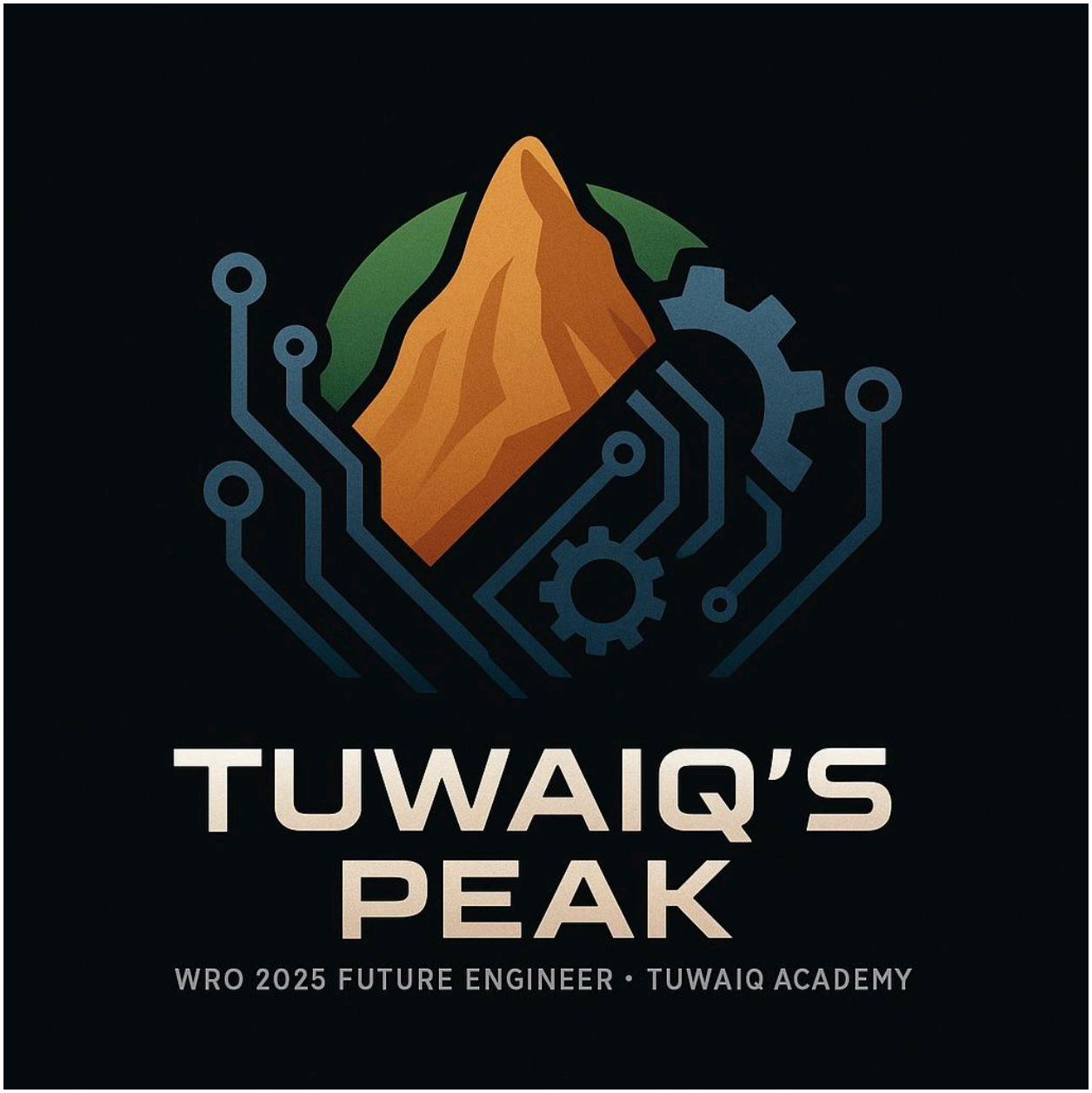
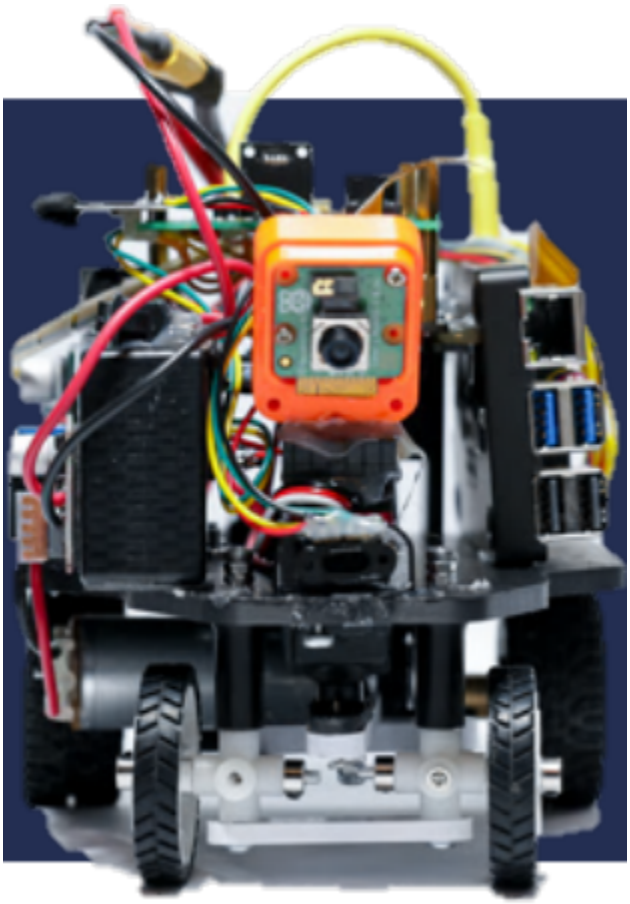
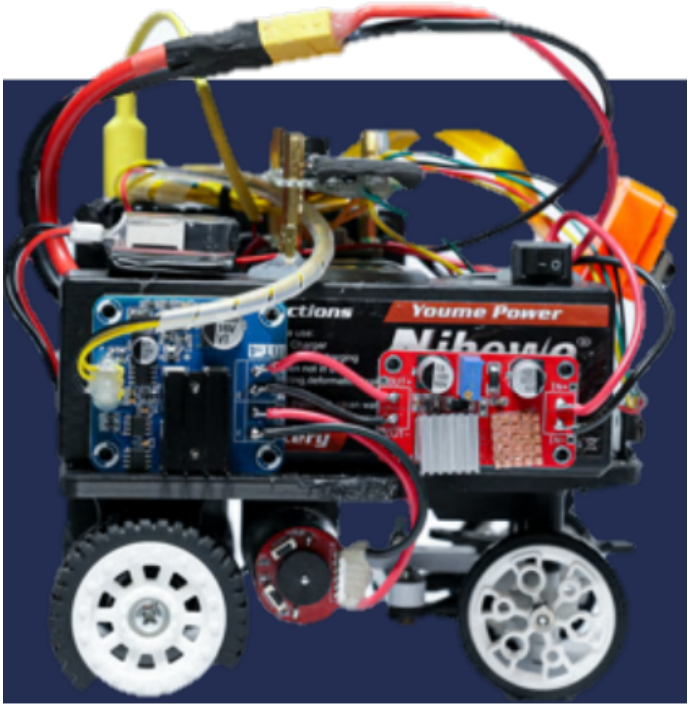
Tuwaiq’s Peak
#3262



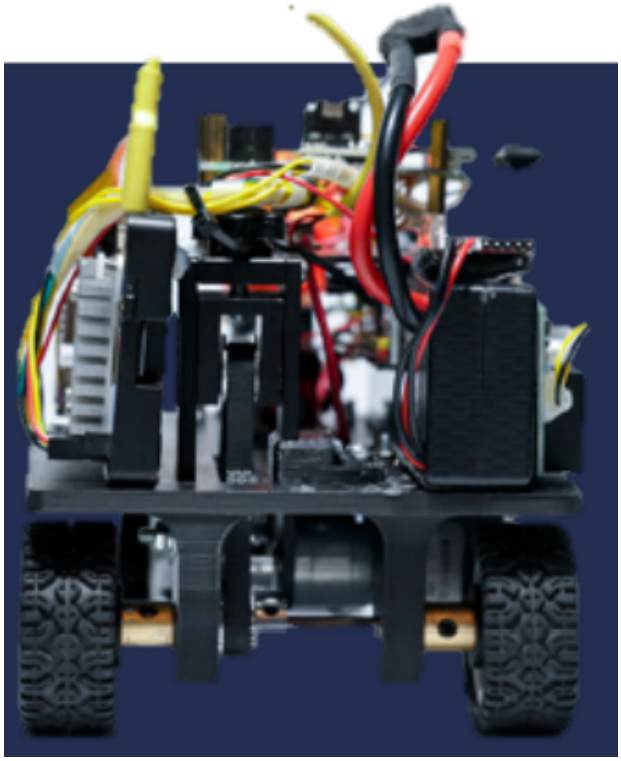
Robot Pictures



Front

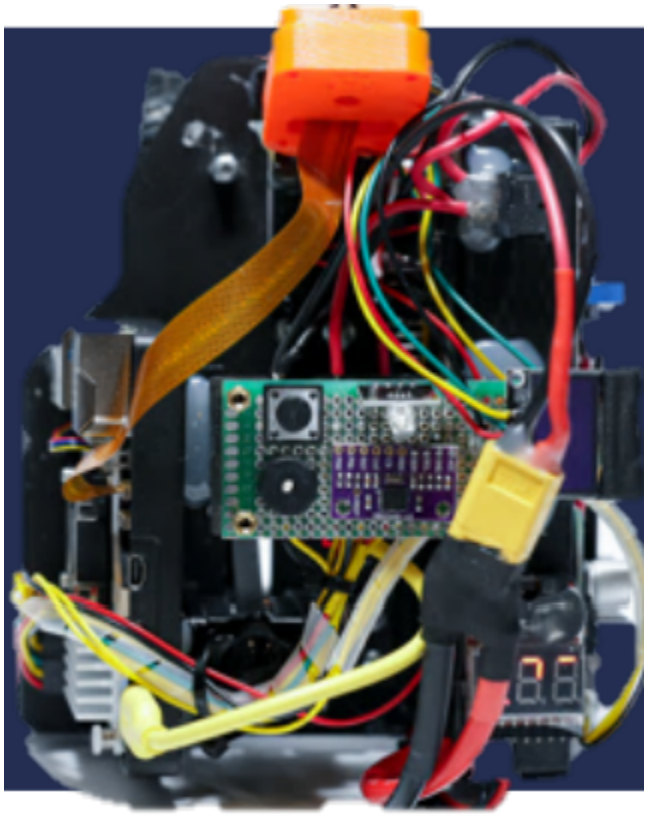


Right



Back

Bottom



Top

Left

Overview

The creation of an autonomous robot for the WRO Future Engineers Challenge is an ambitious project that blends advanced technology, innovative engineering, and strategic problem-solving. The goal was to design a high-performance, adaptable robot capable of operating in complex environments, addressing real-world challenges, and excelling in competitive conditions.

At the heart of the system lies the **Raspberry Pi 5**, which serves as the robot's central processor, handling real-time computations and decision-making. Supported by the **Camera Module 3** and **OpenCV-based vision algorithms**, the robot can detect obstacles, navigate dynamically, and perform precision maneuvers such as automated parking. Additional components, including **high-torque motors**, a **gyroscope**, and **ultrasonic sensors**, contribute to its stability, responsiveness, and accuracy—ensuring dependable performance across a variety of tasks.

This project adopts a **multidisciplinary approach**, combining principles of mechanical engineering, electronics, and software development. Through iterative design, prototyping, and testing, each subsystem was refined for optimal performance and smooth integration. Key design choices—such as component selection, chassis construction, and power management—were made based on thorough experimentation and data-driven evaluation.

Beyond competition readiness, the project aims to advance the field of robotics by demonstrating innovative solutions in design, implementation, and validation. This report highlights the development process, challenges, and accomplishments that led to a robust, competition-ready robot embodying both precision engineering and creative innovation.

Team Photos

Ibrahim Alangari
3D Designer, Hardware

Reda Albin Ahmed
Strategic Designer,Coder

Robot's Components

Electrical Components:

Raspberry Pi 5:

The **Raspberry Pi 5** is a single-board computer (SBC) that serves as the main controller and processing unit of our robot.
It combines powerful computing performance with a compact design, enabling it to handle real-time tasks such as motor control, sensor data processing, and computer-vision analysis simultaneously.
The board runs Raspberry Pi OS (based on 64-bit) and communicates with other electronic components—such as the PCA9685 servo driver, BTS7960 motor driver, and various sensors—through its GPIO, I²C, and SPI interfaces.

Specification	Value / Description
Processor (CPU)	64-bit Quad-core ARM Cortex-A76 @ 2.4 GHz
GPU	VideoCore VII (supports dual 4K displays)
Memory (RAM)	8 GB LPDDR4X

Storage	microSD card 32GB
Networking	Gigabit Ethernet, Wi-Fi 5, Bluetooth 5.0
USB Ports	(2x)USB 3.0 + (2x)USB 2.0
GPIO Header	40 pins
Camera Support	(2x) CSI-2 ports (for Camera Module 3)
Display Support	(2x) micro-HDMI (4K 60 fps each)
Power Input	USB-C 5 V / 5 A recommended
Operating System	Raspberry Pi OS (based on 64-bit)
Dimensions	85.6 mm × 56.5 mm × 17 mm
Weight	around 45-50 grams

Advantages

- **High performance:** Fast CPU/GPU enables AI and computer vision processing.
- **Multiple interfaces:** (GPIO, I²C, etc...) for connecting sensors and drivers.

- **Supports camera and display modules:** Ideal for vision-based robotics.
- **Full computer environment:** Runs Linux and Python libraries (OpenCV, TensorFlow, etc.).
- **Compact and energy-efficient:** Fits easily on small robots.
- **Active community:** Excellent documentation and online support.

Disadvantages

- **Needs adequate cooling:** Can heat under heavy load; requires fan or heatsink.
- **Requires a stable 5 V / 5 A power source:** Voltage drops may cause restarts.
- **Longer boot time:** Takes several seconds to start OS.
- **More complex setup:** Requires software installation and configuration.
- **Higher cost:** More expensive than microcontrollers like Arduino.

2. DC-DC Step Down Converter

The DC-DC Step Down Converter is an essential part of our robot's power management system, responsible for maintaining stable voltage levels across different subsystems.

Model: Mini560 DC-DC Step Down Converter
Description:

- **Output Voltage:** Adjustable outputs of 3.3V, 5V, 9V, or 12V to meet the varied power needs of connected components.

- **Current Capacity:** Delivers up to 8A, ensuring sufficient power for high-load devices.

- **Voltage Regulation:** Efficiently steps down higher input voltages to stable, lower levels for reliable operation of sensitive electronics.

- **Compact Design:** Small and lightweight (7g), making it easy to integrate within the robot's limited space.

- **Efficiency:** High conversion efficiency reduces energy loss and heat buildup, improving overall system performance.

Component Role: Provides stable, regulated voltage to electronic subsystems, ensuring consistent and reliable power delivery throughout the robot.

3. Raspberry Pi Active Cooler:

Overview:
The Raspberry Pi Active Cooler is a dedicated cooling solution for the Raspberry Pi 5, designed to prevent thermal throttling during demanding operations such as image processing and AI computations. It ensures the CPU and GPU maintain optimal performance during prolonged use.

Key Specifications and Features:

- Cooling Type: Active fan-based cooling system

- Compatibility: Optimized for Raspberry Pi boards, including the Raspberry Pi 5

- Size: Compact form factor that mounts directly onto the board without obstructing GPIO access

- Power Source: Powered directly from the Raspberry Pi via GPIO pins

Advantages:

- Effectively prevents overheating, ensuring stable and consistent performance

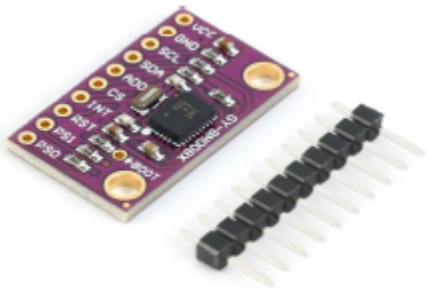
- Compact and easy to install

- Ideal for continuous, high-load tasks where thermal efficiency is critical

Disadvantages:

- Occupies GPIO pins for power, limiting their use for other components
- The fan may introduce slight noise during operation

4. Gyroscope (BNO085)



Overview:
The BNO085 is an advanced 9-axis motion sensor that integrates an accelerometer, gyroscope, and magnetometer into a single compact module. It delivers accurate motion tracking and orientation data, making it well-suited for robotics, navigation, and motion analysis applications.

Key Specifications and Features:

- **Sensor Type:** 9-axis (3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer)
- **Communication Interfaces:** Supports both I2C and SPI connections
- **Output Data:** Provides real-time measurements of acceleration, angular velocity, and magnetic field strength
- **Operating Voltage:** 3.3V–5V

Advantages:

- Integrates three sensing technologies in one module for precise 3D motion and orientation tracking
- High accuracy and minimal drift ensure dependable performance
- Compact and easy to incorporate into embedded systems or robotic designs

Disadvantages:

- Requires calibration for optimal accuracy
- Processing sensor fusion data may increase computational load

5. OLED Display

Overview:
The OLED display is a compact, high-contrast screen used to present system information or visual output. It's widely utilized in DIY and embedded electronics projects due to its vivid display quality and ease of integration.

Key Specifications and Features:

- **Display Size:** Approximately 0.96 inches
- **Resolution:** 128×64 pixels
- **Interface:** Supports I2C or SPI communication
- **Operating Voltage:** 3.3V–5V

Advantages:

- Excellent contrast and brightness, ensuring clear visibility in different lighting environments
- Energy-efficient, making it suitable for battery-powered applications
- Simple communication interface, easily compatible with controllers such as the Raspberry Pi

Disadvantages:

- Small display area limits the amount of information shown
- Viewing angles may be narrower compared to larger or more advanced display types

6. Push Button

Overview:
A momentary push button switch that generates a signal only while being pressed, commonly used for user input in embedded systems and IoT applications.

Key Specifications and Features:

- **Type:** Normally open (NO)
- **Operation:** Push-to-make mechanism

- **Design:** Compact and space-efficient
- **Durability:** Rated for millions of actuations

Advantages:

- Provides a simple and dependable method for user interaction
- Easy to implement and integrate into various electronic designs

Disadvantages:

- Limited to basic input functions such as on/off control
- Mechanical components may experience wear after extensive use

7. Buzzer

Overview:
The buzzer is an audio output device used to generate sound alerts or feedback in electronic systems. It's commonly implemented for alarms, status indicators, or user notifications.

Key Specifications and Features:

- **Type:** Active buzzer
- **Operating Voltage:** 3V–5V

- **Output:** Produces clear, audible tones

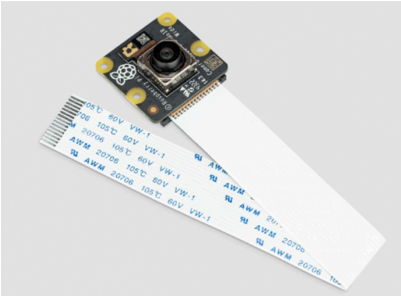
Advantages:

- Easy to connect and incorporate into electronic projects
- Delivers instant sound feedback for alerts or events

Disadvantages:

- May produce noise that's too loud or distracting in certain applications
- Limited to basic sound signaling without tonal variation

8. Raspberry Pi Camera Module 3 Wide



Overview:
The Raspberry Pi Camera Module 3 Wide is an enhanced camera designed for the Raspberry Pi, featuring a wide-angle lens and superior image quality. It's well-suited for applications that require extensive visual coverage, such as robotics, surveillance, and aerial imaging.

Key Specifications and Features:

- **Resolution:** 12 megapixels
- **Field of View:** Wide-angle for expanded scene capture
- **Video Performance:** Supports up to 1080p at 60 frames per second
- **Interface:** CSI (Camera Serial Interface) connection

- **Lens Type:** Fixed-focus wide-angle lens

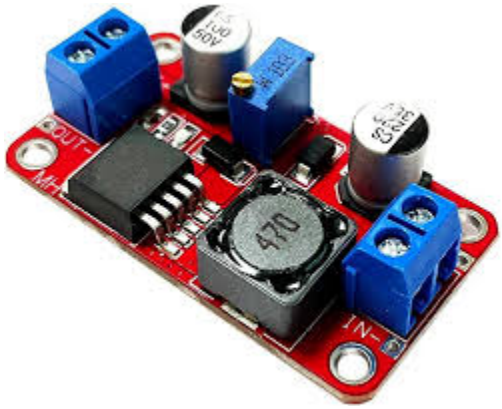
Advantages:

- Provides broad visual coverage, ideal for monitoring and navigation tasks
- Delivers high-resolution, detailed imagery
- Smooth video output at high frame rates for real-time applications

Disadvantages:

- Fixed focus restricts flexibility for close-range imaging
- Image quality may decrease in low-light environments without additional illumination

9. Step-Up Converter XL6019



Overview:
The XL6019 is a DC-DC boost converter designed to step up lower input voltages to higher, stable output levels. It's ideal for powering components that require more voltage than the input source can provide.

Key Specifications and Features:

- **Input Voltage Range:** 5V–32V

- **Output Voltage:** Adjustable from 5V to 35V
- **Current Capacity:** Up to 4A
- **Efficiency:** Reaches up to 90%

Advantages:

- Efficiently boosts voltage for high-demand electronic systems
- Adjustable output enables flexible application across various circuits

Disadvantages:

- Requires proper configuration to maintain stable output at high current levels
- May produce heat when operating under heavy load

10. RGB LED

Overview:
The RGB LED is a versatile light-emitting diode capable of producing various colors by combining red, green, and blue light. It's commonly used in electronics projects for visual indicators, lighting effects, or simple color displays.

Key Specifications and Features:

- **Size:** Standard 5mm package
- **Operating Voltage:** Approximately 3.2V–3.4V per color channel
- **Control Method:** Supports PWM (Pulse Width Modulation) for precise color blending

Advantages:

- Can generate a wide spectrum of colors through RGB mixing
- Compact and easy to implement in electronic designs
- Energy-efficient with low power consumption

Disadvantages:

- Offers limited color accuracy compared to full-color display modules
- Requires precise control signals to achieve consistent color output

11. Battery Nihewo 6500 mAh, 90C

Overview:
The Nihewo 6500 mAh 90C LiPo battery is a high-capacity power source built for demanding applications such as robotics, drones, and other high-performance electronic systems. It provides strong, consistent power delivery for components requiring high current output.

Key Specifications and Features:

- **Capacity:** 6500 mAh
- **Voltage:** 7.4 V
- **Discharge Rate:** 90C, enabling high current draw

Advantages:

- High discharge capability supports energy-intensive devices
- Large capacity allows for extended operating time

Disadvantages:

- Sensitive to improper charging and excessive heat
- Requires careful handling and maintenance to maintain safety and lifespan

12. Servo Motor max 40kg max torque

Overview:
This high-torque servo motor delivers up to 40 kg-cm of torque, making it ideal for demanding applications such as robotic arms, automation systems, and other mechanisms requiring strong and precise movement.

Key Specifications and Features:

- **Torque:** 40 kg-cm
- **Operating Voltage:** 6V–12V
- **Rotation Range:** 180°
- **Applications:** Suitable for robotics and mechanical systems needing accurate, powerful actuation

Advantages:

- Provides high torque output for lifting or moving heavy components with precision
- Well-suited for large-scale robotic or mechanical applications

Disadvantages:

- Bulkier and heavier than standard servos
- Higher power consumption due to increased torque capacity

13. Servo Motor Driver PCA9685PW

Overview:

The PCA9685PW is a servo motor driver capable of controlling up to 16 servos at once with

high precision. It's widely used in robotics, automation, and systems that require coordinated movement across multiple actuators.

Key Specifications and Features:

- **Channels:** Supports up to 16 servo outputs
- **Communication Interface:** I2C protocol
- **Control Method:** PWM (Pulse Width Modulation) for accurate servo positioning
- **Operating Voltage:** Typically 5V

Advantages:

- Simplifies multi-servo control with minimal wiring complexity
- I2C interface ensures easy integration with controllers such as the Raspberry Pi

Disadvantages:

- Channel limit of 16 may restrict larger-scale applications
- Systems with many servos may need extra power management to maintain stable operation

14. DC Motor with Encoder 620 RPM

Overview:
This DC motor with an integrated encoder enables precise control of rotation speed and direction, making it well-suited for robotics, automation systems, and other applications that require accurate motion feedback.

Key Specifications and Features:

- **Speed:** 500 RPM
- **Operating Voltage:** 12V (typical)
- **Encoder:** Provides real-time feedback for speed and position monitoring
- **Applications:** Commonly used in robotics, CNC systems, and industrial automation

Advantages:

- Encoder integration ensures accurate and stable control of speed and direction
- Delivers reliable performance across a wide range of motion control applications

Disadvantages:

- Requires external circuitry or a controller to interpret encoder signals
- Higher current consumption may increase power supply requirements

15. IBT-4 Arduino DC Motor Driver

Overview:
The IBT-4 motor driver is designed to control high-power DC motors, providing reliable performance for applications that demand higher voltage and current levels. It is commonly used in robotics, electric vehicles, and other systems requiring powerful motor control.

Key Specifications and Features:

- **Motor Voltage Range:** 5V–36V
- **Current Capacity:** Up to 43A peak
- **Interface:** Compatible with Arduino and various microcontrollers
- **Control Method:** PWM-based control for managing motor speed and direction

Advantages:

- Capable of driving large, high-torque motors efficiently
- Simple to integrate with popular microcontroller platforms

Disadvantages:

- Generates significant heat during heavy operation, requiring proper cooling
- High current handling necessitates careful wiring and safety precautions

16. TOF200C-VL53L0X

Overview:

The **TOF-200C-VL53L0X** is a **Time-of-Flight (ToF) distance sensor** that measures precise distances by calculating the time taken for a laser pulse to reflect off an object and return. Compact and highly accurate, it is widely used in robotics, automation, and object detection applications for real-time ranging and obstacle avoidance.

Key Specifications and Features:

- **Measurement Range:** Up to 200 cm (2 meters)
- **Accuracy:** ±3% under optimal lighting conditions
- **Interface:** I2C communication protocol
- **Operating Voltage:** 2.6V–5V
- **Field of View (FOV):** Approximately 25°
- **Response Time:** Typically 30–50 ms for real-time applications

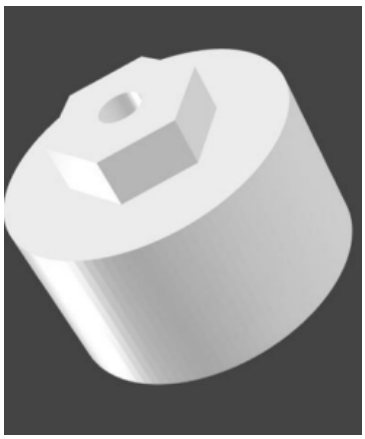
Advantages:

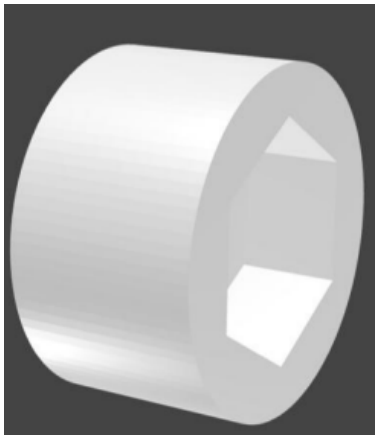
- Provides high-precision, real-time distance measurements
- Compact and lightweight, ideal for small robotic systems
- Low power consumption, suitable for continuous operation
- Immune to most ambient light interference

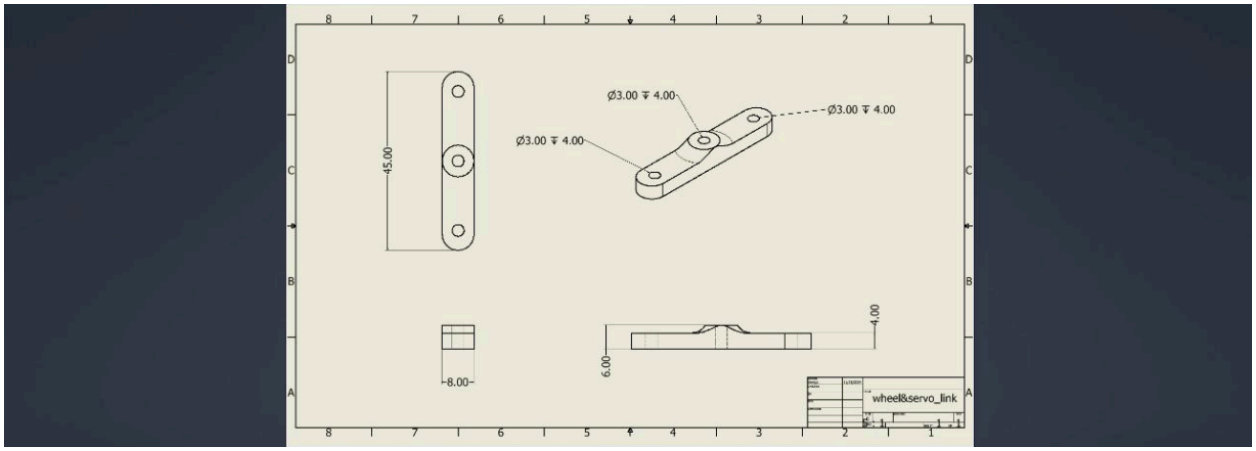
Disadvantages:

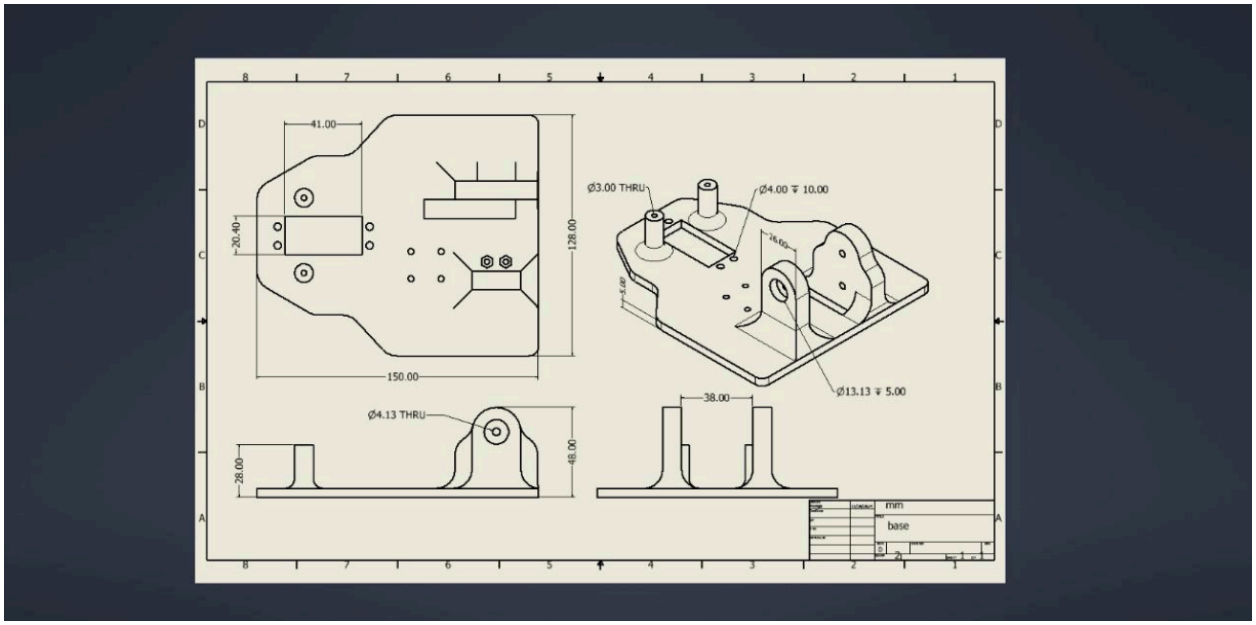
- Limited range compared to longer-distance LiDAR sensors
- Accuracy can be affected by highly reflective or dark surfaces
- Requires careful alignment for optimal measurement performance

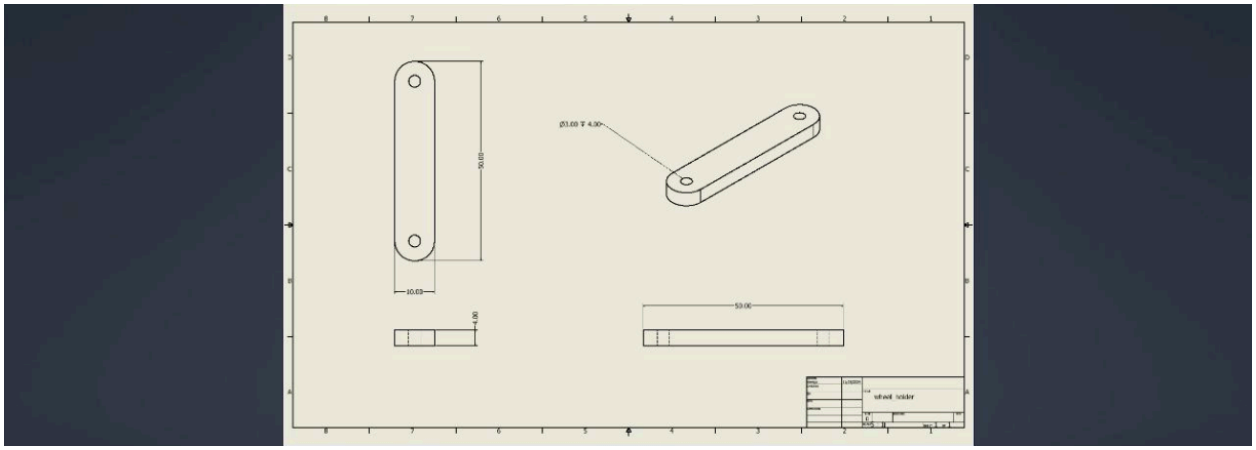
Mechanical Components

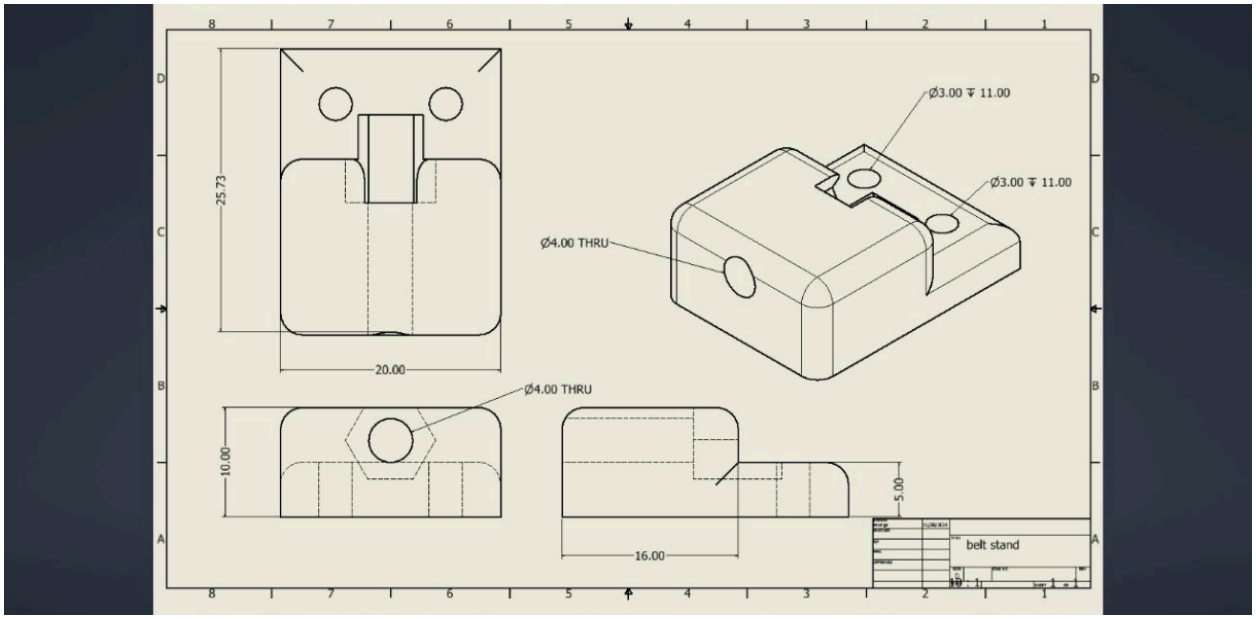












Mobility Management

The robot's drivetrain integrates **DC motors** to provide forward and reverse propulsion, while **servo motors** handle the steering mechanism for accurate directional control. The inclusion of **motor encoders** enables real-time feedback on speed and position, allowing for precise motion regulation and stable navigation. This coordinated system ensures smooth acceleration, controlled turning, and reliable maneuvering across various tasks and environments.

Designing the Mechanical Alignment:

1. The team started by envisioning the robot's mechanical structure, prioritizing optimal component placement to achieve proper balance and stability. Initial sketches were created to outline the design concept while studying the alignment control system—understanding its critical role in maintaining movement stability, steering accuracy at both low and high speeds, and minimizing vibration throughout operation.
2. **Key factors in alignment:**
    a. **Weight Distribution and Balance:**
    Key components such as the battery and Raspberry Pi were positioned near the center of the chassis to lower the center of gravity and enhance overall stability. Using a small digital scale, each part was individually weighed before final assembly to achieve proper weight balance. The total assembled weight of the robot is **1,387 grams**.
    b. **Motor Placement:**
    The DC motors and wheels were mounted symmetrically to ensure equal torque distribution and stable, straight-line motion. The rear wheels are interconnected using a **steel pillar** (specified size) to maintain rigidity and alignment.
    c. **Servo Integration:**
    The steering system was precisely aligned to allow smooth and controlled rotation within the servo motor's operational limits, effectively translating theoretical steering concepts into practical implementation based on referenced design principles.
3. **Iterative Design and Prototyping**

    The initial CAD model went through several refinement stages to resolve potential alignment challenges, including optimizing mounting points, joint placements, and screw hole locations for structural accuracy and ease of assembly.

    Alignment prototypes were produced using 3D printing to test component fit and mechanical functionality. Based on the results of physical testing, necessary adjustments were implemented to enhance precision and overall system performance.

4. **Precision Measurements**
    1. All dimensions were carefully measured to ensure precise component fitting. The front wheel assembly, secured by screws, spans 8 centimeters from left to right, with a central alignment point at 4 centimeters, while the rear wheel spacing measures 12 centimeters.

2. Design tolerances were intentionally included to permit minor adjustments during assembly, ensuring proper alignment without compromising the robot's structural strength.

Obstacle Management

The team adopted a systematic and data-driven strategy for obstacle navigation, integrating advanced computer vision algorithms, real-time sensor data, and optimized decision-making logic. The following outlines the approach used to handle obstacle detection and avoidance:

1. Strategy for Navigating Challenges

a. Detection with OpenCV:
The Raspberry Pi Camera Module 3 was used for real-time vision processing with OpenCV to identify object colors and shapes.

c. Real-Time Feedback:

- **Gyroscope:** Maintained orientation and corrected deviations after turns.
- **Encoders:** Monitored wheel rotations for accurate distance and angle control.

d. Testing and Refinement:
Extensive testing under varied lighting and obstacle conditions led to fine-tuning of vision algorithms for improved detection and navigation accuracy.

2. Flowcharts for Movement and Decision Logic

a. Purpose:
Flowcharts were created to visualize the robot's decision-making process, ensuring logical consistency and easier implementation. They outlined steps for obstacle navigation, movement decisions, and parking routines.

b. Key Components:

- Inputs: Camera data (color and position), gyroscope, and encoder feedback.

- Decisions:
    - Line detection → adjust course
- **Outputs:** Motor actions for movement, turning, or stopping.

c. Development:
The flowcharts were refined through several iterations to handle edge cases and shared across the team to support clear communication and effective debugging.

4. Testing and Iterative Refinement

a. Testing Scenarios:
The robot was evaluated under diverse conditions to assess its obstacle-handling performance, including courses with differently colored obstacles, low-light environments, and unexpected path changes such as sharp curves.

b. Feedback and Adjustments:
Color detection thresholds were refined to enhance vision accuracy, while gyroscope sensitivity and encoder calibration were adjusted to achieve smoother and more precise movement control.

Design & Prototyping

The robot's design and prototyping phase followed an iterative, structured process focused on refining subsystems and integrating them into a cohesive, high-performance design. Development began with defining the robot's overall structure and requirements, which were divided into smaller modules for targeted testing and improvement. This approach enabled early issue detection and continuous enhancement across prototypes.

Component selection was crucial—**the Raspberry Pi 5** served as the main controller, and **the Camera Module 3** provided the vision system essential for tasks like obstacle detection and navigation. These components were chosen for their performance, reliability, and compatibility, ensuring efficient image processing and real-time decision-making.

Initial prototypes faced wiring and layout challenges, which were resolved through organized cable management, modular connections, and improved structural layouts. These refinements enhanced reliability, simplified maintenance, and reduced assembly time.

Over successive iterations, the robot evolved from a basic prototype into a polished, competition-ready system. The team improved chassis stability, weight balance, and overall durability while optimizing motor, sensor, and power system integration.

This systematic design process resulted in a robust and adaptable robot—reliable for competition use and flexible for future upgrades.

1. Development of CAD Designs

a. Design Tools and Software

The team utilized professional CAD platforms such as **Fusion 360** and **SolidWorks** to create detailed models of the robot's components and structure. These tools enabled:

- Accurate 3D modeling of individual parts
- Simulation of mechanical interactions and stress points
- Realistic visualization of the complete robot assembly

b. Detailed Schematics

- **Component Models:** Each key part—such as the chassis, motor mounts, brackets, and wiring channels—was designed with precise measurements to ensure perfect alignment during assembly.
- **Layered Design:** The robot's structure was organized into multiple layers separating electronics, sensors, and motors to simplify maintenance and minimize interference.
- **Exploded Views:** Detailed exploded diagrams illustrated how each component fit together, supporting a clear understanding of the assembly process.

c. Iterative Design Process

The CAD models were continuously refined through multiple iterations based on testing feedback. Adjustments were made to improve fit, balance, and durability while resolving issues related to weight distribution and component interference. Simulations within the CAD environment helped predict system behavior and guided further design optimizations.

2. Structural Integrity and Stability

a. Reinforcement Features

High-stress areas, including motor mounts and load-bearing joints, were strengthened with extra material to withstand operational forces. Cross-beams and support brackets were strategically positioned to minimize flexing and maintain structural rigidity.

b. Weight Distribution

The design focused on placing heavier components—such as the battery and motors—close to the robot's center of gravity for better stability. Lightweight materials were used in non-critical sections to reduce total weight while preserving overall durability.

3. Assembly Instructions

a. Modular Design

The robot was engineered for modularity, enabling each subsystem—such as the drivetrain, electronics, and steering—to be assembled, removed, or replaced independently. This approach simplified maintenance and part replacement.

b. Step-by-Step Guide

Comprehensive assembly instructions were created, outlining:

- **Required Tools:** Including screwdrivers, Allen keys, and wrenches.
- **Fastening Details:** Guidance on securing parts with bolts, nuts, and clips.
- **Cable Management:** Wiring diagrams showing proper routing to avoid tangling or interference with moving parts.

c. Labeling and Markings

All CAD components were clearly labeled to correspond with physical parts. Alignment markings on the chassis and mounts ensured precise positioning during assembly, improving accuracy and consistency.

Coding

Code Explanation 1:1 – Autonomous Navigation System

This Python program controls an autonomous robot using a **Raspberry Pi**, **camera**, **motors**, **sensors**, and **servos** to navigate and avoid obstacles.

1. Libraries and Initialization
Modules like `cv2`, `numpy`, and `Picamera2` handle image processing and camera input. Custom modules manage motor, servo, and sensor control. The camera is configured with a set resolution for real-time video capture.

2. Steering and Sensor Logic
Gyroscope data maintains orientation, while encoder feedback ensures accurate movement. Steering angles are logged for performance tracking and debugging.

3. Stuck Detection
The `stuck_checker` class compares consecutive frames to detect if the robot is stationary. If frame similarity exceeds a threshold, recovery actions such as reversing or steering adjustments are triggered.

4. Main Logic
Motors, servos, and sensors are initialized. Each frame from the camera is processed to calculate steering offsets and track boundaries. Steering angles are smoothed using buffered values before being applied to the servo.

5. Obstacle Avoidance
When obstacles or "stuck" conditions are detected, the robot reverses or adjusts direction automatically. Motor speeds are dynamically modified to maintain control.

6. Lap Counting
Lap progress is monitored using wall points and gyroscope feedback to verify orientation. After completing a set number of laps, the robot stops automatically.

7. Recovery and Cleanup
Upon exit or error, the system halts motors, centers the servo, and safely releases all resources.

Key Highlights:

- Smooth steering using averaged servo angles

- Automatic recovery when stuck
- Steering data logging for analysis
- Lap tracking with gyroscope alignment

Code Explanation 1:2 – TinyML Obstacle Detection System

This Python script uses a **TinyML model** from **Edge Impulse** to detect obstacles in real time using the **Raspberry Pi Camera**.

1. Libraries:

- `cv2`, `numpy` – image and numerical processing
- `edge_impulse_linux.image` – runs the Edge Impulse model
- `Picamera2` – captures frames from the Raspberry Pi Camera

2. Obstacle Detection (get_obstical_cordinates):
This function detects and returns obstacle type and position.

- **Preprocessing:** Converts BGR → RGB, resizes, and crops the image for model input.
- **Model Inference:** The TinyML model identifies obstacles and returns bounding boxes.
- **Filtering:** Detections with confidence below **0.993** are ignored.
- **Coordinate Mapping:** Converts detection points back to real image coordinates and assigns class labels (e.g., green = 0, red = 1).

3. Main Function:

- Initializes the camera at reduced resolution for faster processing.
- Loads the `.eim` model using `ImageImpulseRunner`.

- Continuously captures frames, processes them through the model, and displays detected objects with labels and confidence scores.

- Pressing **'q'** ends the detection loop safely.

4. Key Features:

- **Real-time inference** optimized for Raspberry Pi

- **High accuracy** through confidence-based filtering

- **Visual debugging** with bounding boxes

- **Scalable** to handle multiple detections per frame

5. Application:
Ideal for autonomous robots that identify colored obstacles and react accordingly. Can be expanded by integrating with motor control, tuning inference speed, or logging detections for analysis.

Code Explanation 1:3 – Real-Time Path and Obstacle Detection System

This Python script enables **real-time computer vision** for path tracking and obstacle detection, designed for **autonomous navigation** using a Raspberry Pi Camera.

1. Core Components

- **Libraries:** `cv2`, `numpy` (image processing), `json`, `os` (file handling), and `Picamera2` (camera interface).

- **Color Ranges:** Predefined HSV thresholds for detecting key colors like black, green, and red, used to identify the track and obstacles.

2. HSV Configuration

- **load_hsv_from_json():** Loads HSV color ranges from a JSON file, allowing easy recalibration without modifying the code.

3. Detection Logic

- **detect_color_contours():** Segments the image by color, reduces noise, and finds contours matching target regions.
- **generate_path():** Determines the track's centerline by locating white pixel midpoints.
- **filter_path_points():** Smooths path data with a moving average and curve fitting to minimize noise.
- **find_bottommost_contour():** Detects the nearest obstacle at the bottom of the frame.

4. Frame Processing (process_frame):

- Converts images to HSV and grayscale formats.
- Defines **Regions of Interest (ROI)** for focused path and obstacle detection.
- Detects red/green obstacles and calculates steering offset (`offset_steer`) based on the track's position.
- Visualizes paths and detected objects for debugging.

5. Main Function:

- Initializes the camera at reduced resolution for faster processing.
- Continuously captures frames, processes them, and displays results with overlays.
- Adjusts ROI dynamically based on detected contours.
- Ends safely on pressing **'q'**.

6. Key Features:

- Adjustable HSV settings via JSON file.
- Accurate color-based path and obstacle detection.
- Smoothed path estimation for stable navigation.
- Real-time visualization and adaptive ROI for varying environments.

7. Applications & Improvements:
Ideal for **autonomous robots**, **drones**, or **vehicle guidance systems**. Future upgrades could include faster parallel processing, obstacle prioritization, and automatic HSV calibration for changing light conditions.

Explanation of the Servo Control Script

This Python script controls servo motors through the **Adafruit PCA9685 PWM Driver** via the **I2C interface**. It defines a `ServoController` class and a helper function `create_servo`, offering precise and flexible servo control for robotics and automation applications.

1. Key Components

- **Libraries:**
    - `adafruit_pca9685` – manages the PCA9685 servo driver.
    - `busio`, `board.SCL`, `board.SDA` – set up I2C communication.
- **ServoController Class:** Handles initialization, angle control, and movement.
    - Attributes: I2C setup, servo channel, pulse range (`min_pulse`, `max_pulse`), reversal factor, and centering offset.
    - **Methods:**

    - `set_angle(angle)`: Converts an angle (–90° to 90°) to a PWM pulse width.
    - `sweep(start, end, step, delay)`: Moves the servo gradually across a range.
    - `center()`: Returns servo to neutral (0°).
    - `unlock()`: Frees the I2C bus for other devices.
- **Helper Function (`create_servo`)**: Simplifies setup with customizable parameters like channel, pulse range, and reverse direction.

2. Features

- **Precision:** Accurate angle-to-pulse mapping for smooth motion.
- **Flexibility:** Supports reversed orientation and adjustable pulse limits.
- **Automation:** Sweep function enables scanning and calibration tasks.
- **Reusability:** Modular structure allows easy integration into larger systems.

Example Use Case:

```
servo = create_servo(channel=0)

servo.center()

servo.sweep(start=-90, end=90, step=10, delay=0.5)

servo.set_angle(45)

servo.unlock()
```

Applications:

- **Robotics:** Arm or wheel control.
- **Automation:** Positioning systems (e.g., blinds, valves).
- **Education:** Teaching servo motion and control concepts.

This script provides a **robust, modular, and precise** foundation for servo control in real-world robotics and automation projects.

Explanation of the BNO085 Sensor Script

This Python script manages the **BNO085 9-DOF IMU sensor**, handling initialization, data acquisition, and orientation tracking via I2C communication. It's designed for robotics, navigation, and motion-tracking applications.

1. Core Components

Libraries:

- `busio`, `board` – set up I2C communication on the Raspberry Pi.
- `adafruit_bno08x` – interfaces with the BNO085 sensor.
- `math`, `numpy` – for trigonometric and vector calculations.
- `time` – for timing operations.

Classes and Functions:

- **GyroError:** Custom exception for gyroscope issues.
- **BNO085Sensor:** Main class for managing sensor functions.
- **creat_bno_sensor():** Factory function returning a configured sensor instance.

2. BNO085Sensor Class

Attributes:

- I2C address (`0x4B`), sensor instance, and angle tracking variables (roll, pitch, yaw).
- Time and previous readings for accurate angle integration.

Initialization:

- Configures accelerometer, gyroscope, magnetometer, and rotation vector.
- Scans for I2C devices if initialization fails.

Sensor Configuration:

- **enable_features():** Activates motion and orientation sensors.
- **reset_sensor():** Resets the IMU state.

Data Processing:

- **quat_to_euler():** Converts quaternion data into Euler angles.
- **update_angles_from_gyro():** Integrates gyroscope data with a complementary filter to reduce noise.

3. Data Retrieval

get_sensor_data():
Reads and returns sensor outputs, including:

- **Acceleration (m/s²)**
- **Angular velocity (rad/s)**
- **Magnetic field (µT)**

- **Euler angles (°)**
- **Integrated gyro angles**

4. Real-Time Operation

The script continuously reads sensor data, printing the **yaw angle** (Z-axis rotation) while keeping it normalized between **–180° and 180°**.

5. Key Features

- **Automatic Initialization:** Scans for connected I2C devices.
- **Accurate Orientation Tracking:** Combines gyroscope and quaternion data.
- **Noise Reduction:** Uses a complementary filter for smoother angles.
- **Modular Design:** Easily reused across different projects.

6. Example Usage

```
if __name__ == "__main__":
    sensor = creat_bno_sensor()
    try:
        while True:
            data = sensor.get_sensor_data()
            if data:
                yaw = data['euler'][2]
                print(f"Yaw: {yaw:.2f}°")
            time.sleep(0.1)
```

```
except KeyboardInterrupt:
    print("Exiting...")
    sensor.close()
```

Applications:

- **Robotics:** Direction and heading control.
- **Drones:** Stabilization and flight orientation.
- **IoT:** Motion and position tracking.

7. Error Handling & Testing

- Detects initialization or I2C communication errors and lists available devices for debugging.
- Underwent systematic performance testing focused on stability, precision, and consistency—ensuring the sensor met competition-level reliability.

This script provides a **robust, modular, and accurate** foundation for integrating the BNO085 into autonomous systems requiring precise motion and orientation tracking.

Testing Procedures and Metrics

Speed Evaluation:
The robot's speed was tested across different surfaces to ensure efficient navigation without losing stability. Consistent performance was verified through multiple terrain trials.

Obstacle Detection Accuracy:
Using the **Raspberry Pi Camera Module 3** and **OpenCV**, the robot was tested with various obstacle colors, shapes, and lighting conditions. Its ability to distinguish red and green blocks and respond correctly was closely evaluated.

Adjustments and Improvements:

- **Vision System Optimization:** Camera resolution and FPS were increased for sharper, faster image processing, improving detection accuracy in complex conditions.

- **Obstacle Avoidance Refinement:** Decision logic was enhanced through iterative testing to handle overlapping or partially visible obstacles, ensuring reliable autonomous navigation.

Acknowledgements

We would like to express our deepest gratitude to all those who supported our project. A special thanks to our mentors, whose guidance and expertise were invaluable. Additionally, we extend our appreciation to Tuwaiq Academy for providing us with the resources and opportunities to grow.

Github

https://github.com/Tuwaiqs-Peak-WRO/WRO-Future-Engineers-2025-Tuwaiq-s-Peak/tree/main

Open challenge