endarrerit possible. Un cop un jugador està en fora de joc hi haurà

casos en què no serà una infracció i es continua amb el joc sense cap conseqüència mentre que per tots els altres casos serà infracció i per tant es para el joc i la pilota passa a ser possessió de l'equip rival. Analitzarem si un jugador de l'equip atacant està en posició de

fora de joc, i seguidament si es infracció o no, cada cop que un jugador de l'equip atacant toca la pilota.

Figure 2.1: Situació de fora de joc

Això significa que si un jugador està en posició de fora de joc però l'origen ve d'un jugador de l'equip defensor, no és infracció. Quan es detecta que un jugador està en posició de fora de joc s'ha de determinar si és infracció o no. Els casos excepcionals en els quals un jugador en fora de joc no comet infracció són:

- L'origen de la passada és des de fora la banda
- Si el jugador atacant està en el mateix camp. Llavors directament diem que no està en posició de fora de joc i per tant no pot cometre infracció.

Si no es dóna cap d'aquestes excepcions i el jugador està en posició de fora de joc aquesta esdevé en infracció.

Casos com el llançament des de la cantonada, per exemple, no tindrà cap atacant en fora de joc, ja que la pilota es troba el més a prop de la porteria contrària i tots els atacants estaran segur darrere d'aquesta. Un cop vist el concepte de fora de joc veurem i analitzarem

els diferents sistemes de detecció de posició de fora de joc i l'avaluació de si esdevé en infracció, veient on estan les limitacions actuals per poder presentar una proposta de solució que vagi més enllà d'aquestes limitacions i en millori la qualitat.

2.2 Projectes relacionats

2.2.1 Àrbitre únic

Segons el Reglament de les normes de joc de la FIFA en tots els partits disputats sota aquest reglament tenim dos assistents d'àrbitre (AR) que s'encarreguen de determinar si es produeix una infracció de fora de joc. Això però no succeeix en tots els partits de futbol reglamentaris.

A Catalunya, per exemple, la Federació Catalana de Futbol estableix diferents categories, de menys nivell a més nivell: Tercera regional, segona regional, primera regional, preferent,

tercera catalana, segona divisió B, Segona divisió estatal i Primera divisió. En el Reglament

de Comitè d'Àrbitres de la Federació Catalana de Futbol s'estableix que només a partir de Primera regional existeix l'obligatorietat de tenir dos àrbitres assistents. Això vol dir que per a lligues menor no hi ha àrbitres assistents. Llavors la detecció de posicions de fora de joc i l'avaluació de la infracció o no la porta a terme l'àrbitre principal. [7] Aquest mètode

té grans deficiències. Bàsicament la posició de la càmera (l'àrbitre principal) sobre la imatge i la percepció i qualitat de les distàncies dels elements de l'escena.

La distància i l'angle (darrere en comptes d'en paral·lel amb l'últim defensor) en el que estan fa que la qualitat de la percepció de distàncies sigui baixa. Això es va veure en certa mesura en l'article Referees' visual behaviour during offside situations in football on amb un treball de camp d'analitzar diferents situacions de fora de joc amb una petita població determinen que l'angle i la distància de la càmera són transcendents a l'hora de determinar la situació de fora de joc.

2.2.2 Assistant referee (AR)

A les categories professionals del fubtol español com competicions FIFA compten amb els anomenats assistents d'àrbitre (AR, assistant referee, utilitzat en estudis sobre capacitat visual. Aquests es col·loquen en paral·lel a la cadascuna de les línies de banda respectivament

i a l'altura de l'últim defensor de l'equip defensor que tenen assignat. Llavors una millora que té aquest sistema en relació al sistema de l'àribtre únic és:

- La posició dels àrbitres al ser en paral·lel a la línia de banda i a l'altura de l'últim jugador facilita la percepció de la diferència de posicions dels diferents jugadors sobre l'eix de les abscisses, veient-ho en una única dimensió.
- La distància de l'àrbitre assistent sobre la jugada a analitzar serà menor; estant sempre dins del marge de la longitud del camp. Per tant en el pitjor cas la jugada estarà a l'altra punta del camp, 45 metres mínim segons el reglament FIFA, i un cas mitjà (cas més freqüent) d'uns 20-30 metres aproximadament.

Els desavantatges que presenta aquest sistema són:

Segons l'article del Doctor Francisco Marhuenda Visual sequence in a game of offside in football explica que el temps que triga un AR a determinar si una acció de joc és fora de joc o no són uns 260 ms (hi ha 3 processos de captació d'imatges de 60 ms cadascun subdividits en dos subprocessos: fixació, 30 ms, i processat uns altres 30 ms. A més a més hi ha una latència en el primer procés d'uns 80 ms). [8]

Treballarem sobre processadors Arduino MKR1000 que treballen amb una freqüència de 48 MHz. Això és un temps de cicle de 21 ns. Això ens permet aconseguir amb programes prou grans (si cada instrucció són 10 cicles en mitjà, fent una aproximació, i volem un temps d'execució menor de 260 ms podem tenir programes de fins a 1 milió d'instruccions i això

tenint en compte execució d'instruccions en sèrie no tenint en compte la segmentació del processador i per tant el seu IPC que augmentaria encara més aquest número).

En l'article vist anteriorment, Referees' visual behaviour during offside situations in football , també determinen que si hi ha nous elements dins dels 35º del centre de la càmera augmenta el risc d'error en la percepció i presa de decisió de la infracció de fora de joc. Amb un sistema alternatiu amb la càmera en un altre punt (sobre el camp) es podria eliminar aquesta variable de l'equació.

Figure 2.2: Àrbitre principal durant un partit de futbol.

Figure 2.3: Perspectiva d'un àrbitre assistent.

2.2.3 VAR.

A partir del 2016 la International Football Association Board juntament amb la FIFA van introduir durant dos anys de proves l'àrbitre assistent de vídeo (VAR) que consisteix en un conjunt d'àrbitres que es troben veient el partit per televisió i complementen l'arbitratge de l'àrbitre principal. El Mundial de Rússia de 2018 va ser la seva prova de foc i actualment ja s'utilitza també en "La Liga". [12] El sistema s'utilitza només si es donen 4 accions concretes

del joc: Gols, penals, expulsions directes (vermella directa només) i confusió d'identitat d'un jugador. L'aplicació del sistema és el següent. En produir-se l'acció a analitzar tant l'àrbitre

principal com els àrbitres de vídeo poden sol·licitar revisió. Els àrbitres de vídeo analitzen per televisió. Al poder re arbitrar-se situacions permeten esmenar error que o bé per dificultat

de veure la diferència de posició en temps real o bé per un propi error de percepció esdevenen equivocacions de la senyalització de la infracció. Un primer gran problema del sistema és la precisió de l'ull humà per detectar les distàncies. Un conjunt d'investigadors de Cinesiologia biomèdica i psicòlegs de la Universitat de Leuven a Bèlgica van redactar un article basat en un treball de camp on van agafar una població d'àrbitres (analitzant les variables de cadascun i fent grups) i posaven 63 situacions de detecció de fora de joc virtuals fetes per ordinador i des d'un punt de visió superior on veien tot el camp (imatges semblants a les analitzades en el VAR). Aquest estudi determina que hi ha una desviació de 3 mm en la posició del jugador atacant per l'ull humà tot i no sent una situació de temps real i des de la perspectiva esmentada anteriorment. Un altre gran desavantatge que té el sistema és que

només s'aplica en un dels 4 casos descrits anteriorment. Això fa que molts fora de joc que no es produeixen en aquestes situacions siguin susceptibles de ser percebuts erròniament (per les causes comentades anteriorment). Finalment no és un sistema en temps real. Necessita

parar el joc per re arbitrar la situació concreta perdent qualitat en el mateix joc. Veiem doncs que la implementació del VAR mitiga molt errors que es podien donar en el passat però té clares limitacions. Aquest és l'estat de l'art dels sistemes actuals de percepció de fora de joc.

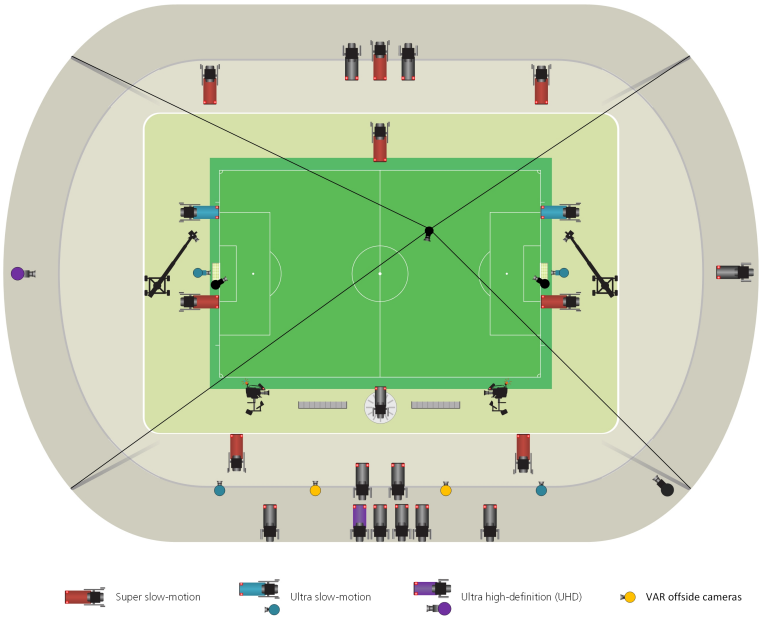


Figure 2.4: Situació de les càmeres per al VAR durant el Mundial de Rússia 2018

Chapter 3

Formulació

3.1 Proposta de solució

La nostre solució serà un sistema que sigui capaç de detectar situacions de fora de joc en temps real. Aquest sistema està integrat per 3 subsistemes:

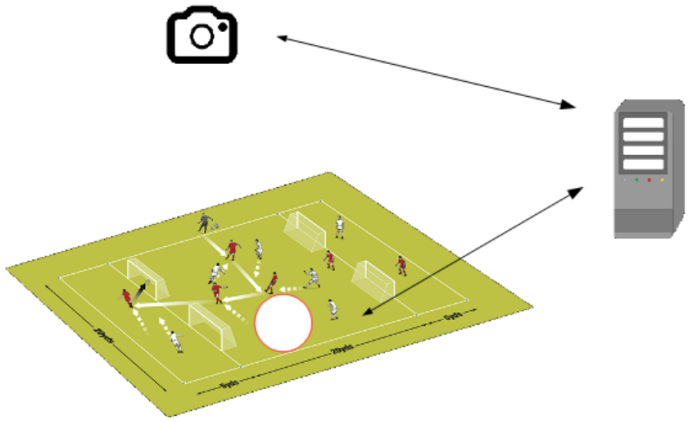


Figure 3.1: Iconograma del Sistema

- El primer de tots és una càmera posicionada al mig del camp a una altura que permeti tenir dins de l'objectiu tot el camp. Amb això i mitjançant visió per computadors aconseguirem tota la informació: Posicions de cadascun dels jugadors, equips al qual pertanyen...
- Després tindrem dins de la pilota un sistema encastat format per un detector de posició connectat a un processador que agafarà la posició de la pilota cada cert temps (en funció de la freqüència del processador). Amb això podrem aconseguir dues posicions en un determinat temps (tenint una successió de posicions x0,x1,x2...) i la diferència d'aquestes posicions dividit per aquest determinat temps ens dóna la velocitat de la

pilota. Així podem calcular una successió de velocitats (v0, v1,v2...) i la diferència de les velocitats entre aquest temps ens dóna l'acceleració de la pilota.

- Finalment tindrem un tercer element que serà el node encarregat de rebre la informació i processar les dades, tant del sistema de la càmera com el de la pilota, i en funció de si l'acceleració rebuda per la pilota supera un cert threshold analitzar les dades i determinar si és posició i infracció de fora de joc o no.

3.2 Objectius

Els objectius teòrics i docents del TFG que haurem complert un cop tinguem sistema implementat i funcional seran els següents:

1. Assimilació de conceptes de la teoria de Visió per Computadors (Preprocessament, processament, segmentació, histogrames, detecció de blobs...)
2. Programació, implementació dels conceptes anteriors, en MATLAB i les llibreries de Visió per Computadors Computer Vision System Toolbox.
3. Programar sobre plaques Arduino programes de processament d'informació (o bé les dades de la càmera, o bé de l'acceleròmetre de la pilota)
4. Connexió de diferents dispositius amb les plaques Arduino i comprendre i implementar correctament del protocol de comunicació que utilitzem.
5. Comunicació entre diferents nodes utilitzant la tecnologia Wifi. Crear una xarxa de nodes.
6. Programar conscientment de l'arquitectura un programa en temps real (STR) i per tant entendre els conceptes teòrics de STR i les seves polítiques de planificació. A més a més, el valor afegit que aconseguirem implementant el nostre sistema respecte a els anteriors sistemes que hem vist seran els objectius pràctics del projecte:
7. És un sistema en temps real. Amb això aconseguim que s'analitzin totes les possibles situacions de fora de joc que es donen durant el partit a diferència del VAR que només podia tornar a revisar aquelles jugades que compleixen un dels 4 casos esmenats abans.
8. A més a més no para el joc donant-li qualitat al sistema i al propi joc
9. La diferència de posicions es detecta mitjançant Visió per Computadors de manera que la limitació de l'ull humà queda superada per la dels processadors.
10. El fet que el sistema sigui de baix cost, com veurem més endavant, podria aplicar-se a totes les categories de futbol professional i formatiu eliminant la causa de l'ús del sistema de l'àrbitre únic en lligues menors.

Aquest objectius teòrics els intentarem assolir a través dels següents objectius pràctics:

1. Desenvolupar un sistema autònom que sigui capaç de captar fotografies, guardar-les i enviar-les per xarxa.

2. Desenvolupar un sistema autònom que sigui capaç de captar informació de sensors externs i enviar-les per xarxa.

3. Desenvolupar un sistema autònom que pugui rebre informació de la xarxa i pugui processar-la

4. Desenvolupar software de visió per computadors capaç de detectar, donada la imatge d'un partit, les posicions dels jugadors i la pilota.

5. Desenvolupar un mòdul capaç d'intercomunicar software escrit en C++ amb software escrit en MATLAB

6. Integrar tots els sistemes autònoms per tal d'aconseguir un sistema integrat que permeti la detecció automàtica del fora de joc.

7. Desenvolupar el mateix sistema però orientat a l'execució multithreading.

Chapter 4

Abast del projecte

4.1 Requeriments

Els requeriments del projecte vénen donats principalment pel programari software necessari per a desenvolupar els programes que utilitzarà el nostre sistema i els recursos hardware necessaris per implementar-ho.

4.1.1 Software

Podem dividir la implementació del nostre software en mòduls. Aquests bàsicament seran: Un mòdul per a cadascun dels 3 sistemes esmenats anteriorment, un mòdul que implementi les funcionalitats de la xarxa i comunicació entre sistemes i els mòduls de visió per computadors. Els requeriments software seran doncs els programes necessaris per a desenvolupar aquest programari en els llenguatges i llibreries que decidim utilitzar.

Els llenguatges que utilitzarem seran MATLAB i C++. En l'apartat 9.1 expliquem i justifiquem els llenguatges i llibreries utilitzades de forma extensa i acurada. El programari per a desenvolupar software en aquests llenguatges són MATLAB R2017b i qualsevol editor de text com Notepad++.

Notepad++ és un software de programari lliure i com a tal podem obtenir-lo sense cap cost directament de la xarxa. MATLAB R2017b no és un programari lliure. Cada estudiant de la Facultat d'Informàtica de Barcelona té accés a una llicència d'estudiant per a MATLAB R2017. Així doncs el cost de MATLAB R2017b serà gratuït.

4.1.2 Hardware

El nostre sistema està format de 3 subsistemes. Cadascun d'aquests tenen un hardware associat que forma part dels requeriments hardware del sistema. En l'apartat 8.1 explicarem el hardware utilitzat en cadascun dels nodes de forma extensa. Aquí simplement esmenarem quins són requeriments que haurà de complir el hardware que triem.

- Per al Node Principal necessitem qualsevol computador que sigui capaç de conectar-se a la xarxa. També és un requeriment el fet que tingui la suficient potència per processar els algorismes de visió per computadors dins dels deadlines establerts.
- Per altre banda el Node de la Càmera necessitarà una computador encastat (SBC) que sigui capaç de conectar-se a la xarxa i pugui comunicar-se amb una càmera com a perifèric d'entrada.
- El Node de la Pilota també haurà de ser un computador encastat i que pugui connectar-se a la xarxa. A més a més haurà de poder comunicar-se amb els perifèrics d'entrada que ens permetin aconseguir la informació referent a la velocitat i acceleració de la pilota com per exemple acceleròmetres o giroscopis.
- Finalment els perifèrics d'entrada ens han de permetre fer fotografies per a detectar la posició dels jugadors a la gespa i un sensor per a calcular la diferència d'acceleració de la pilota produïda en cada moment. Aquests requeriments es poden assolir gràcies a una càmera i un aceleròmetre que es puguin comunicar amb els nostres sistemes encastats.

El mercat ofereix diferents opcions per a complir aquests requisits. Per al Node Principal utilitzarem el Computador particular del desenvolupador mentre que per als dos Nodes encastats utilitzaran una Raspberry Pi Zero. La explicació i justificació com hem comentat anteriorment es poden trobar en l'apartat 8.1 d'aquest mateix document. Allà també s'exposaran els perifèrics d'entrada.

4.2 Obstacles

EL projecte compta amb un gran nombre de riscos que haurem de tenir en compte abans i durant el desenvolupament del sistema. Llavors analitzarem un a un els riscos que poden anar sorgint i la solució que implementarem:

4.2.1 Sistema de càmera

Si no podem aconseguir col·locar la càmera en un camp de futbol real a una altura concreta per poder veure tot el camp

Solució : Utilitzar imatges per defecte que permetin veure la funcionalitat del programa de detecció de posicions dins la imatge

Solució : Utilitzar un camp més petit i altres elements a manera de jugadors com figures d'acció.

4.2.2 Sistema de pilota

Si per problemes de logística o robustesa física del producte (si en xutar la pilota es trenca el sistema)

Solució : Fer una prova de concepte (PoC) utilitzant una pilota més petita o bé un globus o qualsevol altra esfera que pugui portar dins el sistema i treballar sobre el camp reduït esmentat abans.

4.2.3 Sistema comunicació

En cas que la tecnologia utilitzada per a transmetre la informació des dels subsistemes de la càmera o la pilota al node de processament (Wifi, Bluetooth...) Solució : Connectar els dos sistemes al node de processament (un servidor) per cable.

4.3 Metodologia

En aquest apartat determinarem la metodologia de treball que seguirem per al desenvolupament del nostre sistema. També tindrem en compte les eines de seguiment i d'avaluació del projecte.

Com a metodologia de treball viem que hem segmentat el projecte en subprojectes relativament independents. Això fa que puguem tenir un valor de producte de forma incremental: No cal que tot el producte estigui acabat per tenir un producte amb valor. Això lliga amb les metodologies iteratives que permeten desenvolupar funcionalitats o subsistemes iterativament, on cada iteració ja comptempla la part de disseny i testeig de la implementació.

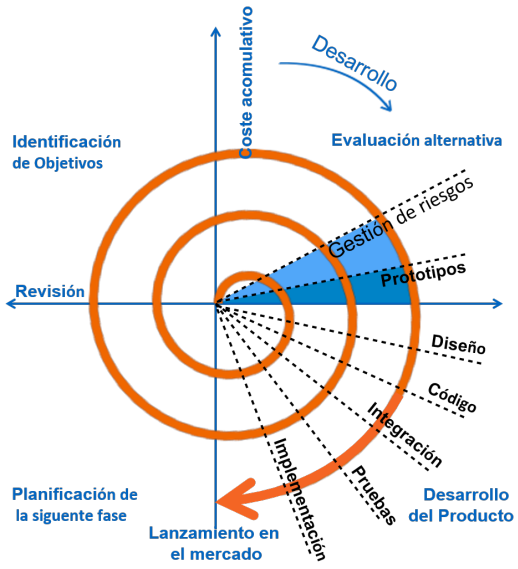


Figure 4.1: Gràfic referent a la Metodologia Espiral

Això fa que la nostra metodologia de treball s'enfoqui cap a la Metodologia Espiral. A més a més això ens facilitarà el testeig i avaluació del projecte (eina de seguiment i avaluació), ja que a cada iteració s'ha de provar la funcionalitat del sistema. També remarcar que com a eina de seguiment plantegem reunions amb el director de projecte de forma mensual o cada dues setmanes que ratifiquin cadascuna de les iteracions del projecte.

Com a eines d'avaluació desenvoluparem tests que sabent la sortida que hauria de tenir el nostre sistema comprovi que la resposta calculada és igual a l'esperada. Això tant poden ser amb imatge editades o bé amb imatges captades en temps real (a cada iteració i en funció de les funcionalitats implementades podrem provar un conjunt de casos de test o uns altres, havent-los comprovat tots al final).

Remarcar que hem seguit la metodologia espiral amb certs matissos donada les necessitats específiques del projecte. La Metodologia espiral itera funcionalitat a funcionalitat i dins de caascuna d'aquestes divideix les fases: identificar requisits, dissenyar, implementar... Nosaltres hem agrupat tot l'analisis de requisits i disseny de totes les funcionalitats a l'inici del desenvolupament i després si que hem anat funcionalitat a funcionalitat de forma incremental seguit la idea de la metodologia Espiral.

4.4 Viabilitat

4.4.1 Viabilitat tècnica

Amb el que hem vist de requeriments i tenint controlats els riscos podem tenir un producte complet on en funció dels riscos tindrà menys o més abast (Camp real o reduït, pilota real o prova de concepte...)

Un dels grans avantatges de descentralitzar el nostre sistema en diferents subsistemes permet aïllar els problemes i riscos que puguin anar sorgint i que no afectin la totalitat del sistema.

4.4.2 Viabilitat econòmica

Un cop vists els requeriments Software i Hardware veiem que els costos de desenvolupar i implementar el projecte són relativament econòmics tenint en compte els costos de la tecnologia com el VAR a Sudamèrica que ronda els 4 milions de despeses anuals.

Els costos del Software són zero gràcies a la llicència d'estudiant de MATLAB R2017b atorgada al desenvolupador com a estudiant de la Facultat d'Informàtica de Barcelona. Respecte al Hardware els costos del sistema complet, més endavant hi ha un estudi exhaustiu dels costos un cop preses les decisions sobre quins components concrets escollit, rondarien els $900, desglossats en els tres sistemes:

El preu del sistema de càmera està format pel preu d'aquesta que oscil·la entre els $25.99 de l' ArduCAM Mini Camera 2 MP OV2640 i els $65.00 de la OpenMV Cam M7 Image Sensor. De microcontrolador els preus ronda els 40$ de la gran majoria de productes de la categoria de "Enhanced Features" o "Internet of Things" Arduino. FInalment el mòdul de connexió podria costar els $10 del AZDelivery Esp8266 01 ESP-01 Wi-Fi. Un total de $120 aproximadament.

El preu del sistema de la pilota comparteix els costos tant del microprocessador com el del mòdul de connexió ($50). La diferència ve donada pel dispositiu de detecció de posició. Aquests costen sobre els $100 d'un detector capacitor KG-3080NFPKGS2T, KG5069, sumant tot un total de $150

Finalment el Servidor consta dels $600 d'un ordinador convencional, sobretaula, portàtil... que pot costar vora els $600

4.4.3 Viabilitat financera

La viabilitat financera és total donada la possibilitat d'adquirir la gran majoria de recursos Hardware als laboratoris que disposa la Facultat d'Informàtica de Barcelona al Departament d'Arquitectura de Computadors i el cost zero del Software explicats anteriorment.

Chapter 5

Planificació temporal

El temps per a implementar el nostre sistema és de 4 mesos amb data de venciment el dia de la presentació d'aquest.

Llavors primer de tot especificarem les tasques de les quals consta el projecte per tal de poder estimar quant de temps ens portaran i alhora veure les dependències i precedències que es generen entre elles. D'aquesta manera també podem plantejar solucions davant les possibles desviacions que sorgeixin en qualsevol de les tasques (analitzarem també el risc de desviació de cadascuna de les tasques). Remarcar que prendrem freqüentment com a unitat de treball un dia que l'estimarem amb 4 hores de treball.

5.1 Especificació de les tasques

Comencem doncs analitzant les diferents tasques que conformen el projecte i un anàlisis del temps que ens portaran entre d'altres:

5.1.1 Gestió del Projecte

Una part important del projecte és planificar el projecte en tota la seva complexitat: estudi del projecte, viabilitat, planificació temporal... posant-ho tot per escrit de manera metòdica i estructurada. Tot això ho treballem durant l'assignatura de GEP. L'estructuració de la Gestió del projecte es divideix en 5 lliurables cadascun amb la seva corresponent càrrega de treball: [3]

Això doncs dóna un total de 75 hores de treball dividides en 37,5 hores dedicades a llegir i assimilar tots els continguts i documentació donats a l'assignatura i rubricats en forma de lliurables i unes 37,5 hores més de dedicació personal al Treball de Final de Grau.

Per tant a més a més d'aquestes hores computades aquí hauríem de tenir en compte hores d'aprenentatge dirigit referents a documentació i mòduls que no estan associats a lliurables com és el cas del mòdul 1 sobre les Eines TIC de suport a la Gestió de Projectes i Equips

Tasques de GEP	Càrrega de Treball
Lliurable 1: Definició de labast i contextualització.	Aprenentatge dirigit: 9.00 h Aprenentatge autònom: 15.50 h
Lliurable 2: Planificació temporal.	Aprenentatge dirigit: 5,00h Aprenentatge autònom: 3,25h
Lliurable 3: Gestió econòmica i sostenibilitat.	Aprenentatge dirigit: 5,00h. Aprenentatge autònom: 4,25h
Lliurable 4: Revisió de les competències del Treball de Final de Grau (propi de cada especialitat).	Aprenentatge dirigit: 5.00h Aprenentatge autònom: 4,00h
Lliurable 5: Document final recull de tots els lliuraments anteriors	Aprenentatge dirigit: 8,00h. Aprenentatge autònom: 10,25h

Table 5.1: Taula amb la càrrega de treball de GEP.

5.1.2 Anàlisis dels Sistemes

Com hem vist en apartats anteriors la implementació del nostre sistema consta de 3 sistemes encastats: Sistema de vídeo, sistema de la pilota i el node del servidor. Per a cadascun dels sistemes hem hagut de fer un estudi exhaustiu tant del concepte teòric sobre el qual apliquem el nostre sistema encastat. Per exemple en el cas de sistema de vídeo l'apliquem per aconseguir mitjançant visió per computadors informació de la imatge del camp amb els jugadors. Aquí hi ha una part d'estudi dels conceptes teòrics de la Visió per Computadors i alhora hi ha un estudi del mateix sistema per poder connectar els diferents dispositius d'aquest com serien la càmera el processador i el mòdul de connexió, amb les especificacions pròpies del hardware de cada sistema.

5.1.2.1.0 Analisis del Sistema de Vídeo

Com hem comentat abans aquí tenim dos camps d'estudi: Teoria de Visió per Computadors: Estudiar els conceptes relacionats amb la teoria de Visió per computadors com el preprocessat, la segmentació, la detecció i identificació de Blobs. A més a més això s'implementa amb MATLAB pel que també necessitem estudiar els diferents mètodes que implementen les llibreries del Computer Vision System Toolbox el qual inclou MATLAB.

Disseny i especificacions del sistema: Necessitem primer de tot dissenyar el nostre sistema. Això vol dir veure d'entre totes les opcions de hardware quina és la més eficient en el nostre procés específic (diferent en cadascun dels sistemes encastats). Un cop dissenyat el sistema amb els dispositius concrets caldrà llegir l'especificació d'aquests per tal de poder després, en les tasques d'implementació, connectar tots els dispositius mitjançant el processador.

5.1.2.2.0 Anàlisis del Sistema de la Pilota

Aquí solament necessitem la part de Disseny i especificacions del sistema, ja que el concepte teòric és relativament simple i amb una càrrega d'hores negligible pel que no requereix una tasca independent.

5.1.2.3.0 Anàlisis del Node Servidor

En el nostre node Servidor sí que dividim en dues subtasques d'anàlisis els següents camps:

Algoritme de detecció del fora de joc: Els sistemes actuals que detecten el fora de joc i valoren si és infracció o no són humans, àrbitres auxiliars per exemple, i no digitals. Això fa que nosaltres a l'implementar un programa que en temps real estigui constantment analitzant-ho i determinant si hi ha infraccions de fora de joc, haurà d'aplicar un algoritme encara no dissenyat. En aquest apartat dissenyarem el nostre algoritme per després poder implementar-lo.

Disseny i especificacions del sistema: És la mateixa idea que els sistemes anteriors. Aquest serà el que ha de rebre la informació dels altres dos sistemes a més a més de tenir la necessitat d'assolir un rendiment màxim per poder donar la informació en temps real i per tant estudiar a fons quina és la tecnologia i llenguatges més adients (un Servidor LAMP convencional, o bé un sistema específic sense soroll de sistema ...)

5.1.3 Implementació

Un cop tinguem analitzat cadascun dels sistemes tocarà implementar-los. De manera individual i tenint en compte a la càrrega de treball els riscos que poden aparèixer i com actuar-hi.

Per al sistema de càmera aquesta càrrega de treball anirà dirigida a programa el codi que permeti obtenir la informació dels jugadors cada cop que la càmera executi la fotografia.

El sistema de Pilota per altra banda es centrarà a comunicar el dispositiu de moviment al nostre processador i que aquest processi (calculi l'acceleració en funció de la seqüència de posicions guardades i que va adquirint i el temps entre cadascuna) aquesta informació.

Finalment en el sistema del servidor haurem de programar el codi que, donat tots els inputs, determini si existeix algun jugador atacant en fora de joc i alhora dictamini si és infracció o no.

5.1.4 Assemblatge

Finalment tindrem la tasca d'Assemblar els tres sistemes com un únic sistema amb la tecnologia pertinent. Aquí haurem de tenir en compte els contratemps que puguin sorgir si la tecnologia no és factible d'aplicar i solucions (ja exposades anteriorment en l'apartat de riscs).

La major part de la càrrega de treball consistirà a comunicar mitjançant els mòduls de comunicació i les tecnologies pertinents els tres sistemes i que el servidor les rebi (sense deadlines perquè sigui un sistema en temps real funcional).

5.1.5 Test

En quant al testeig del projecte diferenciem dues etapes de testeig: Per una banda, cadascuna de les tasques d'implementació porten associades tasques de testeig. Això segueix el principi d'integració contínua que ens permet aïllar i solucionar errors a fases inicial de manera que posteriorment en la tasca d'Assemblatge tinguem un nivell de certesa major i per tant menys risc de tenir errors i desviacions.

Finalment amb cadascun dels sistemes funcionals, amb els tests passats, i ensablat toca testejar el projecte final, l'Assemblatge. Aquí hi trobem 2 casos clars: El sistema en la seva generalitat funcioni correctament per a un gran conjunt de casos d'ús o per altra banda que no hi funcioni. El segon cas és el cas pitjor i planteja una càrrega de treball extra per a solucionar-ho.

El que farem llavors és: Si ens trobem en el segon cas la càrrega de treball serà íntegra a solucionar-ho. Per altra banda si ens trobem en el cas òptim que tot el sistema és funcional (una de la gràcia de tenir un sistema tan modular és que s'ha testejat prèviament cadascun dels sistemes de manera que hi ha un principi d'integració contínua que facilita eliminar aquest possible risc de fallades de sistema en gran quantitat i crítics) dedicarem aquesta càrrega de treball extra a millorar l'eficiència del sistema (pe: Fent el Servidor més específic o programa tant el programa del servidor com el de visió per computador de manera conscient de l'arquitectura...) Tant per al testeig individual de cadascun dels sistemes com per al del projecte ensablat hem de tenir en consideració a l'hora d'estimar la càrrega de treball el següent: El disseny dels tests, la implementació dels mateixos i el temps i l'anàlisis d'aquests tests. A més a més fent que com més cops haurem d'iterar a causa d'errors del sistema augmenta el risc i la càrrega a causa de tornar a analitzar resultats.

5.2 Temps per tasca

Un cop vistes les tasques, a més a més de les seves possibles desviacions, i en què consisteixen podem fer-nos una idea de la càrrega de treball que ens portaran. La data d'inici del treball va ser a inicis d'Agost mentre que la data de finalització establim el dia 1 de Gener. No és la data de presentació del treball però ens reservem aquest diferencial per a un cas de desviació crítica. La previsió, encara comptant amb les desviacions, hauria de complir que la data de finalització del projecte fos l'1 de Gener.

Remarcar que això és comptant el cas òptim i sense tenir en compte les possibles desviacions, les quals les veurem i tindrem en compte més endevant. Tenint en compte tot això, tenim:

Tasca	Càrrega de Treball
Anàlisis Sistema de Vídeo	16h
Anàlisis Sistema de la Pilota	16h
Anàlisis Node Servidor	16h
LLiurable 1	25h
Implementació Sistema de Vídeo	30h
LLiurable 2	8h
Testeig Sistema de Vídeo	12h
LLiurable 3	9h
Implementació Sistema de la Pilota	30h
LLiurable 4	8h
LLiurable 5	18h
Testeig Sistema de la Pilota	12h
Implementació Node Servidor	30h
Testeig del Node Servidor	12h
Assemblatge	15h
Testeig sistema complert	20h

Table 5.2: Taula amb la càrrega de treball de GEP.

5.3 Desviacions

Els temps de dedicació estipulat per a les diferents tasques fins al moment són sense comptar les possibles desviacions que poden sorgir a causa dels riscos a l'hora d'implementar cadascun dels sistemes.

Veient que tant el factor de risc de cadascun dels subsistemes com el temps donades aquestes desviacions varia i és concret de cada subsistema anirem cas per cas esmentant tant el risc com la possible solució, exposats anteriorment, i detallant-ne el temps que ens portaria i per tant la seva desviació i els seus costos.

5.3.1 Sistema de càmera

Veiem que aquí teníem dos factors de risc: Primerament el hardware de la càmera. La imatge ideal és des de l'aire amb l'enfocament cap al camp i en l'escena hi hauria tot el camp. Això però depenent de les dimensions del terreny de joc pot comportar una altura que no ens permet situar-hi allà la càmera o bé impliqui una alternativa amb la seva corresponent desviació temporal i de costos.

Donat aquest risc la solució que proposem és capturar camps i escenes més petites. Això comporta una desviació temporal molt petita, 1 dia de treball que hem determinat en 4 hores i un cost addicional a les hores de treball nul donat que no necessitem cap recurs nou.

5.3.2 Sistema de pilota

A més a més del hardware també tenim el software del sistema que compta com a part amb més risc de desviació donada la seva complexitat el programa de detecció d'informació de l'escena basat en Visió per Computadors.

La solució a les possibles dificultats de desenvolupar el programa amb la màxima qualitat en un entorn real tracta en utilitzar imatges per defecte o editades per tal de simplificar la complexitat dels algoritmes utilitzats en el programa basat en Visió per Computació. Això comporta una desviació petita però existent, ja que hauríem de comptar el temps d'edició de les diferents imatges, 2 dies de feina sent 8 hores. Com a costos addicionals a les hores de treball aquí també seria nul donat que tant el programari d'edició d'imatges que utilitzaríem com les mateixes imatges són de domini públic.

5.3.3 Sistema de pilota

En aquest sistema veiem que el principal factor de risc era la impossibilitat física d'utilitzar una pilota real de futbol com a suport físic del sistema donat o bé als problemes de comunicació que pot comportar com pel que fa a la robustesa donat que la pilota està sent colpejada contínuament. La solució proposada era fer una prova de concepte (PoC) utilitzant un suport físic que no perjudiquen sensiblement a la comunicació del sistema ni la força de colpeig posés en perill la integritat del sistema. Això a la pràctica consisteix a substituir la pilota per un embolcallat de paper i que el colpeig al sistema no fos real sinó controlat i amb una força que no trenqués el sistema. Això igual que les solucions anteriors no impliquen cap cost addicional a les hores de treball i molt poca desviació temporal, 4 hores d'un dia treballat, ja que no impliquen afegir ni modificar res del sistema.

5.3.4 Sistema comunicació

Finalment també existia un risc a l'hora d'implementar els dispositius addicionals a cada sistema que permet comunicar-los entre ells. Això pot ser que donada la tecnologia, Wifi per exemple, no fos capaç d'implementar-se. La solució proposada és utilitza connexió Ethernet on simplifica la complexitat del protocol i la seva implementació.

Llavors en aquest cas de la desviació temporal i el seu cost, 3 dies de treball són 12 hores, ara necessitem un nou recurs que serien els cables Ethernet. Tenint en compte que a més a més un camp de futbol hem vist que pot tenir una llargada de més de 100 metres haurem de comprar cables d'una llargada molt important. Això fa que un equip bobina de cable vermell categoria 7 s/ftp 200 m lliure d'halògens taronja costi 150 €. Com necessitem conectar el sistema de la pilota i de la càmera al nodo servidor fa que necessitem comprar-ne dos, 300 €.

5.4 Dependències i precedències

Per tal de mostrar les dependències utilitzarem un graf de dependències. Remarcar com ja s'ha comentat el valor afegit de treballar amb mòduls tan potencialment independents permetent així aïllar els problemes i rebaixar el risc i les desviacions final del projecte.

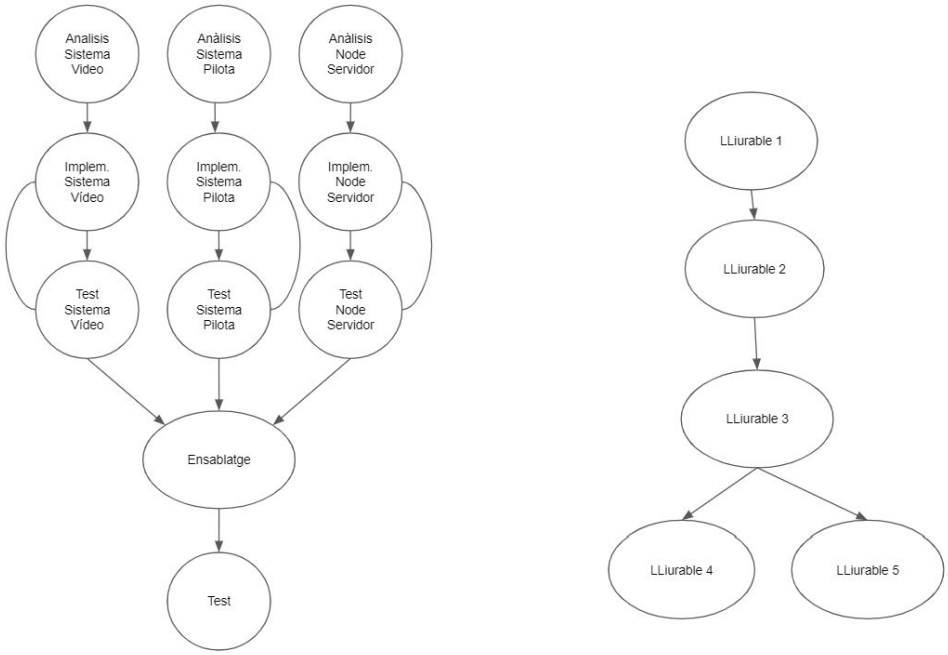


Figure 5.1: Graf de tasques i les seves dependències.

També veiem com el testeig opera de forma iterativa sobre la implementació de cadascun dels sistemes: Fins que el testeig no valida el sistema torna a la tasca d'implementació.

5.5 Diagrama de Gantt

A continuació de lapartat de Rols apareix el diagrama de Gannt donada lexplicació anterior.

5.5.1 Rols

Els rols de cadascuna de les tasques. En ser un projecte que no es realitzar en conveni amb cap empresa la totalitat de les tasques van a càrrec del desenvolupador. Amb tot, la distribució de les tasques que hem dissenyat i les diferents etapes marcarien diferents rols si això s'extrapola a un sector amb més personal o més especialitzats.

Figure 5.2: Diagrama de Gannt

Diferencia els rols de desenvolupador i tester atès que a cada iteració i com a tasca rellevant del projecte es dedica a fer un testeig, cada sistema d'un tipus concret, per tant aquests tests s'han de dissenyar, desenvolupar testejar i finalment posar en ús. La feina dels Desenvolupadors constaria llavors en dissenyar 'implementar els diferents sistemes.

El fet que cada sistema tingui tecnologies notablement diferenciades (Visió per Computadors i sistemes encastats tenen una base teòrica sensiblement diferent) fa que pogués haver-hi especialistes en cadascun dels equips de desenvolupament de cada sistema.

Remarcar també que al ser un projecte de Treball Final de Grau l'usuari no deixa de ser també el propi desenvolupador. Però si es desenvolupés el projecte per a sortir al mercat s'hauria de tenir en compte el rol d'usuaris. Aquests serien tant els àrbitres com els assistents d'àrbitres i tot el seu equip i probablement un nou sector de treballadors en el món del futbol professional que s'encarregués de gestionar i controlar les tecnologies aplicades al futbol. El temps que implicaria formar als primers seria nul, ja que el sistema és totalment autònom i per tant és una caixa negre per als usuaris. Si hi hagués algun problema amb el producte en l'àmbit del software o de fiabilitat del sistema haurien de ser els propis desenvolupadors del projecte els que s'encarreguessin d'arreglar-ho, ja que ells seran qui tinguin accés al codi font del producte.

Chapter 6

Gestió econòmica

Del pressupost general del projecte remarcar que gran part del pressupost anirà destinat als recursos hardware per implementar els diferents subsistemes encastats, a diferència de projectes informàtics orientats a desenvolupament software o sistemes d'informació els quals no requereixen costos addicionals en hardware més enllà dels ordinadors on desenvoluparan.

6.1 Costos

6.1.1 Recurs humans

A l'hora de pressupostar el nostre projecte desglossem els costos en funció dels diferents tipus de recursos. En l'apartat de recursos humans, cost del personal que ha treballat en el desenvolupament del projecte, tenim en compte que el desenvolupador en aquest cas concret al no ser per empresa sinó per la mateixa universitat el cost del desenvolupador és zero.

Alhora haurem de calcular el cost de les hores dedicades tant per al director del projecte com del responsable de GEP del projecte. Amb el mateix informe que anteriorment ens ha dit el preu per hora del desenvolupador junior traurem el cost del desenvolupador sènior, donada la seva experiència.

Prendrem com a referència del preu per hora del desenvolupador el que estipula l'estudi del mercat laboral HAYS [16] i que coincideix amb l'estudi fet per la Universitat Ramon LLull, ESADE, per a Infojobs [17] sent de 42.000 € l'any bruts. Això li hem d'aplicar un 28% de seguretat social que l'empresa, la universitat en aquest cas, ha de pagar. Per tant són 53.760 € bruts anuals de costos. Això dividit entre les 12 mensualitats (no tindrem en compte que fossin 14) són 4480 €. Posant que un mes laborable té 20 dies, a 8 hores diàries, són 28 € l'hora.

Remarcar que en ser un projecte individual i sense conveni amb cap empresa el desenvolupador s'encarrega de la tasca d'analitzar, dissenyar, desenvolupar i testejar el projecte, no hi ha un equip de treball és individual.

Donat que tenim una previsió de reunir-nos un cop cada mes unes 2 hores amb el director del projecte per portar el seguiment, això fa unes 8 hores més unes 6 hores addicionals de suport telemàtic. Referent al tutor de GEP i tenint en compte que GEP consta de 37,5 hores dedicades desenvolupar els lliuraments i les rúbriques (entenent el temps de lectura i aprenentatge dels conceptes) estimem que una tercera part del temps, 12 hores, són de treball per al projecte.

Les hores del desenvolupador surten de fer el sumatori de les hores totals comptabilitzades a l'apartat anterior de Gestió Temporal.

Recurs humà	Dedicació (hores)	Preu hora (€/any)	Cost total (€)
Director del projecte	14	28 €	280 €
Tutor GEP	12	28 €	240 €
Desenvolupador	270	0 €	0 €
Total:	296	-	540 €

Table 6.1: Gestió dels recursos.

6.1.2 Recursos software

Recurs software	Preu (€)	Unitats	Vida útil	Amortització(€/any)
MATLAB R2017b	0€	1	1 any	-
MATLAB Compiler	1150€ [4]	1	1 any	1150 €
MATLAB Runtime	0€	1	1 any	-
Sublime Text 3	0€	1	-	-
GCC Compiler	0€	1	-	-
Microsoft Windows 10 Home	145 € [3]	1	3 anys	48,3 €
Microsoft Office 365 Hogar	99 € [3]	1	3 anys	33 €
Google Drive	0€	1	-	-
ATENEA i Racó FIB	0€	1	-	-
Total:	244 €	-	-	1231 €

Table 6.2: Costos software.

Per la part de recursos Software haurem de tenir en compte: [18][19]

Eines de desenvolupament tot el programari necessari per desenvolupar els nostres 2 programes software: El programa encarregat d'obtenir les dades sobre els jugadors mitjançant visió per computadors i el mateix programa que determini si hi ha fora de joc i si és infracció. Com a eines de desplegament tot el programari ofimàtic i de sistemes d'informació per a portar el projecte amb la facultat.

La part de l'equip personal amb connexió a Internet a més a més dels propis del mobiliari i energia de l'entorn de treball del desenvolupador, donat que es porta a terme sense conveni amb cap empresa, també els tindrem en compte aquí.

6.1.3 Recursos hardware

Recurs software	Preu (€)	Unitats	Vida útil	Amortització(€/any)
ARDUINO YÚN REV 2	49,90 € [5]	2	3 anys	33,26 €
ARDUINO MKR WIFI 1010	27.90 € [5]	3	1 any	83,7 €
Arduino MKR IoT Bundle	64,90 € [5]	1	3 anys	21,63 €
Càmara Pixy (CMUcam5)	73,00 €	1	1 any	73,00 €
MEMSIC 2125 dual-axis acelerometer	54,22 €	1	1 any	54,22 €
KATANA II Slimbook	799,00 €	1	3 anys	266 €
Total:	1174,62€	-	-	531 €

Table 6.3: Costos hardware.

Finalment una de les parts més rellevant del projecte i també del pressupost és els dispositius hardware que implementen el sistema: Sistema de càmera, sistema de pilota i el node servidor. [20]

La justificació de la decisió dels diferents components hardware s'exposa més endavant en l'apartat de decisions. Remarcar també que aquest conjunt de components poden tenir modificacions que tenim en compte en l'apartat Contingències i desviacions. Aquí estan els components estàndards ni el cas pitjor ni el cas òptim on el fet d'implementar millores de rendiment pot implicar nova tecnologia i recursos hardware addicionals.

La vida útil dels nostres dispositius l'hem treta tenint en compte el cicle de vida de les seves memòries (pe: ARDUINO YÚN REV 2 Write/Erase Cycles: 10,000 Flash/100,000 EEPROM). Per altra banda els sensors estan establerts amb una vida útil d'un any.

6.1.4 Costos indirectes

Els costos indirectes del projecte són nuls atès que el mobiliari i l'entorn de treball del desenvolupador són necessaris per a la producció del projecte i per tant són considerats com costos directes.

6.2 Imprevistos i contingències

En aquest apartat tindrem en compte els imprevistos i contingències que poden succeir durant el projecte siguin desviacions o per altra banda casos ideals que poden portar a desenvolupar millores del sistema, donat per exemple pel compliment dels temps sense desviacions temporals.

Primerament tindrem en compte la desviació temporal a l'hora d'implementar el projecte. Anteriorment en l'anàlisi de riscos veiem que varia sensiblement en funció de la tasca (el sistema encastat de càmera tenia un risc alt en comparació al node servidor) pel que treballarem amb un risc mitjà prenent una mitjana de tots els riscos (i tenint en compte que on més risc hi ha és en la part més complexa). Amb tot tindrem en compte un 20% de contingència. Remarcar que en el cas que no hi hagi aquesta desviació aquest tant per cent esdevindrà beneficis del projecte.

Pel que fa a la desviació dels recursos hardware prenem el cas òptim que no hem tingut desviacions ni problemes durant el desenvolupament dels sistemes pel que podem implementar millores. La més significativa serà aplicar el programa de detecció de forma de joc sobre un clúster de procesadors específics. Tenint en compte que el nou sistema comptaria amb 4 Factory Price Cortex A9 i.MX6DL ARM Development Board Support Embedded per tal de fer el clúster, seria un total de 740 € (185 per placa de desenvolupament). Això representa

un 89% de contingència respecte al sistema encastat del node servidor i un I en el pressupost general. La part de software representa un risc nul i per tant cap desviació atès que ja comptem amb tot el programari o és indispensable com el cas de MATLAB Compiler.

6.3 Pressupost

Tenint en compte tots els costos desglossats anteriorment i les possibles desviacions, obtenim un pressupost final:

Recurs	Unitats	Amortització (€ / any)	Temps (hores)	Total (€)
Costos directes				
Gestió del projectes				
Microsoft Windows 10 Home	1	48,3 €	.	48,3 €
Microsoft Office 365 Hogat	1	33 €	.	33 €
MATLAB Compiler	1	1150 €	.	1150 €
Sistemes encastats				
ARDUINO YÚN REV 2	2	33,26 €	.	33,26 €
ARDUINO MKR WIFI 1010	3	83,7 €	.	83,7 €
Arduino MKR IoT Bundle	1	21,63 €	.	21,63 €
Càmara Pixy (CMU-cam5)	1	73,00 €	.	73,00 €
MEMSIC 2125 dual-axis acelerometer	1	54,22 €	.	54,22 €
KATANA II Slimbook	1	266€	.	266€
Recursos humans				
Director del projecte	14	28 €	.	280 €
Tutor GEP	12	28 €	.	240 €
Imprevistos	20% (6)	28 €	.	168 €
Imprevistos i contingències				
Factory Price Cortex A9 i.MX6DL ARM	4	380 €	.	380 €
Costos indirectes				
LLum	4	8	.	32 €
Quota ADSL	1	60	.	60 €
Total acumulat 2921 €				
Contingència 20% acumulat 584 €				
Total sense IVA 3505.2 €				
Total amb IVA 21% 4241,29 €				

Table 6.4: Pressupost.

6.4 Matriu de sostenibilitat

6.5 Impacte social

El nostre producte fa referència una de les majors indústries del món com és l'esport i concretament el futbol. Això fa que un producte amb tant de valor afegit sobre un fenomen social tan rellevant comporti un notable impacte social.

El més rellevant probablement seria acabar amb la confrontació en els partits de futbol basada en els errors de les decisions arbitrals durant els partits. Hem vist que la detecció de fora de joc amb els sistemes actuals tenen unes limitacions físiques, el temps de reacció de l'ull humà o l'augment de la possibilitat d'error quan hi ha molts elements dins del camp de visió de l'àrbitre. Això fa que puguin haver-hi errors arbitrals humans.

Amb la tecnologia i la precisió d'un sistema de visió per computadors aquest problema s'aboleix. A més a més les persones tenen consciència i per tant sempre hi ha el factor de risc d'equivocació premeditada i subjectiva per part de l'àrbitre. El sistema manca de consciència i és totalment objectiu i imparcial. Això faria que es mires el futbol amb molta més justícia i a més a més portaria més ètica, ja que els jugadors sabent que el sistema és imparcial haurien de reconèixer quan caiguessin en posició antireglamentària.

De manera directa, tot i no ser ni de bon tros el principal motiu, el fet d'aplicar justícia i baixar aquesta confrontació disminuiria la violència originada pel futbol i potenciaria la fraternitat entre simpatitzants de diferents equips.

A més a més cal remarcar que com a conseqüència, al final, l'ús d'aquestes tecnologies i el seu manteniment portaria lloc a crear un nou sector de treball dins de la indústria de l'esport, del futbol en el nostre cas, que seria la de tècnics de sistemes on hi formarien part des dels encarregats del manteniment físic i integritat del sistema, de problemes de comunicació, de cadascun dels mateixos subsistemes... creant una gran quantitat de llocs de treball.

-	PPP	Vida útil	Riscs
Ambiental	19,2 Kw/h = 60 (W/h) x 1/1000 (Kw/w) x 4(h/dia)x 20(dias/meses)x 4(meses)	3,3 V x 100 mA = 33 mW.Prenent que consumeix aquesta energia cada hora = 333 mW/hora	Si per algun factor la vida del nostre producte fos més curta shaurien de produir més productes en el temps que els predecessors haguessin estat vius. Això implica costos energètics en la construcció i de CO2 de les indústries. Alguns daquests factor podrien ser factor climatologics que trenquen la integritat del producte: Pluges, nevades...
Econòmica	La viabilitat econòmica i financera és completa donada a dos factors: Al estar modulat en subsistemes la simplificació dun daquests, o dun conjunt, en baixa el cost. Per altra banda, la gran part de material el podem aconseguir gràcies a cursar el grau a la Facultat Informàtica de Barcelona.	Preu 0,14 €/Kwh. 333mW /hora x 0.14 €/kWh /1000000 = 0,000044 €/hora -> 3 subsistemes = 0,000132 €/h x 168 h/dia = 0,03 €/dia. A més a més pensar que només ha destar encès durant el partit per tant a la setmana màxim 2 partits.	Podria no sortir al mercat donat a que a certs sectors no els interessi eliminar la polèmica ja que perden ingressos, o bé si el producte només fos docent i no adaptat a les condicions reals dun partit. Si larraïament del sistema fos petit el seu preu es devaluaria i acabaria sent menor que el cost de producció.
Social	Per una part la realització personal de desenvolupar un producte personal i aplicar els coneixements tècnics a altres passions cotidianes. De la mateixa manera explicar a companys i jugadors com aquesta tecnologia ajuda i mostrar la informàtica com una cosa atractiva	Per una part el fet daplicar un sistema informàtic precís i imparcial en comptes dun humà imprecís (visió humana) i potencialment subjectiu (consciència) potencia la justícia en lesport i rebaixa la confrontació. Alhora crearia llocs de treball de tècnics de sistemes en aquest sector.	Que els propietaris del sistema es corrompessin o fossin extorsionats per a modificar el codi de forma que no fos imparcial. Per altra banda si el sistema aixeca altes expectatives i el resultat acaba sent dolent això generaria sentiments de frustració que radicarien en més confrontació i desconfiança en el progrés tecnològic.

Table 6.5: Gestió dels recursos.

Chapter 7

Marc teòric

Aquest apartat exposarà l'arquitectura del sistema: Explicació de cadascun dels components de cada Node, la topologia de la xarxa i la comunicació entre nodes, la interfície gràfica per a establir paràmetres de configuració del sistema i l'entorn sobre el qual implementarem el sistema.

7.1 Sistemes en Temps Real

Un sistema en temps real (STR) és un sistema on el seu correcte funcionament ve donat perquè les sortides es processin abans d'un temps determinat [21] Un STR està definit per una llista de:

- Els esdeveniments externs que pot atendre. Aquest són les passades dels jugadors que determinen si s'ha de processar el fora de joc o no. Per a la PoC, el temps en el qual una passada es pot donar el determinarem en 2 segons. Cada 2 segons haurem de tornar a inspeccionar l'escena i la velocitat dels jugadors determinaran la regió total on potencialment pot estar el jugador passats els 2 segons i la que s'haurà de processar per cadascun .

- La resposta lògica que ha de produir el sistema. Aquesta no és més que retornar un '1' o un '0' en una variable lògica en funció si s'ha produït fora de joc o no. La sortida física d'aquesta variable lògica, que dependrà de la tecnologia física utilitzada com podria ser un led, juntament amb el temps computacional de la resposta serà el temps de la resposta lògica.

- Els deadlines que s'estableixin. Per a la nostra PoC establirem un deadline de 3 segons entre l'arribada d'un senyal de control de fora de joc i la sortida de l'existència o no del fora de joc.

Existeixen dos grans tipus generalitzats de STR:

- Hard STR: Són els STR on l'incompliment d'un deadline durant l'execució del sistema implicarà un funcionament incorrecte del sistema. Un exemple seria un marcapassos.

- Soft STR: Són els STR on l'incompliment d'un deadline durant l'execució del sistema simplement implica una degradació de la qualitat de la resposta però en cap cas un funcionament incorrecte. Un sistema de vídeo en temps real n'és un exemple.

El nostre STR doncs és un Soft STR. Si el nostre sistema processa la resposta després del deadline simplement s'afegirà un retard en la resposta i no trencarà el sistema. Això és degut al fet que un cop finalitzada la sortida de la resposta el sistema seguirà rebent les dades en temps real i per tant si es produeix una nova situació de fora de joc serà capaç de processar-la i respondre amb la resposta dins del deadline.

Una altre cosa però és si el sistema té una fallada abrupta com podria ser la caiguda del sistema d'alimentació. En aquest cas el sistema no és capaç autònomament de tornar a funcionar a causa de la necessitat d'engegar el sistema manualment i el prerequisit d'engegar el sistema abans del xiulet inicial, que justificarem més endavant, per al correcte funcionament del sistema.

Referent al concepte de l'OS del STR (RTOS) només remarcar que al ser un sistema on cada node treballa sobre una única tasca no és necessària una planificació de tasques i per tant no ens hem de preocupar de la inversió de prioritats ni l'algorisme a aplicar per a planificar les tasques.

7.2 Visió per Computadors

7.2.1 Segmentació i Thresholding

En visió per computadors, la segmentació és el procés de dividir tots els elements de la imatge en grups. Això s'aconsegueix creant imatges binaritzades on cadascun dels seus píxels només poden tenir 2 possibles valors, '1' o '0', a partir d'imatges en color. L'obtenció de les imatges binaritzades facilita el càlcul en el posterior processament de la imatge. [21][22]

El mètode per aconseguir que un píxel(i,j), on i,j representen una fila i una columna respectivament dins de la matriu associada a una imatge, d'un valor de color determinat en la imatge de l'entrada passi a valer o '1' o '0' a la imatge de la sortida és seleccionar un únic valor únic de color anomenat threshold (T). Important determinar quin model de color utilitzarem per tal de tenir tots els valors del procés de thershold sota el mateix format.

Aquest valor està en el format del color determinat (pe: RGB, HSV...). Aquests dos formats de color, els quals són els més estàndards (nosaltres treballarem en el format RGB), són formats definits pe valors de tres components. Si treballem en el format RGB, doncs, el valor T=[tR,tG,tB].

La segmentació que aplicarem, la qual acabarà de definir el procés de threshold que estàvem explicant anteriorment, és coneguda com la segmentació per background i foreground.

- En la segmentació de background per cada píxel (i,j) de la imatge d'entrada el valor del pixel(i,j) a la sortida val '0' si pertany al fons i '1' altrament.

- Per altre banda en la segmentació per foreground per cada píxel (i,j) de la imatge d'entrada el valor del pixel(i,j) a la sortida val '0' si pertany a imatges en primer pla i '1' altrament.

Veiem doncs que la relació entre aquestes dues segmentacions, sempre i quan només hi hagi dos grups d'elements: el fons i elements sobre d'ell, és la negació. Una és la negació de l'altre.

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) \geq T \\ 0 & \text{otherwise} \end{cases} \qquad g(x,y) = \begin{cases} 1 & \text{if } f(x,y) < T \\ 0 & \text{otherwise} \end{cases}$$

(a) Segmentació per background. (b) Segmentació per foreground.

Figure 7.1: Segmentació per background i foreground.

Seguint doncs amb el procés de threshold, si volem aplicar una segmentació per background, el valor T serà el valor que identifica el fons de la imatge. A partir d'aquí determinarem un valor anomenat desviació, d, el qual determinarà un interval I = [T-desviació,T+desviació]. Per a la nostre T en format RGB haurem d'expressar el valor desviació amb 3 paràmetres d = [dR,dG,dB] i per tant: IR in [TR-dR,TR+dR], IG in [TG-dG,TG+dG] i IB in [TB-dB,TB+dB].

Finalment, si el valor del píxel(i,j) està dins d'aquest interval I (cada component està dins de l'interval corresponent a la seva component) el píxel(i,j) a l'imatge sortida valdrà '0'. Altrament valdrà '1'.

Aleshores aquest concepte l'aplicarem per implementar el nostre thresholding per background i poder aïllar els elements de la gespa. Ir(x,y), Ig(x,y) i Ib(x,y) són el valor R,G,B respectivament de la coordenada(x,y). Rpeak, Gpeak i Bpeak corresponen al valor RGB mitjà de la gespa. Finalment Rth,Gth i Bth són els thresholds per a cadascuna de les components. Si la diferència entre la component i el peak és més petita que el threshold significa que està dins de l'interval definit anteriorment. En aquest cas el pixel de la màscara binaritzada valdrà 1. Altrament valdrà '0'.

Les últimes dues comparacions són per assegurar-nos que el color serà verd. Això passarà sempre que el valor de les dues altres components sigui menor a la component G. [27]

$$B(x,y) = \begin{cases} & |I_R(x,y) - R_{peak}| < R_{th} \\ & |I_G(x,y) - G_{peak}| < G_{th} \\ 1 \quad : \quad & |I_B(x,y) - B_{peak}| < B_{th} \\ & I_G(x,y) > I_R(x,y) \\ & I_G(x,y) > I_B(x,y) \\ 0 \quad : \quad & \text{otherwise} \end{cases}$$

Figure 7.2: Aplicació del thresholding al sistema

7.3 Geometria

7.3.1 Espai afí i distància euclidiana

Per a especificar formalment el nostre sistema necessitarem el concepte d'espai afí.

Un espai afí és una tupla (A, E, D) formada per un conjuntament A, un espai vectorial n finitament generat sobre el cos K i una aplicació tal que: [28]

$$\begin{array}{rcl} \delta : \mathbb{A} \times \mathbb{A} & \longrightarrow & E \\ (p,q) & \longmapsto & \delta(p,q) \end{array}$$

En el nostre cas A que determina un punt dins de l'espai. La primera component tindrà un rang (0, número de fileres de la imatge) la segona (0, número de columnes de la imatge). Així doncs A x A determina 2 punts en l'espai (podrien ser el centre de la capsa contenidora de dos jugadors) l'espai de sortida serà l'espai vectorial E2 que determina el vector des d'un jugador i un altre.

Aquest concepte l'utilitzarem per a calcular la distància entre un jugador i la pilota. El punt A farà referència a la coordenada (i,j) del pla cartesià referent al centre de la capsa contenidora del jugador mentre que el punt B serà la coordenada (ii,jj) del pla cartesià associada al centre de la capsa contenidora de la pilota. Amb la condició 1. i 2. sabem que el valor sempre serà positiu i l'ordre de les diferències no afectarà a la distància calculada.

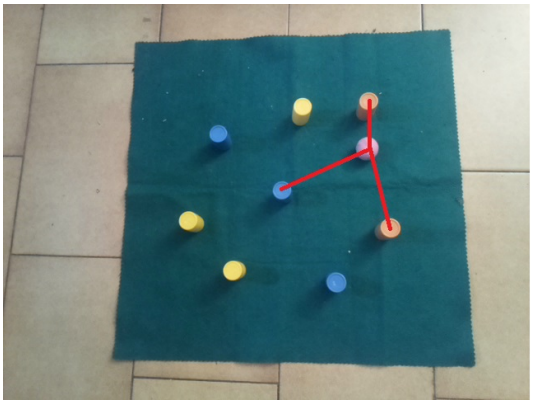


Figure 7.3: Les línies vermelles marquen la distància entre elements

7.4 Xarxes

7.4.1 Sockets

Els sockets són una abstracció a través de la qual que ens permetrà establir un canal de comunicació on enviar i rebre dades entre diferents nodes per la xarxa mitjançant funcions des del nivell aplicació com poden ser funcions en C++. [29]

Estan identificats de forma única gràcies 3 paràmetres:

- Una adreça única (IPv4 en el nostre cas).
- El protocol de comunicació a nivell de transport del protocol TCP/IP: UDP o TCP
- Un port

Això determina dos tipus de sockets en funció del protocol seleccionat:

- Stream: Sockets identificats pel protocol de transport TCP. És orientat a la connexió. Llavors el canal es defineix establint la connexió entre els dos nodes. Les dades que han de ser enviades s'emmagatzemen en un buffer de transmissió, s'envien per la xarxa, i es reben en un buffer de recepció. El receptor rebrà tots els paquets i en ordre sense perdre'n cap.
- Datagrama: Sockets identificats pel protocol de transport UDP. No està orientat a la comunicació per tant no s'ha de configurar la comunicació entre els sockets dels nodes i els missatges enviats es poden perdre en la comunicació sense cap mecanisme de comprovació o també poden arribar en fora d'ordre. El paquets entre ells no tenen relació. La mida màxima d'un paquet 65535 bytes.

En el nostre cas nosaltres utilitzarem Streams donat a que la pèrdua d'una dada originaria un resultat incorrecte del sistema: Si perdem el byte de BallTrigger no s'avaluarà la situació de fora de joc o si perdem bytes en la transmissió de les imatges l'algorisme de detecció inicial i seguiment té moltes probabilitats de fallar.

Aleshores en la comunicació entre nodes mitjançant sockets s'estableixen dos rols: Client i Servidor.

Mentre el Servidor és qui espera la petició del client i respon amb les dades corresponents el Client és qui inicia la comunicació i fa les peticions al Servidor. Aquest a l'hora d'establir la comunicació ha de conèixer l'adreça i port del Servidor.

En el nostre cas els Nodes Pilota i el Node Càmera són els que tenen el rol de Servidor mentre que el Node Principal és qui té el rol de client i per tant es qui inicia la comunicació amb els dos Servidors coneixent les seves direccions IP prèviament i els seus ports que en aquest cas estan assignats arbitràriament pel desenvolupador.

Aleshores l'establiment d'un canal de comunicació mitjançant streams amb aquest model Client Servidor és el mostrat en la següent imatge:

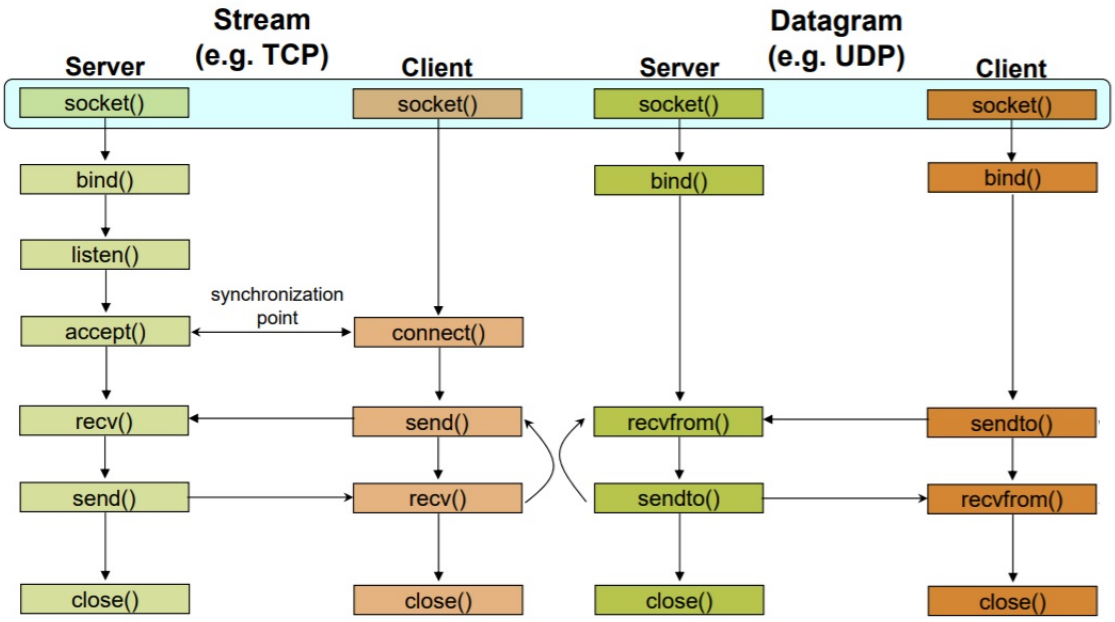


Figure 7.4: Diagrama amb el flux de les funcions C de comunicació per sockets

- socket() és l'encarregat de crear el socket en cadascun dels nodes respectivament. Aquí es determina quin tipus de protocol i tipus de socket utilitzarem. Retornen un file desriptor que identifica el socket per a poder utilitzar-lo en les següents funcions. Fixem-nos que no se li assigna l'adreça IP per tant aquest socket encara no està associat a cap IP ni cap port.
- Anàlogament, la funció close s'encarrega de tancar el socket de cada node alliberant el port utilitzat. Se l'hi ha de passar l'identificador del file descriptor per saber de quin socket es tanca.

- Com hem vist al diagrama anterior ara el Servidor crida a les funcions bind() i listen() en aquest ordre. La funció bind() és l'encarregada de associar el socket creat prèviament i identificat pel file descriptor al port i l'adreça IP que nosaltres vulguem. Un cop ja tenim aquesta associació la funció listen() es l'encarregada d'esperar a que el Client iniciï la comunicació amb ell. Es bloquejant fins que no rep aquesta petició. El Node Pilota i el Node Càmera seran els encarregats de cridar aquestes funcions.

- Ara el Client és l'encarregat d'establir comunicació amb aquests sockets mitjançant la funció listen(). Se li especifica aquí sí l'adreça IP i el port del socket concret per el qual volem establir el canal de comunicació. Així doncs haurem de cridar la funció listen() tants cops com canals com vulguem tenir des del Node Principal (3 en el nostre cas. Un per al socket creat per al BallTrigger i un altre per al canal associat al Button Trigger per als canals amb el Node Pilota i el canal per on enviar les imatges per als canals amb el Node Càmera).

- Finalment ja es pot enviar i rebre informació mitjançant les funcions recv() ni send() respectivament com marca el diagrama.

Amb aquestes funcions implementarem la comunicació entre els Nodes del sistema. Crearem dos mòduls els quals explicarem més endavant anomenats Speaker i Listener que estaran associats als rols de Servidor i Client respectivament i que implementaran les diferents funcionalitats a més alt nivell per a poder comunicar els nodes mitjançant els conceptes i les funcions explicades en aques apartat.

Chapter 8

Arquitectura del Sistema

Aquest apartat exposarà l'arquitectura del sistema: Explicació de cadascun dels components de cada Node, la topologia de la xarxa i la comunicació entre nodes, la interfície gràfica per a establir paràmteres de configuració del sistema i l'entorn sobre el qual implementarem el sistema.

El funcionament del sistema serà el següent: Tindrem 3 nodes: El Node Principal, el Node de la Càmera i el Node de la Pilota on la càmera i l'acceleròmetre estan associats als últims dos respectivament. Aleshores la càmera estarà constantment fent fotografies del camp amb els jugadors i la pilota. Amb això el Node Principal calcularà la posició de cada jugador i el posseïdor de la pilota a cada moment. Quan hi hagi una passada l'acceleròmetre enviarà un senyal al Node de la Pilota i aquest al Node Principal. Finalment el Node Principal al rebre el senyal calcularà amb la informació dels jugadors i la pilota si en aquest moment hi ha hagut fora de joc o no.

8.1 Nodes

Aquí definirem cadascun dels nodes els quals conformaran el nostre sistema i el seu hardware.

El Node Principal serà l'encarregat de, tenint tota la informació necessària, determinar si existeix o no situació de fora de joc en un moment determinat. Serà qui rebrà les imatges i les diferents senyals de control dels demés nodes i també és on es llençaran els algorismes de visió per computació. El node està composat per un Intel Core i7-4790 de 3.60 GHz i 4 nuclis de 2 threads per nucli (8 unitats de procés) i 16 GB de memòria RAM. L'elecció d'aquest hardware respecte d'altres ve pel fet de ja tenir-lo prèviament i implicar un cost 0 a la pràctica a més a més de ser el processador dels disponibles amb més capacitat de processament.

El Node de la Càmera serà el node encarregat de prendre fotografies dels jugadors i la pilota sobre el terreny de joc a cada instant. Està integrat per una Raspberry Pi Zero i un Càmera RasPi Cam v2. Per a poder fer les fotografies des d'adalt està adherit a un cilindre, on situat per sobre de la superfície de la gespa, podrà prendre les imatges satisfactòriament.

Figure 8.1: Node Principal Intel Core i7-4790

Vam seleccionar la Raspberry Pi Zero per sobre d'altres SBC descrits anteriorment per dues raons:

- Les Raspberry Pi treballen amb un OS, instalat prèviament pel desenvolupador, mentre que altres SBC com Arduino no en tenen. Això podria afegir soroll a l'execució del software en el sistema però seria ínfim. Per altre banda el fet de treballar amb sistemes operatius basants en Linux ens simplifica enormement la compilació del software utilitzant els compiladors de gcc i g++. Per altre banda podrem incloure i utilitzar les diferents llibreries orientades a C fàcilment.

- El model Zero de Raspberry Pi té integrat un mòdul Wifi 802.11 n. Això juntament amb la programació en C i el sistema operatiu orientat en Linux fa que ens sigui notablement simple connectar-nos a la xarxa amb els nostres dispositius encastats. Alhora ens simplifcarà la programació dels mòduls de connectivitat a la xarxa per parlar amb els altres Nodes.

Un cop seleccionada la Raspberry Pi Zero com hardware del Sistema de la Càmera directament hem seleccionat la RaspiCam v2 com a càmera per al nostre sistema. La fàcil integració amb la Raspberry Pi i la facilitat d'utilitzar-la mitajnçant llibreries, a més a més del seu cost reduït en comparació a altres opcions, justifiquen la seva selecció.

Finalment el Node de la Pilota serà l'encarregat d'enviar les senyals de control al Node Principal per a que aquest efectuï les accions corresponents. Aquests senyals són el BallTrigger i el ButtonTrigger. Està compresa per una Raspberry Pi Zero, un acceleròmetre i

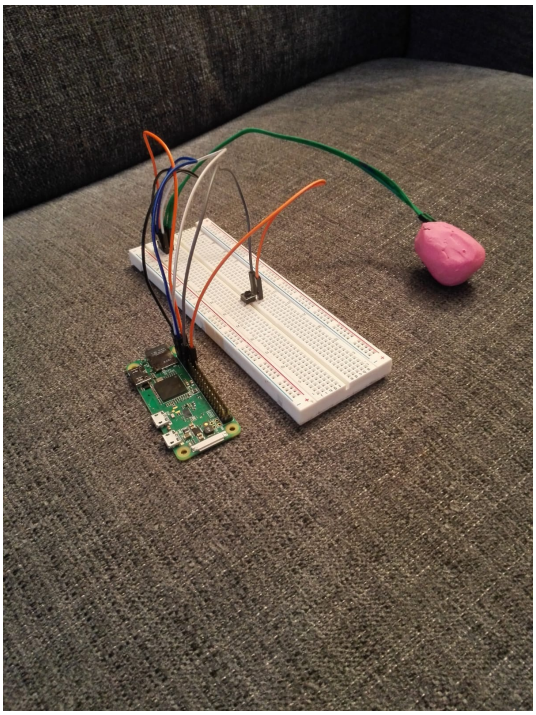


Figure 8.2: Node Càmera amb la RaspiCam v2

giroscopi MPU6050 i un botó polsador de 2 pins Push Button Tact Switch de 6x6x9. La selecció de la Raspberry ha sigut justificada prèviament.

Figure 8.3: Node de la Pilota amb la pilota i el botó de seguiment

8.2 Topologia

La topologia del sistema defineix la comunicació dels diferents elements del sistema entre ells. Aquesta ve representada per la topologia en xarxa, les connexions per xarxa entre els diferents nodes i els missatges que s'hi envien, i la connexió dels perifèrics d'entrada del

sistema als diferents nodes: la càmera l'acceleròmetre i el botó de seguiment. La topologia de la xarxa està representada pel següent esquema:

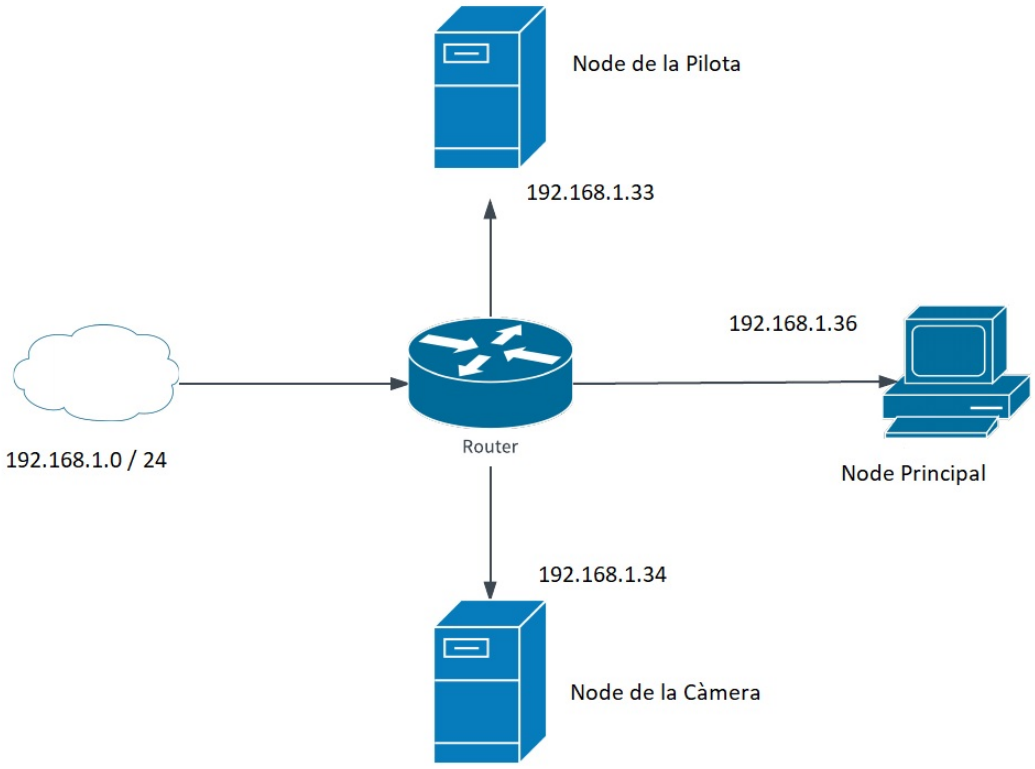


Figure 8.4: Topologia de la xarxa del sistema

El node de la càmera i el node de la pilota estan interconnectats amb el node principal. Pel canal de comunicació entre el Node Principal i el node de la càmera s'enviaran els següents missatges: El Node Principal podrà enviar al Node càmera peticions de càmera. Aquest missatge implica un 1B en la transmissió. Per altre banda el Node càmera podrà enviar imatges al Node Principal. El missatge que conté la imatge són 900 KB.

Entre el Node Principal i el Node de la pilota s'establiran dos canals de comunicació: El canal del BallTrigger i el canal del ButtonTrigger. El comportament és el mateix per als dos: El Node Principal pot enviar peticions en forma de missatges cap al Node pilota que consten d' 1B i el NodePilota podrà enviar el BallTrigger i el ButtonTrigger, per cadascun dels canals respectivament, cap al Node Principal amb un altre missatge d' 1B.

Quant als dispositius d'entrada estan connectats de la següent manera:

La càmera està connectada al Node Càmera per la interfície ZIF 15 de la Raspberry Pi Zero a través del protocol MIPI CSI-2. Aquest protocol, estàndard de la connexió i comunicació entre els dispositius mòbils i les càmeres, té un ample de banda de 4 Gbps (1

per cadascuna de les 4 línies). La llibreria RaspiCam s'encarregarà de la gestió del protocol a baix nivell.

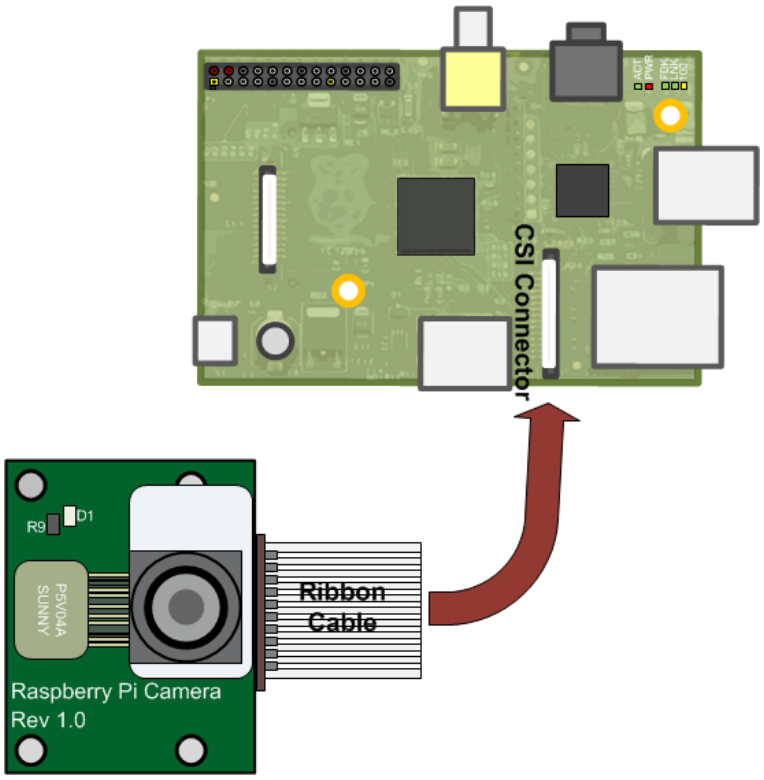


Figure 8.5: connexió RaspiCam i Raspberry Pi

El botó de seguiment està connectat al Node de la Pilota mitjançant els pins GPIO de la Raspberry Pi Zero que el conforma. Concretament la terra prové del PIN 14 (numeració física, veure la següent imatge) i la sortida cap al Node de la Pilota està connectada al GPIO 17, el PIN 11.

L'acceleròmetre MPU6050 està connectat al Node Càmera a través dels pins del GPIO i es comuniquen a través del protocol I2C. Per a poder comunicar-se via I2C la connexió dels pins ha de ser la següent:

El PIN 1 i el PIN 6 del GPIO de la Raspberry Pi Zero es connecten als pins VCC i GND del MPU6050 (Veure següent imatge). Representen l'alimentació de 3.3V i la terra respectivament. El PIN 3 de la GPIO, el qual representa el SDA en el protocol I2C, va connectat al pin SDA del MPU6050. Finalment el PIN 5, que representa el SCL en el protocol I2C, va connectat al pin SCL del MPU6050.

8.3 Interfície gràfica

Aquí descriurem la interfície gràfica que voldrem implementar al noste sistema i la sortida del mateix.

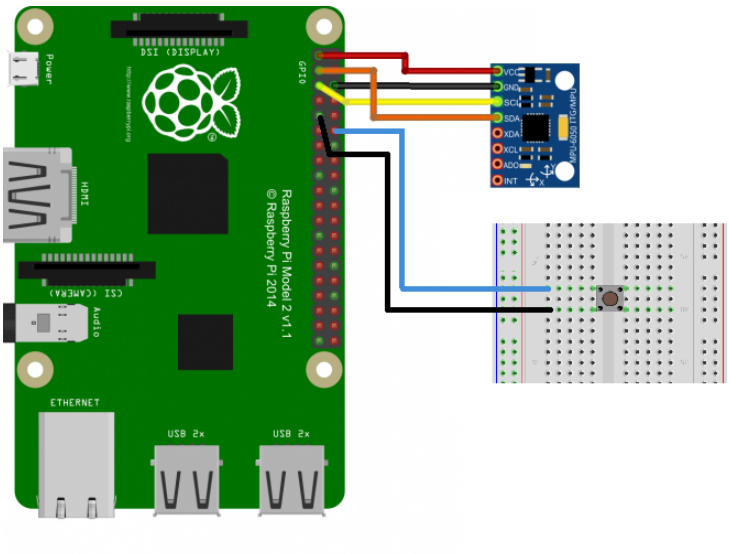


Figure 8.6: connexió Botó de Seguiment i MPU6050 i Raspberry Pi

L'interfície gràfica ens serviria per establir de forma fàcil i parametritzada aquells valors que poden canviar en diferents execucions del sistema i que el sistema no pot aconseguir de forma autònoma. Bàsicament comptemplavem dues idees: Una UI petita escrita en C++ per a no tenir problemes de comunicació entre llenguatges o bé un fitxer .xml on poder canviar els valors dels valors desitjats.

Vam veure però que això no era necessari degut a que realment les dades del sistema que poden canviar simplement són les direccions IP dels diferents nodes (quan el router perd l'alimentació torna a assignar les direccions IP).

Per altre banda tenim la sortida del sistema. Auqesta podria ser molt variada i alhora complexa: Des de crear una aplicació mòbil connectada al sistema, un sistema mecànic que fes més visual la sortida del sistema... Però finalment hem implmentat la sortida estàndard pel terminal. Aquesta és la que hem estat utilitzant durant tot el desenvolupament. La segona implementació que voliem fer constaava en encendre un Led. Aquí haurem de tenir en compte el temps que el Led està encés, ja que com hem definit anteriorment en un STR la sortida del sistema és la que està associada amb el deadline i si aquest temps és massa gran podríem perdre senyals durant l'execució del sistema. A l'apartat 10.6 hi ha una imatge amb la sortida del sistema.

8.4 Entorn

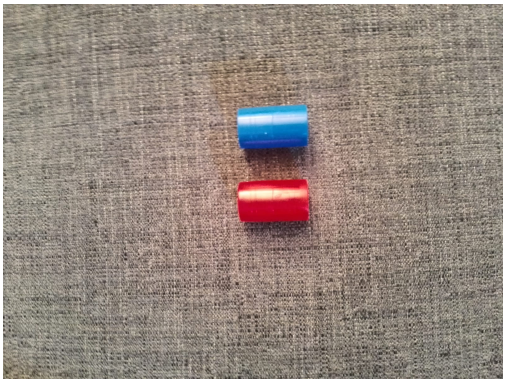
En aquest apartat explicarem l'entorn sobre el qual hem desenvolupat la PoC i les seves característiques.

La xarxa per on es comunicaran tots els nodes té un ample de banda de 1 MB/seg. El

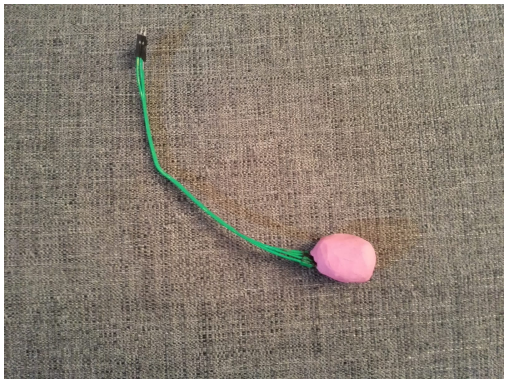
Router encarregat de enroutar els diferents paquets cap als nodes és un Router Mitrastar HGW-2501GN-R2.

En aquest punt diferenciarem dos entorns diferents: L'entorn de desenvolupament i l'entorn de la demostració. Per a l'entorn de desenvolupament:

- La superfície que simularà el terreny de joc està conformada per una taula de joc. És de color verd uniforme i les dimensions són de 49x49 cm. La càmera està centrada a l'estora i a una altura de 62 cm respecta a la taula de joc, que està estirada al terra.
- Com a jugadors, a l'entorn de desenvolupament, utilitzarem cubilets de parxís. la seva estructura cilíndrica simulen l'estructura d'un jugador. Inicialment, a més a més. l'homogeneïtat d'un mateix color per tota la superfície semblava simplificar la detecció dels equips. Per qüestions de disseny que s'explicaran més endavant això no ha sigut necessari.
- Com a pilota hem utilitzat plastilina per a embolcallar l'acceleròmetre i posteriorment donar-li la forma esfèrica. Que sigui mel·lable ens permetia tenir la pilota sense preocupar-nos de les dimensions de la pilota. Els cables els hem hagut de pintar de verda, ja que sinó afegien soroll a l'algorisme de visió per computadors.



(a) Cubilets com a jugadors.



(b) Pilota amb l'MPU6050 integrat.

Figure 8.7: Elements de l'entorn de desenvolupament

Chapter 9

Desenvolupament

En el següent apartat exposarem el procés de desenvolupament de tot el sistema: Des dels recursos utilitzats fins a una explicació exhaustiva de la implementació de cadascun dels elements del sistema.

9.1 Recursos utilitzats

Primerament enumerarem cadascuna de les eines que hem utilitzat per desenvolupar el sistema així com la justificació del per què les hem seleccionades: Llenguatges utilitzats, llibreries incloses i tot el programari necessari.

9.1.1 LLenguatges utilitzats

9.1.1.1.0 C++

És el llenguatge principal del sistema. Utilitzat per a la implementació de la xarxa, els fitxers Main de cadascun dels nodes així com les funcions auxiliars per a cadascun d'ells, com prendre fotografies amb a la càmera o aconseguir l'acceleració detectada per l'acceleròmetre.

És el llenguatge en el que estan desenvolupades totes les llibreries que necessitem, a més a més de la simplicitat per compilar en les Raspberry amb els compiladors de gcc i g++ i desenvolupar el sistema de compilació mitjançant fitxers Makefile. Per tot això es va decidir utilitzar C++ com a llenguatge principal per a implementar el sistema.

9.1.1.2.0 MATLAB

És el llenguatge utilitzat per a la implementació dels algorismes de detecció dels jugadors i la pilota basats en visió per computadors. Inicialment vam tractar amb diferents llenguatges com OpenCV++ per tal de facilitar la comunicació entre els algorismes de visió per computadors i les demés funcionalitats en C++.

Però les poques funcionalitats prèviament implementades juntament amb el poc rendiment en l'execució dels programés contrastava amb les numeroses llibreries de MATLAB per a visió per computadors així com la facilitat per cridar-les i el gran rendiment de les execucions.

9.1.2 LLibreries

9.1.2.1.0 MATLAB Engine API per a C++

MATLAB Engine API per a C++ és una llibreria que ens proporciona una interfície entre el llenguatge C++ i els programés basats en MATLAB. Permet des de un programa en C++ córrer programés en MATLAB. Suporta la MATLAB Data API, que s'explica seguidament.

És una llibreria indispensable per a nosaltres gràcies a la qual podem integrar els algorismes de visió per computadors al nostre sistema de detecció de fora de joc.

9.1.2.2.0 MATLAB Data API

La MATLAB Data API ens permet enviar els paràmetres a les funcions en MATLAB i rebre'n els seus resultats des dels nostres programés en C++. Va lligada a la MATLAB Engine API per a C++ i ambdues són necessàries degut que la gran majoria de les nostres funcionalitats en MATLAB necessiten arguments i retornen resultats.

9.1.2.3.0 RasPiCam

És una llibreria desenvolupada pel grup d'investigació Aplicacions de la Visió Artificial (A.V.A) que permet utilitzar les càmeres de Raspberry Pi sota la llicència BSD. D'una manera molt simple i amb 4 funcions auxiliars hem pogut implementar les funcionalitats d'inicialització de la càmera així com fer fotografies i establir-ne diferents paràmetres com podria ser la mida (640x480 ...)

9.1.2.4.0 WiringPi

WiringPi és una llibreria per a C, C++ , desenvolupada per Gordon Henderson, orientada a l'accés dels PINs GPIO dels SoC BCM2835, BCM2836 i BCM2837 utilitzats en les Raspberry Pi. Està sota la llicència GNU LGPLv3.

Implementen diverses funcions que permeten l'accés als diferents pins ja sigui com entrades o com a sortides. Això ens ha permès implementar diverses funcionalitats per als nodes basats en Raspberry Pi: Integrar dispositius d'entrada i sortida com botons, leds i l'acceleròmetre MPU6050.

9.1.3 Programari

9.2 Implementació

En aquest apartat veurem la implementació de cadascuna de les funcionalitats principals del sistema així com el disseny d'aquestes i els algorismes utilitzats.

9.2.1 Mòdul Speaker

El mòdul Speaker està format pel fitxer 'speaker.cpp'. La seva funcionalitat és oferir funcionalitats per a poder enviar dades a través d'un socket cap a un altre node de la xarxa.

Bàsicament hi ha 3 tipus de funcionalitats

- Les funcions de configuració del socket. start speaking() es l'encarregat d'obrir el socket i lligar-lo a un port concret. meeting() es l'encarregat d'esperar a que algú estableixi connexió amb aquest socket i l'accepta. stop speaking() simplement tanca el socket.
- Els Request: Són funcionalitats que envien '1' pel socket retornat en meeting(). Són utilitzades per demanar explícitament algun servei a un altre node. Són req ball(), req camera() entre d'altres.
- Les de Speak: Són funcionalitats que envien la variable determinada per la funció pel socket retornat en meeting(). Són les funcionalitat que envien la informació rellevant de cada sistema: speak button(), speak pass() entre d'altres. Menció especial a speak img(). Obre un canal associada la imatge, 'top.ppm' concretament, i guarda en un buffer tots els bytes de la imatge. Seguidament els envia pel socket retornat per meeting().

9.2.2 Mòdul: Listener

El mòdul Listener està format pel fitxer 'listener.cpp'. Aquest s'encarrerga d'escoltar les dades enviades a través d'un socket per un altre node de la xarxa. Les funcionalitats implementades són les següents:

- Les funcions de configuració del socket: start listening() és l'encarregat de, un cop un node ha opert un socket, conectar-se a aquell socket especificant el port establert per aquest i la seva adreça IP. Aquí veiem dues coses rellevants: Primer haurem d'inicialitzar els sockets des dels nodes que implementen el mòdul Speaker, Node de la Pilota i Node de la Càmera, per a poder obrir el mòdul amb l'start listening() dels sockets, el Node Principal. La IP s'ha de especificar per tant serà necessari conèixer les direccions IP prèviament i posar-les dins la funció per connectar-se correctament. Retorna el canal associat al socket. stop listening() simplement tanca el canal.
- Les funcionalitats de Recv: Són funcionalitats que escolten el valor enviat pel node Speaker de la variable especificada per la funció. Serveixen per sincronitzar el sistemes i simplement llegeixen aquesta variable. Un exemple seria recv camera().
- Les funcionalitats de Listen: Són les funcions que escolten el valor enviat pel node Speaker de la variable especificada per la funció. Serveixen per obtenir els valors dels perifèrics d'entrada dels demés nodes. El Node Principal és qui les utilitza. Un exemple és listen button(). Menció especial a listen img() la qual escoltarà la imatge enviada pel node Speaker. Crea el fitxer obrint un canal i amb un buffer escolta les bytes enviats per la funció speak img(). Finalment els escriu al fitxer utilitzant el canal obert i un cop acabat el tanca.

9.2.3 Mòdul MPU6050

El mòdul MPU6050 implementa les funcions necessàries per tal de llegir les dades enviades pel dispositiu MPU6050 però a més a més també implementa les funcionalitats per inicialitzar i llegir el botó de seguiment. Aquestes hi són al fitxer 'mpu6050.cpp'. Podríem diferenciar les següents funcionalitats:

- La configuració i inicialització del MPU6050. setupMPU6050() inicialitza la comunicació amb el dispositiu activant l'interfície I2C i deshabilita l'Sleep Mode. A més a més fa una primera lectura per tenir el valor inicial amb el qual començar a calcular els següents diferencials d'acceleració. La sensibilitat respecte l'acceleració a la que està escrit l'MPU6050 és per defecte 16384 LSB/g i això ho tenim en compte a l'hora de normalitzar l'acceleració.

- isTrigger() és la funció encarregada de llegir l'acceleració en aquell moment, calcular el diferencial d'acceleració respecte l'anterior acceleració i si aquesta supera el threshold preestablert retornarà '1', s'ha produït una passada, o '0' altrament. Abans de retornar el resultat actualitzarà l'actual valor de l'acceleració amb la obtinguda en aquest moment. Remarca l'ús de la variable 'retorno' la qual després de calcular un diferencial d'acceleració per sobre del tirgger posa aquesta variable a '1' i el següent trigger l'ignora.

  Això és degut a l'efecte que es produeix quan és rep una passada. Al rebre la pilota i impactar amb la bota (per exemple) es genera el trigger degut a la desacceleració abrupta de la pilota. Però el contacte amb la bota a la recepció fa que la pilota reboti i en aquest rebot el diferencial d'acceleració és prou gran com per enviar un altre trigger. Aquest trigger l'haurem d'ignorar ja que no està associat al concepte de passada o recepció.

- Finalment isButton() s'encarrega de retornar '1' si el botó és premut i '0' si no ho és. El botó està connectat de tal manera que si no és premut el circuit s'obre i no es produeix diferència de potencial. Si està premut es tanca el circuit i es genera la diferència de potencial que està associat a la constant 'HIGH'.

9.2.4 Mòdul Càmera

El Mòdul Càmera està implementat al fitxer 'camera.cpp' i té només dues funcionalitats: configurar inicialment la càmera i prendre les fotografies. Remarcar que la gran part de funcionalitats en aquest mòdul vénen donades per la llibreria RaspiCam. Aleshores:

- setup camera() inicialitza la càmera. Això és posar els valors que vulguem als atributs de l'objecte RaspiCam per poder prendre les fotografies correctament. Destaquen el format del color, RGB en el nostre cas, i les dimensions de la imatge, 640x480px en el nostre cas. Aquesta última decisió ve donada per la relació resolució i temps de transmissió. És prou gran com per tenir la definició òptima per al processament d'imatges i es prou petita com per no tardar excessivament en la transmissió de la imatge i no arribar al deadline.

- La funcionalitat photo() s'encarrega de fer la fotografia. Utilitzant les funcionalitats de RaspiCam fa la fotografia i la guarda en el directori des d'on es llença aquest codi amb el nom 'top.ppm'.

9.2.5 Mòdul Intercomunicació

El nostre sistema està programat en C++ però per contra els nostres algorismes de detecció inicial i seguiment estan programats en MATLAB. És clar doncs que necessitem una interfície per a poder comunicar-nos enter els processos que corrin codi en C++ i els processos que corrin codi de MATLAB. En el fitxer 'interlenguage.cpp' tenim la implementació d'aquesta utilitzant les llibreries de MATLAB Engine y MATLAB DataArray.

Bàsicament implementa tres funcions:

- iniMATLAB() s'encarrega d'obrir una sessió MATLAB on gràcies a les API podrem llençar els nostres programes de MATLAB.
- Per altre banda tenim endMATLAB() la qual tanca aquesta sessió oberta.
- getPlayersMatrix() és l'encarregat de llençar l'algorisme de detecció inicial o de seguiment i retornar la PlayersMatrix. Te dos paràmetres d'entrada: Un enter que determina quin dels dos mètodes hem d'executar i el vector PlayersMatrix que hem d'entrar al mètode de seguiment en cas d'executar-lo. Remarcar que podem passar els valors de PlayersMatrix passats com a paràmetre als diferents programes de MATLAB gracies a la llibreria de MATLAB DataArray així com agafar el resultat d'aquests programes i retornar-los en llenguatge C++.

9.2.6 Detecció inicial

Per a poder detectar les diferents situacions de fora de joc primerament necessitem captar la posició inicial de cadascun dels jugadors així com a quin equip pertanyen. De la mateixa manera necessitem detectar la informació inicial referent a la pilota. Amb aquesta detecció inicial posteriorment podrem anar seguint els diferents elements (jugadors i pilota).

En el fitxer 'ini.m' s'implementa aquesta detecció mitjançant la funció 'getPlayersMatrix'. Com a entrada utilitza la fotografia 'top.ppm'. Aquesta és la fotografia presa pel node de la càmera just en el moment abans del xiulet inicial. Com a sortida retorna un vector 'res' de 36 elements amb tota la informació rellevant en el següent format:

- res[0]: Conté l'índex 'i' del jugador que controla la pilota.
- Seguidament tenim els 4 jugadors de l'equip A i els 4 jugadors de l'equip B en el següent format:
- res[(4*i)+1] Límit superior de la capsa contenidora del jugador 'i'.
- res[(4*i)+2] Límit inferior de la capsa contenidora del jugador 'i'.

- res[(4*i)+3] Límit esquerre de la capsa contenidora del jugador 'i'.
- res[(4*i)+4] Límit dret de la capsa contenidora del jugador 'i'.
- On 'i', l'índex del jugador, va des de 0 fins a 7, ambdós inclosos. Des de 0 fins a 3 són els jugadors de l'equip A i del 4 fins al 7 els jugadors de l'equip B.
- res[33],res[34],res[35] són les components R,G i B respectivament de la detecció del color RGB de la gespa.

Els nostres sistemes de visió per computadors, tant el de detecció inicial com el de seguiment, estaran basats en la detecció de Blobs. Per a aconseguir això necessitarem una imatge binaritzada associada a 'top.ppm' on els jugadors i la pilota valguin 1 metre que cada píxel de la superfície on buscarem aquests blobs, la gespa, valgui 0. Això li'n direm 'FieldMask'.

Figure 9.1: Imatge inicial top.ppm

Hem de tenir en compte que la càmera capta una superfície més extensa que no només la gespa de tal manera que existeixen uns marges que delimiten i localitzen la gespa. També hem observat que els píxels de la gespa són el 40% dels píxels totals i el quadrat que conforma la superfície de la gespa està centrat respecte al centre de la càmera.

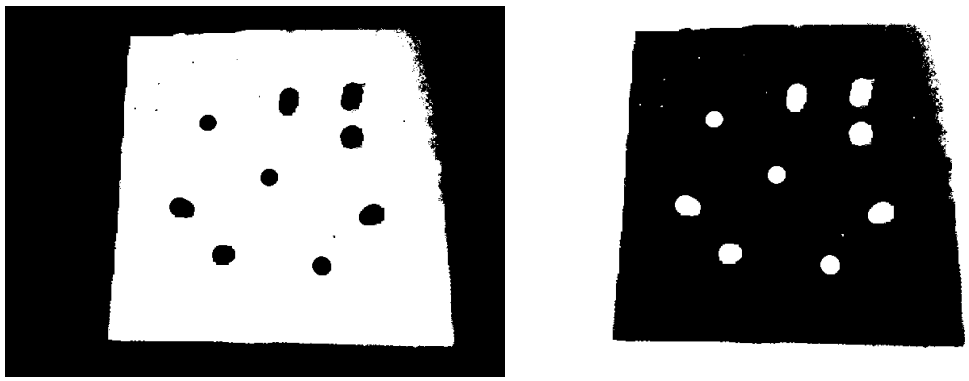
Amb tot això, per a aconseguir el 'FieldMask' i els marges de la gespa, basat-nos en l'article 'Where are the ball and players?' [X], implementarem les següents tres funcionalitats:

- Aconseguirem el valor R,G,B mitjà de la imatge 'top.ppm' a partir dels histogrames dels valors R,G,B. Al ser els píxels de la gespa el 40% dels píxels totals, el valor màxim de cadascun dels histogrames farà referència al valor R,G,B mitjà de la gespa. Els anomenarem RPeak, GPeak, BPeak. [Y]
- Aplicarem un processat de threshold sobre la imatge 'top.ppm': Amb el valor obtingut iterarem sobre cadascun dels píxels on si el valor R,G,B del píxel (i,j) està dins de

l'interval marcat per RPeak, GPeak, BPeak i un diferencial, el píxel a FieldMask' valdrà 0. Altrament valdrà 1. Així aconseguirem el 'FieldMask'

- Per trobar els límits que delimiten la gespa utilitzarem el següent algorisme: Per trobar el límit superior comencem a iterar des de la primera fila. Recorrem totes les columnes i comptem quants píxels valen 0. Si no superen més del 50% no haurem trobat la gespa. Per tant augmentem de fila i repetim el procés fins a trobar la primera fila on el número de píxels que valen 0 sigui igual o superior al 50% . Aquest serà el llindar. Anàlogament ho fem per al límit inferior i per als laterals.Són les funcions top field(), bottom field(), left field() i right field().

Ara ja tenim la imatge binaritzada i els límits de la gespa pel que podem començar a detectar els diferents Blobs.



(a) PlayersMask. (b) FieldMask.

Figure 9.2: Imatges binaritzades.

Iterarem sobre tots els píxels de la gespa utilitzant els marges calculats anteriorment. Amb això mirem si el píxel(i,j) del 'FieldMask' val 1. Mentre no valgui 1 seguim avançant. Quan trobem un píxel que val 0 hem trobat un Blob i ara l'hem de recorre per tal de trobar els límits superior, inferior i laterals de la seva capsa contenidora cridant la funció Blob(i,j) on i,j són la fila i la columna respectivament del píxel processat. Com que estem fent el recorregut per fileres el primer píxel que detectem del Blob serà el límit superior. A partir d'aquí l'algorisme serà el següent:

- Mirarem els píxels veïns. Començant pel píxel(i+1,j), si el seu valor en 'FieldMask' val 0 vol dir que és part del Blob, però hi hauran dues condicions més que haurem de mirar. Primer hem de veure que el píxel que estem analitzant no es troba fora de la gespa degut a algun píxel residual. Això ho fem amb la funcio 'in of bounds()'. Per altra banda píxel processat ja no l'hem d'analitzar més. Això ho implementarem amb una nova matriu anomenada 'Processed' de la mida de 'top.ppm' i amb tots els valors a 0. Cada cop que processem un píxel(i,j) el valor de Processed(i,j) val 1.

- Si compleix totes les condicions significa que el píxel(i+1,j) és part del Blob. Al ser el veí inferior establim 'j' com el límit inferior de la capsa i seguim buscant més Blobs cridant recursivament a Blob(i+1,j).

- Això ho farem també per als demés veïns (i,j+1) i (i,j-1).

Iterant sobre tota la matriu aconseguirem un vector amb tots els Blobs sobre la gespa.

Finalment passarem un procés de Post-processament sobre el vector de Blobs per tal d'eliminar els Blobs residuals i quedar-nos només amb els 8 jugadors.

Per això aplicarem els següents filtres:

- Eliminarem el Blob si el nombre de píxels que el comprenen és més petit que un llindar establert empíricament. Així eliminem aquells Blobs residuals originats de petits píxels, o grups de píxels, que formen part de la gespa però no estaven dins del llindar del threshold.
- Eliminarem el Blob si l'altura de la capsa contenidora és el doble de la seva amplada i viceversa. Hem analitzat empíricament que les capses contenidores dels jugadors tendeixen a ser quadrades per tant qualsevol capsa contenidora que sigui pronunciadament rectangular no serà de cap jugador.

Amb això ja tenim els 8 Blobs corresponents als jugadors d'ambdós equips. Finalment per poder afegir cada jugador al vector res com hem explicat anteriorment necessitem saber a quin equip pertany cadascun.

Aprofitant la pre-condició que s'estableix en el moment de la detecció inicial pel reglament de la FIFA [X], tots els jugadors de l'equip A es troben a una meitat del camp mentre la resta de jugadors de l'equip B es troben a l'altra meitat. Com el nostre algorisme de detecció de Blobs fa el recorregut per files començant per la posició (0,0), anirà trobant els Blobs ordenats per files. Així tindrem tots els jugadors d'un equip a les primeres 4 posicions i la resta a les 4 últimes sense necessitat de cap algorisme amb un potencial temps computacional elevat com podria ser l'Homografia.
Aquest vector resultant serà guardat com a 'currentPlayersMatrix'

9.2.7 Seguiment

Amb la situació inicial de tots els elements(la posició de cada jugador donats els límits de les capses contenidores, l'equip al que pertanyen, i el posseïdor de la pilota) ara podem aplicar un algorisme de seguiment per detectar al llarg del temps les noves posicions dels jugadors i quin jugador té la possessió de la pilota. Aquest programa està implementat al fitxer 'tracker.m'

El sistema de la càmera fa la fotografia i envia la imatge cada Tcam al Node Servidor. En un entorn real cada imatge rebuda seria processada per l'algorisme de seguiment ja que els jugadors es poden moure autònomament i canviar de posició en Tcam.

En el nostre entorn de la PoC en canvi, els jugadors no es poden moure autònomament. Si volem canviar de posició els hem de moure manualment. Les imatges preses durant el canvi de posició dels jugadors no es rellevant degut al soroll afegit per la mà que canvia el jugador de posició. Així doncs utilitzarem un botó per discretitzar el temps i decidir quan aplicar l'algorisme de seguiment:

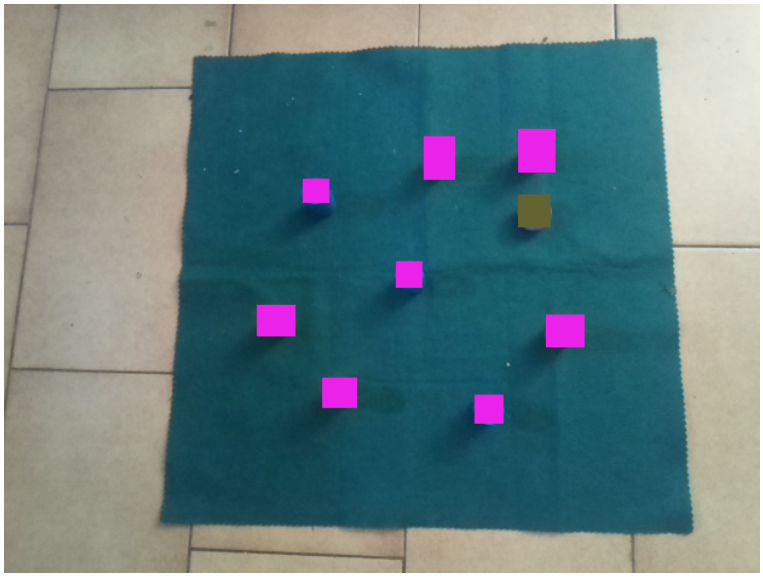


Figure 9.3: Imatge final de debug amb tots els elements detectats

- Mentre el botó no estigui premut l'algorisme de seguiment no s'aplicarà. En aquest temps es podrà canviar la posició dels jugadors ja que a cada Tcam s'actualitza 'top.ppm'. Quan el botó es prem s'executa l'algorisme de seguiment. La imatge que es processi no ha de tenir el soroll originat del moviment manual dels jugadors.

L'algorisme de seguiment té com a entrades la imatge 'top.ppm' i el vector 'currentPlayersMatrix'. La sortida d'aquest serà un vector de 36 posicions amb el mateix format que el descrit anteriorment a l'algorisme de detecció inicial. El principi d'aquest algorisme és el següent:

En un entorn real aquest seguiment es fa cada Tcam, on Tcam és un temps tan petit com sigui possible. Això fa que el moviment del jugador en un interval de temps tan curt sigui molt petit.

Llavors donada l'última posició captada pel sistema amb les capses contenidores de cada jugador, buscarem la nova posició del jugador mirant a totes les potencials posicions on hi pot ser, anomenada zona de cerca, determinades pel rectangle amb el mateix centre que la capsa contenidora del jugador , amb els límits dret i inferior més grans i el superior i l'esquerra més petits.
Aplicarem doncs l'algorisme de detecció de Blobs començant per la posició corresponent al vèrtex superior esquerra de la calculada zona de cerca calculada per a cada jugador.

Per poder aplicar l'algorisme de detecció de Blobs necessitem de nou tenir la imatge binaritzada i amb els límits de la gespa trobats. El valor Rpeak , Gpeak i Bpeak que s'utilitzaran en el threshold ens venen donats per l'entrada ja que ho hem calculat en el moment inicial i no varia respecte el temps. Amb aquesta memorització passant el valor cada cop per l'entrada ens estalviem temps de computació. per obtenir la imatge binaritzada

'FieldMask' i els límits ho calculem com anteriorment.

Amb això obtindrem els nous límits de les capses contenidores de cada jugador. Aplicant l'algorisme de detecció de la pilota i de posseïdor de la pilota ja tenim el vector resultat res. Aquest vector resultant actualitzarà el vector 'currentPlayersMatrix'.

9.2.8 Detecció de la Pilota

Tant en l'algorisme de detecció inicial com en l'algorisme de seguiment és necessari determinar qui és el posseïdor de la pilota. El jugador posseïdor de la pilota serà aquell amb la distància Euclidea mínima entre el centre de la capsa contenidora del jugador i el centre de la capsa contenidora de la pilota.

Necessitarem llavors detectar la posició de la pilota. Utilitzarem l'algorisme de detecció de Blobs aplicat en els algorismes explicats anteriorment. De la mateixa manera treballarem sobre el 'FieldMask' calculat en cadascun dels algorismes.

La pilota és invariable durant el temps i el seu disseny sol ser uniforme en tota la seva superfície de tal manera que sempre tindrà el mateix histograma amb un cert marge.
Seogns la normativa FIFA [X] a l'inici del partit abans del xiulet inicial, quan apliquem l'algorisme de detecció inicial, la pilota estarà al centre del terreny de joc. Aleshores en la detecció inicial anirem al centre de la imatge i el Blob que detectem serà la pilota. Aconseguirem els valors R,G,B de l'histograma del píxels que conformen la pilota i els guardarem com RBall, GBall i BBall.
Aquests valors són els que utilitzarem per discriminar de tots els Blobs detectats en l'algorisme de seguiment quin d'ells és la pilota.

Un cop detectada la pilota haurem de calcular les distàncies Euclides des del centre de la capsa contenidora de la pilota fins el centre de les capses contenidores de cadascun dels jugadors. Aquell jugador el qual la distància des del seu centre sigui la menor serà el posseïdor de la pilota.

9.2.9 Sistema de la Pilota

El sistema de la Càmera és l'encarregada d'enviar si l'acceleròmetre integrat a l'interior de la pilota supera el límit d'acceleració establert que determina que s'ha originat una passada o una recepció de la pilota, que anomenarem BallTrigger. També és el node on tindrem integrat el botó que ens permetrà llençar l'algorisme de seguiment, el qual anomenarem ButtonTrigger.

La implementació consta del fitxer Main, al fitxer 'main ball.cpp' juntament amb les funcionalitats explicades anteriorment de Listener i Speaker per poder comunicar-se amb el Node Servidor a través de la xarxa i les funcionalitats del mòdul MPU6050 per poder llegir la informació de l'acceleròmetre.

El codi Main s'encarrega de les següents funcionalitats:

- Inicialitza dos sockets de comunicació amb el Node Principal: Per un enviarà i rebrà l'informació referent al BallTrigger mentre per l'altre serà el canal de comunicació per al ButtonTrigger. També inicialitza l'acceleròmetre.
- Ara entra en un bucle infinit on primer espera la petició del Node Principal per rebre el BallTrigger. Un cop rebut n'envia el valor detectat. Seguidament espera la petició del Node Principal per rebre el ButtonTrigger i un cop rebut n'envia el valor corresponent.
- Si surt del bucle infnit tanca els sockets i finalitza l'execució.

9.2.10 Sistema de la Càmera

El sistema de la càmera s'encarrega de prendre les fotografies del camp constantment i enviar-les al Node Principal.

Aleshores els fitxers que implementen aquesta funcionalitat són el Main, al fitxer 'main camara.cpp' juntament amb els que implementen les funcionalitats de Listener i Speaker per a poder comunicar-se amb el Node Principal, especialment la funcionalitat d'enviament de la imatge, i les del mòdul Càmera per poder prendre les fotografies i inicialitzar la càmera. El comportament del Main és el següent:

- Inicialitza el socket per on es comunicarà amb el Node Principal i també la càmera.
- Entrem en el bucle infinit on primerament pren la fotografia. Abans d'enviar la imatge amb la funció speak img() espera rebre si el NodePrincipal fa una petició explícita de tornar a realitzar la fotografia.
  Això es donarà quan el Node Principal hagi detectat que hi ha una passada o una recepció i per tant voldrà tornar a fer la fotografia per detectar l'actual posició dels jugadors i de la pilota i el seu corresponent posseïdor. Si no es fes aquesta petició explicita la fotografia associada a la passada o la recepció es faria quan la pilota està en moviment i per tant no es detectaria correctament qui n'és el receptor.
  Si hi ha petició tornarà a fer la fotografia. Després, tan si hi ha hagut fotografia com si no, enviarà la imatge al Node Principal.
- Si surt del bucle infnit tanca els sockets i finalitza l'execució.

9.2.11 Node Principal

El Node Principal és l'encarregat de rebre els senyals dels diferents sistemes per a poder actualitzar correctament la posició actual i anterior de tots els jugadors juntament amb el posseïdor de la pilota actual i anterior i poder determinar, sempre que hi rebi un BallTrigger, si existeix situació de fora de joc.

El fitxers que conformen el sistema són el Main 'main node' juntament amb les funcionalitats de Listener i Speaker per poder rebre informació del diferents nodes i les funcionalitats

pròpies com la funció de detecció de fora de joc Offside() i les seves funcions auxiliars. Especialment el mòdul d'Intercomunicació amb el qual podrem córrer els algorismes de visió per computador i alhora passar-ne i rebre'n les dades corresponents.

El comportament del Main en el Node Principal és:

- Estableix comunicació amb els sockets oberts prèviament de la Càmera, els dos oberts al sistema de la pilota per al BallTrigger i del ButtonTrigger. A més a més inicialitza MATLAB a través del mòdul Intercomunicació.

- Si tot s'ha inicialitzat correctament haurem d'aconseguir les dades inicials dels jugadors. Per això necessitarem la imatge en aquest moment inicial i llençar l'algorisme de detecció inicial. Això ho fem enviant la petició de càmera al sistema de la càmera. Seguidament escoltarà i rebrà la imatge que s'ha captat en el sistema de la càmera. Ara obtenim la PlayersMatrix cridant a l'algorisme de detecció inicial i guardem l'anterior PlayersMatrix a oldPlayersMatrix i l'acabada de processar a la variable PlayersMatrix.

- Un cop inicialitzades totes les dades entrem en el bucle infinit. Ordenarem al sistema de la càmera que faci la fotografia i l'ha enviï. Ho fa tal i com hem explicat anteriorment. Un cop rebuda, envia la petició per rebre el ButtonTrigger i el rep. Si val '1' llença l'algorisme de seguiment i actualitza oldPlayersMatrix amd l'actual PlayersMatrix i la PlayersMatrix amb la PlayersMatrix processada.
  Finalment envia la petició de BallTrigger al sistema de la pilota i en rep el valor. Si val '1' aleshores envia explícitament la petició per a que el sistema de la càmera faci la fotografia amb la posició de la pilota ja estabilitzada, justificada anteriorment, llençant de nou l'algorisme de seguiment i actualitzant les PlayersMatrix per finalment córrer l'algorisme de Offside. Si retorna 1 la sortida del sistema determina que hi ha hagut fora de joc. Altrament no hi ha sortida. Torna a iterar així contínuament.

9.2.12 Offside

La funció Offside() és una funció pròpia del Main del Node Principal que s'encarrega de decidir si en el moment actual i l'anteriorment processat hi ha situació de fora de joc. Dediquem un capítol exclusiu per a aquesta funció, pròpia del mòdul explicat anteriorment, degut a la seva rellevància.

Com a entrades Offside() té la oldPlayersMatrix i PlayersMatrix. La sortida serà '1' si es produeix una situació de fora de joc entre els dos moments representats per les dues PlayersMatrix, mentre que valdrà '0' si s'hi produeix.
Remarcar que les dades referents a la oldPlayersMatrix fan referència al moment de la passada mentre que les dades de PlayersMatrix fan referència al moment de la recepció.
L'algorisme segueix els següents passos:

- Primer estableix quin és l'equip passador i quin és l'equip receptor. Mirant el primer element de la oldPlayersMatrix i PlayerMatrix obtindrem l'índex del passador i receptor

respectivament. Com a les PlayersMatrix primer tenim els jugadors de l'equip A i després el de l'equip B, si l'índex és inferior a 13 farà referència a un jugador de l'equip A. Altrament farà referència a un jugador de l'equip B.

- Només hi ha fora de joc si la passada és entre jugadors diferents (si fossin el mateix jugador es tractaria d'una auto-passada) del mateix equip. Per tant si el passador i el receptor són d'equips diferents o bé el passador és el mateix jugador que el receptor ja no hi haurà fora de joc i retornarem '0'.

- Si són diferents jugadors del mateix equip la següent condició a analitzar és si la passada s'efectua cap endavant o cap endarrera. En cas que sigui cap endarrera no hi ha fora de joc.
  Això ho podem saber comparant els límits adequats de la capsa contenidora del passador i del receptor. Important remarcar que els límits a mirar seran diferents si es tracta de l'equip A o l'equip B donat que l'equip B ataca direcció cap amunt i l'equip A ataca direcció cap abaix.
  Si la passada s'efectua cap enderrera retorna '0'.

- Si la passada és cap endavant analitzarem l'última condició: Si el receptor en el moment de la passada estava més avançat que el defensa més endarrerit s'esdevé situació de fora de joc. Igual que abans, té rellevància quin és l'equip atacant per utilitzar uns límits o uns altres. Finalment si estava més avançat retornarem '1' i altrament retornarem '0'.

Chapter 10

Testing

Aquí s'explicarà el sistema de testeig per a cadascun de les funcionalitats principals així com els resultats obtinguts. S'exposarà el mètode de testeig i l'ubicació dels datasets pujades repositori. La verificació del test ha sigut manual.

10.1 Algorismes visió per computadors

Per als algorismes de visió per computadors hem testejat l'algorisme de forma aïllada donada la imatge a testejar. Aquestes, les quals conformen el dataset de testing per a aquests algorismes, es troben dins del repositori al directori test/PlayerDetection.

les imatges anomenades 'a [num].png' són les primeres imatges testejades encarades a la detecció de Blobs. Les imatges 'm [num].png' ja proven totes les funcionalitats dels algorismes com detectar la gespa dins la imatges...

Finalment les imatges de format .ppm són les imatges usades per testejar l'algorisme un cop ja s'ha integrat dins del tot el sistema.

També tenim imatges 'ball*' les quals s'ha utilitzat per a testejar la detecció del color mitjà de la pilota.

10.2 MPU6050

El testeig del mòdul MPU6050 i de la implementació del Mòdul Pilota es va testejar prova-assaig.

No hi és el dataset però si remarcar l'utilitat del resultats d'aquest per a poder arreglar els errors del mòdul MPU6050 referents a la variable 'retorno', entre d'altres.

Es va testejar aïllat del sistema només en el Node Pilot i finalment és va provar en tot el sistema integrat.

Figure 10.1: Exemple de les imatges *.ppm

Figure 10.2: Exemple de les imatges ball*

10.3 Botó

Per a testejar la funcionalitat del Botó vam crear un driver de testeig anomenat 'button.cpp' aquest simplement implementa la funcionalitat de isButton() de forma aïllada.

Per a testejar la funcionalitat del botó de seguiment el dataset utilitzat també és a mode de prova-assaig.

10.4 Offside

El driver 'main debug.cpp' és on testejarem la funció Offside(). Té definit inicialment la oldPlayersMatrix i la currentPlayersMatrix i així pot executar directament l'algorisme de

forma independent al Node Pilota i al mòdul Intercomunicació.

La implementació de dirver 'main debug()' simula el comportament d'un sowtware de testeig convencional. Itera sobre tot el dataset de testeig i per a cada cas, testcase, llença la funcionalitat Offside() i compara amb el valor esperat guardat en memòria i precalculat manualment pel tester prèviament i treu per la sortida estàndard el resultat del testcase (si la sortida es redirigeix cap a un fitxer tenim els logs de l'execució).

10.5 Intercomunicació

10.6 Sistema

El testeig de tot el sistema integrat ha sigut una tasca costosa degut a l'entorn. Des d'un ordinador remot hem obert 3 connexions SSH per treballar amb els tres nodes. Inicialitzant cadascun dels nodes hem anat testejant cadascuna de les funcionalitats dins del sistema.

Les dificultats es trobaven a l'hora de fer un canvi en algun fitxer per solucionar algun error al codi. Al ser nodes remots diferents el procés era pujar el canvi al repositori, descarregar-lo a cadascun dels nodes i tornar a compilar el sistema.

En el repositori remot es troba la branca Debug la cual té totes les funcions de sortida (printf i fprintf) utilitzades per debuggar el sistema integrat des dels terminals.

També efectuat a mode de prova-assaig remarcar l'utilitat d'aquest per trobar l'error de sincronització entre nodes explicat anteriorment.

La següent imatge mostra l'entorn de testeig del sistema integrat des de l'ordinador remot:

Figure 10.3: Captura de l'entorn d'execució del sistema

Chapter 11

Planificació final i desviacions

Aquí comentarem la planificació final del desenvolupament del projecte i les desviacions que s'hi han produït.

En la planificació inicial es tenia com a objectiu presentar el projecte la setmana del 28 de Gener. Finalment l'exposició es durà a terme la setmana del 22 d'Abril, concretament el 23 d'Abril.

Tenim una desviació de gairebé 3 mesos respecte als 6 mesos estimats per al desenvolupament i presentació del projecte. Aquestes desviacions vénen donades pels següents punts i s'exposen en la següent taula:

- A causa de la impossibilitat de complir les hores de desenvolupament marcades a la planificació inicial des de principis de Novembre fins a mitjans de Desembre. La causa d'això va ser l'augment d'hores de treball en la feina contractual del desenvolupador. Aquesta va ser la principal causa per posposar la presentació del projecte a la setmana del 22 d'Abril.

- A causa del canvi de disseny en els algorismes de visió per computadors. Com s'ha comentat anteriorment s'ha arribat a dissenyar i desenvolupar, no fins a la implementació final, fins a 3 algorismes diferents per a la mateixa funcionalitat. Aquest temps de pensar i dissenyar el nou algorisme, així com el temps invertit a adquirir els coneixements per poder fer els dissenys esmentats anteriorment (especial incís al temps invertit en aprendre conceptes com l'Homografia el qual no ha tingut cap aplicació en la implementació del algorismes). Concretament:

  - El primer algorisme que vam implementar tractava d'aconseguir totes les posicions dels jugadors dins del camp i els seus respectius equips donada una imatge lateral, típica de les retransmissions televisives dels partits, com a entrada. Un cop el vam implementar ens vam adonar dels diferents problemes que tenia així com ineficiències: Primerament si hi havia oclusió dels jugadors, podia arribar a ser 12 impossible detectar els jugadors o en la majoria dels casos complicava l'algorisme, ja que es tractava com un mateix Blob. Alhora necessitàvem llençar les línies de fuga per obtenir les posicions en coordenades reals. Això era costós d'implementar i també computacionalment.

- La segona implementació que vam fer és situar la càmera sobre del camp de manera que no és necessari llençar línies de fuga per aconseguir les posicions significatives dels jugadors (tot i que existeix un petit error donat per la vista axonomètrica de les càmeres en comptes de la idíl·lica ortogonal). El problema de la càmera superior és la impossibilitat de detectar l'equip de cadascun dels jugadors donat que els color de la samarreta, element de referència per a saber de quin equip és cadascú, no es pot captar de manera significativa amb la càmera superior.
- Com a solució es va plantejar posar una càmera situada a l'esquerra del camp i una a una de les porteries captant el camp de manera frontal. La idea és detectar en cadascuna de les dues càmeres tant la posició dels jugadors detectats com els seus respectius equips. Finalment triangular totes les posicions dels jugadors de la càmera lateral i de la frontal i si una d'aquestes es correspon amb les posicions de la càmera superior, podem assignar aquell equip al jugador. Està basat en el concepte de les vistes de Planta perfil i alçat donat un objecte.

Això tenia una implicació important en la planificació temporal degut a que no és un algorisme trivial d'implementar. Aleshores es va acabar implementant l'algorisme de detecció inicial descrit anteriorment utilitzant les precondicions de la situació inicial abans del xiulet. Amb això ja sabem a quin equip pertany coadascun dels jugadors inicialment i per tant no és necessària les implementacions anterior. Aquest replantejament de l'algorisme de visió per computadors ha sigut un dels grans culpables del canvi de dates, com veurem a l'apartat de Planificació final.

- Degut als temps d'adquisició del harware específic per a poder seguir desenvolupant i testejant el sistema, parcialment i en la seva totalitat. Un exemple són els mòduls MPU6050 i mòdul Càmera els quals estan implementats mitjançant llibreries suportades només en les arquitectures Raspberry. Fins el moment que no vam poder adquirir les diferents Raspberry el desenvolupament d'aquests mòduls, entre d'altres, va estar bloquejat.

- Degut a altres factors que han tingut una afectació, menor però al cap i a la fi rellevant, al temps de desenvolupament com la generació de tot el sistema de Build o el temps addicional degut a l'entorn de testeig de tot el sistema integrat.

Això va modificar la nostra planificació que va quedar de la següent manera:

També un gran canvi significatiu respecte a la planificació inicial ha estat el hardware sel·leccionat. Per desconeixement de tot el hardware que el mercat posava a disposició vam optar per hardware que posteriorment, com s'ha justificat aquí, es va refusar i escollint nou hardware.

Això modifica la taula de costos hardware que havíem vist en l'apartat 6.1.3:

- Raspberry Pi Zero en comptes d'Arduino. El preu unitari de la Raspberry Pi Zero són 26,70 €. 3 Rapsberry Pi Zero són 80,7 €.

Setmana	Tasca
Gener	.
31/12/2018	Modificar el software de Visió per Computador principal, top .
14/01/2018	Implementar el software de Visió per Computador left i bottom
21/01/2018	Acabar dimplementar la funcionalitat genèrica de tot el Sistema de Visió per Computador
28/01/2018	Implementar el Sistema de la Pilota
Febrer	.
04/02/2019	Implementar el Sistema de la Pilota
11/02/2019	Comunicar procés en C amb un procés en MATLAB al MainNode
18/02/2019	Crear els mòduls Speaker i Listener per comunicar Sistemes per sockets
25/02/2019	Implementar la funcionalitat de detecció de fora de joc al MainNode i acabar dimplementar Speaker i Listener
Març	.
04/03/2019	Implementar la funció de tracking i re factoritzar el codi de Visió per Computador per millorar-ne el rendiment
11/03/2019	Millorar la Performance del software de Visió per Computador mitjançant la Parallel Toolbox de MATLAB
18/03/2019	Finalitzar la implementació de tot el Sistema.
25/03/2019	Documentar Memòria
Abril	.
01/02/2019	Documentar Memòria
08/02/2019	Documentar Memòria
15/02/2019	Preparació de lexposició
22/02/2019	Exposició

Table 11.1: Planificació final.

- MPU6050 és finalment l'acceleròmetre seleccionat. El cost unitari és de 6 €.
- La RaspiCam v2 és la càmera seleccionada com a conseqüència de canviar de SBC. El preu unitari és de 26.99 €. Dues càmeres són 53.98 €
- Tota la resta de eïnes necessàries per a poder treballar amb el Sistema de la Pilota com seria la placa protoboard, les resistències, el botó de seguiment... Això és el kit ELEGOO de components electrònics. Val 13.99 €

Si fem la suma això són 157.67 €. Si fem la comparació amb els costos hardware inicials són notablement més barats. Això es degut a que les Raspberry Pi Zero ja tenen integrats el mòdul Wifi i per tant no s'ha de comprar el mòdul apart. A més a més el cost de la Raspberry Pi Zero és la meitat que l'Arduino. La càmera i l'acceleròmetre també són més barats.

Fixem-nos que en els costos hardware posem l'ús de 3 Raspberry Pi Zero en comptes de dues. Això és perquè durant gran part del desenvolupament vam tenir dissenyat l'algorisme de visió per computador per utilitzant 2 Raspberry Pi. Això també passa per a les càmeres.

Chapter 12

Conclusions

12.1 Assoliment dels objetius inicials

Tenint en compte els objectius marcats per el projecte en l'apartat 3.1 del document i un cop desenvolupat el sistema podem dir:

- L'objectiu 1 tractava de l'aprenentatge i implementació dels conceptes relatius a la visió per computadors. Aquests s'ha explicat en l'apartat del Marc teòric i s'han utilitzat per a dissenyar els algorismes de detecció inicial i de seguiment.
- L'objectiu 2 era treballar amb MATLAB i les corresponents llibreries. Clarament s'ha assolit aquest objectiu aprenent MATLAB, primerament, i tenint com a resultat el 50% del codi font desenvolupat en MATLAB.
- Els objectius 3,4 i 5 parlaven del desenvolupament de software en els dispositius encastats a més a més de la connexió amb els diferents perifèrics. 2 dels 3 nodes que conformen el nostre sistema, el Sistema de la Pilota i el Sistema de la Càmera, són dispositius encastats, Raspberry en comptes de les Arduino especificades en la planificació inicial, i el seu software s'ha desenvolupat amb llibreries pròpies d'aquests dispositius a més a més del seu testeig. A més a més hem pogut implementar mòduls per a la connexió i comunicació amb aquests perifèrics com són la càmera RaspiCam, l'acceleròmetre MPU6050, el Botó de Seguiment...
- Els objectius 6,7 i 8 feien referència a les propietats d'un STR i la seva implementació. No hem hagut d'implementar cap política de planificació de tasques al STR com hem vist i justificat anteriorment. Per altra banda sí que hem aconseguit un sistema que funció en temps real i dintre d'uns deadlines clars. Tot això ho tenim exposat a l'apartat de Testing.
- La resta d'objectius són orientats al desenvolupament del producte real. No els podem aplicar degut a que nosaltres hem acabat desenvolupant una PoC.

A més a més també ens havíem marcat uns objectius pràctics que estan associats al producte final que hem desenvolupat.

Podem dir que hem aconseguit desenvolupar un sistem autònom que és capaç de detectar situacions de fora de joc i comunicar-ho al exterior.

Hem sigut capaços de desenvolupar cadascun dels nodes autònoms on donats els inputs necessari són capaços de treure la sortida esperada. A més a més hem aconseguit integrar-los gràcies als mòduls de comunicació epr la xarxa i de intercomunicació. Hem implementat un sistema que es capaç de detectar situacions de fora de joc en temps real aprofitant les precondicions avans de l'inici del

12.2 Assoliment de les competències tècniques

A més a més dels objectius marcats en l'apartat 3.1 també vam establir uns objectius referents a les competències tècniques especifiques d'un Treball de Fi de Grau de l'especialitat d'enginyeria de computadors. Seguidament enumerarem aquestes competències i justificarem si des del punt de vista del desenvolupador aquestes competències s'han assolit (Remarcant que aquesta és la justificació del desenvolupador. El director i els vocals hauran de ser els qui acabin determinant l'assoliment o no d'aquestes competències):

CEC1.1: Dissenyar un sistema basat en microprocessador/microcontrolador.

Aquesta competència és anàloga als objectius 3,4 i 5 de l'apartat 3.1 d'Objectius. Com ja hem explicat anteriorment tant el Sistema Pilota com el Sistema de la Càmera són dispositius encastats amb microprocessadors i hem desenvolupat el seu corresponent software.

CEC2.4: Dissenyar i implementar software de sistema i de comunicacions.

Els 3 nodes del nostre sistema són capaços de comunicar-se per la xarxa i enviar o rebre dades en funció de les nostres necessitats. Això està implementat en els Mòduls Listener i Speaker que hem explicat anteriorment i els resultats de la seva correcta implementació estan explicats en l'apartat Testing.

CEC3.1: Analitzar, avaluar i seleccionar les plataformes hardware i software més adients per al suport d'aplicacions encastades i de temps real.

Per una banda hem argumentat el perquè no era necessari aplicar un planificador de tasques en els Main (OS) de cadascun dels nodes. A més a més l'implementació dels mètodes Req i Recv en els mòduls Listener i Speaker així com l'aplicació en els Main de cada node ens ha permès assolir els deadlines marcats. Així com també l'establiment i justificació dels deadlines establerts i el seu compliment.

CEC3.2: Desenvolupar processadors específics i sistemes encastats; desenvolupar i optimitzar el software d'aquests sistemes.

Aquí podríem parlar dels mòduls MPU6050 i del mòdul Càmera els quals estan associats al Node de la Càmera i al Node de la Pilota i que per tant han estat orientats, dissenyats i programats com a sistemes de propòsit específic. Per altra banda decisions com la baixada de dimensions de les imatges a transmetre la justificació de l'existència de l'algorisme de seguiment en contraposició al de detecció inicial, la memorització de càlculs com la del color RGB de la gespa i així només hem de calcular-lo un cop... Són decisions orientades al rendiment del sistema. Com a treballs futurs però l'apartat del rendiment té un gran pes.

12.3 Treballs futurs

Finalment i un cop acabat el desenvolupament del projecte es donen diversos treballs futurs que es podrien aplicar a aquest projecte, i sobretot a projectes futurs basats en aquest projecte, que millorarien l'eficiència i l'abast del projecte actual. Bàsicament són:

- Augmentar el rendiment i el temps de resposta del sistema modificant el software i convertir-lo en software multithreading. Això passa per aplicar llibreries orientades la programació en multithreading com Pthreads, OpenMp per al software desenvolupat en C++ i la Parallel Toolbox per aplicar-ho també en els algorismes de visió per computadors desenvolupats en MATLAB.

- Desenvolupar un sistema orientat a l'entorn real basant-se en el desenvolupament d'aquest projecte.

Aquest treball futur de desenvolupar el sistema per a un entorn real es pot dividir en diversos objectius o tasques futures:

- El disseny d'un nou sistema de BallTrigger basat en els principis explicats, dissenyats i implementats en el sistema de la Pilota i en l'algorisme de detecció inicial. Amb això aconseguiríem arribar al màxim de casos possibles orientats a l'entorn real a més a més de millorar la precisió de la funcionalitat.

- Dissenyar la integració de més càmeres en el Node de la Càmera i adaptar el software: tant de detecció inicial i seguiment com la comunicació del Node amb les diferents Càmeres. Amb això augmentaríem la precisió eliminant l'error que origina la perspectiva de la càmera en la detecció de les posicions dels jugadors la qual és una de les fonts de imprecisió més grans de tot el sistema.

- Implementar un mecanisme de recuperació del sistema en cas de caiguda abrupte per tal de poder tornar a ser funcional sense la necessitat de les precondicions inicials del partit. Seguir utilitzant els algorismes basats en aquestes precondicions inicials en l'inici del sistema, ja que pel simple fet d'executar-se el nou software tindrà un temps més costós que l'actual, però que en cas de caiguda del sistema es pogués aplicar.

Bibliography

[1] KPMG El fútbol profesional en España genera 7.600M de euros
https://home.kpmg.com/es/es/home/sala-de-prensa/notas-de-prensa
/2015/05/futbol-profesional-espana-genera-7600m-euros.html

[2] Belda Maruhenda, F. (2004) Can the human eye detect an offside position during a football match?
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC535985/

[3] Normativa del Treball de Final de Grau de la Facultat Informàtica de Barcelona (2017)
https://www.fib.upc.edu/sites/fib/files/documents/
estudis/normativa-tfg-gei-final.pdf

[4] Història de la FIFA
https:/es.fifa.com/about-fifa/who-we-are/ the-laws/index.html

[5] FIFA Laws of the game 2018/2019 [en línea]:
https://resources.fifa.com/image/upload/ laws-of-the-game-2018-19.pdf?
cloudid=qafar0qbviwls7vqabkl

[6] Reglament oficial de la Federació Catalana de Futbol 2018 (2018)
http://files.fcf.cat/documentos/2018/ Reglament 20CTA 2001 07 18.pdf

[7] Normativa de joc de la Fédération Internationale de Football Association , FIFA (2004)
https://es.fifa.com/mm/document/afdeveloping/ refereeing/law 11 offside
es 47386.pdf

[8] Belda Maruhenda, F (2004). Visual sequence in a game of offside in football.
https://www.bmj.com/rapid-response/2011/ 11/01/visual-sequence-game-offside-football

[9] Bart Gilis, Werner Helsen, Peter Catteeuw, and Johan Wagemans (2008) Offside Decisions by Expert Assistant Referees in Association Football. Perception and Recall of Spatial Positions in Complex Dynamic Event s. Disponible en:
https://core.ac.uk/download/pdf/34405617.pdf

[10] Luis, V.; Canelo, A; Morenas, J.; Gómez-Valadés, J.M. Gómez, J.S. (2012). Referees' visual behaviour during offside situations in football.
https://revistas.uam.es/rimcafd/article/ viewFile/rimcafd2015.58.008/1455

[11] Reglament Laws of the game 2016 de la FIFA (2016)
https://img.fifa.com/image/upload/datdz0pms85gbnqy4j3k.pdf

[12] Implementació del VAR en el Mundial de futbol de Rússia 2018 (2018).
https://football-technology.fifa.com/es /innovations/var-at-the-world-cup/

[13] Implementació del VAR en el Mundial de futbol de Rússia 2018 (2018).
https://football-technology.fifa.com/es /innovations/var-at-the-world-cup/

[14] Félix Díaz, J (2017) ¿Cuánto cuesta el VAR y quién lo paga?, Marca.
http://www.marca.com/futbol/primera-division/2017/
11/15/5a0cb0f3268e3e9b2a8b461b.html

[15] Karthik Muthuraman,P Joshi,Suraj Kiran R (2018) Vision Based Dynamic Offside Line Marker for Soccer Games.
https://arxiv.org/pdf/1804.06438.pdf

[16] HAYS (2018) Guia del mercado laboral 2018.
https://www.esic.edu/empleabilidad/pdf/recursos /guia-hays-2018-sectores-y-salarios.

[17] InfoJobs · ESADE (2018) ESTADO DEL MERCADO LABORAL EN ESPAÑA Informe.
https://nosotros.infojobs.net/wp-content /uploads/2018/05/Informe-Anual-InfoJobs-ESA

[18] Microsoft > Productos [En línea]
https://products.office.com

[19] MathWorks Products and Prices (2017).
https://es.mathworks.com/support/pricing/2017/euro-es.html

[20] Arduino store [En línea].
https://store.arduino.cc/

[21] Forsyth, David A. (2002) Computer Vision: A Modern Approach Richard Hartley, Andrew Zisserman (2000) Multiple View Geometry in Computer Vision, Second Edition Yoshinori Ohno, Jun Miura, and Yoshiaki Shirai Dept. of Computer-Controlled Mechanical Systems, Osaka University 2-1, Yamadaoka, Suita, Osaka (2014) " Where are the ball and players? : Soccer Game Analysis with Color-based Tracking and Image Mosaick"
https://www.researchgate.net/publication/ 2427922 Where are the ball and players Soccer Game Analysis with Color-based Tracking and Image Mosaick

[22] Manuel Stein, Halldor Janetzko, Andreas Lamprecht, Thorsten Breitkreutz, Philipp Zimmermann, Bastian Goldlucke, Tobias Schreck, ¨ Member, IEEE, Gennady Andrienko, Michael Grossniklaus, Member, IEEE, and Daniel A. Keim, Member, IEEE (2017) Bring it to the Pitch: Combining Video and Movement Data to Enhance Team Sport Analysis
https://core.ac.uk/download/pdf/130998571.pdf

[23] Svein Arne Pettersen , Dag Johansen , Havard Johansen , Vegard Berg-Johansen Vamsidhar Reddy Gaddam , Asgeir Mortensen , Ragnar Langseth , Carsten Griwodz , Hakon Kvale Stensland , Pal Halvorsen (2013) Soccer Video and Player Position ataset University of Konstanz(2018) Football through the eyes of a computer
https://phys.org/news/2018-06-football-eyes.html

[24] FIFA Football Stadiums, Lighting and power supply
https://resources.fifa.com/image/upload /guide-the-artificial-lighting-for-football-

[25] Murat Duru¸s (2014) Ball Tracking and Action Recognition of Soccer Players in TV Broadcast Videos
http://mediatum.ub.tum.de/doc/1145077/870316.pdf

[26] MATLAB API for C++ (2018)
https://es.mathworks.com/help/matlab/Cpp-api.html

[27] Where are the ball and players?: Soccer Game Analysis with Color-based Tracking and Image Mosaick Sunghoon Choi, Yongduek Seo, Hyunwoo Kim, Ki-Sang Hong, Dept. of EE, Pohang University of Science and Technolog; Dong, Pohang, 790-784, Republic of Korea
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.3849rep=rep1type=pdf

[28] Apunts Geometria Afí. Facultat de Màtimàtiques Universitat Politècnica de Catalunya.
https://drive.google.com/file/d/1bMxe-qi0364PGKDuxginERQ-4ysaYgaQ/view

[29] Apunts Geometria Afí. Facultat de Màtimàtiques Universitat Politècnica de Catalunya.
https://drive.google.com/file/d/1bMxe-qi0364PGKDuxginERQ-4ysaYgaQ/view

[30] Introduction to Sockets Programming in C using TCP/IP. Professor: Panagiota Fatourou
https://www.csd.uoc.gr/ hy556/material/tutorials/cs556-3rd-tutorial.pdf