PrepaTec Eugenio Garza Lagüera – Grupo Estudiantil AIZER

Robocup Junior Soccer Open
Technical Description Paper

Roberta Camila Garnica Juárez	A01199588@tec.mx
Íker Cossío García	A01283181@tec.mx
Fátima Chantal González Solís	A01384777@tec.mx
Alejandro Salinas Salas	A01287454@tec.mx

I. Introducción

1.1 Grupo Estudiantil AIZER

Grupo Estudiantil AIZER de la PrepaTec Eugenio Garza Lagüera participará por segunda vez en el TMR bajo el nombre de ‘AIZER’. Es un grupo estudiantil que comenzó sus actividades como un grupo a la segunda mitad del año pasado. Comenzó realizando clases para personas que tuvieran el interés de participar en una competencia de robótica y finalmente a inicios de el año 2024 comenzaron con el desarrollo de dos robots para Robocup Junior Soccer Open.

1.2 Experiencias previas

Iker Cossío y Alejandro Salinas participaron previamente en el Torneo Mexicano de Robótica 2023 donde consiguieron el primer lugar y el reto técnico obteniendo el pase al mundial de Robocup2023 en Burdeos, Francia.

1.3 Objetivos y metas

AIZER se propone con este TMR2024 impulsar el nivel en las competencias de Robocup Junior Soccer, implementando tecnologías que no habían sido implementadas en el TMR y que están inspiradas en lo visto en el mundial de Robocup2023.

II. Diseño y mecánica

2.1. Herramientas

Tras una evaluación entre diversas herramientas para el desarrollo CAD, se tomaron en cuenta AutoCad, Solidworks y Onshape. Durante la evaluación se tomaron en cuenta factores como curva de aprendizaje, costo, complejidad, soporte y colaboración. Solidworks y AutoCAD contienen la desventaja de no estar disponibles para todos los sistemas operativos, lo que ponía una barrera de entrada bastante complicada. Además que la colaboratividad con estos dos programas hacía más complicado hacer un diseño en conjunto con todos los miembros del equipo. Onshape fue la herrramienta que elegimos ya que dos miembros del equipo ya tenían experiencia con la aplicación y fue potenciada en parte con el modo en el que no se requiere una instalación para utilizar el programa. Relacionado a los costos, al AIZER ser un grupo estudiantil de la PrepaTec, estos tres softwares estan incluidos en nuestras matrículas por lo que no fue un factor a considerar.

2.2 Piezas desarrolladas

Se desarrollaron las bases para adaptarse al diseño electrónico de las PCBs y las necesidades de visión de programación. Se tomaron en cuenta las medidas de la cancha para crear piezas de alto nivel de complejidad como una cámara omnidireccional.

2.3 Kicker

La idea de AIZER era llevar tanto Kicker como dribbler, sin embargo por presupuesto y tiempo nos vimos limitados a incluir unicamente un Kicker de 20N, 12V ya que creemos que de acuerdo a simulaciones es más conveniente un Kicker ya que se incrementan la cantidad de disparos a gol, a diferencia de un dribbler que a pesar de permitirnos un mejor control, lo podiamos obtener gracias al espejo omnidireccional.

2.4 Espejo omnidireccional

Tras observar a otros equipos utilizar un espejo omnidireccional nos propusimos crear el nuestro, diseñamos nuestra pieza con forma de un cono curvo en lugar de hipérbole ya que nos permitia ver toda la cancha sin importar en qué lugar nos encontráramos. Diseñamos las piezas de aluminio y las manufacturamos con un proveedor en China. Posteriomente mandamos a cromar 2 piezas y a pulir otras 2 piezas para comparar.

Tipo de espejo	Costo	Tiempo	Precisión	Color	Resistencia
Cromado	350	1 semana	+1.5mm	Colores brillantes y menos deformación de imagen	Resistente a rayones
Pulido	250	2 dias	Sin pérdida de precisión	Colores más opacos y con más deformación	No resistente a rayones

III. Desarrollo de eléctrica

3.1 Herramienta

Anteriormente habíamos utilizado Eagle de Autodesk por su capacidad para crear proyectos más grandes y confiables. Pero en esta edición del TMR nos decidimos por utilizar Fusion360 ya que cuenta con características más modernas a diferencia de Eagle.

3.2 Placas

Se crearon 4 placas distintas.

- Control
- Motores
- Sensores de luz
- Atmega

en las cuales se dividían el trabajo de programación de una manera estructurada, y sin cables lo que permitía un armado más veloz y un tiempo de respuesta más rápido en caso de un error.

3.3 Componentes usados

3 Motores Pololu HP 9.68:1 12V 48CPR Encoder	$800 c/u
3 Llantas omnidireccionales GTF de 50mm	$800 c/u
3 Controladores de motores DRV8256E Single brushed DC motor driver carrier	$200 c/u
1 ATMEGA328P-PU	$200c/u
1 Regulador de voltaje a 8A	$100c/u
1 BNO055	$900c/u
1 Raspberry Pi Pico W 2	$250c/u
1 Batería LIPO 2200mAh 50C 11.1V	$500c/u
1 Jetson nano 4GB	$3600c/u
1 Raspberry pi camera module 3	$1100c/u
1 Arducam IMX708	$700c/u
1 espejo omnidireccional	$1500c/u
4 placas de eléctrica	$3600
1 Solenoide 20N, 12V	$350c/u
1 Cable microusb 15cm	$100c/u
1 Tubo de neumática	$15c/u
1 Kg PLA	$500c/u
9 LDR	$8c/u
9 LED ROJO ULTRABRILLANTE	$3c/u

IV. Programación

4.1 Herramientas utilizadas

Para crear la lógica del robot, se utilizaron los entornos de desarrollo integrados de Jetbrains como Clion, Pycharm y Webstorm. Además de herramientas como nano y vim para realizar cambios pequeños a los programas.

4.2 Hardware y procesadores

El robot consta de 3 procesadores principales, 1 ATMEGA328P-PU para la deteccion de líneas, 1 Raspberry PI Pico W 2 para controlar los motores y comunicarse con la Jetson Nano de 4GB que se encarga de la lógica y de la visión de las dos cámaras del robot.

4.3 Lenguajes de programación

El robot está programado exclusivamente en C++ a excepción del nodo de comunicación por serial de la jetson nano que utiliza python para comunicarse. Utilizamos ROS para comunicar los nodos y crear un diseño más estable, fácil de debugear y con mayor posibilidad de crecimiento.

4.4 ROS en Robocup Junior Soccer Open

Implementar o no implementar ROS fue una decisión difícil ya que tendría más reto implementarlo. Al final nos decidimos en utilizarlo ya que era una manera fácil de unir los programas que ya habíamos diseñado y nos permitía debugear el robot de manera remota y con una facilidad que no habíamos conseguido con microprocesadores.

Los nodos de ros de nuestro robot se dividen de la siguiente manera

- /frontcamera: Se encarga de analizar y detectar objetos en la camara frontal y ser enviados al nodo de estrategia mediante un tipo de dato Float32MultiArray
- /omnicamera: Se encarga de detectar la pelota y enviar su información a través de un Float32MultiArray al nodo de estrategia
- /strategy: Basándose en los datos recibidos de la Raspberry Pi Pico W 2 y las cámaras creamos estrategias de defensa y ataque y enviamos las decisiones en un formato de UINT64 con un contenido de máscara de bits que luego es interpretado por la raspberry pi pico
- /rpi-serial.py: Utilizando los datos recibidos del nodo de estrategia, se encarga de enviarlos a la raspberry pi pico y además, se encarga de actualizar los datos como el IMU y el estado de los sensores de luz