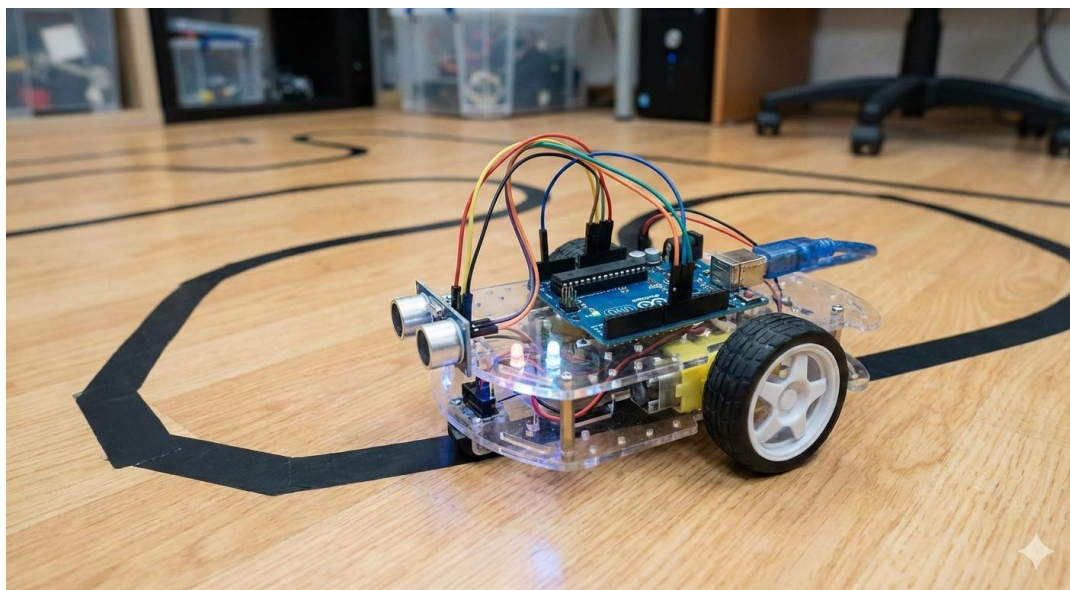




REMOTE WORKSHOP FOR PROGRAMMING

Arduino-Based Autonomous Robotics Systems



Practice 1 - Autonomous Line-Following Robot

Version_20251102

Student Name:

Delivery Date:



This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



OBJECTIVE

The objective is for students to become familiar with autonomous robotic systems using Arduino microcontrollers. To achieve this, the guided exercise involves designing a chassis for a line-following robot and programming the Arduino to implement autonomous navigation algorithms.

Students will work remotely on design and programming tasks, preparing all components that would be needed to deploy on a physical robot. This practice focuses on the theoretical and software aspects of robotics development.



1. PREVIOUS KNOWLEDGE.

For this exercise, the student should be able to:

- Understand basic electronics concepts (voltage, current, digital signals).
- Have basic knowledge of C/C++ programming syntax.
- Understand the concept of inputs and outputs in microcontrollers.
- Be familiar with Arduino IDE environment and basic sketch structure.
- Have basic 3D design knowledge (TinkerCAD or similar CAD software).
- Understand flowcharts and algorithm design principles.
- Be familiar with servo motor operation and PWM signals.

2. THEORETICAL RESOURCES.

- Remote Workshop Manual - Line Following Robot
- Annex A - 3D Parts Technical Drawings
- Annex B - Wiring Diagram
- Arduino IDE Reference Documentation
- TinkerCAD Online Platform (<https://www.tinkercad.com>)

3. DESCRIPTION OF THE PRACTICE.

1. The task is to design and program an autonomous line-following robot:

- Design a 3D printable chassis that can house an Arduino MEGA, servo motors, and line sensor array.
- Program the Arduino to read IR sensor values and detect black lines on a light surface.
- Implement a search algorithm: when no line is detected, the robot rotates to find a line.
- Implement a follow algorithm: when a line is detected, the robot moves forward along the line.
- Add audio feedback using a buzzer to indicate different robot states.

2. The system will include the following components:

- Arduino MEGA 2560 microcontroller board.
- 2x Continuous rotation servo motors (TS90D 9g) for wheel drive.
- 5-channel IR line sensor array for line detection.

- Piezo buzzer for audio feedback.
- Battery pack for power supply.

3. Pin Configuration:

Component	Pin	Type
Left Servo Motor	D9	PWM Output
Right Servo Motor	D10	PWM Output
Buzzer	D8	Digital Output
Line Sensor OUT1 (Left)	A0	Analog Input
Line Sensor OUT2	A1	Analog Input
Line Sensor OUT3 (Center)	A2	Analog Input
Line Sensor OUT4	A3	Analog Input
Line Sensor OUT5 (Right)	A4	Analog Input

4. Important Programming Considerations:

- Continuous rotation servos use 90 as stop, values below 90 rotate one direction, above 90 rotate the other.
- Line sensor returns HIGH when detecting black (low reflectivity) and LOW on white (high reflectivity).
- A threshold value must be calibrated for reliable line detection.

4. SAFE USE OF THE EQUIPMENT IN THE LABORATORIES.

The activities proposed in this workshop are performed remotely and do not require physical interaction with electrical equipment. However, students should:

- Use proper posture when working at a computer for extended periods.
- Take regular breaks to prevent eye strain.
- Save work frequently to prevent data loss.
- If testing code on physical hardware later, always disconnect power before making wiring changes.

5. ACTIVITIES

Part A: 3D Chassis Design (TinkerCAD)

- **Access TinkerCAD:** Go to <https://www.tinkercad.com> and log in or create a free account.
- **Create new 3D design:** Click 'Create new design' and name it 'LineFollower_[YourName]'.
- **Design the base plate:** Create a rounded rectangle approximately 120mm x 80mm x 5mm for the main chassis.
- **Add servo motor mounts:** Create rectangular cutouts (23mm x 12mm) on each side for the TS90D servo motors.
- **Create Arduino mount:** Add standoffs or mounting holes for the Arduino MEGA (101.5mm x 53.3mm footprint).
- **Add sensor bracket mount:** Design a front extension to mount the 5-channel line sensor at 10-15mm height from ground.
- **Add battery compartment:** Include space for the battery pack on the underside or rear of the chassis.
- **Export design:** Export as STL file for 3D printing documentation.

Part B: Arduino Programming (Arduino IDE)

- **Open Arduino IDE:** Launch the Arduino IDE software or use the online version at <https://create.arduino.cc>.
- **Create new sketch:** Create a new sketch and save it as 'LineFollower_v1'.
- **Include Servo library:** Add '#include <Servo.h>' at the top of your sketch.
- **Define pin constants:** Create constants for all pins (LEFT_SERVO=9, RIGHT_SERVO=10, BUZZER=8, SENSORS A0-A4).
- **Create servo objects:** Declare Servo objects for left and right motors.
- **Write setup() function:** Attach servos to pins, configure buzzer pin as OUTPUT, set sensor pins as INPUT.
- **Write readSensors() function:** Create a function that reads all 5 analog sensor values and returns line detection status.
- **Write moveForward() function:** Set both servos to move robot forward (opposite directions due to mirrored mounting).
- **Write rotate() function:** Set servos to rotate robot in place to search for line.
- **Write main loop() logic:** Implement: if line detected → moveForward(), else → rotate().
- **Add buzzer feedback:** Use tone() function to play different sounds for line-found vs searching states.
- **Verify code:** Click the checkmark button to compile and verify your code has no errors.

Part C: Algorithm Flowchart

- **Open flowchart tool:** Use draw.io (diagrams.net), Lucidchart, or paper.
- **Create start block:** Begin with initialization (setup servos, pins).
- **Add main loop:** Show the continuous loop structure.
- **Add decision diamond:** 'Line Detected?' with Yes/No branches.
- **Add action blocks:** 'Move Forward' for Yes, 'Rotate/Search' for No.
- **Connect back to loop:** Show the continuous cycle.

Activity 1: Pin Configuration and Variable Declaration

Complete the addressing table required for the line-following robot:

Component	Pin	Variable Name	Description

Activity 2: Arduino Code Development

Write the complete Arduino code for the line-following robot. Your code must include:

- All pin definitions and variable declarations
- setup() function with proper initialization
- A function to read sensor values and determine if line is detected
- Functions for robot movement (forward, rotate)
- Main loop() implementing the search and follow algorithm
- Buzzer feedback for different states

Activity 3: Algorithm Flowchart

Create a flowchart showing the complete algorithm for the line-following robot. Include:

- Initialization steps
- Main program loop
- Decision points (line detected yes/no)
- Actions for each branch

Activity 4: 3D Chassis Design

Design a 3D printable chassis for the line-following robot using TinkerCAD or similar software. Your design must include:

- Base plate with appropriate dimensions (~120mm x 80mm)
- Mounting cutouts for two servo motors
- Mounting area for Arduino MEGA
- Front bracket for line sensor array
- Space for battery pack

Activity 5: Code Simulation and Testing

Using TinkerCAD Circuits or Wokwi Arduino Simulator:

- Create a virtual circuit with Arduino MEGA, servos, and simulated sensors
- Upload your code to the simulator
- Test the logic by simulating sensor inputs
- Take a screenshot showing your simulation working

Activity 6: Documentation

Write a brief technical description (100-200 words) explaining:

- How your line-following algorithm works
- Why you chose your specific design approach
- What improvements could be made

6. Learning Outcomes.

- a) Designs a functional 3D chassis for an autonomous robot using CAD software.
- b) Writes Arduino code implementing sensor reading and motor control.
- c) Implements a line-following algorithm with search and follow behaviors.
- d) Creates technical documentation including flowcharts and pin mapping.
- e) Uses simulation tools to verify code logic before hardware deployment.
- f) Applies knowledge of servo motors, sensors, and microcontroller I/O.

7. Evaluation Criteria

- a) The 3D design includes all required mounting points and appropriate dimensions.
- b) The Arduino code compiles without errors and follows good programming practices.
- c) The algorithm correctly implements search (rotate) and follow (forward) behaviors.
- d) Pin mapping is correct and matches the hardware specifications.
- e) Flowchart accurately represents the program logic.
- f) Documentation clearly explains the design decisions and approach.
- g) Simulation demonstrates working code logic.

8. Evaluation tools.

Evaluation Item	Weight	Assessment Method
Pin configuration table (Activity 1)	10%	Completeness and accuracy
Arduino code (Activity 2)	30%	Code review + compilation test
Algorithm flowchart (Activity 3)	15%	Logic correctness + clarity
3D chassis design (Activity 4)	20%	Design review + printability
Simulation (Activity 5)	15%	Simulation screenshot + functionality
Documentation (Activity 6)	10%	Quality of explanation
TOTAL	100%	

EXTRA

You can check the programming in the following link created for the VETWORc project.

<https://mechatronica.netlify.app/>

SUBMISSION REQUIREMENTS

- Complete this document with all activities filled in
- Attach Arduino code file (.ino)
- Attach flowchart image or PDF
- Provide TinkerCAD share link or export STL file
- Include simulation screenshots