RoboCupJunior Open Weight Soccer 2024

Team Description Paper

The Pythons

Ahmed Wael, Ahmad Zaki, Abdullah Ayman, Adam Mohamed

Mentors: Adham Amr, Amr El Shabacy, Youssef Attia

Sidi Gaber Language School for Boys, Alexandria, Egypt

Innova Stem Education

info@thepythons.tech

Abstract: This paper provides a detailed explanation of the mechanical, electrical and software designs of the pythons' team in order to compete in the RoboCupJunior 2024 in Germany after taking the second place in the national competition willing to do our best in this competition. As it is our first year in the soccer category, we underwent a lot of experiences despite the unfortunate events that have been happening lately, we have always been passionate about our work. The changes from the last competition were substantial to improve the efficiency and looks. On the hardware's side, we now use a multi-layer PCB to achieve better stability instead of a PCB with ordinary wiring, we implemented a new design for the dribbler to achieve better ball control.

Table of Contents:

1. Introduction

1.1 Team Background

THE PYTHONS Robotics Team is a project created in 2018 at Alexandria, Egypt by a group of high school students came up to build robots to compete in the national competitions which we came up with a positive history. Our team tends to continue this journey of learning even if it was the hard path and try to benefit from our past experiences and put it all in our work hoping that we can do our best in this competition. We participated in the national competition four times, and we count:

- Fourth place in Rescue Line RCJ 2018
- Second place in WRO 2022
- First place in Rescue Maze RCJ 2023
- Second place in Soccer Open Weight RCJ 2024

This team consists of 2 Software Developers (Ahmad Zaki & Ahmed Wael), Electrical (Abdullah Hegazy) and Mechanical (Adam Mohamed).

1.2 Team Photo

Figure 1: Team Photo

1.3 Highlights

Our robot now works with a multi-layer board as its main base which is all the important components are attached to it. The design was made with Eagle CAD and Solid Works. Many improvements were made: the hardware now is more practical as the design is more efficient and elegant than before, we used:

- 1 PCB for the absence of ordinary connectors achieving more stability and a better look.
- 8 IRs sensors distributed all over the lower plate in order to achieve better line detecting.
- New dribbler design to achieve more ball controlling
- Hyperbolic mirror

From the software side now, we have:

- We have changed the control system from the Arduino Mega to the Raspberry Pi Pico which led us to change the software structure to adapt to the new control system.

2. Robot

2.1 Hardware

2.1.1 Mechanical Design

The robot's chassis has been designed in SolidWorks software. The robot is built upon three layers:

Base Plate: The lowest plate is CNC laser cut out of FR4 which consists of woven fiberglass and a flame-retardant epoxy resin which means it can withstand the weight of the whole robot.
Four 12V DC motors are fixed in the shape of "×" combined with the Omni Wheels the robot is able to move holonomically.
Between this plate and the second PCB plate are the IR sensor modules used to detect the out-line. These are attached to both plates.

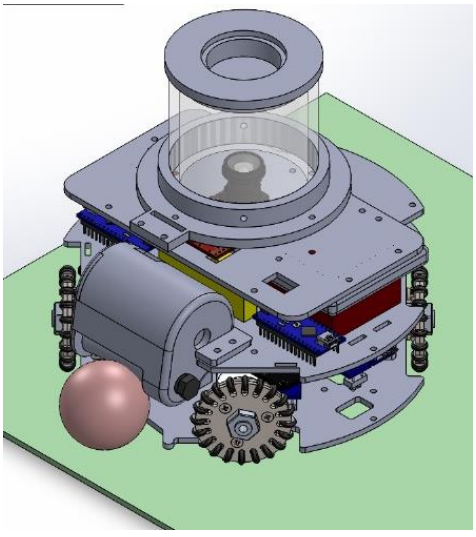

Figure 2: Robot design in SolidWorks software

Second Plate: The second plate is the main plate that combines both the mechanical design with the electrical system. This components on this plate will be discussed in the electrical section.

Third Plate: The third plate is the smallest plate out of the three. It is printed of 3mm Acrylic. All the components related to the vision are attached to this plate, such as: the OpenMV Cam H7, the chrome-plated hyperbolic mirror, the transparent acrylic tube and its holder. A 3D-printed PLA material custom holder for the battery (which rests on the second plate) is also fixed to this plate to prevent the battery from sliding around.

Dribbler: The dribbler that is used to control the ball. The dribbler's body is 3D-printed PLA material and fits the 6mm Acrylic gears (2 gears with ratio 3:1) and the 2200 KV brushless motor inside. A curved rubber mould has been custom-made to make the ball self-align to be in front of the center of solenoid shaft.

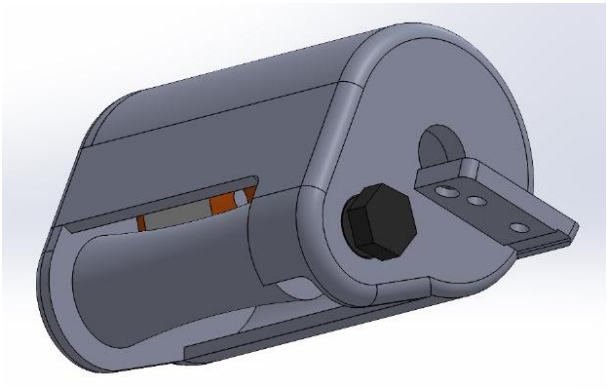

Figure 3: Robot dribbler in SolidWorks software

Mirror: We developed an omni vision system utilizing a camera paired with a hyperbolic mirror. The mirror is made from stainless steel and electroplated with chrome. There were multiple equations for the surface of the hyperbolic mirror, but after simulating the equations on Desmos software, we settled on the one we deemed to be the best. We machined the mirror from a stainless steel block then electroplated with chrome to make it more durable and reflective.

Figure 4: Stainless Steel Mirror

2.1.2 Electrical Design

2.1.2.1 Components

Table 1: Components list

Function	Part	Qt.
Control	Raspberry Pi Pico	1
	Arduino Nano	2
Power	11.1V 5200 mAh Lithium Polymer Battery	1
	Logic Level Converter (3.3V, 5V)	2
Movement	Metal Gear Motor 25Dx65L	4
	Cytron 10A Dual Motor Driver	2
Cameras	Openmv H7	1
Orientation	10 DOF IMU (MPU9250+BMP280)	1
Kicker	12V Push-Pull Solenoid	1
	Mosfet Module	1
	Boost Circuit	1
Dribbler	A2212 1000KV Brushless Motor	1
	ESC 30A	1

MICROCONTROLLERS

Raspberry Pi Pico: A lightweight controller that is compatible with the Arduino Libraries. It is the master that commands two other secondary boards. It decides what to do based on the detected inputs. It is connected to logic converters to step down the signals from the Arduino nano from 5v to 3.3v and vice versa and the motor drivers, ESC and the camera are connected to it directly.

Arduino Nano (ATmega328): The Arduino Nano is an open-source breadboard-friendly microcontroller board based on the Microchip ATmega328P, microcontroller. We are using 2 boards the first one is connected to IMU(Compass) and connected to the Pico via UART communication and the other one is connected to the IR line sensors and to the Pico via parallel communication.

CAMERA

OpenMV H7: We mounted one of them in each robot ,with the hyperbolic shaped mirror to see the entire field and track the goals and the ball by their color and they are connected to the Raspberry Pi Pico via UART communication.

POSITIONING SENSORS

10 DOF IMU (MPU9250+BMP280): This sensor module which uses a combination of MPU-9250 (3 axis Gyro, 3axis Accelerometer, digital Compass) and BMP280 barometric pressure sensor. We are using it as a compass to keep the robot heading always towards the opponent's goal and it is connected to one of the nanos that is connected to the Pico via UART communication.

IR Sensors (TCRT5000 Reflective IR Sensor): The TCRT5000 Reflective IR sensor photoelectric switch is perfect for line tracking robot. It includes an infrared emitter and phototransistor in a leaded

package which blocks visible light. It has a compact construction where the emitting-light source and the detector are arranged in the same direction to sense the presence of an object by using the reflective IR beam from the object. We are using it for detecting the outline of the field so we can keep the robot from getting out of the field during the match all the IR sensors are inserted in the main plate in a circular shape (2 sensors opposing each other) aiming that we could have the best line detection. All of the IR sensors are connected to the other Arduino Nanos which are connected to the Pico by parallel communication and powered up by a 7805 voltage regulator.

Power Supply: This robot power supply is a 12V LiPo battery & 5200mah current as the main source of power. We are using two 7805 voltage regulator one to power up the Raspberry pi Pico and the other for their sensors and the power of the camera is taken from a cell of the battery

2.1.2.1 PCB Design

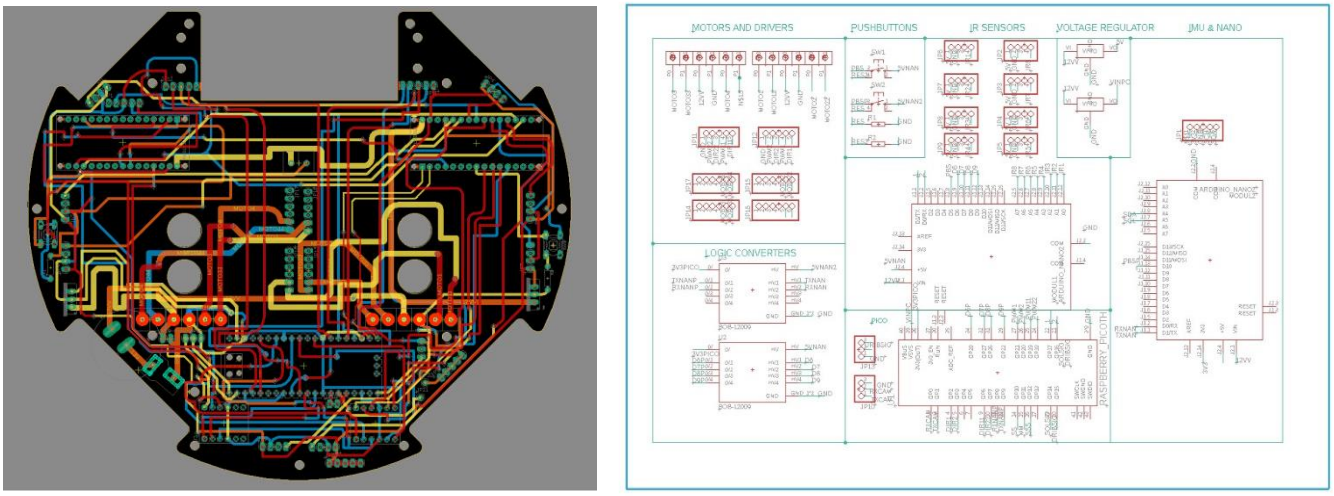

Figure 5: PCB design on Eagle Software

2.2 Software

Overview: We tried to create a software system with modules that can be changed individually which allows ease of debugging and quick changes if necessary.

Movement: The motor configuration allows the robot to move using holonomic motion. This allows the robot to move in any direction instantly, not having to turn then moving forward. You can just give the motors the needed speeds and directions to make the robot move directly towards a target. We have developed an equation for each motor to get its speed and direction. The equations were deduced after we have worked for quite a bit of time using if conditions and assigning the speeds according to ratios between the wanted angle and which quadrant it is on, but this was too hard to debug. The motion equations depend on three parameters: turn, which is how much we want to turn the robot (used in correcting heading), forward and side (both are calculated by getting the sin and cos of the wanted angle, respectively and multiplying it by the maximum speed). You get the equations by first assuming one of the directions for each motor to be positive (we assumed that the clockwise direction is the positive one). Then you see for each of the movements whether the directions of movement are the same or not, as seen in figure 3. So, the equations will be:

$$Upper\ Left = turn + forward + side$$
$$Upper\ Right = turn - forward + side$$
$$Down\ Left = turn + forward - side$$
$$Down\ Right = turn - forward - side$$

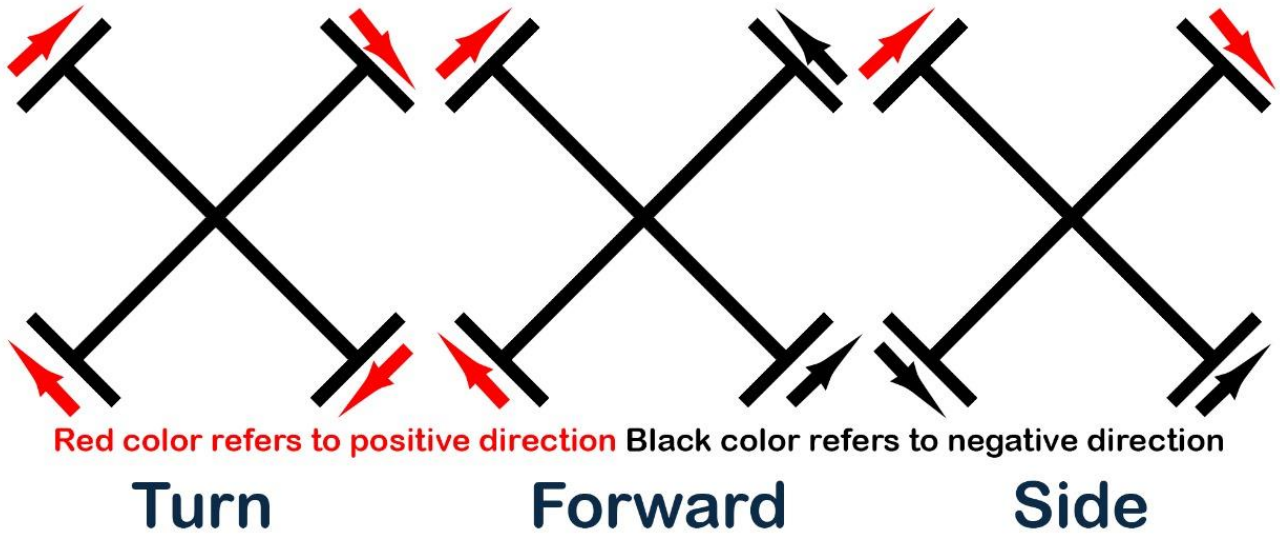

Figure 6: Robot motion visualization

Vision: The algorithm detects the color of the ball as blobs then it gets the largest blob to neglect errors in camera view. Then it calculates the angle of the detected blob with respect to the center of the camera. For example, in figure 4, the angle of the ball is 41°. The angle is calculated from the left normal of the view clockwise. The radius of the ball from the center of the robot is calculated. The angle between the dribbler (white cross) and the ball is calculated, if the angle is between 180 & 360, the robot will move according to the dribbler angle so that the ball becomes in front of the robot. Otherwise, the camera angle is used. Then the state of the robot is sent from the camera to the Pico together with the selected angle. Each frame the position of the ball is saved and if the ball is out of sight, the last location of the ball is then checked to see whether the ball was last seen in the front center of the robot, so it will most likely be controlled by the dribbler. After that the robot starts to search for the opponent's goal. When the perpendicular distance to the goal is near enough, the OpenMV Cam sends a signal to the Raspberry Pi Pico to shoot.

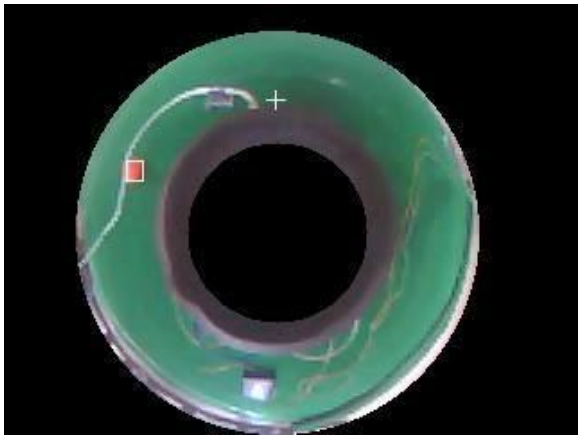

Figure 7: View from camera in robot

IMU:

Calibration: Our calibration process is divided into two parts, gyroscope calibration and magnetometer calibration. As for the gyroscope calibration we used an already built library that calculates the offsets for the readings then we assign those offsets to the registers of the sensor. This process is done once for each sensor module and these values stay constant. Then for the magnetometer calibration we tried using a software called MotionCal which calculates the offsets for soft-iron & hard-iron. The hard-iron offsets are errors in the reading caused by nearby magnets for example: dc motors, while soft-iron offsets are caused due to nearby magnetic fields generated by current flowing in wires. The hard-iron offsets are three values, one for each axis. While the soft-iron offsets are a matrix. For an ideal IMU sensor, it will be an identity matrix.

Usage: The algorithm calculates the heading angle of the robot by getting the raw readings of the z-axis of the gyroscope and integrating it to estimate the yaw angle of the robot. Then it gets the magnetometer readings, subtracts the magnetometer offsets and calculates the tan inverse of the x and y magnetometer readings to estimate the yaw angle with respect to the north and then subtracts the initial reading from it. Then we apply a complementary filter between the gyroscope and magnetometer readings to achieve stable and responsive readings for the robot's heading.

Line Sensors:

Calibration: The algorithm for the calibration is based on getting the minimum and maximum values from each of the 8 sensors and create a range for each sensor for which a line is detected. For performing a calibration, the robot should be placed on a line and click on the calibration button. The robot will start spinning in place and save the readings for 20 seconds. The minimum and maximum values for each sensor are saved to the EEPROM of the Arduino Nano.

Usage: The already saved values are loaded at the start of the code and each sensor reading is compared to its saved values to check whether it detected a line or not. Then another function decides how to modify the forward and side motion parameters to avoid going out of bounds.

Sweeping Routine: When the robot doesn't know where the ball is, the robot will start trying to find the ball by going forward till the front IR sensors read a line then it will reverse till the back IR sensors read a line and so on until the ball is found

3. Performance

3.1 Testing

Hardware testing: We start the hardware testing with a checklist to ensure that all mechanical and electrical subsystems are working properly. This process is usually done after changing anything within the hardware as connecting or disconnecting any components or repairing any hardware damage. There is also a testing phase at the start of the algorithm to test all the sensors and if there are any errors the code is aborted and the sensor with the problem is stated to check it for any wiring issues.

Software testing: Every time we test the algorithm, we first move the robot manually and check that the data readings are as expected. If they are correct, we run the robot autonomously, otherwise we start by pinpointing the issue at hand and debugging the code till it works.

4. References

Our GitHub Repository
Our Website