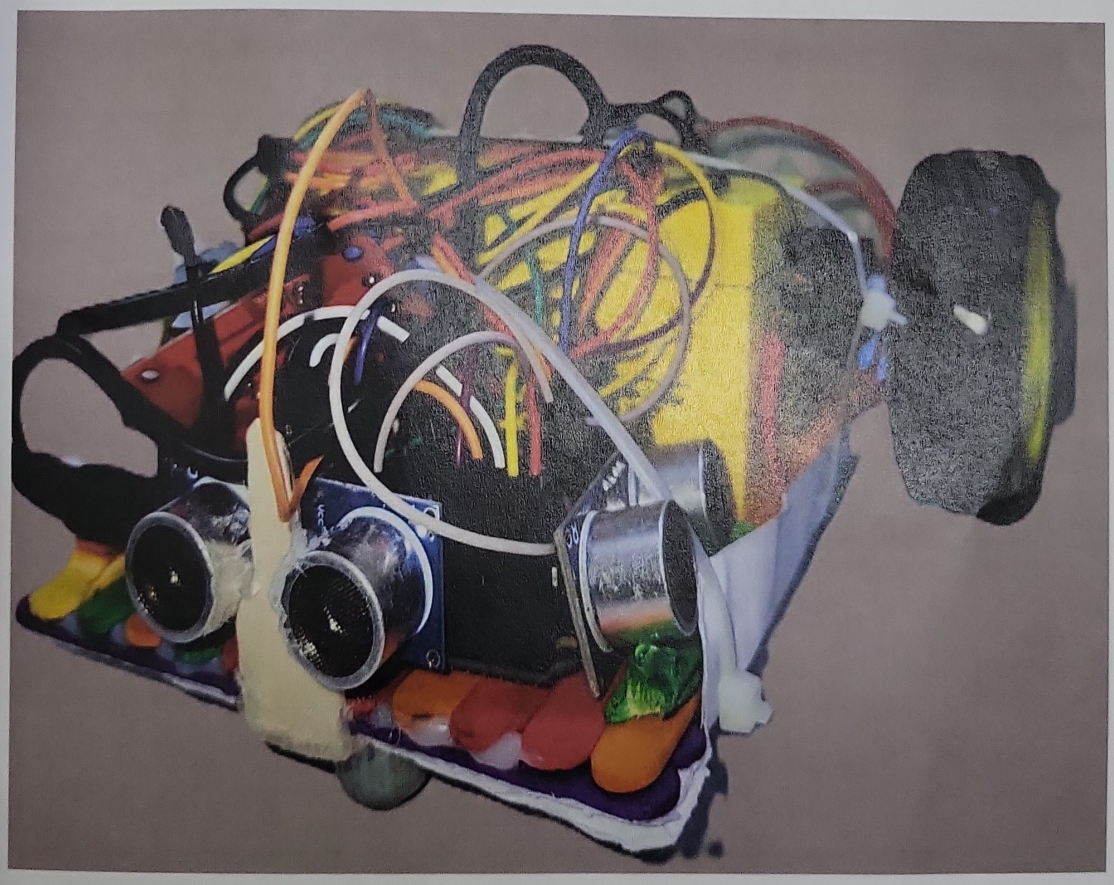
Arduino | Maze Solving Robot

Welcome, This is my team Lexus and our first robot "Lexus v1.0".

This Robot was designed to solve a simple Maze.



1. Introduction

The Arduino Maze Solving Robot project focuses on the integration of analog electronics concepts with microcontroller-based robotics to solve a maze autonomously. The robot utilizes various analog and digital sensors, actuators, and control algorithms to navigate through a predetermined path in a maze, demonstrating practical applications of analog electronics, signal conditioning, and control systems in robotics.

2. Objectives

- To design and implement a maze-solving robot using an Arduino microcontroller.
- To explore the application of analog electronics components such as sensors, signal processing circuits, and motor drivers.
- To demonstrate autonomous decision-making in a constrained environment.

3. System Overview

3.1 Hardware Components:

- Arduino Microcontroller: Serves as the control unit for processing inputs and controlling outputs.
- Infrared (IR) Sensors: Used to detect paths and obstacles within the maze.
- Motors & Motor Drivers: Provide movement and control for the robot's wheels.
- Power Supply Unit: Supplies power to the microcontroller and connected devices.
- Chassis: Holds all the components together in a stable structure.

3.2 Software Components:

- Arduino IDE: For writing and uploading code to the microcontroller.

- Algorithms: Utilizes maze-solving algorithms such as the right-hand or left-hand rule, Breadth-First Search (BFS), etc., for decision-making.

4. Working Principle

The robot uses a combination of analog and digital electronics principles to sense and react to its environment. The primary functions are:

- Sensor Input: IR sensors provide analog signals that are converted to digital signals using an Analog-to-Digital Converter (ADC). These signals help detect walls, paths, and intersections.
- Control Logic: The microcontroller processes sensor input using a predetermined algorithm to make navigation decisions.
- Motor Control: Analog and digital signals are used to control the speed and direction of the motors, allowing the robot to move and turn as required.

5. Applications of Analog Electronics

5.1 Analog Sensors for Path Detection

The robot relies on analog IR sensors to detect variations in reflectance. These sensors play a critical role in distinguishing paths and obstacles within the maze, highlighting a key application of analog electronics in real-time feedback systems.

5.2 Signal Processing and Filtering

Analog signal conditioning circuits are used to reduce noise and improve the accuracy of the sensors. Operational amplifiers (op-amps), resistors, and capacitors are utilized for amplification and filtering.

5.3 Motor Control

Motor drivers convert control signals into appropriate analog voltages to drive the motors, illustrating the interface between analog and digital systems in robotic motion control.

6. Design and Implementation

6.1 Circuit Design

The maze-solving robot's circuit includes IR sensors connected to the Arduino's analog input pins. Motor driver ICs (such as L298N) are interfaced with the microcontroller to drive the motors. Proper decoupling capacitors and resistors are used for stability.

6.2 Programming

The control algorithm is programmed using C/C++ in the Arduino IDE. The code enables the robot to make decisions based on sensor inputs and control motor movement accordingly.

7. Results and Analysis

The robot successfully navigates a given maze using the programmed logic and analog sensor data. The performance of the robot depends on factors such as the sensitivity of the IR sensors, motor speed control, and noise reduction in analog signals.

8. Conclusion

This project illustrates how analog electronics can be integrated with digital systems to create intelligent, autonomous devices. The Arduino-based maze-solving robot showcases practical applications in the fields of robotics, automation, and analog signal processing.

Bibliography

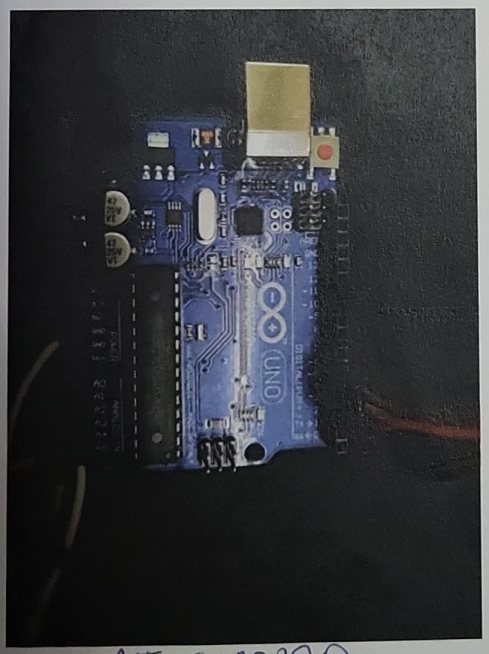
1. Banzi, Massimo, and Shiloh, Michael. Getting Started with Arduino . O'Reilly Media, 2020.

2. Villamizar, Diego. Robotics with Arduino . Packt Publishing, 2018.
3. Mims III, Forrest. Getting Started in Electronics . Master Publishing, 2003.
4. Rashid, Muhammad H. Spice for Circuits and Electronics Using PSpice . Prentice Hall, 1995.
5. Online Sources:
- Arduino Documentation: https://www.arduino.cc
- Maze Solving Algorithms: https://www.robotics.com/maze-solving

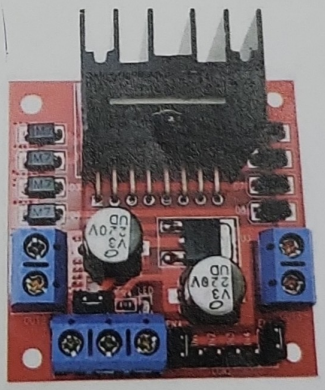
Step 1: Parts

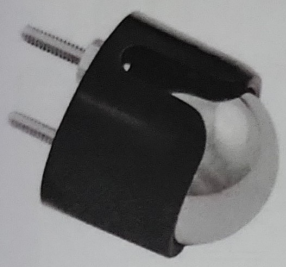
Parts List:

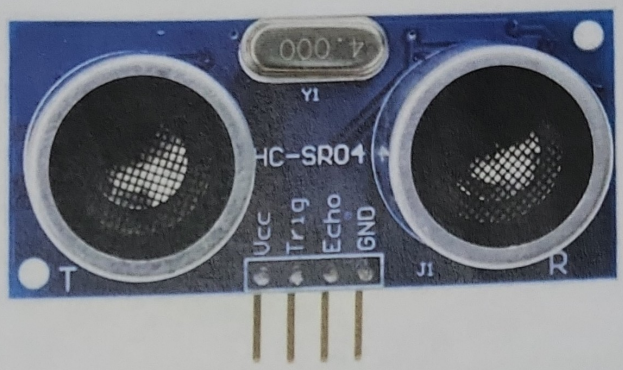
1. Arduino UNO
2. 12v DC motors (x2)
3. Wheels (x2)
4. Motor Driver (L298N)
5. Distance Sensor (Ultra Sonic)
6. Wires (Jumper wires 1-pin male-male)
7. 12v Battery (1300 mAh)
8. Switch



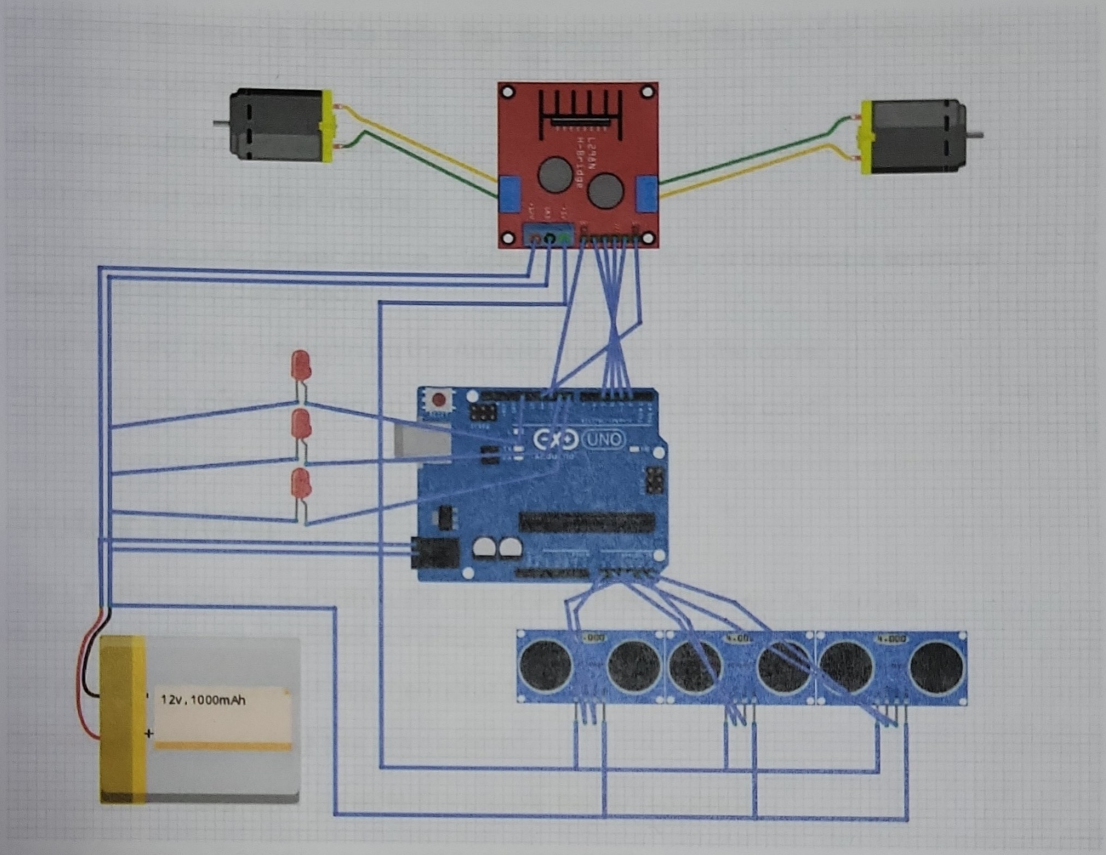
IC-ATMEGA238P

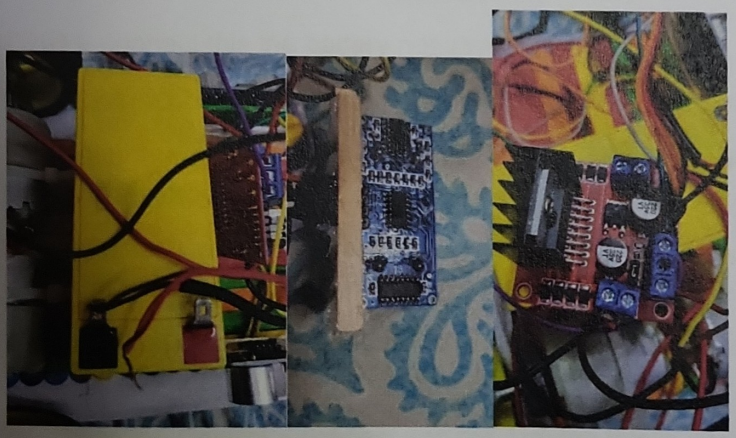





4.000
Y1
HC-SR04
Vcc
Trig
Echo
GND
T
J1
R

Step 2: Building/Wiring





Sensors

Lets talk about "The Ultrasonic sensor"

An Ultrasonic sensor is simple radar that measures the distance of an object by using sound waves.

Ultrasonic Sensor connections:

1. GND: connect this to the Ground.
2. VCC: connect to the power source 5 voltages.(**Alert! if you connect it to more than 5v it will be damaged**)
3. Echo: connect this to any pin on the Arduino. (match it to the code)
4. TRIG: connect this to any pin on the Arduino. (match it to the code)

Motor Driver

The **L298N** H-bridge: it controls the speed and direction of two DC motors. it can be used with motors that have a voltage of between 5 and 35V DC.

1. DC motor 1 "+" > connect this the motor #1
2. DC motor 1 "-" > connect this the motor #1
3. 12v jumper > keep this connected to enable the 5v regulator.
4. Power Source > Connect your battery positive here
5. GND > connect this the battery negative
6. 5v output (if 12v jumper in place) > connect the sensors here
7. DC motor 1 enable jumper > Remove the jumper and connect it to the Arduino this is used to control the speed of motor 1 (match it to the code).
8. IN1 Direction Control > connect it to the Arduino this is used to control the direction of motor 1 (match it to the code).
9. IN2 Direction Control > connect it to the Arduino this is used to control the direction of motor 1 (match it to the code).
10. IN3 Direction Control > connect it to the Arduino this is used to control the direction of motor 2 (match it to the code).

11. IN4 Direction Control > connect it to the Arduino this is used to control the direction of motor 2 (match it to the code).
12. DC motor 2 enable jumper > Remove the jumper and connect it to the Arduino this is used to control the speed of motor 2 (match it to the code).
13. DC motor 2 "+" > connect this the motor #2
14. DC motor 2 "-" > connect this the motor #2

Battery

I used 12v Battery with 1300 mAh.

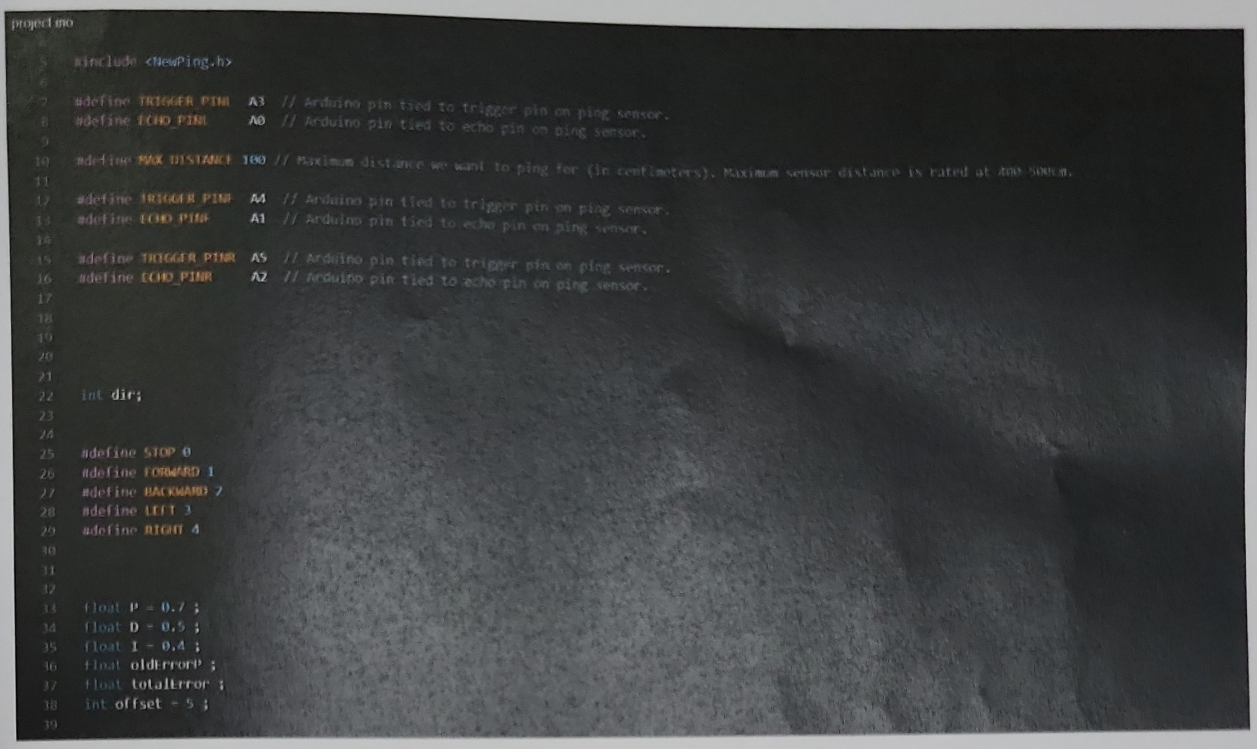
Note:

Remember to connect all grounds to a common Ground to the battery negative.

Step 3: Coding

Code flow:

1. **defining the pins**
2. **defining output and input pins**
3. **check sensors' readings**
4. **use sensors' reading to define walls**
5. **check first route (if it was left then follow the left wall, if it's right follow the right wall)**
6. **Use PID to avoid hitting the walls and to control motors' speed**



```
project.ino

#include <NewPing.h>

#define TRIGGER_PIN  A3  // Arduino pin tied to trigger pin on ping sensor.
#define ECHO_PIN     A0  // Arduino pin tied to echo pin on ping sensor.

#define MAX_DISTANCE 100 // Maximum distance we want to ping for (in centimeters). Maximum sensor distance is rated at 400-500cm.

#define TRIGGER_PINR  A4  // Arduino pin tied to trigger pin on ping sensor.
#define ECHO_PINR     A1  // Arduino pin tied to echo pin on ping sensor.

#define TRIGGER_PINR  A5  // Arduino pin tied to trigger pin on ping sensor.
#define ECHO_PINR     A2  // Arduino pin tied to echo pin on ping sensor.




int dir;


#define STOP 0
#define FORWARD 1
#define BACKWARD 2
#define LEFT 3
#define RIGHT 4


float P = 0.7 ;
float D = 0.5 ;
float I = 0.4 ;
float oldErrorP ;
float totalError ;
int offset = 5 ;

```

1- first_turn = false ;

2- rightWallFollow = false ;

3- leftWallFollow = false ;

first_turn = true ;

rightWallFollow = true ;

leftWallFollow = false ;

if you want it to follow the left wall:

first_turn = true ;

rightWallFollow = false ;

leftWallFollow = true ;