# Line-Following RC Car Project

# Milestone 1: Project Planning & System Design

---

**Major:** **Tutorial:**

**Student 1 name:** **ID:**

**Student 2 name:** **ID:**

**Student 3 name:** **ID:**

**Student 4 name:** **ID:**

**Student 5 name:** **ID:**

# Table of Contents

## 1. Project Handling Strategy

### 1.1 Team Organization

The project will be divided into three main phases: planning, implementation, and testing. Each team member will be assigned specific responsibilities across hardware design, software development, and integration testing.

### 1.2 Development Approach

We will follow an incremental development methodology:

- **Phase 1:** Core functionality (basic line following with 3 sensors)
- **Phase 2:** Code Enhancement (smooth turning algorithms, sensor data optimization)
- **Phase 3:** Robustness (lost-line detection and recovery)

### 1.3 Testing Strategy

Testing will be conducted in stages, starting with individual peripheral verification (ADC readings, PWM output), followed by integrated subsystem testing (sensor array response, motor control), and concluding with complete system testing on various track configurations.

---

## 2. Constraints & Priorities

### 2.1 Technical Constraints

- **Microcontroller:** ATmega32 (limited to its integrated peripherals)
- **Power Supply:** Must operate on battery power (voltage regulation required)
- **Sensor Range:** IR sensors effective distance is 2-10mm from surface
- **Motor Control:** PWM frequency must be compatible with DC motor specifications
- **Response Time:** System must react to line deviations within 50-100ms

### 2.2 Physical Constraints

- Car chassis dimensions must accommodate all components
- Weight distribution affects turning performance
- Sensor mounting height critical for accurate line detection
- Wire management to prevent interference with mechanical parts

### 2.3 Priority Hierarchy

1. **Critical:** Reliable line detection and basic following capability
2. **High:** Smooth turning without losing the line
3. **Medium:** Lost-line detection and recovery mechanism
4. **Low:** Speed optimization and advanced features

---

## 3. Full System Description

### 3.1 System Overview

The line-following RC car is an autonomous embedded system that uses infrared sensors to detect and follow a black line on a white surface. The ATmega32 microcontroller processes sensor data and controls two DC motors through PWM signals to maintain the car's position on the line.

### 3.2 Subsystem Descriptions

**3.2.1 Sensor Subsystem** An array of 5 IR sensors is mounted at the front of the car, positioned perpendicular to the ground. Each sensor consists of an IR LED and a phototransistor that detects reflected light. Black surfaces absorb IR light (low ADC reading), while white surfaces reflect it (high ADC reading). The sensors are connected to the ATmega32's ADC channels for continuous monitoring.

**3.2.2 Motor Control Subsystem** Two DC motors drive the left and right wheels independently. An L298N (for example) motor driver receives PWM signals from the ATmega32's Timer0 and Timer2 peripherals, enabling variable speed control. The differential speed between motors allows the car to turn while maintaining forward momentum.

**3.2.3 Decision and Control Logic** The microcontroller continuously reads the sensor array and determines the line position relative to the car's center. Based on this position, it calculates appropriate PWM duty cycles for each motor. The control algorithm implements proportional correction where deviation magnitude determines turning intensity.

**3.2.4 User Interface** A physical push button connected to an external interrupt pin serves as the start/stop switch. An LED indicator connected to a digital output pin provides visual feedback: steady on when following the line, blinking when lost-line recovery is active, and off when stopped.

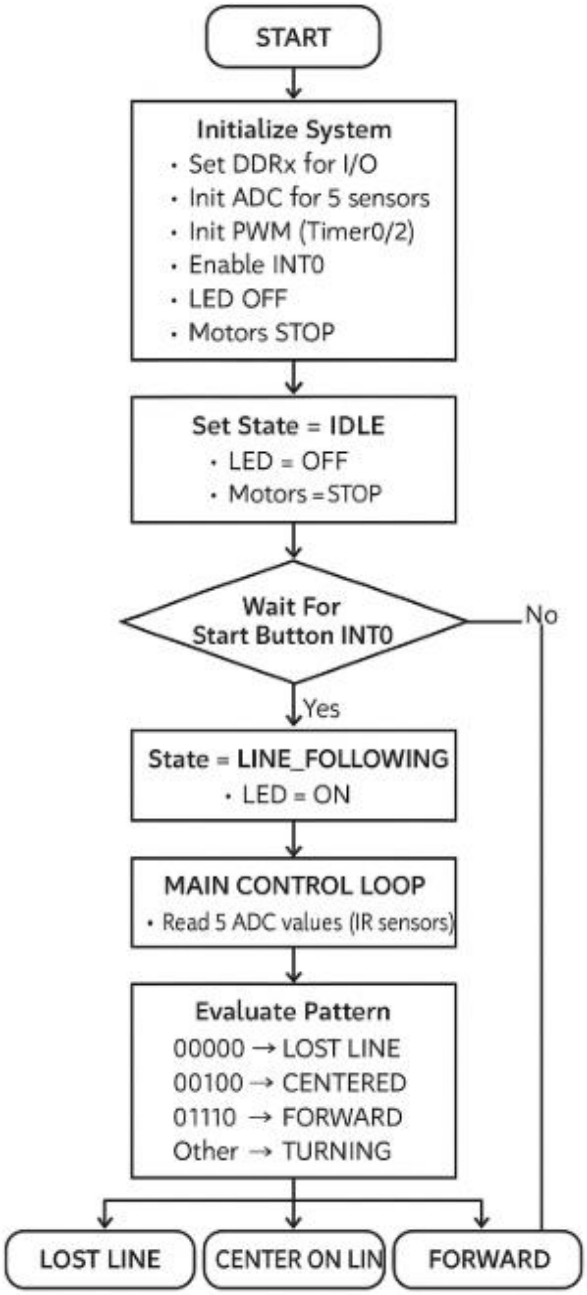
**3.2.5 Lost-Line Recovery** When all sensors detect white (line lost), the system enters recovery mode. It executes a search pattern: continuing forward briefly, then performing a sweeping turn in the direction of the last known line position. This continues until at least one sensor reacquires the line.

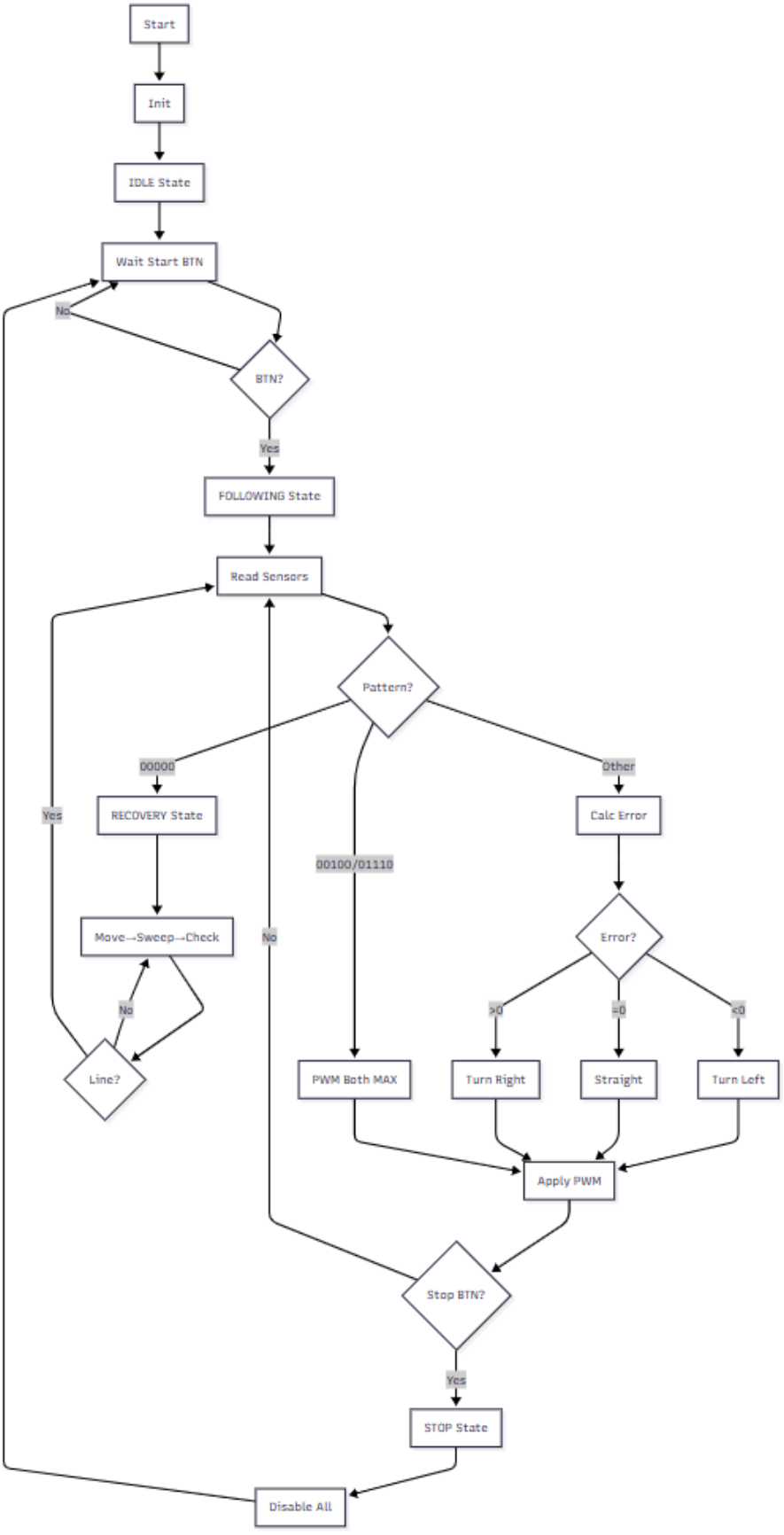
### 3.3 System Operation

Upon power-up, the system initializes all peripherals and enters standby mode. When the start button is pressed, the system begins reading sensors at regular intervals. The control algorithm updates motor speeds based on sensor data, keeping the center sensors over the black line.

---

**4. Flowchart Visualization**




Start
Init
IDLE State
Wait Start BTN
No
BTN?
Yes
FOLLOWING State
Read Sensors
Pattern?
00000
Other
00100/01110
Yes
RECOVERY State
Calc Error
Move→Sweep→Check
No
Error?
No
>0
=0
<0
Line?
PWM Both MAX
Turn Right
Straight
Turn Left
Apply PWM
Stop BTN?
Yes
STOP State
Disable All

- The system begins by reading the line-sensor output using the microcontroller's ADC. The analog voltage is converted into a digital value used to determine the robot's position relative to the line.

- The ADC value is compared against preset threshold levels to classify the robot's alignment as Center, Left, or Right.

- Based on the detected position, the control logic adjusts the motor outputs using PWM signals generated by Timer0 and Timer2. Duty cycles are modified by updating the output compare registers tied to each timer.

- In the Center condition, both motors receive balanced PWM duty cycles to maintain straight-line movement.

- In the Left condition, the PWM duty cycle on one motor is reduced while the other is increased, causing a corrective right turn.

- In the Right condition, the PWM duty cycles are adjusted inversely to produce a left correction.

- After applying the appropriate PWM adjustments, the system loops back to perform another ADC reading, forming a continuous closed-loop control cycle

---

### 5. Components List with Prices

| Component | Specification | Quantity | Unit Price (EGP) | Total (EGP) |
|---|---|---|---|---|
| **Microcontroller** | ATmega32 | 1 | 120 | 120 |
| **IR Line Sensors** | TCRT5000 Module | 5 | 15 | 75 |
| **DC Motors** | 6V Geared Motor (1:48 ratio) | 2 | 35 | 70 |
| **Motor Driver** | L298N Module | 1 | 65 | 65 |
| **Robot Chassis** | 2WD Car Kit with wheels | 1 | 150 | 150 |
| **Battery Holder** | 4x AA Battery Holder | 1 | 12 | 12 |
| **Batteries** | AA Rechargeable 1.2V | 4 | 20 | 80 |
| **Voltage Regulator** | 7805 (5V regulator) | 1 | 5 | 5 |
| **Push Button** | Tactile Switch | 1 | 2 | 2 |
| **LED** | 5mm Green LED | 1 | 1 | 1 |
| **Resistors** | 10kΩ, 220Ω, 330Ω | 1 pack | 10 | 10 |
| **Capacitors** | 100µF, 10µF, 0.1µF | 1 pack | 8 | 8 |
| **Crystal Oscillator** | 16 MHz | 1 | 8 | 8 |
| **PCB/Breadboard** | Universal PCB or breadboard | 1 | 25 | 25 |
| **Connecting Wires** | Jumper wires & headers | 1 pack | 20 | 20 |
| **Mounting Hardware** | Screws, nuts, spacers | 1 set | 15 | 15 |
| **Miscellaneous** | Black/white tape | - | 20 | 20 |
| | | | **TOTAL** | **686 EGP** |

## 6. Summary

This milestone establishes the complete planning foundation for the ATmega32-based line-following RC car project. The system architecture integrates sensor input, motor control, and decision-making logic to achieve autonomous operation. With a comprehensive component list, clear development phases, and structured approach documented, the project is ready to proceed to the implementation stage of hardware assembly and software development.