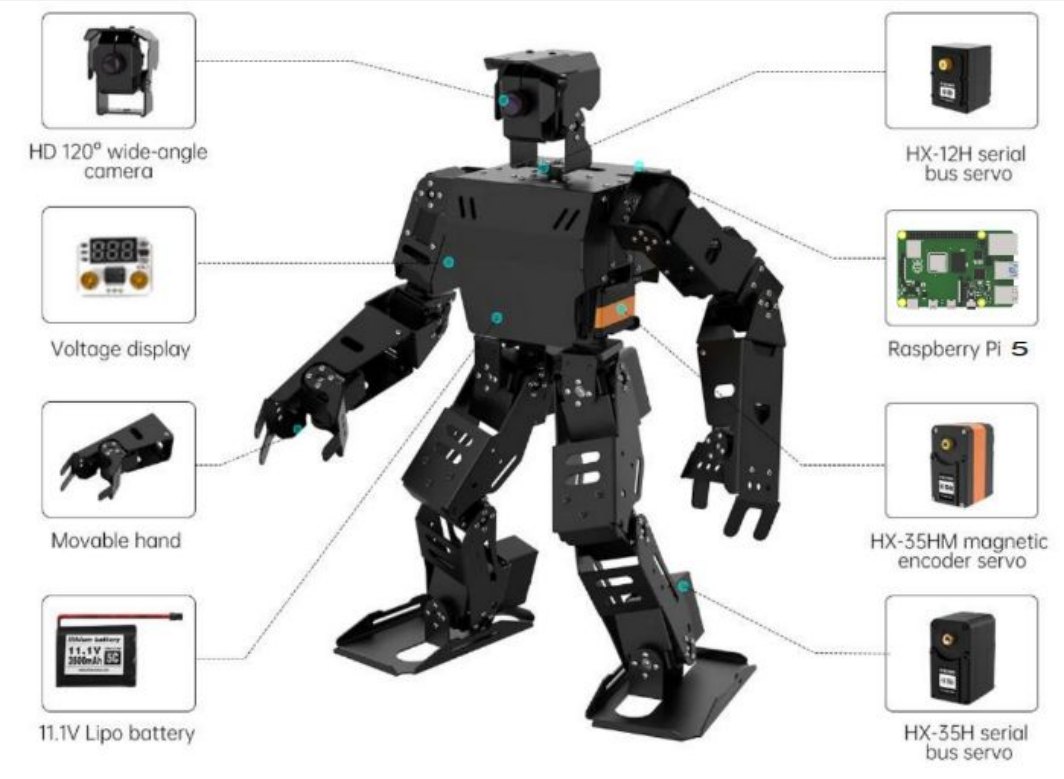
Ainex Competition Training

Presenter : syahmi

Ainex robot introduction



Network Configuration

Set AiNex to Client Mode (Wi-Fi)

Follow these steps to switch AiNex from AP mode to Client mode:

- Edit the config files to enable **Client (LAN) Mode**
- Enter and save your **Wi-Fi SSID and password**
- Set a **static IP address**
- Restart the **Wi-Fi service**
- Reconnect using **VNC with the new IP**

Step by step on how to switch to client mode

https://github.com/syahmisanab/marker/tree/main/script/network_configuration

ROS

ROS (Robot Operating System) is an open-source framework that helps robots:

- **Think** – process data, make decision
- **Communicate** – share info between parts (sensors, motors, vision)
- **Run code in modules** – reusable blocks called **nodes**

🔍 **Why ROS for AiNex?**

- Built-in tools for:
  - Motion control
  - Camera streaming
  - Sensor data processing

- Makes complex behaviors like **line following** or **arrow detection** easier to manage
- Used in **real-world robotics**, from research to industry

ROS concept

- Nodes
- Messages and Topics
- Services
- ROS Master
- Parameters
- Stacks and packages

ROS basic command

- roscore
- rosnode
- rosrun and roslaunch
- rostopic
- rqt

Python basic (ros)

- Running Python Scripts
- Reading & Modifying Code
- Control Logic
- Variables & Data
- Functions
- Imports & Libraries

Linux system

- File Navigation
- File management
- Script
- Permissions
- Editing & Viewing
- Terminal Shortcuts
- System command

Docker Basics (for AiNex)

What is Docker?

- A lightweight virtual container
- Runs the Ainex software in a controlled, isolated environment
- Think of it like a pre-packaged robot brain you can turn on/off

Why?

- No Need to Install Ubuntu on Raspberry Pi
- Self-Contained ROS Environment
- Separates ROS from Host OS

Ainex controller

The **AiNex Controller** is a PC software tool used to:

- Control and calibrate servos in real time
- Design and edit action groups (e.g., walking, waving)
- Test motion sequences before using them in actual scripts
- Adjust servo positions and deviations with sliders or manual input
- Save and load motion files used in walk_ready, stand, etc.

Gait Control Code (Sprint Movement)

- Uses GaitManager to command the robot to **walk forward, reverse, and turn**
- Adjust movement using parameters:
  - x → forward/backward speed
  - y → lateral movement (not used here)
  - yaw → rotation (turning left/right)

- Includes safe start/stop of gait

Step by step on how to run code with explanation

https://github.com/syahmisanab/marker/tree/main/hurocup/Gait_Control

Sprint game Code

- Controls a full movement cycle: **walk forward → reverse**
- Built using GaitManager with timed actions
- Code structure is simple and editable:
  - Change **speed** (x) and **duration** (time.sleep())
- Uses **OpenCV to detect a line**, then triggers reverse action

Step by step on how to run code with explanation

https://github.com/syahmisanab/marker/tree/main/hurocup/Sprint_game

Line Following – Visual Patrol

- Uses **OpenCV color detection** to track a black line on the ground
- Defines **Regions of Interest (ROIs)** to focus on specific parts of the camera feed
- Processes frames to detect the line's position and adjust movement
- Script can be **tuned for different lighting and surfaces** by adjusting thresholds

Step by step on how to run code with explanation

https://github.com/syahmisanab/marker/tree/main/hurocup/Line_Following

Arrow detection Node

- Uses **OpenCV** to recognize arrow shapes: **left**, **right**, and **forward**
- Applies **7- or 9-point contour logic** to distinguish arrow direction based on shape
- Publishes results to the ROS topic:
    - Other scripts can subscribe and react to arrow data

Step by step on how to run code with explanation

https://github.com/syahmisanab/marker/tree/main/hurocup/Arrow_Detection

Marathon game

- Combines two key systems
- **Line Following → Arrow Detection → Movement Decision**
- Robot follows the line, **detects arrow direction**, and **turns or walks forward**

Logic includes:

- Turn left/right based on /arrow/shape_direction
- Keep following if arrow points forward

Full mission script to test complete game flow from start to finish

Step by step on how to run code with explanation

https://github.com/syahmisanab/marker/tree/main/hurocup/Marathon_game