Project Name: A Line Following Robot using Arduino Nano.

Objectives:

* To develop an autonomous robot capable of following a predefined path.
* To implement an efficient sensor-based navigation system.
* To use an Arduino Nano for processing sensor inputs and controlling motor actions.
* To achieve stable power management using buck and boost converters.

# Introduction:

A line-following robot is an autonomous robot designed to follow a predefined path, typically marked by a black line on a white surface. It utilizes IR sensors to detect the path and makes real-time decisions to adjust its movement accordingly. This project implements a line-following robot using an Arduino Nano, motor driver, IR sensors, and a combination of buck and boost converters for stable power supply.

The line-following robot operates using an array of IR sensors that detect the black line on a white surface. These sensors provide input to the Arduino Nano, which processes the signals and determines the required movement. Based on the sensor readings:

* If only one sensor detects the line, the robot moves forward.
* If multiple sensors detect the line on one side, the robot turns in that direction.
* If no sensor detects the line, the robot stops.

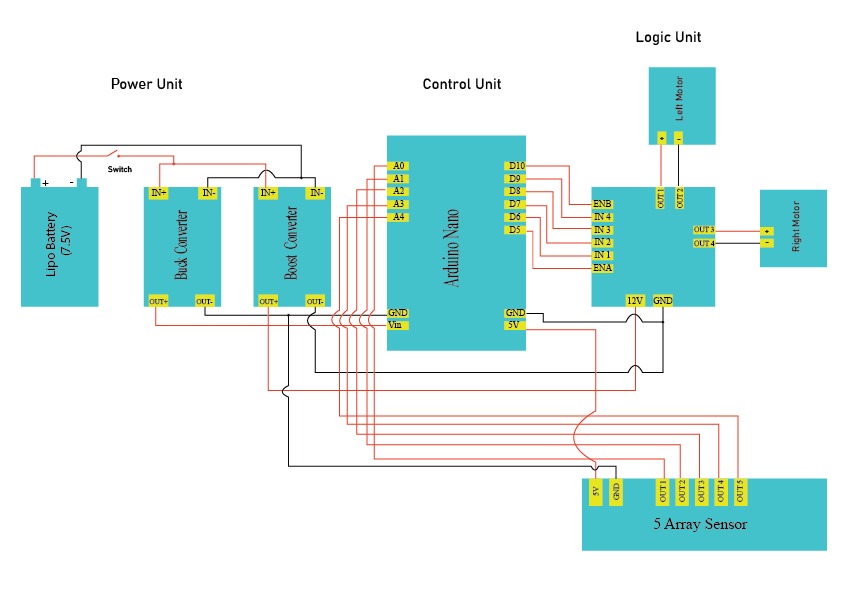
The motor driver (L298N) controls the DC motors based on Arduino's commands, while the buck and boost converters ensure stable voltage levels for different components.

# Apparatus Required:

Table 1.1: Required apparatus for the project.

|  |  |  |
| --- | --- | --- |
| **Apparatus Name** | Rating/Specification | Quantity |
| Arduino Development Board | Nano,5V | 1 |
| Motor Driver | L298N | 1 |
| DC Motor | 6v 100 RPM | 2 |
| TCRT5000 | 5 Channel, 5V | 1 array module |
| Chassis | Robot Base | 1 |
| Wheels | Compatible with Motors | 2 |
| Castor Wheel | Free Rotating | 1 |
| Jumper Wires | Male-Female, Male-Male | As required |
| Switch | 5V | 1 |
| Lipo Battery | 7.4v,1500MAH,2S | 1 |
| Buck Converter | 12V to 5V Step-Down | 1 |
| Boost Converter | 5V to 12V Step-Up | 1 |

# Circuit Diagrams:



|  |
| --- |
|  |
|  |

# Code:

// Motor Driver Pins (L298N)

#define ENA 5 // Left Motor Speed

#define IN1 6 // Left Motor Forward

#define IN2 7 // Left Motor Backward

#define ENB 10 // Right Motor Speed

#define IN3 8 // Right Motor Forward

#define IN4 9 // Right Motor Backward

// Sensor Configuration

#define NUM\_SENSORS 5

int sensorPins[NUM\_SENSORS] = { A0, A1, A2, A3, A4 };

int s[NUM\_SENSORS];

int base[NUM\_SENSORS] = { 1, 2, 4, 8, 16 };

int threshold = 512, sensor, sum;

// PID Control Variables

int baseSpeed = 50; // Adjust for desired speed

float kp = 40, kd = 20;

float error, lastError, PID;

void setup() {

Serial.begin(9600);

pinMode(ENA, OUTPUT);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(ENB, OUTPUT);

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

for (int i = 0; i < NUM\_SENSORS; i++) {

pinMode(sensorPins[i], INPUT);

}

}

void loop() {

readSensors();

calculatePID();

applyMotorControl();

}

// Read sensor values and calculate error

void readSensors() {

sensor = 0;

sum = 0;

int weightedSum = 0;

for (byte i = 0; i < NUM\_SENSORS; i++) {

s[i] = analogRead(sensorPins[i]); // Read sensor value

s[i] = (s[i] > threshold) ? 0 : 1; // Convert: 0 (Black), 1 (White)

weightedSum += s[i] \* (i + 1);

sum += s[i];

}

// If sensors detect black, calculate error; otherwise, maintain last error

if (sum > 0) {

error = ((float)weightedSum / sum) - ((NUM\_SENSORS + 1) / 2);

} else {

error = lastError;

}

}

// Calculate PID values

void calculatePID() {

float pTerm = kp \* error;

float dTerm = kd \* (error - lastError);

PID = pTerm + dTerm;

lastError = error;

}

// Apply motor control based on PID

void applyMotorControl() {

int leftSpeed = baseSpeed - PID;

int rightSpeed = baseSpeed + PID;

leftSpeed = constrain(leftSpeed, 0, 255);

rightSpeed = constrain(rightSpeed, 0, 255);

analogWrite(ENA, leftSpeed);

analogWrite(ENB, rightSpeed);

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

}

# Costing:

Table 1.2: Cost of the components used in this project

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Quantity** | **Price (BDT)** |
| Arduino Nano | 1 | 385 |
| Motor Driver | 1 | 183 |
| DC Motor | 2 | 300 |
| TCRT5000 | 1 array module | 310 |
| Chassis | 1 | 200 |
| Wheels | 2 | 100 |
| Castor Wheel | 1 | 80 |
| Switch | 1 | 5 |
| Lipo Battery | 1 | 1500 |
| Lipo Battery Charger | 1 | 600 |
| Buck Converter | 1 | 100 |
| Boost Converter | 1 | 85 |
| Others | - | 500 |
| **Total** | | **4348.00** |

# Result and Discussion:

The line-following robot (LFR) is designed to follow black lines on a lighter surface. The critical value for detecting black lines is when IR sensor readings fall below 512. In the code, the condition `s[i] = (s[i] > threshold) ? 0 : 1` assigns `1` to readings ≤ 512 (line detected) and `0` to readings > 512 (no line). The PID system utilizes this detection to adjust motor speeds, maintaining alignment with the line.

The threshold of 512 is effective because black lines reflect less light, resulting in sensor readings ≤ 512, while the lighter surface produces readings > 512. This aligns with the LFR’s configuration for a darker line. The `readSensors()` function computes the line’s position via a weighted sum, feeding the error into the PID controller (`kp = 0.2`, `kd = 20`) to steer the robot. It turns left for a negative error (line left) and right for a positive error (line right).Variations in line reflectivity or lighting could require threshold adjustments. Currently, 512 ensures consistent line detection under the tested conditions.

# Conclusions:

The line-following robot (LFR) developed in this project is capable of tracking black lines on a lighter surface using IR sensors and a PID control system. However, its performance is hindered by calibration issues under varying lighting conditions and inconsistencies when the track properties change. Despite these challenges, the project has been highly educational, providing practical experience in electronics and programming. Through working with components such as sensors and motor drivers, we have gained a deeper understanding of robotic systems and control logic. While improvements like adaptive calibration could enhance its reliability, the knowledge and skills acquired during this endeavor have made it a valuable learning experience.

# **References**: