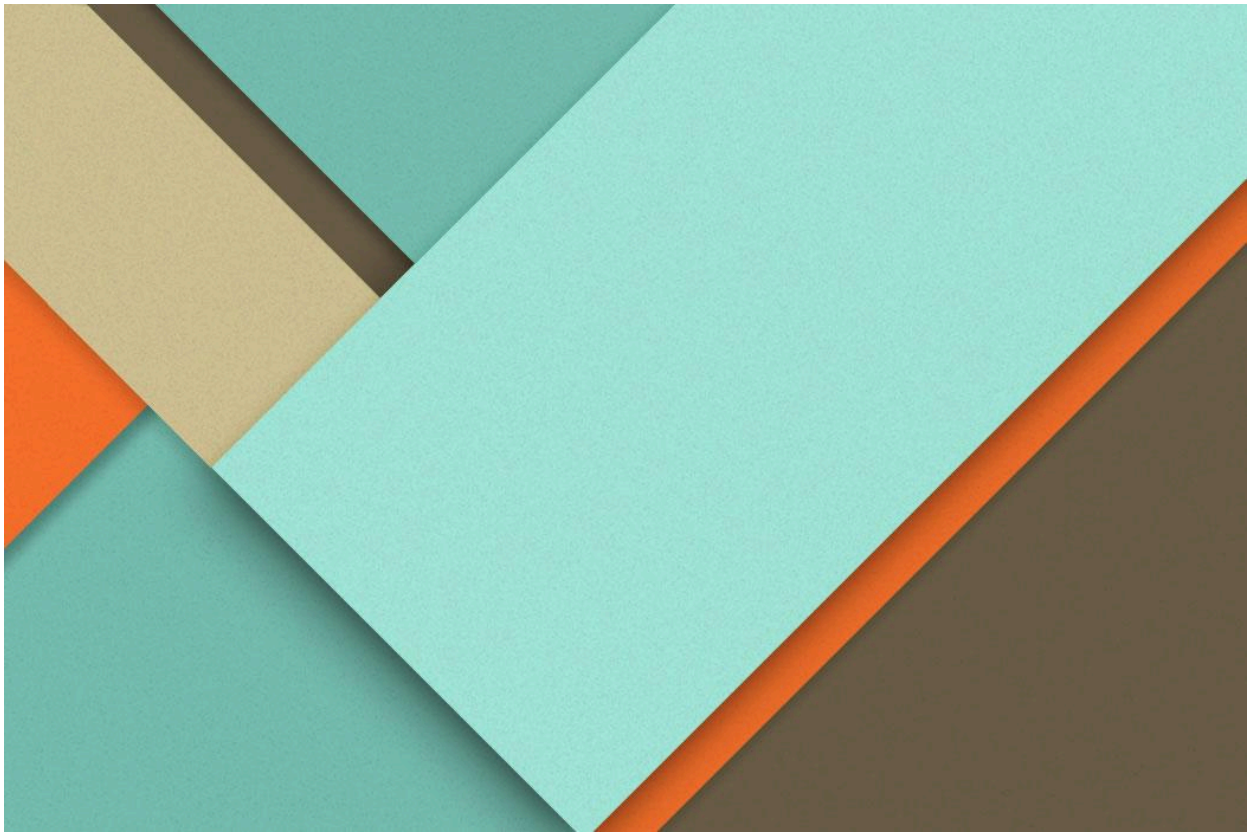


Oak-Labs Documentation

Created by: The Oak-Labs Team
For the: World Robotics Olympiad (WRO) Future Engineers 2025

Table of Content

Mobility Management

Driving System

1.1 RS-365PH Dual-Shaft Motor

The RS-365PH motor powers both rear wheels through a single axle drive, connected by a 2:1 gear ratio gear system. This means that both rear wheels rotate at the same speed, but with twice the torque and half the speed compared to the motor's direct output. The dual-shaft configuration allows stable gear mounting and better torque transfer to the axle.

RS-365PH Motor Specifications:

- **Operating voltage:** 6V to 24V (DEPENDS ON MODEL)
- **No-load speed:** ~4000 RPM at 9V (DEPENDS ON MODEL)
- **Current draw:** 70–250 mA at no load, significantly higher under load

The motor is controlled by PWM signals from the Arduino Uno, which regulate the speed of the rear wheels. The motor driver manages the power supply and controls motor direction.

How It Works:

- **Forward movement:** The RS-365PH drives the rear wheels through the single axle. With the 2:1 gear ratio, the wheels spin at half the motor's output speed but with increased torque, giving smoother acceleration and stronger pulling power.
- **Backward movement:** The motor reverses, causing both rear wheels to rotate in the opposite direction, moving the bot backward.

Since there is no differential, both rear wheels are always locked to rotate at the same speed. This setup provides reliable forward and backward motion, but turning relies heavily on the front-wheel steering system.

1.2 Single Axle and 2:1 Gear Ratio

The 2:1 gear ratio ensures that the rear wheels receive more torque for stronger pushing power, while still rotating at the same speed. This simplifies the drivetrain but makes the bot fully reliant on the front servo steering for turning.

Steering System

1.3 Tower Pro MG995 Servo Motor

The steering system is powered by a Tower Pro MG995 high-torque servo motor, which controls the angle of the front two wheels. This servo provides reliable torque and durable performance compared to smaller micro servos, making it well-suited for steering applications in the bot.

Tower Pro MG995 Servo Specifications:

- **Operating voltage:** 4.8V – 7.2V
- **Torque:** ~10 kg·cm at 6V
- **Speed:** ~0.20s/60° at 6V
- **Control:** PWM signal from Arduino Uno

The MG995 is a metal-geared, high-torque servo with a rotation range of about 0° to 180°. In this bot, it controls the front wheels within a restricted range of angles to steer left or right, ensuring both stability and precision.

How It Works:

- **Left turn:** The servo shifts the front wheels left by rotating to a negative steering angle.
- **Right turn:** The servo shifts the front wheels right by rotating to a positive steering angle.
- **Straight movement:** When the servo is in its neutral (centered) position, the front wheels remain aligned forward, enabling straight movement.

Control Logic:

The robot's **Camera 1** is managed by the Raspberry Pi, which performs color-based detection. When a condition is triggered (e.g., detecting green for left or red for right), the Pi sends a variable to the Arduino Uno. The Arduino interprets this variable and outputs the corresponding PWM signal to the MG995 servo, adjusting the steering angle of the front wheels accordingly.

Components

Component	Function	Power Requirements
RS-365PH Motor	Drives the two rear wheels for movement (forward/back ward).	6V - 24V
Single Axle Drive	Allows rear wheels to rotate at different speeds during turns.	N/A (Mechanical)
Tower Pro MG995 Servo	Controls the angle of the front wheels for steering.	4.8V - 7.2V
Motor Driver	Controls the DC motor's speed and direction.	Motor Side: 11.1V; Logic is 5V

Engineering Principles

Mechanical Engineering Principles

1.1 Single Axle Drive with 1:1 Gear Ratio (Kinematics)

The propulsion system is based on a **single axle drive with a 1:1 gear ratio**, ensuring that both rear wheels rotate at the same speed. This setup simplifies the kinematics of motion, as speed and direction are uniformly applied across the axle. With this arrangement, the bot is optimized for consistent, straight-line motion.

- **Straight-Line Motion:** Since the rear wheels rotate at equal speeds, forward and backward movement is stable and predictable. The 1:1 gear ratio prevents slippage between wheels, maintaining uniform motion.

- **Turning Radius:** Unlike a differential system, the bot cannot vary the rear wheel speeds independently. As a result, the turning radius is entirely governed by the **front-wheel steering**, controlled by the MG995 servo motor. A greater steering angle produces a tighter turning radius, which is essential for maneuvering around obstacles or sharp corners on the track.

1.2 Load Distribution and Torque (Statics and Dynamics)

In this design, **torque is generated by the RS-366PH dual-shaft motor** and transmitted to the rear axle, while stability depends on effective load distribution.

- **Torque and Traction:** The RS-366PH provides sufficient torque to drive the bot forward. Proper weight distribution ensures that the rear wheels maintain traction with the track, preventing slippage and maximizing efficiency.

- **Stability During Turns:** Turning dynamics are influenced by the bot's **center of mass, wheel placement, and weight distribution**. The **Tower Pro MG995 servo**, with its high-torque capability, provides the necessary force to adjust the steering angle without compromising stability during sharp turns.

Electrical Engineering Principles

2.1 Pulse Width Modulation (PWM) for Speed and Steering Control

Both the **RS-366PH motor** and the **MG995 servo** are controlled using **Pulse Width Modulation (PWM)** signals, which regulate power delivery by varying the duty cycle of electrical pulses.

- **Motor Speed Regulation:** By modulating PWM signals sent to the RS-366PH motor driver, the bot's propulsion speed can be dynamically adjusted. This allows acceleration on straight tracks and deceleration when approaching obstacles or curves, ensuring both efficiency and safety.

- **Servo Angle Control:** The MG995 servo receives PWM signals to control its angular position. Depending on input from the vision system, the servo adjusts the front-wheel angle—for example, steering left when detecting green or right when detecting red—thereby guiding the bot accurately through the track.

Software Engineering Principles

3.1 Sensor Data Integration and Decision-Making (Pi–Arduino Communication)

The navigation system relies on **real-time sensor fusion** between the Raspberry Pi and Arduino Uno. Camera 1, controlled by the Pi, performs color detection to identify track markers and blocks. When the Pi detects a specific condition—such as an **orange block**—it sends a variable through serial communication to the Arduino. The Arduino then executes the corresponding action by controlling the **RS-366PH dual-shaft motor** for propulsion and the **MG995 servo** for steering.

- **Obstacle Detection and Response:** Ultrasonic sensors mounted on the left and right sides measure distances to nearby obstacles. If an obstacle is within a set threshold, the Arduino triggers corrective steering adjustments using the MG995 servo to avoid collisions.

- **Color-Based Commands:** The Pi translates visual data into commands (e.g., `'R'` for red → right turn, `'L'` for green → left turn, `'O'` for orange → special maneuver). These commands are transmitted via serial to the Arduino, which executes them after checking a cooldown timer to prevent rapid conflicting inputs.

- **Data Filtering:** To ensure stability, the system uses timed checks and ignores redundant or noisy signals. This helps maintain smooth control, especially on tracks with frequent color markers or environmental disturbances.

3.2 Feedback Control System

The bot employs a **closed-loop feedback control strategy** to constantly adjust its motion based on sensor inputs. This ensures reliable navigation and adaptability to dynamic environments.

- **Closed-Loop Control:** Sensor readings (ultrasonic distances, Pi color detection) are continuously fed into the Arduino's control loop. For example, if the ultrasonic sensor detects a nearby obstacle, the bot slows down or adjusts steering until the path is clear.

- **Command Execution:** When the Pi detects an orange block and sends the `'0'` command, the Arduino interprets this and modifies the servo/motor behavior accordingly. Cooldown logic ensures that each action completes before another command is executed, preventing instability.

Systems Engineering Principles

5.1 Systems Integration

Systems engineering principles ensure that power delivery, motor control, sensor feedback, and decision-making operate as a unified system.

- **Component Interfacing:** Reliable communication between the **Raspberry Pi, Arduino Uno, ultrasonic sensors, cameras, MG995 servo, and RS-366PH motor** is critical. The Pi handles vision-based color detection and transmits commands to the Arduino, which in turn manages actuation through PWM signals. Continuous data exchange between these subsystems ensures smooth navigation.

- **Modularity:** Each subsystem—**propulsion (RS-366PH motor), steering (MG995 servo), sensing (cameras + ultrasonics), and power regulation (buck converters)**—is developed as a modular unit. This approach simplifies debugging, testing, and future upgrades, as improvements can be made to one subsystem without disrupting the functionality of the entire bot.

Power and Sense Management

Power

1.1 Power Source

The main power source for the bot is a 3S 11.1V LiPo battery. This battery provides the necessary power to all sensors and the Arduino Uno, which acts as the main controller for the system. LiPo batteries are known for their high energy density, providing ample power for mobile robotic systems.

Battery specs:

- **Voltage:** 11.1V
- **Current:** 800mAH
- **Configuration:** 3 cells in series (3S)
- **Battery type:** LiPo (Lithium Polymer)

Sensors

This section outlines all the sensors integrated into the 4-wheel bot, their functions, and power requirements.

2.1 Ultrasonic Sensors (HC-SR04)

Three HC-SR04 ultrasonic sensors are mounted on the bot for obstacle detection and distance measurement.

Power Specs:

- Operating Voltage: 5V DC
- Operating Current: 15mA
- Working Frequency: 40kHz

Sensor Roles:

- **Front Ultrasonic Sensor:** Detects the distance to obstacles directly ahead, preventing frontal collisions.
- **Left Ultrasonic Sensor:** Monitors proximity to obstacles on the left side, ensuring the bot does not drift too close to barriers.
- **Right Ultrasonic Sensor:** Tracks distance on the right side, preventing collisions from that direction.

These sensors provide essential data for **collision avoidance and path planning**.

2.3 Webcam (Generic USB Camera)

A generic USB webcam is used for **vision-based navigation and obstacle recognition**. Unlike ultrasonic sensors, the webcam processes visual data to identify colored markers or objects along the track.

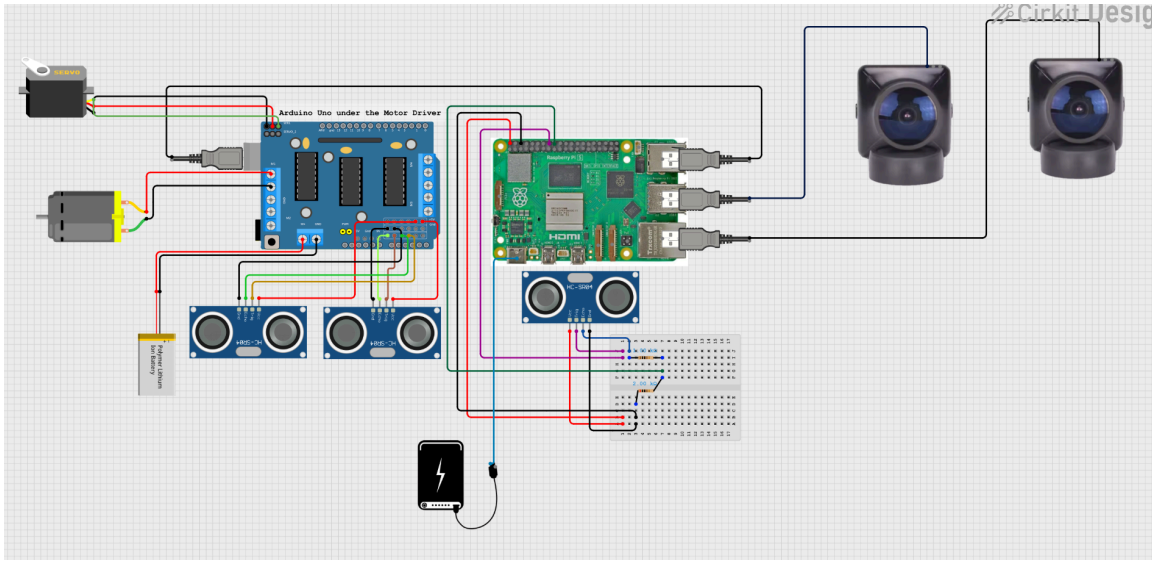
Power Specs:

- Operating Voltage: 5V DC (via USB)
- Current: ~120–200mA (depending on model)

Sensor Roles:

- The webcam captures live video and streams it to the Raspberry Pi, where image processing algorithms identify **color-coded obstacles**.
- **Red marker/object:** Signals the bot to turn right.
- **Green marker/object:** Signals the bot to turn left.

The camera, in coordination with the Raspberry Pi and Arduino, enables **real-time decision-making for navigation and obstacle avoidance**.


Arduino Uno under the Motor Driver
HDMI