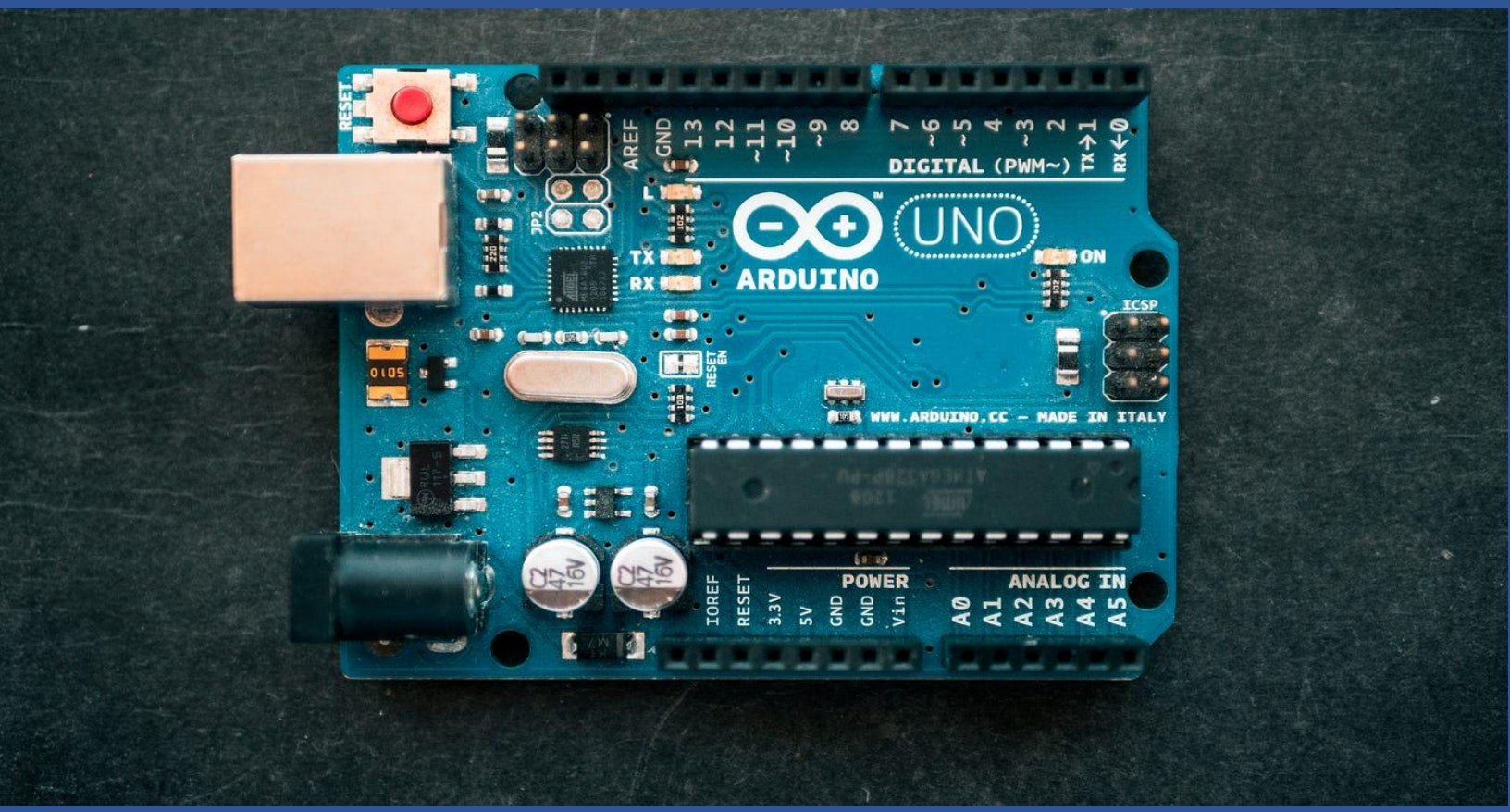
Embedded System Project

2024-2023



Obstacle-Avoiding Robot

By :

- Eslam Hany El Sayed Zoghla
- Adham Ashraf Ibrahim
- Alaa Abd El Maqsoud Muhammad
- Eslam Hamdy Ragab
- Engy Salah Saad
- Omnia Magdy Saber
- Ayat Ashraf Mohamed
- Rahma Ahmed Abdelhamid

Supervised by : Eng. Ghada Refaay

Instructor : Asst.Pro. Ahmed Shalaby

Acknowledgment

We would like to express our deep gratitude and thank you to **Asst.Pro. Ahmed Shalaby** and **Eng. Ghada Refaay**, who did put effort into encouraging us to do a great job, providing our team with valuable information and advising us to be better every time. Thanks for the continued support and kind communication which has had a huge impact in terms of feeling interested about what we are working on. We are truly grateful to them.

Contents

Abstract

The obstacle-avoiding robot project focuses on the development of an autonomous robotic car capable of intelligently navigating its surroundings while evading obstacles. Essential to this project is the integration of sensors, particularly ultrasonic sensors, which enable the robot to detect obstacles and dynamically adjust its path to avoid collisions.

Define your project goals and identify its objectives.

Project Goals:

- Autonomous Navigation: Develop a robot capable of autonomously navigating through varied environments.
- Obstacle Avoidance: Implement a robust obstacle avoidance system using ultrasonic sensors to detect and evade obstacles in real-time.
- Hardware Integration: Combine essential hardware components, including motors, wheels, chassis, and a microcontroller, to create a functional robotic system.
- Real-time Decision-Making: Enable the robot to process sensor data swiftly, allowing it to make informed decisions on navigation adjustments.
- Precise Motor Control: Implement a programming algorithm to ensure precise control over motor functions, enabling accurate and efficient movement.

Objectives:

- Sensor Integration: Connect ultrasonic sensors to the robot to facilitate the detection of obstacles within its vicinity.
- Hardware Assembly: Physically assemble motors, wheels, chassis, and the microcontroller into a cohesive robotic structure.
- Software Development: Create a software algorithm that interprets sensor data, analyzes it in real-time, and triggers appropriate responses for obstacle avoidance.
- Testing and Calibration: Conduct thorough testing of the robot in controlled environments to calibrate and fine-tune its obstacle avoidance capabilities.
- Documentation: Document the project's design, components, algorithms, and results comprehensively for future reference and replication.
- Optimization: Continuously optimize the obstacle avoidance algorithm and motor control mechanisms to enhance the robot's efficiency and responsiveness.
- Demonstration: Present a working demonstration of the obstacle-avoiding robot, showcasing its ability to navigate autonomously and avoid obstacles effectively.

1- INPUT AND OUTPUT TABLE

Inputs	Description
Ultrasonic Sensor Readings	is a device that uses ultrasonic waves to measure the distance between the sensor and an object, the distance in range(20cm,16.8m)

Outputs	Description
Motor Control Signals (in1, in2, in3, in4)	Signals sent to the motor driver to control the direction of the motors (forward, backward, left, right) based on obstacle detection and avoidance decisions.
Motor Speed Signals (ENA, ENB)	Analog signals controlling the speed of the left and right motors to adjust the overall speed of the robot.

2- HARDWARE COMPONENTS

Pin	Function	Description
2 (echo1)	Input for left sensor echo	Receives the echo signal from the ultrasonic sensor on the left side, indicating the time taken for sound to travel to an object and back.
4 (Trig1)	Output for left sensor echo	Sends a brief pulse to trigger the ultrasonic sensor on the left side to start measuring distance.
9 (Echo2)	Input for central sensor echo	Receives the echo signal from the ultrasonic sensor in the front/center, providing information about the distance to an object.
11 (Trig2)	output for central sensor trig	Sends a pulse to trigger the ultrasonic sensor in the front/center to initiate distance measurement.
3 (Echo3)	input for right sensor echo	Collects the echo signal from the ultrasonic sensor on the right side, indicating the time taken for sound to travel to an object and back.
5 (Trig3)	Output for right sensor trig	Generates a brief pulse to trigger the ultrasonic sensor on the right side to start measuring distance.
13 (in1)	Output for right motor control	Controls the direction of the right-side motor (forward).
12 (in2)	Output for right motor control	Controls the direction of the right-side motor (reverse).

10 (in3)	Output for left motor control	Controls the direction of the left-side motor (forward).
8 (in4)	Output for left motor control	Controls the direction of the left-side motor (reverse).
7 (ENA)	Output for Right motor speed control	Adjusts the speed of the right-side motor.
6 (ENB)	Output for Left motor speed control	Adjusts the speed of the left-side motor.

Component	Description
Microcontroller	A programmable device that acts as the brain of the robot, executing the provided code.
DC Motors	Motors responsible for driving the wheels of the robot, enabling its movement.
Motor Driver	A motor driver circuit that controls the direction and speed of the DC motors.
Ultrasonic Sensors	Devices that use sound waves to measure distances; typically used for obstacle detection.
Left Sensor	Measures the distance to the left side of the robot, helping in left-side obstacle detection.
Central Sensor	Measures the distance in the front/center of the robot, aiding in obstacle detection in its path.
Right Sensor	Measures the distance to the right side of the robot, assisting in right-side obstacle detection.

3- Algorithm

void loop() {

Read distances from forward ,left and right sensor then make some check before taking decision.

Check if there is an object in forward.

If object so close of front of car => go Back.

If the is no object in left and right direction.

If Right > Left => go right.

Else=>go left.

If there is object on right and no in left => go Left.

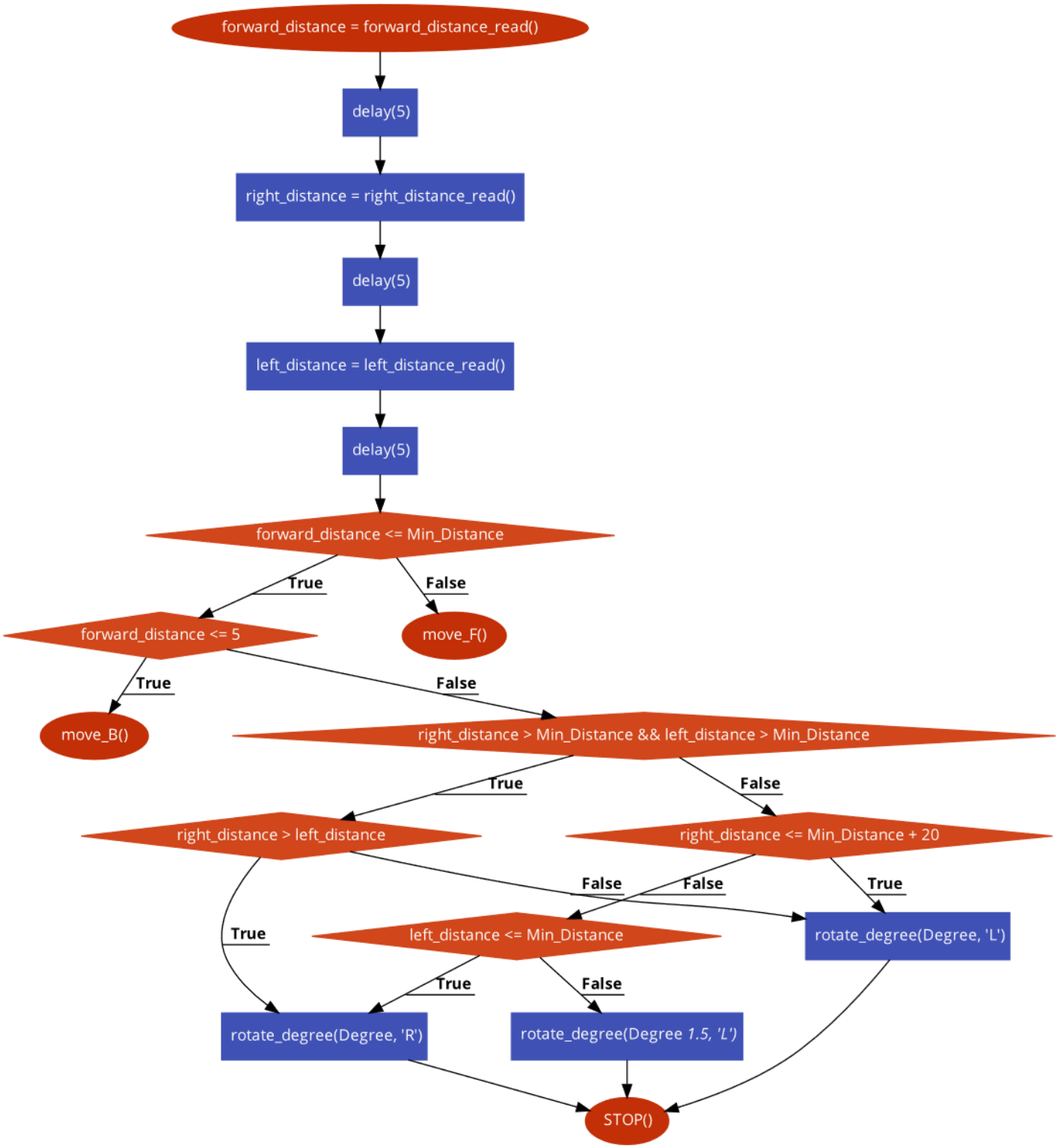
If there is object on left and no in right=> go Right.

Otherwise rotate with ratio degree.

Else move Forword().

}

4- Flowchart


forward_distance = forward_distance_read()
delay(5)
right_distance = right_distance_read()
delay(5)
left_distance = left_distance_read()
delay(5)
forward_distance <= Min_Distance
True
False
forward_distance <= 5
move_F()
True
False
move_B()
right_distance > Min_Distance && left_distance > Min_Distance
True
False
right_distance > left_distance
right_distance <= Min_Distance + 20
False
False
True
True
left_distance <= Min_Distance
rotate_degree(Degree, 'L')
True
False
rotate_degree(Degree, 'R')
rotate_degree(Degree 1.5, 'L')
STOP()

5- Ultrasonic code in C language

```
#include <avr/io.h>
#include <util/delay.h>

#define F_CPU 16000000UL  // Set CPU frequency to 16 MHz
#define SOUND_SPEED 343   // Speed of sound in air in m/s

void initUltrasonic() {
	// Configure the required pins and registers for ultrasonic sensor
	// Assume Trig pin is connected to PORTD, Pin 2 (PD2) and Echo pin is connected to PORTD,
Pin 3 (PD3)

	// Set Trig pin as output
	DDRD |= (1 << PD2);

	// Set Echo pin as input
	DDRD &= ~(1 << PD3);
}

unsigned int measureDistance() {
	// Send a 10 microsecond pulse to trigger the ultrasonic sensor
	PORTD |= (1 << PD2);   // Set Trig pin high
	_delay_us(10);         // Wait for 10 microseconds
	PORTD &= ~(1 << PD2);  // Set Trig pin low

	// Measure the duration of the pulse from the ultrasonic sensor
	while (!(PIND & (1 << PD3))); // Wait for Echo pin to go high
	TCNT1 = 0;  // Reset Timer/Counter1
	TCCR1B |= (1 << CS11); // Set prescaler to 8

	while (PIND & (1 << PD3)); // Wait for Echo pin to go low
	TCCR1B = 0; // Stop Timer/Counter1

	// Calculate the pulse duration in microseconds
	uint16_t pulse_duration = TCNT1;

	// Calculate the distance using the formula: distance = (duration * speed of sound) / 2
	// Divide by 58 to convert microseconds to centimeters
	uint16_t distance = (pulse_duration * SOUND_SPEED) / (2 * 58);

	return distance;
}

int main() {
	initUltrasonic();

	// Enable global interrupts
	sei();

	// Main loop
	while (1) {
		unsigned int distance = measureDistance();
		// Do something with the distance measurement, e.g., send it over UART or perform an
action based on distance
	}

	return 0;
}
```

6- test strategy

The goal of the test strategy is to ensure the proper functionality of the obstacle-avoiding robot, validating its ability to navigate autonomously while avoiding obstacles and responding to user inputs.

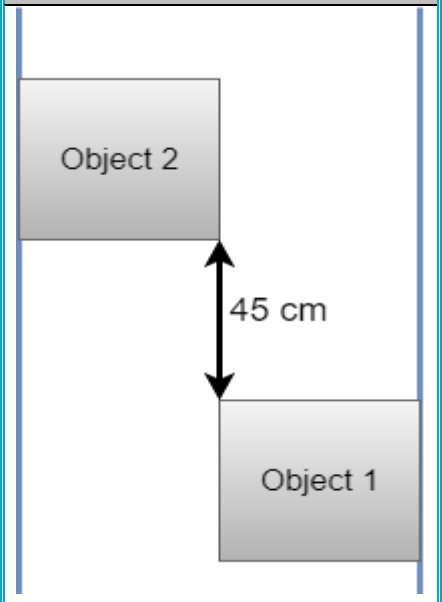
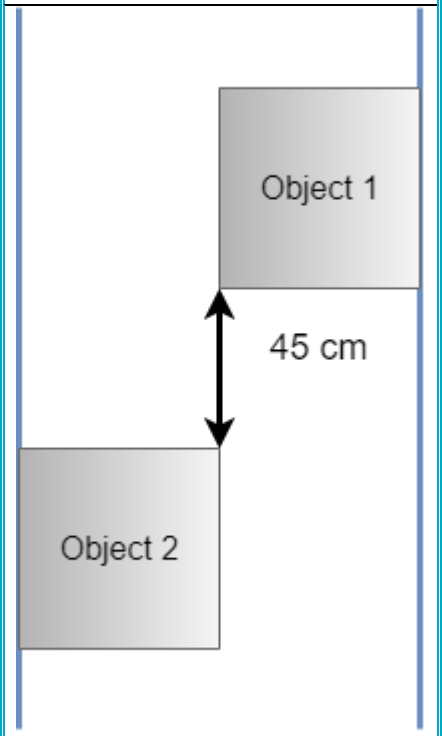
Test Cases:

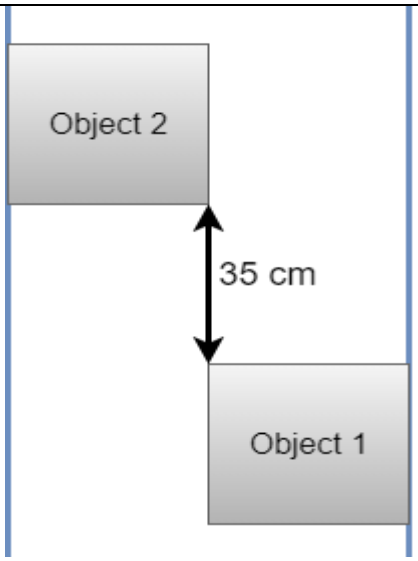
- Ultrasonic Sensor Accuracy:
  - Inputs: Ultrasonic sensor readings (Echo1, Echo2, Echo3).
  - Procedure: Place obstacles at known distances and verify that the sensor readings correspond accurately to the actual distances.
  - Expected Result: Sensor readings match the expected distances within an acceptable margin of error.

- Motor Movement Control:
  - Inputs: Motor control signals (in1, in2, in3, in4).
  - Procedure: Send control signals to the motors for forward, backward, left, and right movements.
  - Expected Result: Motors respond correctly to control signals, resulting in the desired robot movements.
- Motor Speed Adjustment:
  - Inputs: Motor speed signals (ENA, ENB).
  - Procedure: Adjust motor speeds and observe corresponding changes in the robot's overall speed.
  - Expected Result: Motors operate at varying speeds, and the robot's speed adjusts accordingly.

- Obstacle Avoidance Algorithm:
  - Inputs: Ultrasonic sensor readings, obstacle scenarios.
  - Procedure: Introduce obstacles in the robot's path and observe its response to avoid collisions.
  - Expected Result: The robot successfully detects obstacles, adjusts its direction, and avoids collisions.

- End-to-End Autonomous Navigation:
  - Inputs: Real-world environment with obstacles.
  - Procedure: Deploy the robot in a controlled environment and let it navigate autonomously.
  - Expected Result: The robot successfully navigates, avoiding obstacles, and completes a predefined course.

Distance between objects (cm)	Time	Escape	Image
45	5 s	YES	
45	8 s	YES	

35	5 s	YES	
35	7 s	YES