chaBots Phoenix

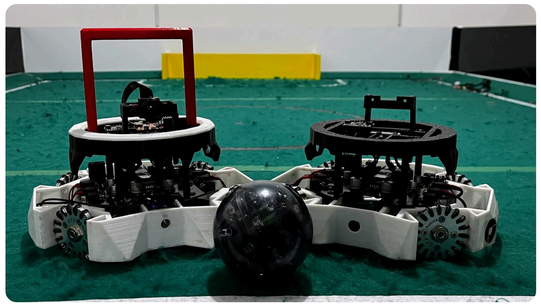
Mexico, Soccer Lightweight League

TEAM MEMBERS

Honoré Villanueva - Low level programmer
Hugo Guerrero - CAD and PCB designer
Alfonso De Anda - High level programmer

LINKS

Linktree for our socials, contact and GitHub



Picture of our robots at our workshop in Mexico.

ABSTRACT

chaBots Phoenix is a team of 3 students from San Luis Potosi, Mexico, with years of experience in competitive robotics, including 2 years competing in Soccer Lightweight League Open Mexico events.

Our robot design process started in 2024 by our team captain, and has had many improvements and iterations since then, including a reliable kicker system used to kick the ball with both robots, a highly responsive line sensor, and better developed PCBs with integrated circuits using only surface mounted components.

We have been working these past 6 months to integrate as many features from both our circuits and our code as possible and have reached very promising results, however we still have many features to adapt and revise. For our robots software, we even developed our own libraries, and made them public for others to have reference and have a better starting point, just as we did with other teams robot documentation.

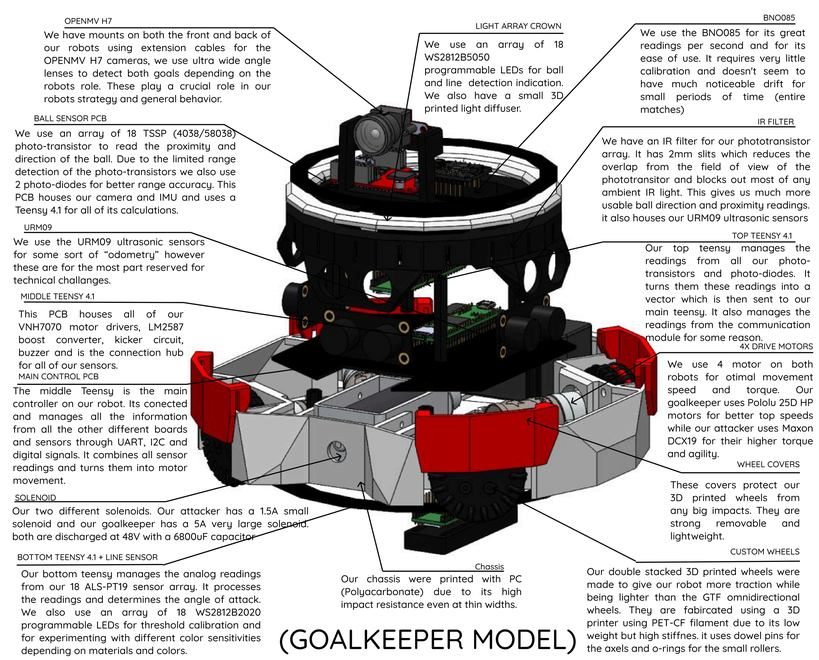
This is our first time at a RoboCup Junior international event, so there are many unknowns for us. We hope to learn from as much as possible from other teams and help other new teams with some of our material.

Team members in the same order and as above, from left to right

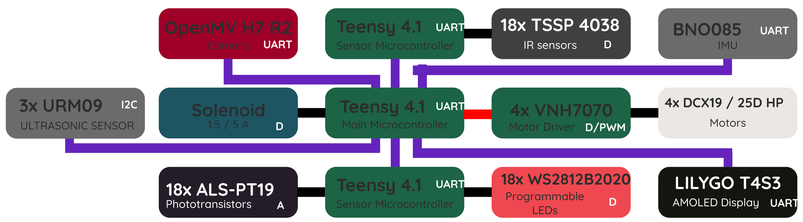
METHOD, ROBOT PRODUCTION & DESIGN

ROBOTS OVERVIEW

This years robot was made for lots of tests as it is our first time going to a RoboCup international event. Most of the PCBs have many unused circuits that we wanted to test in the future. Most of the features on this robot still have lots of space for improvement such as the kicker and wheels but this sets the base for future iterations. Its main objectives were to test features, be easily assembled and reliable. We were able to achieve low center of mass on both robots making them very stable. All of our PBCs are two layered to keep them simple and easily fixable. The following image descibes our robots features.

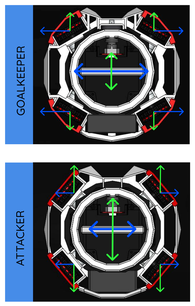


Components Diagram:



DESIGN HIGHLIGHTS

The main highlight for this years design is the different robot design intended for different roles. Most of this comes down to the our motor choice, orientation and weight. For out goalkeeper we chose the Pololu 25D 1:9.7 12V HP motor for better top speeds (~1000RPM) and their shorter form factor which allowed us to place a larger 5A Solenoid kicker. For our main robot we chose the MAXON DCX19 (~800RPM) due to their lighter weight and higher torque/efficiency, these motors allow us to have very good agility thanks to their higher torque and their lower weight allows us for potential use of a dribbler. The real design decision is their uneven angle offset. Our goalkeeper has 60°/120° motor angles, which gives it a heavy bias towards lateral movement speeds. On the other hand, our attacker robot has 100°/80° motor angles giving it a slight bias towards vertical moments. This was all achieved while using the same PCB design on both robots. However we did require two very different chasis designs.



SOFTWARE

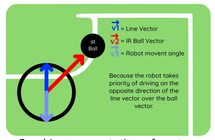
Our program structure is primarily based on **vector calculations** for holonomic movement and sensors detection, this is because of **accuracy** and the **easy adaptation** to our omnidirectional system.

We use one microcontroller for **calculating a vector** for every TSSP reading using the pulse width of the received IR pulse as magnitude, and then summing all of the vectors to get a final one **pointing to an accurate position estimation of the ball.** A similar process is made as well with the Teensy in charge of the line sensors, just this time we don't need a magnitude value. Then, we **send the resulting vectors** angle and magnitude **through UART** to the main microcontroller which decides the motors output based on a **Finite State Machine** to take decisions depending on the robots role.

We use **Arduino IDE** with Teensyduino for our Teensy 4.1 code and **OpenMV IDE** for our camera module, together with a **GitHub repository** for having good and collaborative version control.

Software highlights:

- Self coded sensor **libraries**
- Highly responsive **UART communication**
- Decision making based on **goal** position
- **PD controller** for angular correction
- Organized code in **classes and methods**

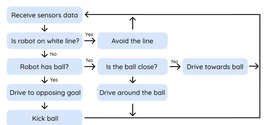


Graphic representation of vectors and decision making

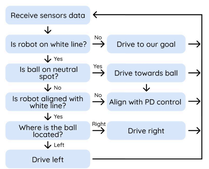
QR code to our public software GitHub repository

Attacker robot:

The biggest challenge for our attacker robot was to drive at high speeds without colliding with the ball nor losing control over it to score. We were able to solve this by optimizing our UART communication algorithm for faster responses, manipulating speeds depending on states, using analog photodiodes for reference of when the ball is on front of the ball and driving towards the opposing goal for consistent scoring.



Attacker robot decision taking flow chart



Defender robot decision taking flow chart

Defender robot:

This robot is developed with the purpose of avoiding the opposing team to score goals, we do this by driving the robot either left or right depending on the ball's position and aligning moving forward and backward with a PD controller in reference of the white line of the penalty area. We use a camera facing towards our goal for the robot to go back to the white line when the robot is out of the defending area.

DATA, RESULTS & DISCUSSION

IDENTIFIED PROBLEMS

Right after our regional event, we focused on solving the following issues we had at the competition: slow reaction times, getting easily out of bounds and not being able to score consistently into the opposite goal.

After days testing, we came to the conclusion that the slow reaction times and getting out of bounds were related to one major problem: the loop frequency times in both sender and receiver microcontrollers was too slow, this because of large delays and unnecessary trigonometric operations in our sensors controllers, and a constant ineffective update of states in our main controller.

We also realized that even though we had a camera to detect both our goal and the opponents, we couldn't use it to stably score goals because of one problem: since we were using a mirror to see both forward and backward, the cameras resolution divides by two, allowing for less precision when searching for the goals. Also, we would have to frequently clean the mirror and check if it was at the correct height.

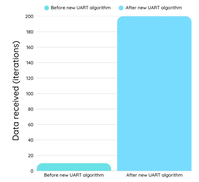
MODIFICATIONS AND TESTING

To address these issues, we implemented a new algorithm for faster UART communication, which significantly reduced the time needed for receiving and interpret the data in the main microcontroller. We also used a new camera system to detect the opposite goal on our attacker robot and our goal in the defense robot.

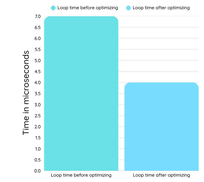
Change implemented	Impact
Replaced expensive trigonometric operations	Increased loop frequency in transmitter Teensy
Reduce the quantity on data being transmitted	Improved reliability and stability on receiver Teensy
Sent the needed data in a compact way	No need of sending more than one byte of data each iteration
Only update not indispensable data every 10ms	Increased loop frequency in receiver microcontroller
Wide angle lens on camera	Increased FOV and resolution

RESULTS

- Latency between communicating microcontrollers reduced by 2500% (100ms to 4ms per cycle)
- Sensor data gets updated faster
- Real time responsiveness when driving the robot



Amount of sensor data received in 1000 milliseconds (one second) before and after



Loop time of the decision taking Teensy before and after optimization

DISCUSSION

These software improvements allowed for smoother ball control and more accurate defensive reactions. Optimized serial communication ensured the main Teensy had up-to-date sensor information at all times, which was especially beneficial during high-speed gameplay situations at the world championship.