Advanced Path-Programmable Line Following Robot

(Suggestion: Replace the placeholder above with a cool picture or animated GIF of your robot in action!)

1. Project Overview

This repository contains the complete firmware for an advanced, highly configurable Line Following Robot (LFR). Designed by a robotics enthusiast with a background in Mechanical Engineering, this project goes beyond simple line following. It features a sophisticated, path-based navigation system that allows it to solve complex tracks with intersections, gaps, inverse lines, and obstacles.

The core of the project is its user-friendly interface, which uses an onboard OLED display and a rotary encoder. This allows for real-time tuning of all performance parameters without needing to re-upload code, making it a powerful platform for learning and competitive events.

Key Features:

- **On-the-Fly Configuration:** A full menu system on an OLED display allows for real-time adjustment of speed, turning, sensor thresholds, and more.
- **Path-Programmable Navigation:** The robot's behavior is dictated by a path[] array, allowing it to execute a pre-programmed sequence of turns and actions to solve any track.
- **Multi-Sensor Fusion:** Utilizes a 6-channel analog IR array for precise line tracking and three ultrasonic sensors for obstacle avoidance and wall following.
- **Advanced Maneuvers:** Capable of handling complex track elements including 90-degree turns, T-junctions, cross-intersections, line gaps, and U-turns.
- **Inverse Line Detection:** Can dynamically switch between following a black line on a white surface and a white line on a black surface.
- **Persistent Memory:** All tuning parameters and path data are saved to the Arduino's EEPROM, so settings are retained even after power loss.
- **Modular Codebase:** The firmware is logically separated into multiple files for easy maintenance and future expansion.

2. Hardware Components

Component	Quantity	Purpose
Microcontroller	Arduino Nano	1

Display	1.3" I2C OLED (SH1106)	1
Motor Driver	L293D	1
Motors	DC Gear Motor	2
Caster	Ball Caster	1
Sensors	6-Channel Analog IR Array	1
	HC-SR04 Ultrasonic Sensor	3
Power	12V Battery Pack	1
	Buck Converter	1
Input	Rotary Encoder with Push-button	1
Safety	Kill Switch	1

3. Firmware Structure

The Arduino code is organized into several logical files for clarity and maintainability:

- **LFR_Advanced.ino**: The main file containing setup() and loop().
- **UI_OLED.ino**: Manages all functions related to the OLED display and menu system.
- **Navigation.ino**: Contains the core line-following and path-execution logic.
- **Sensors.ino**: Handles all sensor reading, processing, and calibration.
- **Hardware_Control.ino**: Contains low-level functions for motor control and input reading.
- **EEPROM_Utils.ino**: Manages saving and loading data to and from the EEPROM.

4. How to Use the Robot

The robot is operated entirely through the onboard menu system.

1. **Power On:** Turn on the robot using the main power and kill switches. The "TechTopia" splash screen will appear.
2. **Enter Menu:** Give the rotary encoder's button a **short press** to enter the main menu.
3. **Navigate:** Turn the rotary encoder knob to scroll through the menu options.
4. **Select:** A **short press** on the knob selects the highlighted menu item.
5. **Go Back / Exit:** A **long press** (hold for >1 second) will exit the current submenu or

return to the splash screen.

Essential First Steps:

1. **Calibrate Sensors:** Navigate to 5) Calibration and select it. The robot will rotate to scan the track surface and automatically determine the thresholds for black and white. This is **essential** for reliable performance.
2. **Set Path (Optional):** Go to 4) Path Adjust to program a specific sequence of actions for the robot to take at intersections.
3. **Adjust Parameters:** Go to 3) Adjustment to fine-tune speed, turning behavior, and timers for your specific track and motors. The most important value to tune is Turn 90 Delay.
4. **Run:** Select 1) Line Follow to start the main run.

5. Path Programming Guide

The robot's intelligence comes from the path[] array. Each number in this array is a command that tells the robot what to do when it encounters a specific event on the track.

Command ID	Name	Description
0	**EMPTY**	Does nothing. Marks an unused path step.
1	**LEFT**	At a simple intersection, forces a left turn.
2	**STRAIGHT**	At a simple intersection, forces the robot to go straight.
3	**RIGHT**	At a simple intersection, forces a right turn.
4	**T-LEFT**	At a T-junction, takes the left path.
5	**T-RIGHT**	At a T-junction, takes the right path.
6	**CROSS LEFT**	At a cross-intersection (+), takes the left path.
7	**CROSS STRAIGHT**	At a cross-intersection (+),

		goes straight.
8	**CROSS RIGHT**	At a cross-intersection (+), takes the right path.
9	**90 DETECT**	Expects a sharp 90-degree turn on the track.
10	**U TURN**	Expects a dead end and executes a U-turn.
11	**LINE GAP**	Expects a gap in the line and continues straight.
12	**WALL FOLLOW**	Expects a gap with a wall and follows the wall.
13	**OBSTACLE**	Expects an obstacle on the path and executes an avoidance maneuver.
14	**CONTINUE**	A special command to chain another command (e.g., 14, 12 means "expect a continue event that leads to a wall follow").
15	**INVERSE**	Expects a marker to switch between normal and inverse line modes.
16	**CAPACITOR**	Expects a special marker (e.g., capacitor charge point in some competitions).

6. Author

Team Vengeance

- Students of Mechanical Engineering, Islamic University of Technology (IUT)
- Robotics | CAD Modeling | Firmware Development

This project is a demonstration of my passion for robotics and my skills in integrating mechanical design with intelligent software systems.