UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TEMA:

Diseño e implementación de robot soccer autónomo con captura de movimientos a través del sistema embebido Raspberry Pi

AUTOR:
Veliz Plua, Jean Carlos

Trabajo de Titulación previo a la obtención del título de
INGENIERO EN TELECOMUNICACIONES

TUTOR:
M. Sc. Palacios Meléndez, Edwin Fernando

Guayaquil, Ecuador

13 de Marzo del 2019

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

CERTIFICACIÓN

Certificamos que el presente trabajo fue realizado en su totalidad por el Sr. **Veliz Plua, Jean Carlos** como requerimiento para la obtención del título de **INGENIERO EN TELECOMUNICACIONES**.

TUTOR

_____

M. Sc. Palacios Meléndez, Edwin Fernando

DIRECTOR DE CARRERA

_____

M. Sc. Heras Sánchez, Miguel Armando

Guayaquil, a los 13 días del mes de Marzo del año 2019

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Veliz Plua, Jean Carlos**

DECLARÓ QUE:

El trabajo de titulación **Diseño e implementación de robot soccer autónomo con captura de movimientos a través del sistema embebido Raspberry Pi"** previo a la obtención del Título de **Ingeniero en Telecomunicaciones**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, a los 13 del mes de Marzo del año 2019

EL AUTOR

_____

VELIZ PLUA, JEAN CARLOS

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **Veliz Plua, Jean Carlos**

Autorizó a la Universidad Católica de Santiago de Guayaquil, la publicación, en la biblioteca de la institución del Trabajo de Titulación: **"Diseño e implementación de robot soccer autónomo con captura de movimientos a través del sistema embebido Raspberry Pi"**, cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

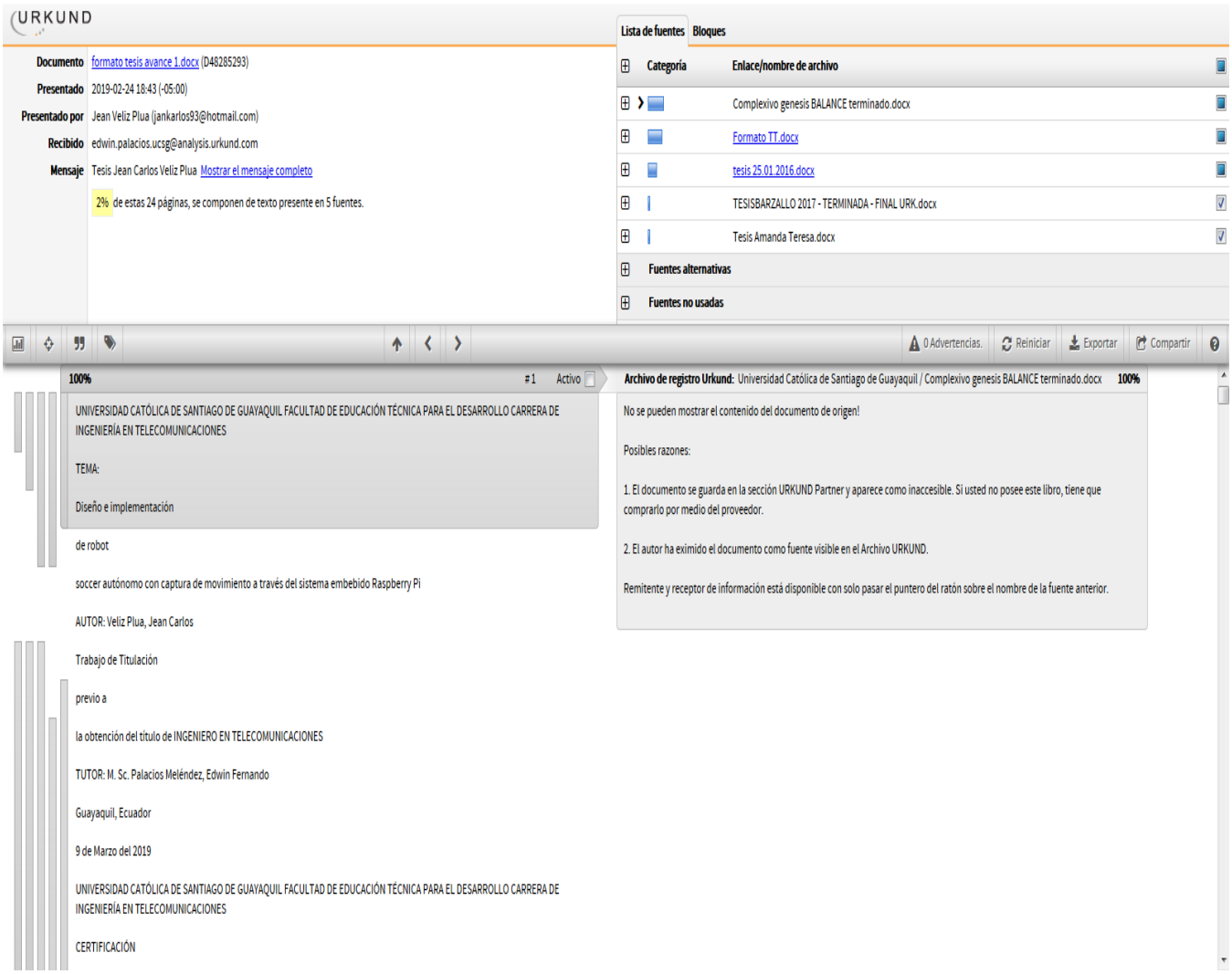
Guayaquil, a los 13 del mes de Marzo del año 2019

EL AUTOR

_____

VELIZ PLUA, JEAN CARLOS

REPORTE DE URKUND



DEDICATORIA

A Dios por ser nuestro padre y por guiarnos en la vida, a mi mami Carolina Plua García la mujer que más amo la que me ayudado a lo largo de mi vida, mi papi Wilson Veliz Olvera un guerrero en todo sentido y la fuente de inspiración para seguir luchando cada día.

Mi hermano Wilson Gabriel Veliz cual fue mi gran ejemplo y mi apoyo emocional desde muy pequeño, mi hermana Guiselle Veliz Plua la mujer que siempre está para mí a pesar de haberme equivocado, mis tías, primas y primos, que con su ayuda y demostración de afecto han contribuido a formar una excelente persona y cada día llenan de felicidad mi corazón.

A mis sobrinos por ser la luz que cambio mi vida, Carlos Eliud mi angelito mi guerrero que nunca se dará por vencido, Dahra, Julian y Carlos Stephano gracias por estar conmigo siempre.

A mis amigos del colegio, a mis amigos de la universidad que sin ellos mi idea de estudiar tuviese otro sentido, Personas como Kevin Robles, Gerson Castillo, Xavier Garzón que estuvieron en los momentos más difíciles a lo largo de mi vida estudiantil.

EL AUTOR

_____

VELIZ PLUA, JEAN CARLOS

AGRADECIMIENTO

A Dios por su gran amor, a cada uno de los miembros de mi familia que me han demostrado que no existe obstáculo alguno que no se pueda vencer, a mis padres que han dado todo por nosotros a lo largos de nuestras carreras, mis hermanos que han estado ahí para aconsejarme siempre.

Mis amigos los cuales han sido un apoyo dentro de mis estudios universitarios al igual que los de bachillerato que de alguna u otra forma colaboraron para la culminación de este trabajo.

Agradezco a cada uno de los docentes que estuvieron a lo largo de este viaje instruyéndonos con sus lecciones de vida y conocimientos, en especial al Ingeniero Fernando Palacios Meléndez por sus consejos, apoyo y enseñanzas dentro del ámbito estudiantil.

EL AUTOR

_____

VELIZ PLUA, JEAN CARLOS

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

f. _____

M. Sc. ROMERO PAZ, MANUEL DE JESÚS

DECANO

f. _____

M. Sc. ZAMORA CEDEÑO, NÉSTOR ARMANDO

COORDINADOR DE AREA

f. _____

M. Sc. CÓRDOVA RIVADENIERA, LUIS SILVIO

OPONENTE

Índice General

Índice de Figuras

Capítulo 1:

No se encuentran elementos de tabla de ilustraciones.

Capítulo 2

Capítulo 3

Índice de Tablas

Capítulo 2

Capítulo 3

Resumen

El presente trabajo de titulación consiste en el diseño e implementación de un soccer autónomo con captura de movimiento a través del sistema embebido Raspberry Pi; cuya razón por la cual este trabajo de titulación fue creado es motivar el desarrollo de nuevos diseños en establecimientos educativos. Dentro del trabajo de titulación se utilizó el método descriptivo e investigativo los cuales permitieron cumplir con éxito los objetivos que se plantearon. Este proyecto a nivel teórico describe el uso y características de cada uno de los componentes utilizados a lo largo de la implementación del prototipo, también se diseñó el código en ambas plataformas como lo son Arduino y Python para el correcto funcionamiento del prototipo. Se diseñó una placa de distribución. Con el diseño de este prototipo permitirá a la institución ingresar a competencias nacionales e internacionales de soccer con características autónomas que se realizan cada año en diferentes partes del mundo.

Palabras claves: CONTROLADOR, OPENCV, DETECCION, MOVIMIENTO, CSV, ARDUINO, RASPBERRY, CAPTURA, IMÁGENES.

Abstract

The present titling work consists of the design and implementation of an autonomous soccer with movement capture through the Raspberry Pi embedded system; the reason why this degree work was created to motivate the development of new designs in educational establishments. Within the titling work, the descriptive and investigative method was used, which allowed to successfully meet the objectives that were set. This project at a theoretical level describes the use and characteristics of each of the components used throughout the implementation of the prototype; this code was also designed on both platforms such as Arduino and Python for the correct operation of the prototype. A plate of distribution was designed. With the design of this prototype, it will allow the institution to enter national and international soccer competitions with autonomous characteristics that take place every year in different parts of the world.

Keywords: CONTROLLER, OPENCV, DETECTION, MOVEMENT, CSV, ARDUINO, RASPBERRY, CAPTURE, PICTURES.

Capítulo 1: Descripción General

1.1. Introducción.

El gran aumento de los estudios tecnológicos realizados en los últimos 15 años ha contribuido al desarrollo de nuevas actividades e implementaciones para diversos tipos de controladores, microcontroladores o pequeños computadores, etc. El uso de los diferentes controladores o SBC aumentan con gran rapidez gracias a su compatibilidad sobre otras plataformas. Debido a las diferentes aplicaciones que poseen están directamente involucrados con la robótica, se sabe que los robots han sido diseñados para cumplir con labores difíciles o complicadas que un humano muy probable no puede realizar.

En el siguiente documento, que se divide en los componentes que conforman el cerebro que controlara el prototipo, comunicación que se utiliza, captura de movimiento y diseño de la placa. El soccer autónomo es un robot capaz de detectar la pelota, ir hacia ella y capturarla para transportarla directo el área. Cuando llegue al área el mecanismo de disparo se activara realizando la anotación de gol, todo este proceso lo efectuara el sistema embebido Raspberry Pi con colaboración de la cámara Logitech y un Arduino UNO.

Las actividades básicas que ejecutara el robot serán las siguientes:

- La cámara se encargara de receptar las imágenes presentadas y transformarlas en datos para almacenarlos en la Raspberry Pi.
- La Raspberry Pi envía la información al Arduino para la activación de los motores.
- El robot se dirige y recoge la pelota para llevarla a la portería contraria.
- Se activa el disparador del robot cuando este se situé a una distancia cercana a la portería.

Por lo general esta clase de robots son utilizados en competencias internacionales, estos robots autónomos físicamente cuentan con: Raspberry PI (SBC), cámara, chasis, Arduino Uno, actuadores (disparador, motores y ruedas).

1.2. Antecedentes.

El año 1997 será recordado como el día en el que la inteligencia artificial y la robótica cambiaron la historia, ese mismo año el robot de la IBM “Deep Blue” derrotaba al campeón mundial de ajedrez una hazaña que le tomo alrededor de cuarenta años a un gran número de científicos a cargo del desarrollo del proyecto que culminaría de una manera exitosa. La misión MARS Pathfinder del año 1997 que fue enviado por parte de la NASA, esta nave no tripulada realizo su aterrizaje con éxito sobre la superficie de Marte. Debido a estos acontecimientos nació la idea de crear un equipo de futbol capaces de vencer al equipo de futbol humanos campeón del mundo.

Durante el mes de Octubre del año 1992 en Tokio se realizó los estudios del impacto de la tecnología sobre los deportes, estudios de viabilidad e impacto social fueron los más destacados incluyendo las reglas de los prototipos y simulaciones para un evento a futuro. El cual se llamaría Robot J-League al no ser bien acogido el nombre se decidió cambiarlo por “RoboCup”.

No fue hasta el año 1997 que se llevó a cabo el primer campeonato de RoboCup en el cual existieron más de 40 equipos inscritos variando entre físicos y simulaciones. Desde aquel año se realiza de manera anual en diferentes sedes del mundo el campeonato.

1.3. Definición del Problema.

La necesidad de crear el primer prototipo autónomo a nivel competitivo para representar al país en competencias a nivel nacional e internacional y motivar la implementación de diferentes prototipos autónomos por parte de unidades educativas como colegios, universidades elevando el prestigio del país en el área de la robótica.

1.4. Justificación del Problema.

Identificando todos los elementos presentes en el problema y procediendo de manera efectiva los pasos requeridos para la elaboración del proyecto el cual consta de diseñar e Implementar un robot soccer autónomo con captura de movimientos a través del sistema Raspberry Pi, se procederá

con el proyecto con el fin de acabar con la necesidad del problema previamente mencionado.

Por lo cual el prototipo de soccer será de gran ayuda a la institución para la participación de eventos de este tipo, además de sus diferentes usos debido a su capacidad de modificar los códigos fuentes y reconocimiento de imágenes que posee. Este proyecto beneficiara a los estudiantes, profesores y facultad para futuras interacciones con diseños de robots autónomos.

1.5. Objetivos del Problema de Investigación.

1.5.1. Objetivo General.

Realizar la implementación, diseño y programación de un robot soccer autónomo con captura de movimientos a través del sistema embebido Raspberry Pi para poder competir a nivel internacional.

1.5.2. Objetivos Específicos.

- Describir los componentes a utilizarse para la implementación del prototipo.
- Diseñar la placa en proteus para la distribución eléctrica de los componentes electrónicos del prototipo.
- Desarrollar el código para la programación del soccer autónomo utilizando Python dentro de la Raspberry Pi.
- Construir el robot prototipo de soccer autónomo.

1.6. Hipótesis.

El aumento de la tecnología e implementación en el Ecuador va en aumento, debido al basto conocimiento de los ecuatorianos y las ganas de superarse día a día a pesar de que el país no cuente con los materiales necesarios. Pero actualmente dentro del país no existen diseños de robots soccer autónomos.

1.7. Metodología de Investigación.

El presente trabajo de titulación dispone del siguiente tipo de investigación aplicada la cual está basada su diseño en el tipo exploratorio

que permite investigar por primera vez todo lo relacionado con el diseño de un prototipo autónomo. Otro método a utilizar en el trabajo de titulación es el método descriptivo, debido al análisis realizado a las piezas que conforman el soccer autónomo previo a su diseño e implementación.

Se utilizarán diferentes métodos a lo largo de la implementación para el cumplimiento del postulado fundamental: probar la hipótesis. También se realizarán pruebas de los componentes implicados en la implementación para su correcto funcionamiento.

Capítulo 2: Fundamentación Teórica

2.1. Origines de los robots y la robótica.

En el año 2000 a.C se consideraban a los robots como artilugios que fueran automatizados de manera mecánica, los primeros autómatas poseían mecanismo coordinado lo cual emulaba una sincronía perfecta.

En el año de 1920, Karel Capek un reconocido escritor publicaría en su novela RUR (Rossum's Universal Robots), Que la palabra “ROBOT” significa siervo, no sería hasta el año 19950 que el escritor Isaac Asimov publicaría su libro llamado “Yo Robot” en la cual detallaría tres leyes fundamentales de la robótica.

1. Los robots son incapaces de lastimar a los seres humanos o estar dispuesto a sufrir un daño eminente.
2. Deberán obedecer a los humanos excepto en caso romper la primera ley
3. Los robots tienen la obligación de protegerse al menos que exista una disputa entre leyes.

Gracias a este libro la palabra Robótica comenzó a popularizarse en Europa. Las primeras implementaciones de robots móviles llegaron de la mano del neurólogo Grey Walter en el año 1948 llamadas “Tortugas de Bristol”, estos prototipos contaban con sensores de luz, válvulas y sensores de contacto, su función principal era el seguimiento de luces para estudiar el comportamiento de estos robots. Algunas fechas que complementan la evolución de los robots y el nacimiento de la robótica son detalladas en el siguiente punto:

- 1954, George Devol patenta el primer robot programable debido a la necesidad de realizar labores repetitivas
- 1961, Dentro de las fábricas de General Motors se implementa el primer robot de Unimate, para el manejo de materiales de fundición.

- 1978, Unimate con ayuda de General Motors crea el robot PUMA (Programmable Universal Machine for Assembly)
- 2000, Honda lanza su robot Asimo (Advance Step in Innovation Mobility) con la visión de ayudar a personas que carecieran de movilidad completa y así motivar a los jóvenes a estudiar carreras de ciencias.

2.2. Raspberry Pi.

El Raspberry Pi es una placa reducida que cuenta con un integrado de bajo costo que fue desarrollado en Reino Unido, con la motivación de estimular el aprendizaje de ciencias computacionales en las universidades y escuelas. En la actualidad existen varios modelos de tarjetas Raspberry Pi, las cuales sus versiones conocidas mundialmente son las 2B y 3B, la placa posee contactos de entrada y salida de uso general (GPIO), mejora su conectividad gracias a los 4 puertos USB.

Hoy en día existen un listado de modelos de PCB embebidos para Raspberry Pi entre los conocidos están:

- Raspberry-Pi Modelo A
- Raspberry-Pi Modelo A+
- Raspberry-Pi Modelo B
- Raspberry-Pi Modelo B+
- Raspberry-Pi 2 Modelo B
- Raspberry-Pi 3 Modelo B
- Raspberry-Pi 3 Modelo B +

2.2.1. Elementos principales de una tarjeta Raspberry Pi.

GPIO en Raspberry Pi

Los pines de GPIO (ver figura 2.1) son interfaces que definen entradas y salidas dentro de la Raspberry Pi realizan la una función semejante a la de los interruptores que permiten encender y apagar en cualquier momento, están ubicados en el borde superior de la tarjeta y permite diferenciarlos por dos maneras

- Numeración GPIO
- Numeración Física

Figura 2. 1: Descripcion de los puertos GPIO
Fuente:(Gordon Chavarría & Molina Ormaza, 2016)

Procesador

Para que la tarjeta Raspberry Pi pueda cumplir con tan exigentes tareas está equipada con un procesador BCM2836 (ver figura 2.2) al cual lo acompañan cuatro núcleos y un procesador multinúcleo mejorando su rendimiento y eficiencia con las aplicaciones

Figura 2. 2: Procesador BCM2836
Fuente: (Soria Andrade, 2017)

Puerto HDMI

El puerto de interfaz multimedia de alta definición permite conectar dispositivos audiovisuales (ver figura 2.3) permitiendo la transmisión de audio

y video de forma digital son usualmente utilizados en televisores, computadoras y videojuegos.

Figura 2. 3: Puerto HDMI de la tarjeta Raspberry Pi
Fuente: (Gordon Chavarría & Molina Ormaza, 2016)

Conector 3.5mm

La Raspberry Pi cuenta con un conector Jack de 3.5mm (ver figura 2.4) la cual nos permite incorporar unos auriculares o salidas de audio como parlantes pero estos puertos poseen unas normas al momento de ser utilizados

Figura 2. 4: Salida Jack 3.5mm
Fuente:(Soria Andrade, 2017)

Socket 10/100 BaseT Ethernet

Es un socket que permite la conexión del cable rj45 10/100 con base T que utilizan 4 hilos (ver figura 2.5) permite conexiones rectas o cruzadas. Su primer versión fue la 802.3 la cual utiliza cableados para edificios.(Gordon Chavarría & Molina Ormaza, 2016)

Figura 2. 5: Puerto ethernet para conexión a internet
Fuente:(Gordon Chavarría & Molina Ormaza, 2016)

Micro USB

La fuente de alimentación de la tarjeta Raspberry Pi es dada por la entrada micro USB (ver figura 2.6) la cual permite el paso de 5 V con una corriente que oscila entre los 800 - 1500 mA. Esto depende de la cantidad de dispositivos conectados a la tarjeta esta aumenta su potencia.

Figura 2. 6: Puerto micro USB para alimentacion.
Fuente:(Gordon Chavarría & Molina Ormaza, 2016)

Puerto USB

El puerto USB es un estándar que se encuentra en todo los ordenadores del mundo el cual viene configurado con el famoso Plug and Play (ver figura 2.7) el cual acepta hasta 127 diferentes periféricos. Permite el paso de energía alimentando así a otros dispositivos.

Figura 2. 7: Puerto USB para conexión de perifericos
Fuente: (Gordon Chavarría & Molina Ormaza, 2016)

2.2.2. Tipos de Raspberry Pi.

- **Raspberry Pi modelo A**

En los inicios de la creación de las Raspberry Pi el modelo A (ver figura 2.1) solo contaba con un puerto USB, no contaba con los controladores de Ethernet y era mucho más barato que el modelo B. A pesar de no contar con un controlador de Ethernet se le podía adaptar un controlador Wi-Fi por USB.

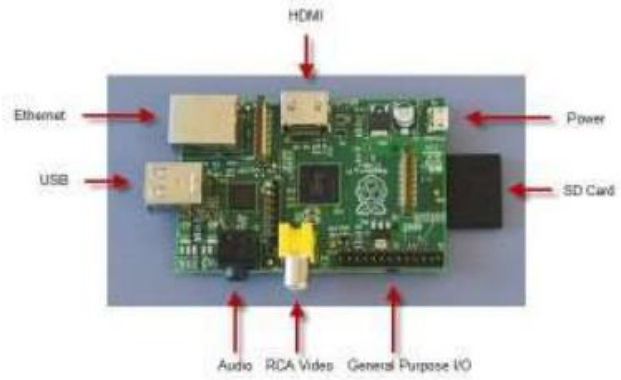


Figura 2. 8: Raspberry Pi Modelo A
Fuente: (Mollocana Alban, 2018)

- **Raspberry Pi modelo A+**

Unas leves mejoras con respecto a su antecesor el modelo A, incluye una ranura para tarjeta micro SD. “La Raspberry Pi A+ (ver figura 2.2) es el modelo más barato que ofrece hoy la fundación. A este precio se ofrece un SoC Broadcom BCM2835 con 256 MB de RAM. No es un dato menor, pues uno de los objetivos de la organización es hacer llegar hardware libre asequible a segmentos de la población que no se puedan permitir pagar los precios del mercado. Por eso el dispositivo ha sido útil para sectores como la educación” (Ayala Martillo, 2015).

Figura 2. 9: Raspberry Pi Modelo A+
Fuente: (Ayala Martillo, 2015)

“ Las mejoras implementadas en la nueva versión A+ pretender llamar la atención del público potencial de Raspberry Pi, tratando de dar un impulso a la línea A”(Ayala Martillo, 2015)

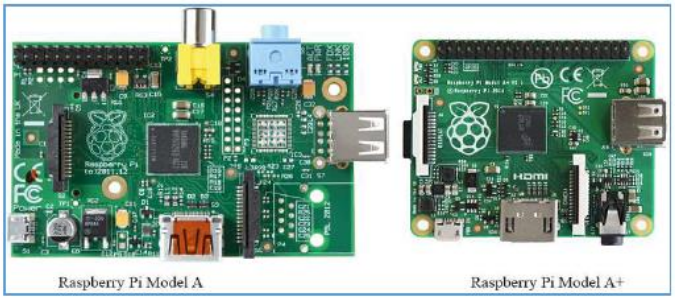


Figura 2. 10: Diferencia entre los modelos A & A+
Fuente: (Ayala Martillo, 2015)

- **Raspberry Pi modelo B**

La Raspberry-Pi Modelo B (ver figura 2.11) diseñada en una placa base de 85 x 54 un tamaño idéntico al de una tarjeta de crédito, posee un chip Broadcom BCM2835 con un 1 GHz de velocidad y 512 Mbyte de memoria RAM.

Este modelo en temas de conectividad, presenta un puerto Ethernet a diferencia de los modelos A y A+ que no disponen. A pesar de poseer los puertos para conectar directamente a un router además de hacer uso de adaptadores inalámbricos Wifi, cuenta con dos puertos USB para conectar un teclado y un mouse.

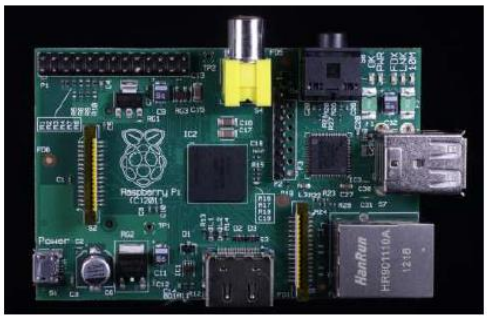


Figura 2. 11: Raspberry Pi Modelo B
Fuente: (Ayala Martillo, 2015)

“El Raspberry-Pi no viene con reloj en tiempo real, por lo que el sistema operativo debe usar a través de internet una configuración de hora en red, o pedir al usuario ingresar la hora en el momento de arrancar el sistema

operativo. Sin embargo se le puede añadir un reloj en tiempo real como lo es DS1307 con una batería mediante el uso de la interfaz I2C". (Ayala Martillo, 2015)

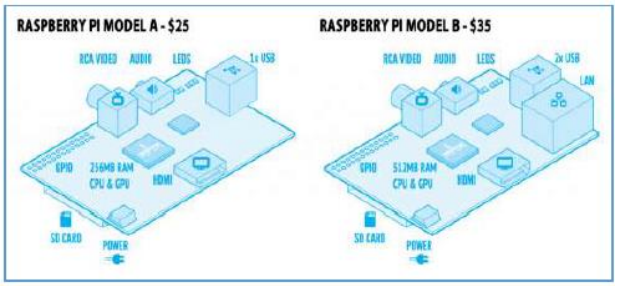


Figura 2. 12: Diferencia de precios entre los modelos A & B
Fuente: (Ayala Martillo, 2015)

- **Raspberry Pi modelo B +**

El modelo B + (ver figura 2.13) presento ciertas mejoras con respecto al año anterior, agregando cierto componentes extras. En la tabla 2.1 permite apreciar las mejoras que se presentó en el modelo B +.

Tabla 2. 1: Características de los Modelos Raspberry Pi B y B+

Raspberry Pi Modelo B +		
Característica	Modelo B	Modelo B +
Almacenamiento	SD	Micro SD
USB	2 Puertos	4 Puertos
Energía	750mA hasta 1.2A a 5V	600mA hasta 1.8A a 5V
Pines GPIO	26 Pines	40 Pines

Elaborado por: Autor

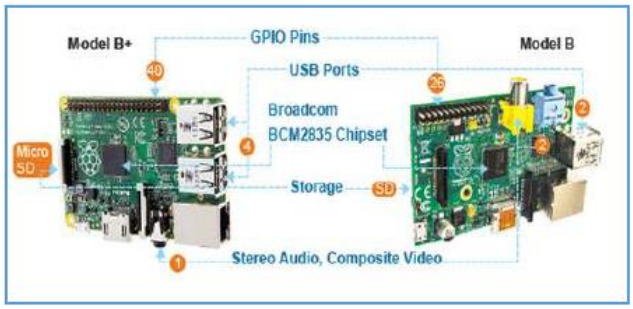


Figura 2. 13: Diferencia de características entre el modelo B+ & B
Fuente: (Ayala Martillo, 2015)

- **Raspberry Pi 2 Modelo B**

El modelo de Raspberry Pi 2 (ver figura 2.14) posee el mismo tamaño que el modelo B+ con su principal característica la cual incluye el aumento de

potencia, resalta una importante alianza con Microsoft para incluir las licencias de Windows 10 desde los modelos de Raspberry-Pi 2 Modelo B. Su procesador de cuatro núcleos a 900MHZ y de 1GB de RAM es seis veces mayor al del modelo B+ que cuenta con 512MB de RAM. Comparte muchas características y componentes entre las que resaltan son:

- 4 Puertos USB
- HDMI
- Ranura microSD
- Conector de audio

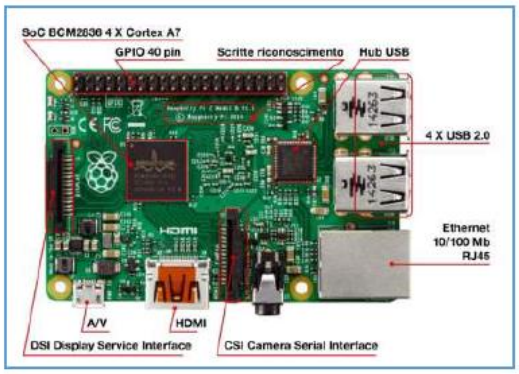


Figura 2. 14: Diseño de la Raspberry Pi 2 B
Fuente: Ayala Martillo, 2015)

- **Raspberry Pi 3 modelo B**

El modelo de Raspberry Pi 3 modelo B posee una distribución idéntica a la anterior versión de acuerdo al número de componentes, con 40 pines GPIO, salida HDMI a 108p, lector de tarjeta micro SD, puerto de red, conectividad Wi-Fi y Bluetooth. Para la implementación del proyecto se va utilizar el modelo de Raspberry Pi 3B (ver figura 2.15).

Figura 2. 15: del modelo Raspberry Pi 3 B
Fuente: (Miranda Sislema, 2018)

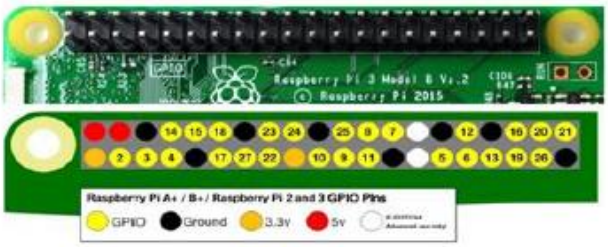


Figura 2. 16: Distribución de los pines en una Raspberry Pi 2 B
Fuente: (Simbaña Quilachamín, 2017)

- **Composición de una tarjeta Raspberry Pi tipo B**

La fuente de vida de la Raspberry Pi radica en su chip integrado Broadcom BCM2835 que almacena un procesador ARM11 con diferentes frecuencias del procesador VideoCore IV, adicional se encuentra una salida de audio y video a través de la entrada HDMI permitiendo la conexión a monitores o pantallas de televisión. Un System on Chip (SoC) es el integrado que llevan incorporado todo los componentes del sistema.

2.2.3. Sistemas operativos y sus principales orígenes.

El sistema operativo de la Raspberry es un software libre basado en Linux, el núcleo principal con el que opera es Raspbian, a pesar de contar con algunas alternativas más. Para una fácil instalación de Raspbian se aconseja descargar desde la página oficial de Raspberry el paquete de NOOBS (New of the Box Software), este paquete contiende todo lo esencial para dar los primeros pasos dentro de la Raspberry.

a) **Aros:** Research Operating System es un sistema operativo de código abierto y gratuito creado en el año 1997 desarrollado por APIs, compatible con los niveles binarios de procesadores x86 que existen en Linux
b) **Fedora:** Creado en el año 2003, desarrollado por un grupo comunitario RHEL posee un instalador llamado “Anaconda” que está escrito sobre Python 2 actualmente fue lanzada la versión Fedora 28.
c) **Gentoo Linux:** Lanzado en el año 2002 creada por Daniel Robbins llamándose Enoch Linux para posterior pasar a llamarse Gentoo Linux
d) **Debian:** Creado en el año 1993 por Ian Murdock, lanzo su primera versión en el año 1996 con el nombre Buzz, la última versión de Debian liberada llena por nombre Stretch en el año 2017.

e) **Linux:** Creado en el año 1969 bajo el nombre de Unix, cuando el proyecto creció y fue llamado Linux dado al acrónimo GNU (GNU is Not Unix).

En la siguiente figura 2.17 se puede ver el entorno de instalación que posee en NOOBS una vez encendida la Raspberry Pi.

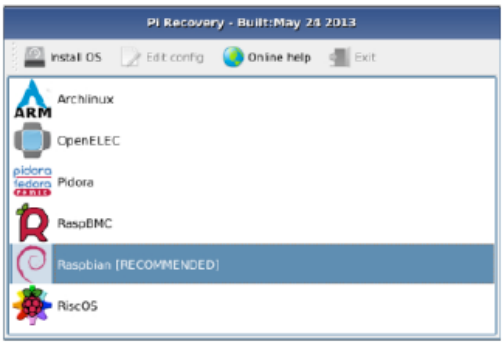


Figura 2. 17: Listado de Sistemas Operativos Disponibles
Fuente: (Gonzalez Garcia, 2015)

2.2.4. Lenguajes de programación utilizados en Raspberry Pi

A continuación se hablara sobre los principales lenguajes de programación utilizados para programar en Raspberry Pi, se analizara cada uno de ellos y se elegirá el que mejor prestaciones presente para el desarrollo del proyecto.

- **Lenguaje C**

En el año de 1990 su estándar fue cambiado al de ISO, el primer estándar conocido fue el ANSI como lo muestra la figura 2. 18 es un lenguaje orientado para sistemas operativos, siendo así el lenguaje de programación más popular para la creación de software libres. (Simbaña Quilachamín, 2017)

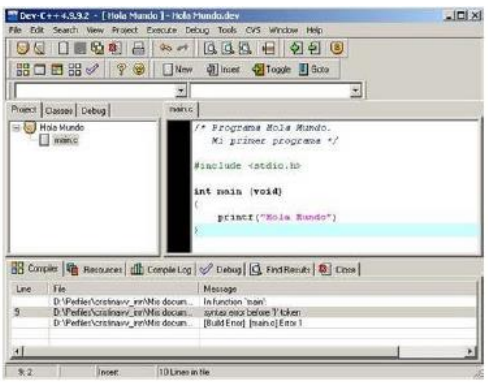


Figura 2. 18: Entorno del Lenguaje C
Fuente: (Simbaña Quilachamín, 2017)

- **Ruby**

Lanzado en el año 1995 es un lenguaje de programación reflexivo y orientado a objetos (ver figura 2.19). Combina una sintaxis inspirada en Python y Perl, se puede considerar que su implementación es distribuida bajo una licencia de software libre. (Simbaña Quilachamín, 2017).

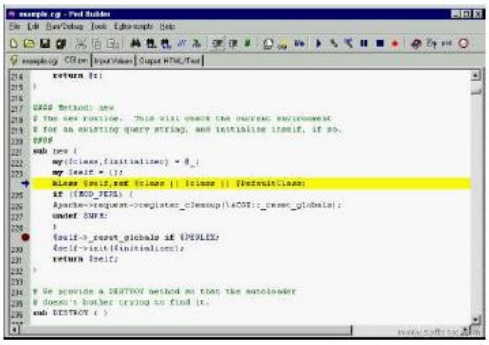


Figura 2. 19: Entorno del Lenguaje Ruby
Fuente: (Simbaña Quilachamín, 2017)

- **Python**

Python es un lenguaje de programación muy útil y fácil de aprender (ver figura 2.20). Cuenta con estructuras de datos de alto nivel que lo hacen más eficiente y enfoque simple orientado a objetos, al poseer una sintaxis dinámica de escritura lo hacen un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de plataformas. (Campuzano Bulgarín & Cedeño Vélez, 2015)

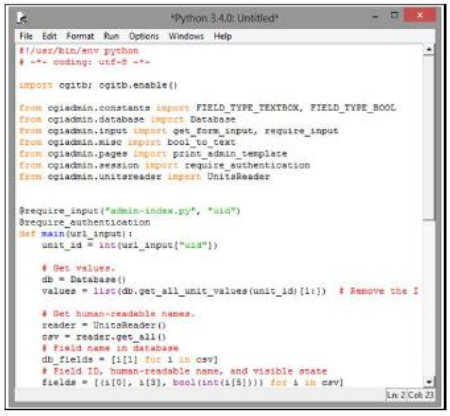


Figura 2. 20: Entorno del Lenguaje en Python
Fuente: (Campuzano Bulgarín & Cedeño Vélez, 2015)

- **Características de Python**

A continuación se identificara las principales características que posee el lenguaje de programación Python creado por Guido van Rossum.

a) Permite crear programas de propósito general y hasta desarrollar páginas web.
b) Python es open source lo que le permite tener compatibilidad con una gama de plataformas disponibles (Windows, Linux, Solaris, Mac).
c) Al ser open source es completamente gratis.
d) Contiene una gran lista de funciones, librerías y datos que ayudan a facilitar las tareas sin la necesidad de empezar de cero.
e) Ofrece la facilidad de insertar otros lenguajes como C/C++ para facilitar el scripting.

2.3. Arduino

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador (ver figura 2.21) y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos con multi propósito. (Martínez Escobar, 2015).

Su hardware está basado en un microcontrolador Atmel AVR, puertos de entrada y salida. Los microcontroladores más comunes y utilizados en esta placa son los Atmega168, Atmega328 y Atmega8 debido al poco costo y su facilidad de uso es utilizado en desarrollo de diversos proyectos de diseño, el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque que es ejecutado en la placa. Se programa en el ordenador para que la placa controle los componentes electrónicos. (Martínez Escobar, 2015).

Figura 2. 21: Partes del arduino Uno
Fuente: (Rodríguez Chicaiza, 2016)

Tabla 2. 2: Tipos de Arduino, Shield y Kits

Placa	Shield	Kits
Arduino Uno	Arduino GSM Shield	The Arduino Starter Kit
Arduino Leonardo	Arduino Ethernet Shield	Arduino Materia 101
Arduino Due	Arduino Wi-Fi Shield	TFT LCD Screen
Arduino Yún	Arduino Wireless SD Shield	USB/Serial Light Adapter
Arduino Tre (En Desarrollo)	Arduino USB Host Shield	Arduino ISP
Arduino Zero (EEUU)	Arduino Motor Shield	Mini USB/Serial Adapter
Arduino Micro	Arduino Wireless Proto	
Arduino Esplora	Shield	
	Arduino Proto Shield	

Elaborado por: Autor

2.3.1. Lenguaje de programación

El microcontrolador montado en la placa del Arduino es programable mediante el lenguaje de alto nivel Processing basado en java y similar a C++. Debido a esto es posible programar en otros lenguajes basados en java o populares en Arduino.

- Matlab
- C
- C++
- Ruby
- Python
- Adobe
- JavaFlash
- Linux TTY
- ProcessingPure
- Instant Reality
- Blitz Max
- Visual Basic
- PERL
- NETVBScript

Esto es posible debido a que Arduino se comunica mediante la transmisión de datos en formato serie, soportado por la mayoría de lenguajes anterior mente citada. Para los formatos que no son soportados es posible

utilizar software intermediarios que traduzca los mensajes enviadas por ambas partes para permitir una comunicación segura y fluida. (Martinez Escobar, 2015)

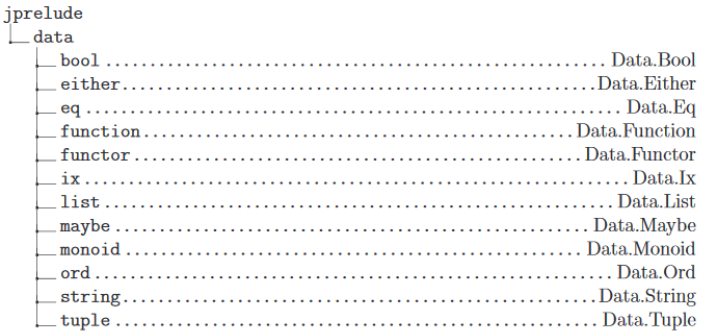


Figura 2. 22: Variables del lenguaje Java
Fuente: (Estrella Heredia, 2016)

2.4. Computer Vision System

El objetivo de la visión por computadora es permitir que los medios artificiales, como las computadoras tengan la capacidad de percibir el entorno externo, puedan comprenderlo, tomen las medidas o decisiones adecuadas, aprendan de esta experiencia para que puedan mejorar su desempeño futuro.

Un sistema de visión artificial es un reflejo del sistema de visión natural, es posible lograr, rastrear ciertos objetos que pueden estar en el camino de un individuo a través de la visión y el aprendizaje

Una subtarea importante en este sistema de visión por computadora es la segmentación de imágenes, que es el proceso de dividir la imagen en un conjunto de regiones, segmentos o números. Esto permite dividir una imagen en regiones significativas, agrupándolas antes de buscar una característica común, como el color, el borde y la forma. La segmentación del color utiliza el color para clasificar las áreas en la imagen que separa los objetos que no tienen el mismo color característico.

Es común que un sistema de visión por computadora tenga como objetivo realizar la reconstrucción del entorno externo (ver figura 2.23), específicamente, los objetos en los que el primer objetivo es lograr la ubicación del objeto con certeza y confiabilidad. El color del objeto es una

característica utilizada para separar diferentes áreas y posteriormente habilitar el uso de un módulo de seguimiento, entonces esta característica fue elegida para ser estudiada e implementada.

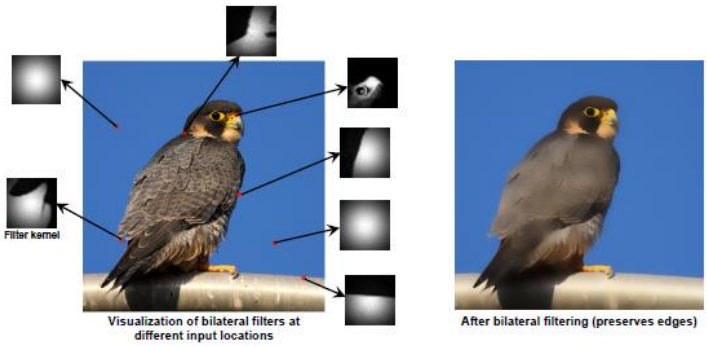


Figura 2. 23: Captura procesada por CVS
Fuente: (Jampani, 2017)

2.4.1. Captura de imágenes

El sistema elegido para la recopilación de información es una cámara web que es detectada de manera automáticamente después de enchufar el puerto USB en la Raspberry Pi. Use el comando lsusb y se mostrará la detección de periféricos. Para ver el software de transmisión de cámara web, que es una forma de transmitir datos multimedia a través de paquetes almacenados temporalmente en la memoria caché, se requiere Raspberry Pi. El software de transmisión ha sido instalado: mjpeg-streamer en Raspberry Pi a través de comandos en la figura 2.24 se aprecia una captura de imagen de forma negativa.

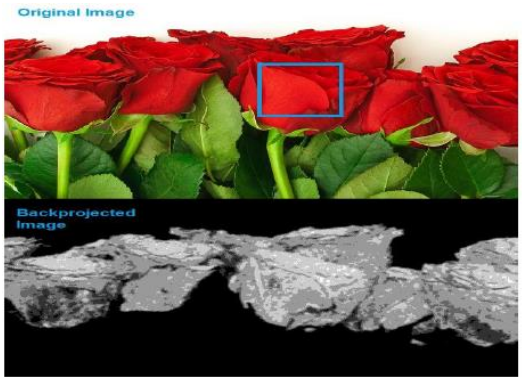


Figura 2. 24: Captura de imagen en escala de grises
Fuente: (Taha Mohamed, 2014)

2.4.2. Detección de movimiento

El puerto de entrada / salida de propósito general de GPIO permitió la fácil instalación de componentes de detección por infrarrojos, como un

fototransistor y un diodo IR (ver figura 2.25), necesarios para diseñar la unidad de detección. La idea depende del método de salto de línea.

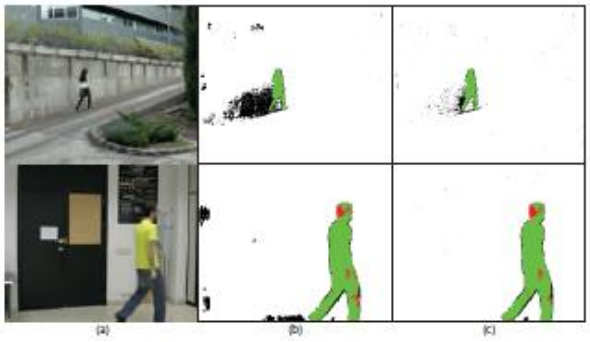


Figura 2. 25: Captura por detencion de imagen
Fuente: (Berjón Díez, 2016)

2.4.3. Por qué se usa OpenCV en Raspberry PI

Al tratar de encontrar una manera de hacer un software de visión artificial que funcione en la Raspberry Pi (ver figura 2.26), se consideró LabView. Sin embargo, aunque LabView tiene una interfaz de programación gráfica fácil.

Hay un problema que hace que sea virtualmente imposible trabajar con otros productos, una razón para ello es que LabView requiere un motor de tiempo de ejecución, lo que significa que no se puede ejecutar como un software independiente en la PC, y mucho menos en un dispositivo integrado como el Raspberry Pi.

En segundo lugar, al programar dispositivos incorporados con MATLAB, el código debe 'traducirse' a un script en C, que luego realiza una edición adicional para que se ejecute en dichas plataformas. Por todos estos motivos, al elegir una biblioteca que permite escribir "código nativo" que se ejecuta exactamente como se predijo, la selección se hizo a favor del OpenCV. El lenguaje de programación implementado en OpenCV está basado en C y C++, Las dos ventajas de usar Python son:

- Facilidad de instalación
- Estructura del módulo de Python

Los códigos escritos en Python son muchos más fáciles de leer, depurar y la variedad de módulos disponibles hace que su programación sea más sencilla y fluida.

Figura 2. 26: Captura de imágenes a traves de OpenCV
Fuente: (Taha Mohamed, 2014)

2.4.4. Proceso de captura de movimiento mediante cámara

Los siguientes pasos demostraran el conjunto de funciones que realizara el sistema. Estas funciones se ejecutan secuencialmente, repetidamente y para valores reales de las características dinámicas del objeto (coordenadas y tamaños), hasta seis veces (o seis variaciones) por segunda vez. Es decir, cada segundo se generan hasta seis valores que se procesarán y compararán, lo que agilizará el proceso (ver figura 2.27).

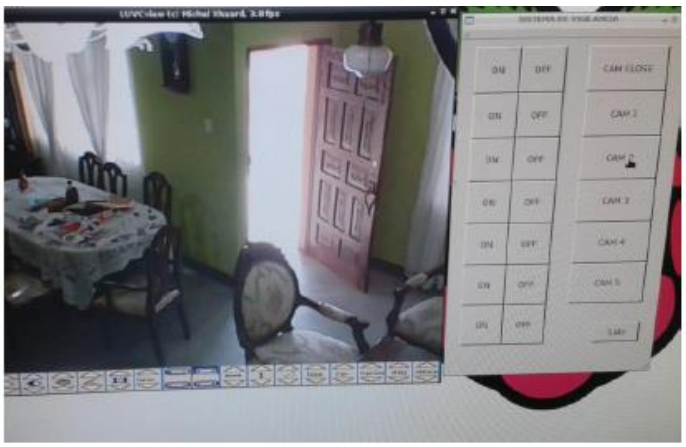


Figura 2. 27: Captura de movimiento desde pantalla principal
Fuente: (Campuzano Bulgarín & Cedeño Vélez, 2015)

1) Primero fue necesario capturar (o recibir) la imagen o, específicamente, el marco que contiene la imagen (marco). El tamaño es de 160x120 píxeles. El marco en general (p. Ej., 640 píxeles de ancho y 480 píxeles de alto) causó ralentizaciones en el proceso de reconocimiento cuando

la imagen se transmitió remotamente. El sistema predeterminado es el color RGB, este sistema de color se representa en el marco de la webcam obtenido a través de los colores básicos: Red (rojo), Green (verde) y Blue (azul). Estos colores se representan en un vector dimensional píxel a píxel, por ejemplo el color rojo se representa con los valores de 0 com (0, 255, 0), respectivamente, representados para cada canal. Es decir cada píxel tiene su valor RGB representado por tres bytes.

2) Después de recibir los datos de la imagen, se llevó a cabo la conversión del sistema de color RGB al color HSV (tono, saturación y valor), ya que este modelo se describe de manera similar al reconocimiento por los colores de los ojos humanos. Dado que el sistema RGB (rojo, verde y azul). Tiene colores basados en combinaciones de los colores primarios (rojo, verde y azul) y el sistema HSV define los colores como tono, saturación y valor, facilitando la extracción de información. Qué convierte la imagen de entrada de un sistema de color.

3) Con la imagen transferida al modelo HSV, fue necesario encontrar los valores correctos de HSV mínimo y color máximo del objeto que se seguirá. Para guardar estos valores, se hicieron dos vectores con un valor mínimo de HSV y HSV en color como valores mínimos: Hue (42) Saturación mínima (62) Brillo mínimo (63) Matiz máximo (92) Saturación máxima (255) Brillo máximo (235). Entonces, el siguiente paso para generar una imagen binaria, la información relevante puede estar limitada solo en el contexto de estos valores. Estos valores son necesarios para limitar el patrón de color del objeto. Se usó una función de comparación de los valores de píxel con los valores estándar del vector insertado. El resultado fue una imagen binaria que proporcionaba solo un valor para cada píxel.

4) Una vez realizada la segmentación, que da como resultado la imagen binaria, se observa que todavía hay ruido en el cuadro. Estos ruidos son elementos que dificultan la segmentación (incluida la obtención del

tamaño real) del objeto. Para corregir (o intentar solucionar) este problema, fue necesario aplicar una transformación morfológica a través de operadores en el marco, de modo que se eliminaron los píxeles que no cumplían con el estándar deseado. Para esto, se utilizó el operador morfológico EROS, que realizó una "limpieza" en el marco, reduciendo el ruido contenido en él.

5) Luego se utilizó para la función "Moments", que calcula los momentos de contorno positivo (blanco) utilizando una integración de todos los píxeles presentes en el contorno. Esta característica solo es posible en un marco ya binarizado y sin ruido, de modo que el tamaño del contorno del objeto no se ve afectado por píxeles perdidos en el marco, lo que obstaculiza y causa redundancia en la información.

Moments = cv2.moments (imgErode, True)

En el caso expuesto, fue necesario encontrar el área del contorno y sus coordenadas de ubicación del marco para realizar los cálculos de reposicionamiento del chasis. El cálculo del área del objeto realiza la suma binaria de positivo, generando la variable M00 y registrada en la variable área

Área = Moments ['m00']

La especificidad del contorno se refiere a un objeto, no a un polígono. Este valor se encuentra en un área aproximada de píxeles positivos (blanco) que componen el objeto. Si este valor es un área nula, se descarta la existencia de un color de objeto tratado (si es el color "green") en el marco. El uso de esta función ayudará a lograr el movimiento del robot acercándose y alejándose del objeto objetivo, tratando de tratar el problema de la profundidad. Es decir, el hecho de que el objeto se acerca o se distancia excesivamente del chasis.

2.5. Driver controlador de motores

Los drivers controladores para motores son módulos especiales que cuentan con integrados (ver figura 2.27) capaz de controlar motores de corriente continua y motores de paso a paso, cuentan también con

reguladores de voltaje y salidas analógicas o digitales dependiendo del modelo del driver estos son algunos modelos de drivers en el mercador.

- Tarjeta controladora de motores TB6560
- Tarjeta controladora de motores EasyDriver V44
- Tarjeta controladora de motores L298D
- Tarjeta controladora de motores L293

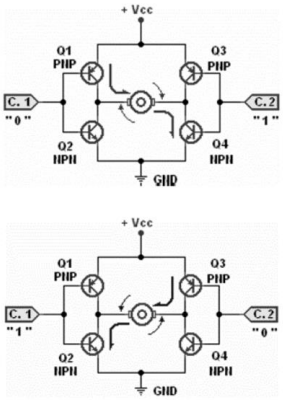


Figura 2. 28: Diagrama de un puente H
Fuente: (Correa Eras & Remache Ortega, 2006)

2.6. Definición de motores DC

Los motores eléctricos poseen la gran capacidad de transformar energía eléctrica en mecánica permitiendo desplazar objetos a su vez si estos motores trabajan en sentido contrario son capaces de crear energía eléctrica es decir funcionan como un generador (ver figura 2.29).

Por lo general todos los motores en DC poseen las siguientes partes para su funcionamiento;

- Parte exterior (cubierta)
- Estator
- Rotor

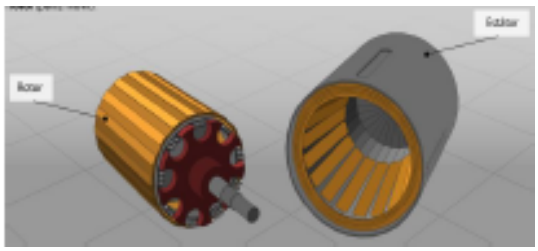


Figura 2. 29: Elementos de un motor
Fuente: (Gualli Cujilema & Véliz Intriago, 2016)

2.6.1. Tipos de motores

Motores con escobillas (ver figura 2.30) son del tipo de motores más comunes que existen en el mercado son utilizados en todo tipo de área, estos cuentan con un anillo de cobre segmentado por la cual la corriente circula a través de una bobina alternando los giros del motor.

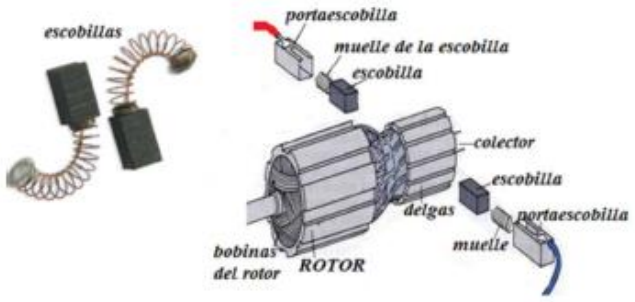


Figura 2. 30: Elementos de un motor de escobilla
Fuente: (Reyes Dominguez, 2017)

Motores sin escobilla (ver figura 2.31) estos motores poseen una gran confiabilidad y eficiencia reduciendo así el ruido producido por la rotación. Gracias a estas características permiten el ahorro de energía

Figura 2. 31: Elementos de un motor
Fuente: (Correa Mora, 2018)

2.6.2. Funcionamiento de los motores DC

Los principales fundamentos del funcionamiento de los motores de corriente DC o alterna está basada en un conductor que está dentro de un campo magnético induce una tensión que hará circular corriente eléctrica por el conductor (ver figura 2.32) el efecto que se genera expondrá al conductor con una fuera perpendicular a ellos siguen la regla de la mano derecha. (Gualli Cujilema & Véliz Intriago, 2016).

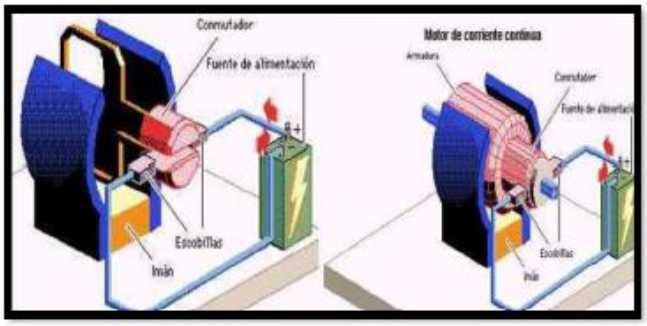


Figura 2. 32: Funcionamiento de un motor DC
Fuente: (Iperty Barros & Cruz Hermenejildo, 2018)

2.7. Descripción de las baterías

Las baterías son dispositivos que son capaz de almacenar energía de una forma electroquímica, existen dos tipos básicos de baterías:

Baterías primarias: su principal característica es que la reacción electroquímica es irreversible, esto indica que una vez agotada su carga no podrá volver a cargar.

Baterías secundarias: es todo lo contrario a una batería primaria debido a que su reacción es reversible permitiendo volver a cargar la batería aunque su eficiencia disminuye con cada ciclo de carga. El principio de funcionamiento de las baterías se basa en la celda electroquímica, estas celdas contienen dos electrodos que son: el ánodo y cátodo

2.7.1. Tipos de baterías

Existen tres características esenciales que definen el gran rendimiento de una batería los cuales son:

1. Cantidad de energía que permite almacenar: El valor Wh se calcula al multiplicar el valor del voltaje nominal por Ah.
2. Máxima corriente de entrega: está definido como un numero fraccionario
3. Profundidad de descarga: este valor es representado de forma porcentual

A continuación se detallara alguno de los tipos de baterías más comunes:

- Plomo-acido(Pb-acido)

Batería de tipo recargable más común debido a la relación costo-desempeño a pesar de ser una de menor densidad de energía

- Nickel-cadmio(NiCd)

Entre su larga lista de ventajas las más destacadas son, tener la mitad del peso, tolerar altas temperaturas y tener las celdas selladas debido a esto posee una tasa baja de auto descarga

- Nickel-hidruro metálico(NiMh)

Es una batería mejorada a partir de la NiCd la cual ofrece una mayor densidad de energía posee una alta tasa de auto descarga. Son usualmente utilizadas en vehículos híbridos

- Ion-Litio(Li-ion)

Es una batería revolucionaria la cual ofrece una densidad de energía tres veces mayor, cada celda posee un voltaje de 3.5 y gracias a su baja tasa de auto descarga la hacen una batería de gran rendimiento

- Polímero-Litio(Li-poly)

La batería de polímero posee celdas de entre 3.7 V debido a esto no se debe permitir una descarga por debajo de los 3 V ya que este evento causara que se deterioren dichas celdas

- Aire-zinc

Baterías capaces de superar con crecer a las populares Ion de litio ya que su fabricación es barata, estas baterías funcionan con oxígeno almacenado dentro de las celdas

- Celdas de combustible

Puede producir aproximadamente 1,2 V son de acción electroquímica con la gran diferencia de poder cargarlas.

2.8. Tipos de comunicación

La necesidad de comunicar las dos placas para la transferencia de datos y lograr que se comuniquen, por ejemplo (ver figura 2.33) compartir los datos de la captura de movimiento. Para poder comunicar nuestra Raspberry Pi con el Arduino, es necesario tener instalado el pySerial dentro de esta librería nos permitirá escribir y leer el puerto de programación serial que posee el lenguaje Python.

Conectando el Arduino con la Raspberry mediante un USB macho a macho se procede a digitar el comando en la terminal de la Pi:

```
ls /dev/tty
```

Figura 2. 33: Comunicación de la Raspberry con el Arduino
Elaborado por: Autor

El resultado mostrado en pantalla será el siguiente: “/ dev / ttyACM0”, esto indica que nuestra Raspberry Pi se está comunicando perfectamente con el Arduino.

2.8.1. Ejemplo de comunicación entre Arduino y Raspberry Pi

Para comprobar si en realidad la comunicación serial está funcionando (ver figura 2.34) correctamente se ingresara el siguiente programa dentro del Arduino:

```
void setup(){
Serial.begin(9600);
}
void loop(){
Serial.println("Hello Pi");
delay(2000);
}
```

Figura 2. 34: Muestra en pantalla de Saludo entre dispositivos
Elaborado por: Autor

Dentro de la Raspberry Pi mediante el editor de Python se ingresara el siguiente comando:

```
import serial
ser = serial.Serial('/dev/ttyACM0', 9600)
```

Figura 2. 35: Importación de los datos del puerto serial
Elaborado por: Autor

Con la siguiente línea de comando se podrá agregar un escucha al puerto serie de la Raspberry Pi.

```
while 1 : ser.readline()
```

Figura 2. 36: Línea de comando para obtener el mensaje del puerto serial
Elaborado por: Autor

Se mostrara en pantalla del Terminal la palabra “HI”.

2.8.2. Ejemplo de comunicación entre Raspberry y Arduino

Para el siguiente ejemplo (ver figura 2.37) se escribirá un código para que nuestra Raspberry Pi envié un número el cual hará que el Arduino se encienda y apague el pin 12 las veces que indique el número ingresado.

```
[sourcecode language="cpp"]

const int ledPin = 12;

void setup(){
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
}

void loop(){
if (Serial.available()) {
light(Serial.read() - '0');
}
delay(500);
}

void light(int n){
for (int i = 0; i < n; i++) {
digitalWrite(ledPin, HIGH);
delay(100);
digitalWrite(ledPin, LOW);
delay(100);
}
}
[/sourcecode]
```

Figura 2. 37: Configuración del puerto serial para encender un LED
Elaborado por: Autor

Dentro de la Raspberry el comando a escribir será el siguiente

```
ser.write('3')
```

Figura 2. 38: Número de pulsos para encendido de un LED
Elaborado por: Autor

CAPÍTULO 3: DISEÑO Y RESULTADOS OBTENIDOS

3.1. Descripción de los componentes

3.1.1. Raspberry Pi 3 modelo B

La Tarjeta Raspberry que se utilizara para la elaboración del prototipo de robot Soccer será el modelo B (ver figura 3.1.) debido a que se realizaron varios estudios y pruebas para poder incluir el CVS en computadoras a pequeña escala que pudiera usar un chasis robótico, en ese momento se eligió la pequeña computadora Raspberry PI 3 modelo B debido a su costo aceptable, tamaño pequeño y valiosas especificaciones con las que cuenta además de ser capaz de embarcar periféricos como puertos USB y también la capacidad de integrar características como actuadores y sensores en un conjunto llamado GPIO de enlaces externos, que incluye salidas de pines/entradas digitales, UART, I2C, SPI , Audio.

Figura 3. 1: Raspberry Pi 3 Modelo B utilizada en el prototipo
Elaborado por: Autor

3.1.2. Arduino UNO

La tarjeta Arduino UNO es una placa electrónica basada en el Atmega328P. Cuenta con 14 pines digitales de entrada/salida (de los cuales 6 se pueden usar como salidas PWM), también cuenta con 6 entradas analógicas, cristal de cuarzo a 16 MHz, una conexión USB, un conector de alimentación y un botón de reinicio como lo muestra la figura 3.2. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o a la corriente con un adaptador de corriente alterna a corriente continua o batería.(Ninabanda Pilco, 2016)

Figura 3. 2: Arduino UNO R3
Fuente: (Ninabanda Pilco, 2016)

A continuación en la tabla 3. 1 se describe las especificaciones técnicas principales del Arduino UNO

Tabla 3. 1 : Características del Microcontrolador ATmega

Microcontrolador	**ATmega328p**
Tensión de Funcionamiento	5V
Voltaje de Entrada(Recomendado)	7-12V
Voltaje de Entrada(Limite)	6-20V
E/S Digitales	14 (6 son de salida)
PWM digital pines I / O	6
Pines de entrada digital	6
Memoria flash	32 KB
Velocidad del reloj	16 MHz
Peso	25 g

Elaborado por: Autor

3.1.3. Cámara web Logitech

La marca de cámara que se usara para la captura de movimiento es una Logitech HD (ver figura 3.4) la cual permite diferenciar el objeto que se desea seguir con gran precisión, además de ser una de las cámaras con mayor compatibilidad con el programa CVS y Raspberry Pi 3.

Figura 3. 3: Cámara Web Logitech HD
Elaborado por: Autor

3.1.4. Fuente de alimentación

Uno de los puntos más importante a la hora de mover el prototipo es el tipo de fuente energética que llevara el robot para esto será necesario el uso de baterías de LiPo (ver figura 3.4) estas ofrecen una mejor densidad de energía y potencia que dará una vida útil más larga, para esto es importante escoger bien el voltaje de la batería que se necesita dependiendo del consumo del sistema, una de las ventajas de estas baterías es su fácil montaje, conexión y carga de la misma.

A continuación en la tabla 3.1 se mostrara algunos tipos de baterías de LiPo con sus respectivos valores.

Tabla 3. 2: Tipos de batería LiPo según su voltaje y corriente.

Número de celdas	Voltaje	Capacidad de carga	Corriente
2 S	7.4 v	35 C	300 mAh
2 S	7.4 v	20 C	500 mAh
2 S	7.4 v	20 C	1000 mAh
2 S	7.4 v	25 C	1500 mAh
2 S	7.4 v	20 C	2200 mAh
2 S	7.4 v	25 C - 30 C	4000 mAh
2 S	9. 9 v	1 c	1500 mAh
3 S	11.1 v	25 C	1000 mAh
3 S	11.1 v	10 C	1000 mAh para Bioloid
3 S	11.1 v	65 C	1400 mAh
3 S	11.1 v	20 C	1600 mAh
3 S	11.1 v	20 C	2200 mAh
3 S	11.1 v	10 C	5200 mAh
4 S	14.8 v	65 C	2200 mAh-Grafeno
4 S	14.8 v	10 C	4000 mAh

Elaborado por: Autor

Figura 3. 4: Batería marca Turnigy de 11.1 V
Elaborado por: Autor

3.1.5. Puente H L298N

El controlador que se usara en el prototipo será el puente H L298N (ver figura 3.5), el cual se puede alimentar de distintas maneras debido a que cuenta con un integrado LM7805.

Permitiendo el paso del voltaje de entre 6 V a 12 V, en el otro caso solo permite el paso de 5 V además de que el consumo d corriente sea mayor a 550 mA. Existen tres tipos de movimientos que permiten interactuar con el entorno, utilizando el software y hardware de manera más completa. Los cambios definirán la ubicación de los objetos y el uso de la profundidad:

- Adelante
- Atrás
- Derecha
- Izquierda

Figura 3. 5: Controlador de Motores L298N
Elaborado por: Autor

3.1.6. Mecanismo de disparo

Para el mecanismo de disparo del robot soccer se utilizara un solenoide (ver figura 3.6) de 12 V a 500 mA con una fuerza de acción de 55 N la cual

nos dará un disparo con mucha potencia a la hora de anotar en el arco contrario.

Figura 3. 6: Solenoide de 12 V para mecanismo de pateo
Elaborado por: Autor

3.1.7. Modulo regulador de voltaje

Debido a que no todo los componentes trabajan a 12 V como en el caso del solenoide, se deberá emplear un regulador para disminuir el voltaje a 5 V que son requeridos por componentes como: Arduino, Raspberry Pi 3. Este módulo permite una entrada de 3 V a 38 V (ver figura 3. 7) para transformarlo a un rango de 1.5 V a 36 V además de poseer un voltímetro digital incluido para mostrar el voltaje de salida que se desea obtener.

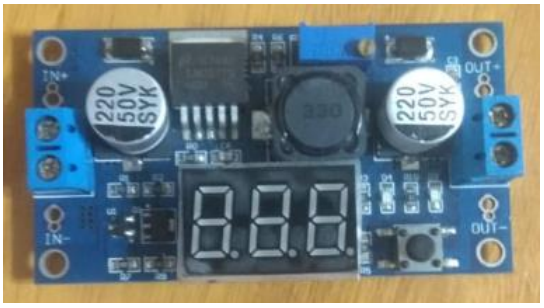


Figura 3. 7: Modulo Regulador de Voltaje DC
Elaborado por: Autor

3.2. Tarjeta de distribución del prototipo

La construcción de esta plata de distribución fue diseñada con el fin de alojar el cerebro del robot, debido a que el Arduino aloja a todo los componentes necesarios del prototipo. Dentro del diseño mostrado en la figura 3.8 se puede observar que cada uno de los pines del Arduino posee una bornera para su conexión la cual se detallara más adelante en el documento.

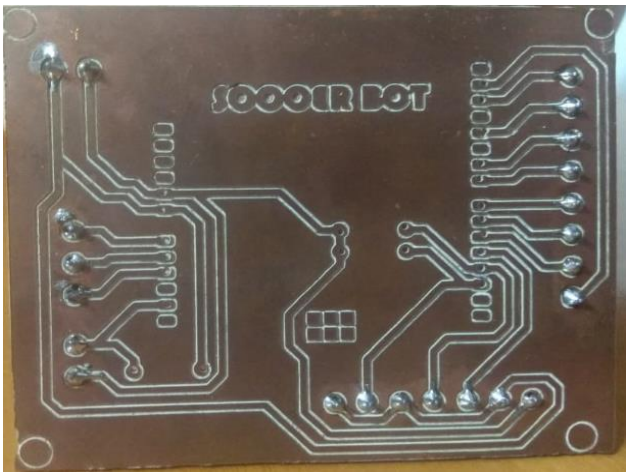


Figura 3. 8: Diseño de la placa principal de distribución
Elaborado por: Autor

3.3. Elaboración de la programación del prototipo

A continuación se definirá los parámetros de programación que tendrá el prototipo de soccer autónomo. Los puntos que se explicaran de forma detallada son los siguientes

- Líneas de Comando
- Definición de los parámetros

3.3.1. Programación de la Raspberry Pi

Es importante declarar o llamar a los módulos que se instalan previamente dentro de nuestra Raspberry pi como lo son OpenCV (cv2), time o numpy los cuales cuentan con funciones requeridas para el correcto funcionamiento del robot soccer. (Ver figura 3.9)

```
import cv2.cv as cv
import cv2 as cv2
import time
import numpy as np
import os
from time import sleep
import RPi.GPIO as GPIO
from threading import Thread
```

Figura 3. 9: Importación de los módulos dentro del código
Elaborado por: Autor

En la figura 3.10 se definirán los parámetros con los cuales la cámara se inicializara mediante el comando “def__init__ (self):”. Una vez encendida comenzara la búsqueda del objeto definido mediante el código “def start”. La definición de los colores a buscar mediante la cámara serán definidos

mediante la línea de comando “def update (self)”. Dentro de estos parámetros se obtendrán los datos más relevantes.

```
class CameraStream:

    def __init__(self):
        print("Initialising Camera")
        ret, unfilteredImage = capture.read()
        imgHSV = cv2.cvtColor(unfilteredImage,cv2.cv.CV_BGR2HSV)
        imgThreshold = cv2.inRange(imgHSV, rangeMinBall, rangeMaxBall)
        imgErosion = cv2.erode(imgThreshold, None, iterations = 3)
        self._moments = cv2.moments(imgErosion, True)
        self._xPos = 0

                def start(self):
                    print("Starting Camera Thread")
                    t = Thread(target=self.update,args=())
                    t.daemon = True
                    t.start()
                    return self

    def update(self):
        while True:
            ret, unfilteredImage = capture.read()
            imgHSV = cv2.cvtColor(unfilteredImage,cv2.cv.CV_BGR2HSV)
            # Looking for the ball - assumed green
            if lookingForGoal == False:
                imgThreshold = cv2.inRange(imgHSV, rangeMinBall, rangeMaxBall)
            # Looking for the goal - assumed blue
            elif lookingForGoal == True:
                imgThreshold = cv2.inRange(imgHSV, rangeMinGoal, rangeMaxGoal)
            imgErosion = cv2.erode(imgThreshold, None, iterations = 3)
            self._moments = cv2.moments(imgErosion, True)
            if self._moments['m00'] > 0:
                self._xPos = self._moments['m10'] / self._moments['m00']
```

Figura 3. 10: Definición de parámetros de inicialización de la cámara
Elaborado por: Autor

Dentro de la asignación de los puertos de salidas GPIO que se ingresaran en la figura 3.11, destacan las salidas para los movimientos de los motores (izquierda y derecha) los cuales serán los pines 9 y 10 respectivamente. Así como la salida del mecanismo de pateo definido en el pin 27 siendo este una salida., mientras que la captura mediante el servo motor definidos en el pin 22 será definida como entrada.

```
forwardMotorLeftActionPin = 9
forwardMotorRightActionPin = 10

kickBallPin = 27

ballIsTrappedPin = 22

lookingForBall = False
GPIO.setmode(GPIO.BCM)
GPIO.setup(forwardMotorLeftActionPin,GPIO.OUT)
GPIO.setup(forwardMotorRightActionPin,GPIO.OUT)

GPIO.setup(kickBallPin, GPIO.OUT)
GPIO.setup(ballIsTrappedPin, GPIO.IN)
```

Figura 3. 11: Asignación de los puertos GPIO de la Raspberry PI
Elaborado por: Autor

Los valores con los cuales se elegirá los colores representativos de los objetos que serán representados dentro de la Raspberry como lo son la pelota que los parámetros han sido asignados mediante el color verde (véase en la figura 3.12). La portería será definida por el color azul al igual que la pelota cuenta con sus propios parámetros.

```
# The HSV range used to detect the coloured object
# Green ball
HminBall = 42
HmaxBall = 92
SminBall = 62
SmaxBall = 255
VminBall = 63
VmaxBall = 235

"""

# Blue goal
HminGoal = 95
HmaxGoal = 113
SminGoal = 65
SmaxGoal = 255
VminGoal = 205
VmaxGoal = 255


# Image capture window parameters
width = 1000 #Default value is 500
height = 500 #Default value
```

Figura 3. 12: Configuración de los parámetros de color de los objetos
Elaborado por: Autor

Los parámetros de captura de imagen se definen en la figura 3.13 donde el ancho y la altura de la cámara así como el centro y mostrar en pantalla la última dirección que fue visto la pelota.

```
# Image capture window parameters
width = 1000 #Default value is 500
height = 500 #Default value

centrePoint = 317.5 #Based on a width of 1000 at max x of 635 -> observational value
height = 500 # Default 120
delay = (1/1000.0)
ballKicked = False
directionChoice = 0
directionSet = False
maxTime = 2
bufferCt = 0
bufferMax = 10
ballFoundToRight = False

lookingForGoal = False
```

Figura 3. 13: Valores de la obturación de cámara
Elaborado por: Autor

En la figura 3.14 se crea un Array en el cual contendrá los valores máximos y mínimos con el cual será identificada la pelota y la portería, adicional la distancia desde la portería hasta la distancia necesaria de pateo.

```
 # Creates a HSV array based on the min and maxium values (as an unsigned int) - For the Green Ball
rangeMinBall = np.array([HminBall, SminBall, VminBall], np.uint8)
rangeMaxBall = np.array([HmaxBall, SmaxBall, VmaxBall], np.uint8)

 # Creates a HSV array based on the min and maxium values (as an unsigned int) - For the Blue Goal
rangeMinGoal = np.array([HminGoal, SminGoal, VminGoal], np.uint8)
rangeMaxGoal = np.array([HmaxGoal, SmaxGoal, VmaxGoal], np.uint8)

# Minimum area to be detected
minArea = 25 # Default area = 50

# Maximum area to be detected - Determines distance from goal
maxArea = 5000
```

Figura 3. 14: Creación de HSV basado en los valores del color
Elaborado por: Autor

Mediante la variable capture = cv2.VideoCapture (ver figura 3. 15) se guardara las imágenes capturadas por la webcam la cual definirá los parámetros de búsqueda mediante la función def findball (directionChoice) la cual permitirá movernos en sentido del reloj o contra reloj para la búsqueda de la pelota.

A continuación mediante def KickBall (): una vez encontrada la portería se activara el solenoide para ejecutar el disparo.

```
capture = cv2.VideoCapture(0)

def RotateClockwise():
    GPIO.output(forwardMotorLeftActionPin, 1)
    GPIO.output(forwardMotorRightActionPin, 0)

def RotateAntiClockwise():
    GPIO.output(forwardMotorLeftActionPin, 0)
    GPIO.output(forwardMotorRightActionPin, 1)

def ForwardFull():
    GPIO.output(forwardMotorLeftActionPin, 1)
    GPIO.output(forwardMotorRightActionPin, 1)

def StopMotors():
    GPIO.output(forwardMotorLeftActionPin, 0)
    GPIO.output(forwardMotorRightActionPin, 0)

def findBall(directionChoice):
    if(directionChoice == 0): #Turn Left
        RotateClockwise()
        print "Looking for the ball to the Left"
    else:
        RotateAntiClockwise() #Turn Right
        print "Looking for the ball to the Right"

def KickBall():
    StopMotors()
    print("Found goal and taking a shot!")
    GPIO.output(kickBallPin, 1)
    sleep(1)
    print("Kicked ball. Did I hit it?")
    GPIO.output(kickBallPin, 0)
    lookingForGoal = False
    sleep(2)

print("Soccerbot initialised. Starting ball search now")

StopMotors()
cam = CameraStream().start()
time.sleep(1.0)
```

Figura 3. 15: Asignación de la dirección de los valores
Elaborado por: Autor

3.3.2. Programación del Arduino UNO

En la figura 3.16 se definirá las variables constantes que se utilizaran dentro del prototipo, se definirá las salidas digitales tales como el mecanismo de pateo, motores y las salidas de comunicación con la Raspberry Pi.

Además de definir los números de los pines utilizados como se puede apreciar en la siguiente imagen donde se nombra las principales funciones como los LDR y comunicación entre Arduino y Raspberry Pi.

```
#include <Servo.h>
int lightResSensitivity = 450; //LDR cut off value
Servo trapperArmServo;

int lightResValue; //Value for light sensor
int maxServoPos = 65;
int maxTrapperArmPos = 65;
int trapperArmServoPos = 0;

const int ldrPin = A4; //LDR analog connection
const int kickSolenoidPin = 5;
const int trapperArmServoPin = 4;
const int ballTrappedOutputPin = 12; // Ball is trapped so sned 3.3V to raspberry pi - via relay
bool ballIsTrapped = false;
const int forwardLeftActionPin = A0;
const int forwardRightActionPin = A1;
const int inputToKickPin = A2; // Input for raspberry pi - if high then kick
int forwardLeftActionPinVal, forwardRightActionPinVal, inputToKickPinVal;
const int ForwardMotorRight = 6;
const int ReverseMotorRight = 7;
const int ForwardMotorLeft = 3;
const int ReverseMotorLeft = 2;
int raspberryPiAnalogHighValue;
bool forwardMotorRightOn, reverseMotorRightOn, forwardMotorLeftOn, reverseMotorLeftOn;
```

Figura 3. 16: Asignación de tipo de variables en Arduino
Elaborado por: Autor

LightResValue contendrá los parámetros de funcionamiento para la activación de los servomotores que permitirán capturar la bola, si estoy valores están por debajo de los 400 no se activara dichos servomotores. (Ver figura 3.17.)

```
Serial.begin(9600);
lightResValue = analogRead(ldrPin); //Read ldr and output a value
SetLightResSens();

trapperArmServo.attach(trapperArmServoPin);
trapperArmServo.write(trapperArmServoPos);

SetMotorBools(false,false,false,false);

raspberryPiAnalogHighValue = 400;
```

Figura 3. 17: Asignación de los parámetros del servo motor
Elaborado por: Autor

La línea de comando pinMode definirá de que tipo será la variable: salida o entrada como se puede comprobar en la figura 3.18 donde se definirán el movimiento de los motores.

```
//Setup for the DC motor movement functionality
pinMode(MotorOnPinRight, OUTPUT);
pinMode(ForwardMotorRight, OUTPUT);
pinMode(ReverseMotorRight, OUTPUT);
pinMode(MotorOnPinLeft, OUTPUT);
pinMode(ForwardMotorLeft, OUTPUT);
pinMode(ReverseMotorLeft, OUTPUT);
pinMode(kickSolenoidPin, OUTPUT);

pinMode(forwardLeftActionPin, INPUT);
pinMode(forwardRightActionPin, INPUT);
```

Figura 3. 18: Definición de la funcionalidad de los motores DC
Elaborado por: Autor

Cuando el prototipo captura la bola esta información es receptada por el Arduino la cual envía un impulso de 3.3V a la Raspberry para confirmar que se encuentra en búsqueda de la portería.

```
    // Receive input from raspberry pi causing ball release and kick towards target
    pinMode(inputToKickPin, INPUT);

    // Ball is trapped and as such send a 3.3V high to the raspberry pi pin
    pinMode(ballTrappedOutputPin, OUTPUT);

    digitalWrite(MotorOnPinRight, LOW);
    digitalWrite(ForwardMotorRight, LOW);
    digitalWrite(ReverseMotorRight, LOW);
    digitalWrite(MotorOnPinLeft, LOW);
    digitalWrite(ForwardMotorLeft, LOW);
    digitalWrite(ReverseMotorLeft, LOW);
  }
```

Figura 3. 19: Configuración del mecanismo de captura y disparo
Elaborado por: Autor

Mediante void loop nos permitirá declarar la función principal que posee el Arduino como lo muestra la figura 3.20, como son las funciones de encendido de los motores, verificación si la pelota es atrapada buscar la portería en caso de no haber capturado la pelota se comenzara la búsqueda nuevamente hasta volver a encontrarla.

```
void loop()
{
  lightResValue = analogRead(ldrPin); //Read ldr and output a value

  forwardLeftActionPinVal = analogRead(forwardLeftActionPin);
  forwardRightActionPinVal = analogRead(forwardRightActionPin);

  if (lightResValue >= lightResSensitivity && !ballIsTrapped)
  {
    Serial.println("Ball is trapped");
    TrapBall();
  }

  inputToKickPinVal = analogRead(inputToKickPin);

  if (ballIsTrapped && inputToKickPinVal >= raspberryPiAnalogHighValue)
  {
    Serial.println("Releasing and kicking ball now: ");
    ReleaseAndKickBall();
  }

 if(forwardLeftActionPinVal >= raspberryPiAnalogHighValue && forwardRightActionPinVal >= raspberryPiAnalogHighValue)
  {
    ForwardMotorLeftMethod();
    ForwardMotorRightMethod();
    goto bailFromUpdate;
  }

  if(forwardLeftActionPinVal >= raspberryPiAnalogHighValue)
  {
    ForwardMotorLeftMethod();
    StopRightMotor();
  }

  if(forwardRightActionPinVal >= raspberryPiAnalogHighValue)
  {
    ForwardMotorRightMethod();
    StopLeftMotor();
  }

  if(forwardLeftActionPinVal <= raspberryPiAnalogHighValue && forwardRightActionPinVal <= raspberryPiAnalogHighValue)
  {
    StopLeftMotor();
    StopRightMotor();
  }

  bailFromUpdate:

delay(1);
}
```

Figura 3. 20: Contenido del programa principal del Arduino
Elaborado por: Autor

3.4. Construcción del prototipo soccer autónomo

3.4.1. Conexiones entre componentes

En la figura 3.21 se puede apreciar un diagrama con los pines de cada elemento conectado al Arduino UNO, así como la Raspberry y su comunicación con el Arduino cada pin ha sido señalado con la finalidad de facilitar su conexión de una manera intuitiva.

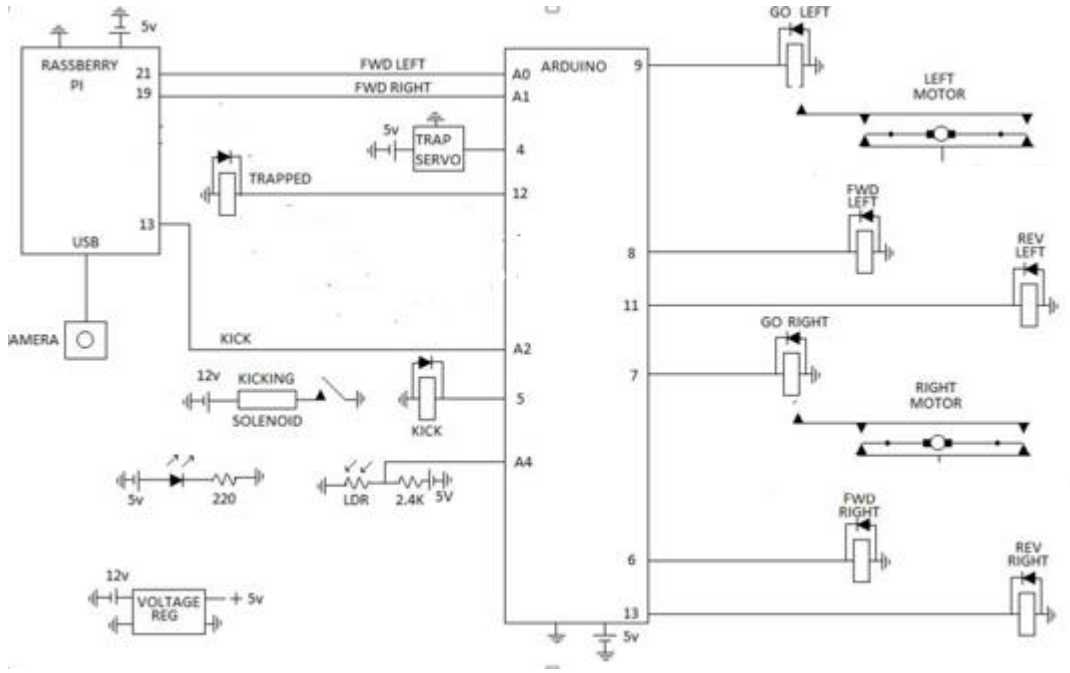


Figura 3. 21: Diagrama de las conexiones del prototipo
Elaborado por: Autor

3.4.2. Diagrama de comunicación del robot

En la figura 3. 22 se detalla a breves rasgos como será tratada la información para el diseño del robot soccer los cuales son ejecutados en el siguiente orden:

Entrada de Información: En este caso la información de entrada será generada por la cámara USB que está conectada a la Raspberry Pi, esta cámara son los ojos del robot, permitiéndole comprobar y analizar las imágenes capturadas en el siguiente proceso.

Proceso de Imágenes : En esta etapa la programación ingresada dentro de la Raspberry Pi previamente se encargara de reconocer la forma y color

del objeto, Al tiempo que enviara pulsos de información a nuestro Arduino UNO.

Movimiento del Robot: Una vez recopilada y procesada la información previamente por la cámara con el CVS que cuenta la Raspberry pi, este indicara al Arduino UNO que deberá poner a funcionar los motores indicándole la dirección en el cual se encuentra nuestro objetivo.

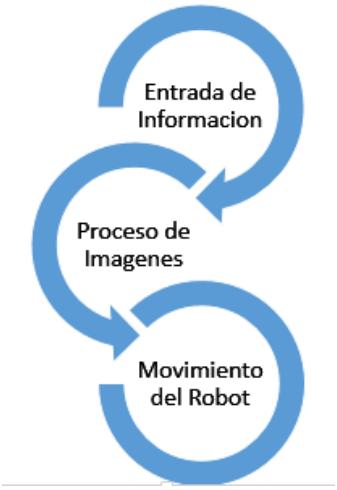


Figura 3. 22: Proceso de comunicación del prototipo
Elaborado por: Autor

3.4.3. Construcción final del prototipo

Con todos los elementos ubicados dentro de una superficie hecha de acrílico se procede a fijar con tornillo cada uno de los elementos para evitar que se muevan con los giros del robot. Como se muestra en la figura 3.23 los elementos de manera fija para proceder a las respectivas conexiones.

Figura 3. 23: Colocación de los componentes del prototipo
Elaborado por: Autor

Con los todos los componentes conectados y fijados de manera correcta (ver figura 3. 24) el diseño final del soccer estará lista para las competencias de soccer autónomo.

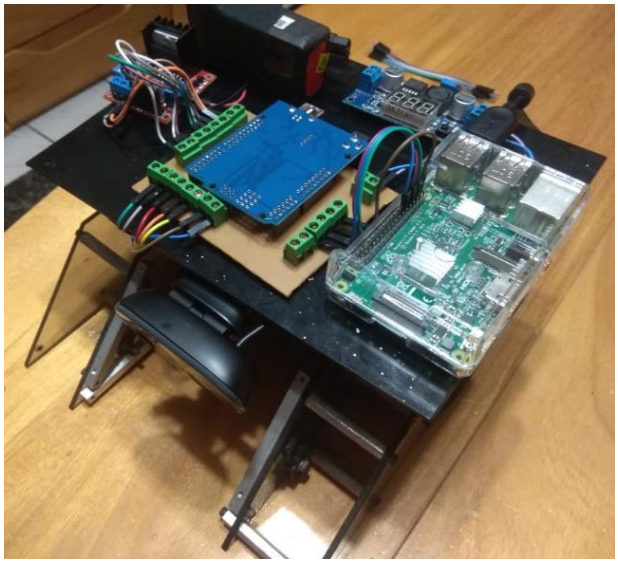


Figura 3. 24: Conexión de los componentes del prototipo
Elaborado por: Autor

CAPÍTULO 4: CONCLUSIONES Y RECOMENDACIONES.

4.1. Conclusiones.

- En el estudio para la selección de elementos se tomó en cuenta el tipo de comunicación efectiva que entre los elementos Arduino y Raspberry Pi para el desarrollo del prototipo del prototipo, así como se escogió cada componente de manera que facilitara su diseño, ya que estos componentes no son difíciles de encontrar y no suponen un gran costo para el presupuesto general; con estos elementos previamente analizados se cumple con las expectativas de calidad y funcionamiento para brindar un correcto desempeño del robot.
- Por medio de la creación de la placa de distribución reducirá el tamaño y espacio del prototipo ya que dentro de esta placa se alojara el Arduino UNO, el cual permitirá un acceso más rápido a sus pines ahorrando el cableado entre componentes y la distribución eléctrica del mismo.
- Para la programación se consideró que sea intuitiva y fácil de entender por parte del Arduino IDE como se muestra dentro de los procedimientos la cual engloba de manera concreta el uso de las variables para una fácil respuesta entre elementos, por parte de la Raspberry Pi se configuro de manera fácil y rápida siendo el principal punto la funcionalidad de grabación y captura de movimiento permitiendo al prototipo diferenciar entre una pelota (escoger un color definido) y una portería.
- Se realizó el diseño y construcción del prototipo utilizando una placa de acrílico la cual reduce el peso del prototipo, se utilizó 2 motores y una estructura alargada para poder dar un centro de gravedad capaz de soportar todo los elementos de la placa.

4.2. Recomendaciones.

- Se recomienda utilizar los elementos originales mencionados en el documento debido a que algunos componentes pueden presentar incompatibilidad en caso de no ser originales, esto es importante para el ahorro de tiempo y gastos innecesarios para la construcción

- Para la programación en Arduino IDE es necesario conocer un nivel básico del lenguaje ya que este no es tan dificultoso dentro de este diseño, mientras que para el lenguaje de programación Python dentro de la Raspberry es necesario conocer los comando para la instalación de paquetes que no contienen por default.
- Es importante al momento de la construcción unir bien los elementos mediante puntos de soldadura para lo cual debe contar con conocimiento de esta herramienta, además de los valores de tolerancia de cada elemento.

Bibliografía

Ayala Martillo, H. (2015). Diseño de aplicaciones de sistemas embebidos basados en tecnología Raspberry y Odroid-U3. Recuperado de http://dspace.ups.edu.ec/handle/123456789/11409

Berjón Díez, D. (2016). Parallel computer vision algorithms for graphics processing units (phd). E.T.S.I. Telecomunicación (UPM). Recuperado de http://oa.upm.es/40621/

Campuzano Bulgarín, A., & Cedeño Vélez, V. (2015). Diseño e implementación de un sistema de seguridad de control local y remoto con dispositivos de vigilancia, desarrollo local y remoto con dispositivos de vigilancia, desarrollado en el software PYTHON, centralizándose en ua tarjeta RASPBERRY PI. Recuperado de http://dspace.ups.edu.ec/handle/123456789/10392

Correa Eras, A. Y., & Remache Ortega, E. P. (2006). Sistema para controlar la velocidad de un motor DC utilizando modulación de ancho de pulso. Recuperado de http://bibdigital.epn.edu.ec/handle/15000/1930

Correa Mora, A. (2018). Implementación de un sistema de transmisión de video en tiempo real utilizando el módulo de comunicación Wolf whoop Q3 5.8 GHz para un robot volador de carreras. Recuperado de http://repositorio.ucsg.edu.ec/handle/3317/10230

Estrella Heredia, S. (2016). Desarrollo de una Librería Utilitaria Funcional sobre una Lenguaje Orientado a Objetos, aplicado a un Caso de Estudio. Recuperado de http://bibdigital.epn.edu.ec/handle/15000/15148

Gonzalez Garcia, I. T. (2015). Diseño e implementación de sistema interactivo de información de Docentes, con Raspberry PI. Recuperado de http://dspace.ups.edu.ec/handle/123456789/10408

Gordon Chavarría, V. J., & Molina Ormaza, J. A. (2016). Diseño e implementación de un Sistema de Circuito Cerrado de Televisión utilizando mini ordenadores Raspberry Pi y cámaras para cubrir el Concurso Ecuatoriano de Robótica dentro de la Universidad Católica de Santiago de Guayaquil. Recuperado de http://repositorio.ucsg.edu.ec/handle/3317/5459

Gualli Cujilema, M. A., & Véliz Intriago, R. A. (2016). Implementación de un robot MegaSumo radiocontrolado y autónomo utilizando microcontroladores PIC. Recuperado de http://repositorio.ucsg.edu.ec/handle/3317/6401

Iperty Barros, V. F., & Cruz Hermenejildo, J. E. (2018). Simulación y automatización de los sistemas variadores de velocidad para motores de corriente continúa. Recuperado de http://dspace.ups.edu.ec/handle/123456789/15735

Jampani, V. (2017). Learning Inference Models for Computer Vision, 53.

Martinez Escobar, S. (2015). Diseño e implementación de un sistema de comunicación integrando tecnologías para el mejoramiento de voz IP interna de la Facultad De Ingeniería De La Universidad Nacional De Chimborazo. Recuperado de http://dspace.unach.edu.ec/handle/51000/610

Miranda Sislema, E. (2018). Diseño de un sistema de rastreo básico en tiempo real con tarjetas de desarrollo de bajo costo para objetos rastreables. Recuperado de http://dspace.espoch.edu.ec/handle/123456789/9265

Mollocana Alban, R. (2018). Sistema domótico de apoyo para personas con discapacidad motriz mediante tecnología móvil y reconocimiento de voz. Recuperado de http://repositorio.uta.edu.ec/jspui/handle/123456789/28012

Ninabanda Pilco, V. (2016). Diseño e Implementación de una radiosonda de transmisión de parámetros meteorológicos para el “programa integrado de monitoreo de control de calidad del aire en la ciudad de Riobamba usando modelos Matemáticos”. Recuperado de http://dspace.unach.edu.ec/handle/51000/3260

Reyes Dominguez, F. (2017). Diseño de un módulo de simulación para el control de posicionamiento y de velocidad de un motor DC utilizando control predictivo con supervisión a través de un sistema SCADA. Recuperado de http://repositorio.ucsg.edu.ec/handle/3317/9245

Rodríguez Chicaiza, G. (2016). Diseño e implementación de un sistema eléctrico inteligente, en una oficina modelo basada en la aplicación de la domótica para el mejoramiento de la eficiencia energética y el

confort. Recuperado de http://repositorio.utc.edu.ec/handle/27000/3089

Simbaña Quilachamín, F. (2017). Sistema de seguridad antirrobo vehicular, utilizando Raspberry Pi con tecnología 3G y una aplicación para smartphone. Recuperado de http://repositorio.uisrael.edu.ec/handle/47000/1425

Soria Andrade, V. (2017). Diseño e implementación de un sistema electrónico para el control, seguridad y rastreo vehicular utilizando un ordenador de placa reducida Raspberry. Recuperado de http://dspace.espoch.edu.ec/handle/123456789/6868

Taha Mohamed, G. (2014). Automated Inspection of Apple Moth : Machine vision using OpenCV and Raspberry Pi. Recuperado de http://www.theseus.fi/handle/10024/78200

DECLARACIÓN Y AUTORIZACIÓN

Yo, **Veliz Plua, Jean Carlos** con C.C: # 092029300-8 autor del Trabajo de Titulación: **"Diseño e implementación de robot soccer autónomo con captura de movimientos a través del sistema embebido Raspberry Pi"** previo a la obtención del título de **INGENIERO EN TELECOMUNICACIONES** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 13 de Marzo de 2019

f. _____

Nombre: Veliz Plua, Jean Carlos

C.C: 092029300-8

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA			
FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN			
TÍTULO Y SUBTÍTULO:	Diseño e implementación de robot soccer autónomo con captura de movimientos a través del sistema embebido Raspberry Pi		
AUTOR(ES)	VELIZ PLUA, JEAN CARLOS		
REVISOR(ES)/TUTOR(ES)	M. Sc. EDWIN F. PALACIOS MELÉNDEZ		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil		
FACULTAD:	Facultad de Educación Técnica para el Desarrollo		
CARRERA:	Ingeniería en Telecomunicaciones		
TITULO OBTENIDO:	Ingeniero en Telecomunicaciones		
FECHA DE PUBLICACIÓN:	13 de Marzo de 2019	**No. DE PÁGINAS:**	66
ÁREAS TEMÁTICAS:	Electrónica		
PALABRAS CLAVES/ KEYWORDS:	**CONTROLADOR, OPENCV, DETECCION, MOVIMIENTO, CSV, ARDUINO, RASPBERRY, CAPTURA, IMÁGENES.**		

RESUMEN/ABSTRACT :

El presente trabajo de titulación consiste en el diseño e implementación de un soccer autónomo con captura de movimiento a través del sistema embebido Raspberry Pi; cuya razón por la cual este trabajo de titulación fue creado es motivar el desarrollo de nuevos diseños en establecimientos educativos. Dentro del trabajo de titulación se utilizó el método descriptivo e investigativo los cuales permitieron cumplir con éxito los objetivos que se plantearon. Este proyecto a nivel teórico describe el uso y características de cada uno de los componentes utilizados a lo largo de la implementación del prototipo, también se diseñó el código en ambas plataformas como lo son Arduino y Python para el correcto funcionamiento del prototipo. Se diseñó una placa de distribución. Con el diseño de este prototipo permitirá a la institución ingresar a competencias nacionales e internacionales de soccer con características autónomas que se realizan cada año en diferentes partes del mundo.

ADJUNTO PDF:	☒ SI	☐ NO
CONTACTO CON AUTOR/ES:	**Teléfono:** +593996607113	E-mail: jankarlos93@hotmail.com
CONTACTO CON LA INSTITUCIÓN: COORDINADOR DEL PROCESO DE UTE	**Nombre:** Palacios Meléndez Edwin Fernando	
	Teléfono: +593-9-68366762	
	E-mail: edwin.palacios@cu.ucsg.edu.ec	

SECCIÓN PARA USO DE BIBLIOTECA	
Nº. DE REGISTRO (en base a datos):	
Nº. DE CLASIFICACIÓN:	
DIRECCIÓN URL (tesis en la web):	