SRI RAMAKRISHNA INSTITUTE OF TECHNOLOGY

(An Autonomous Institution
Approved by AICTE, New Delhi– Affiliated to Anna University, Chennai
Accredited by NAAC with 'A' Grade)

DEPARTMENT OF ELETRICAL AND ELECTRONICS ENGINEERING

PROJECT BASED LEARNING

NAME	SRI KANTESHWAR CS
REGISTER NUMBER	71382203020
YEAR &SEMESTER	III YEAR & 5th SEMESTER
COURSE CODE	20EEP34
COURSE NAME	INTRODUCTION TO RASPBERRY PI
PROJECT TITLE	OBSTACLE AVOIDANCE ROBOT USING RASPBERRY PI PICO

TABLE OF CONTENTS :

ABSTRACT

This project is focused on creating an autonomous robot capable of navigating its environment by avoiding obstacles. At the heart of the system is the Raspberry Pi, which processes input from an ultrasonic sensor to detect objects in the robot's path. When an obstacle is detected, the Raspberry Pi sends commands to the L298N motor driver, which controls the motors to adjust the robot's direction and steer clear of the obstacle. The integration of these components allows the robot to move independently and avoid collisions, making it a useful model for understanding the basics of autonomous navigation and robotics. The project's success can serve as a stepping stone for more complex robotics projects that require higher levels of autonomy and decision-making.

OBJECTIVES:

The primary objective of this project is to design and develop an autonomous obstacle-avoiding robot that can navigate its environment without human intervention.

The key objectives include:

1. To implement real-time obstacle detection using an ultrasonic sensor.

2. To control the movement of the robot using two gear motors powered by Li-ion batteries and managed by an L298N motor driver.

3. To program the Raspberry Pi Pico microcontroller for efficient decision-making in avoiding obstacles.

4. To improve the robot's response time to detected obstacles.

5. To allow for adjustable motor speed for optimized movement and performance in different environments.

6. To demonstrate the feasibility of autonomous navigation for potential applications in robotics and automation.

INTRODUCTION:

Obstacle-avoiding robots represent an essential step toward developing autonomous systems capable of navigating dynamic environments. This project involves designing and implementing such a robot using the Raspberry Pi Pico microcontroller, an ultrasonic sensor, and an L298N motor driver. The robot uses real-time obstacle detection to adjust its movement, ensuring smooth and collision-free navigation. Powered by dual Li-ion batteries and controlled by efficient programming, the system demonstrates intelligent decision-making and adaptability. This project highlights the practical potential of robotics in automation, with applications ranging from smart navigation to autonomous delivery systems.

LIST OF COMPONENTS:

S.NO	Components	Specification	Quantity
1.	RASPBERRY PI PICO	RPI2040	1
2.	MOTOR DRIVER	L298n	1
3.	ULTRASONIC SENSOR	HC-SR04	1
4.	SERVO MOTOR	SG90(180 deg)	1
5.	DC GEARED MOTOR	100 RPM	2
6.	CONNECTING WIRES	-	As Required
7.	LI-ION BATTERIES	3.7V,2500mAh	2

WORKING OF ULTRASONIC SENSOR:

Figure 1:Ultra sonic sensor

1**. Pins:**

-**VCC**: Connects to 5V power supply.

-**GND**: Connects to ground.

-**Trig**: Sends ultrasonic pulse when triggered by the microcontroller.

-**Echo**: Outputs a signal for the time of flight of the echo.

2.**Working**:

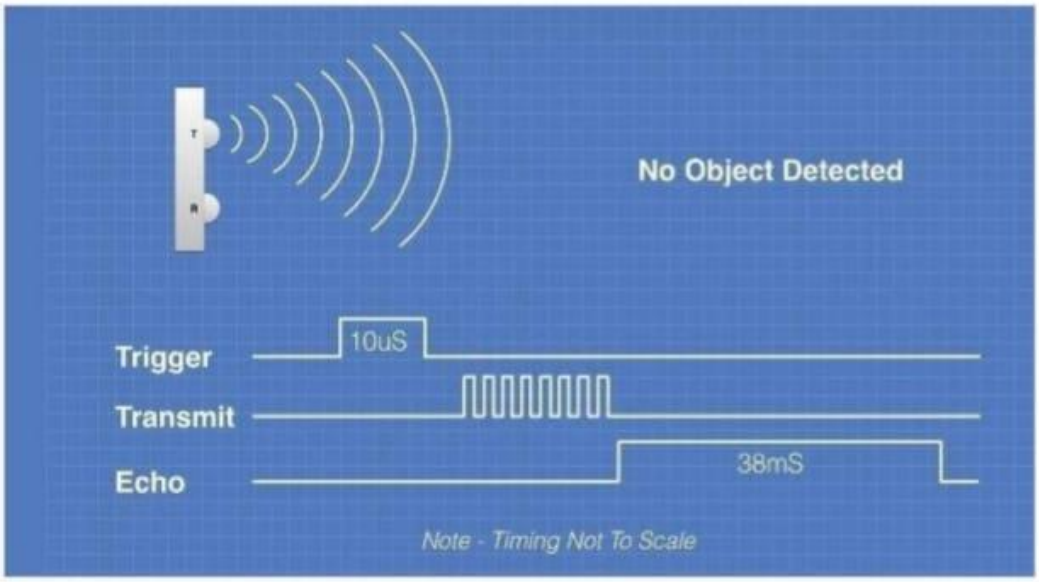
-**Trigger Pulse**: A 10µs HIGH signal on Trig pin emits ultrasonic waves.

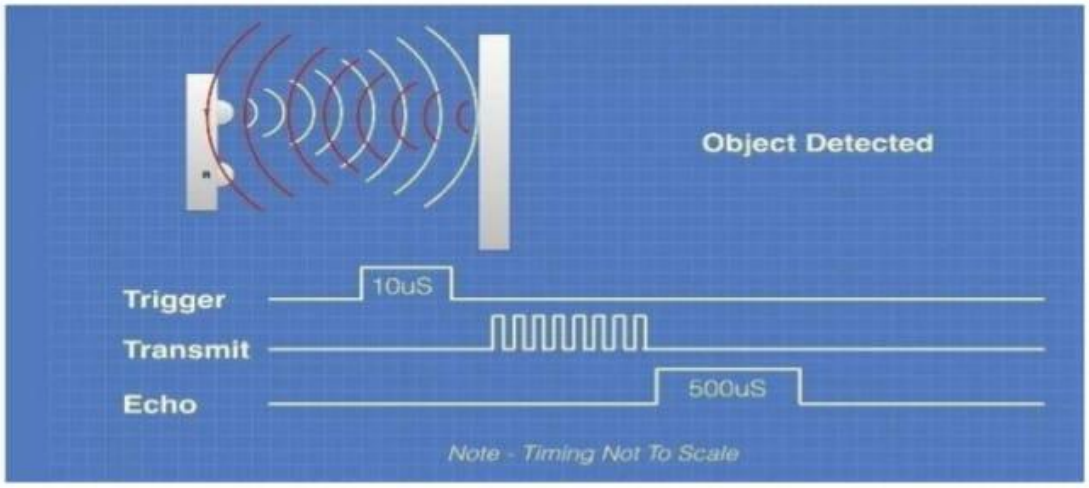
-**Echo Signal**: Echo pin goes HIGH when reflected waves are received.

-**Time Measurement**: Microcontroller measures how long Echo pin stays HIGH.

-**Distance Calculation**: Distance = \(\frac{\text{Time} \times \text{Speed of Sound}}{2}\).

This process provides real-time distance data for obstacle detection.





)istance= (Time x Speed of Sound in Air (343 m/s))/2.

Figure 2 ultra sonic sensor working

WORKING OF L298N:

Figure 3 :L298n driver

1. **Pins**:
    - **Power Pins**:
        - **VCC**: Connects to motor power supply (up to 12V).
        - **GND**: Common ground for power and control.
        - **5V**: Provides 5V output (used for logic circuits if needed).
    - **Control Pins**:
        - **IN1, IN2, IN3, IN4**: Control motor directions (two pins per motor).
    - **Enable Pins**:
        - **ENA, ENB**: Control motor speed using PWM signals.
    - **Output Pins**:
        - **OUT1, OUT2, OUT3, OUT4**: Connect to motor terminals.
2. **Working**:

- **Direction Control**: Input pins (e.g., IN1 and IN2) are toggled HIGH/LOW to set motor rotation direction.
- **Speed Control**: PWM signals applied to ENA (Motor A) or ENB (Motor B) pins adjust speed.
- **Motor Output**: Voltage is supplied to the motors via OUT1-OUT4 based on control signals.
- The L298N motor driver is based on the H-bridge configuration (an H-bridge is a simple circuit that lets us control a DC motor to go backward or forward.), which is useful in controlling the direction of rotation of a DC motor.
-

H-BRIDGE CIRCUIT DIAGRAM

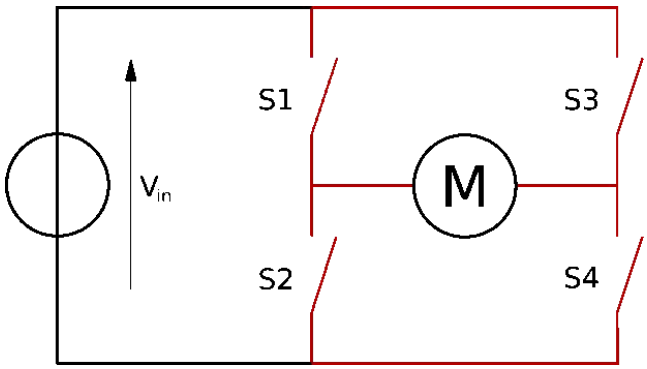


Figure 4:H Bridge ciruit

Change in the direction of rotation of the motor using h-bridge

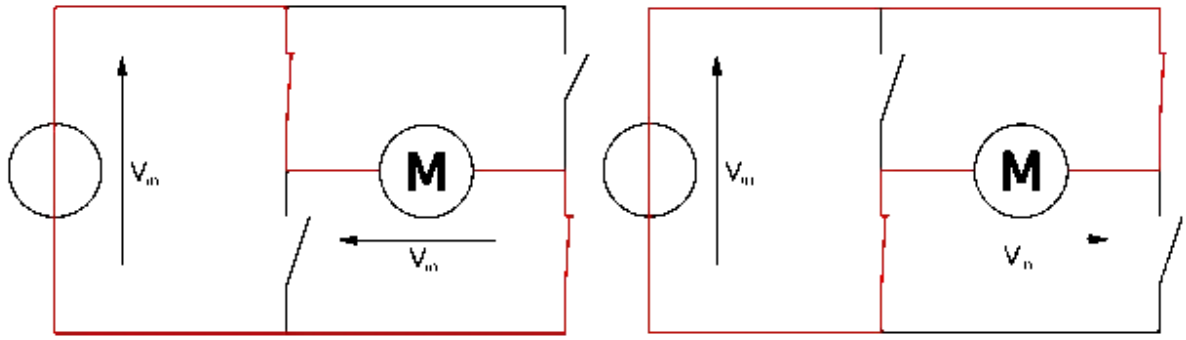


Figure 5:Change in H bridge

When S1 and S4 are ON and S2 and S3 are OFF, the left side of the motor terminal is more +ve than the other terminal. This causes the clockwise rotation of the motor.When S2, S3 are ON and S1, S4 are OFF, the right side of the motor terminal is more +ve than the left terminal. This causes anticlockwise rotation of the motor.

Input	*Input*	*Input*	*Input*	*Output*	*Output*	*Output*	*Output*	*Motors Output*		*Movement*
								Right	*Left*	
Low	High	High	Low	0	Vss	Vss	0	Straight	Straight	Straight
Low	High	Low	Low	0	Vss	0	0	Straight	No mov	Left Turn
Low	Low	High	Low	0	0	Vss	0	No mov	Straight	Right Turn
Low	High	Low	High	0	Vss	0	Vss	Straight	Reverse	Sharp Left
High	Low	High	Low	Vss	0	Vss	0	Reverse	Straight	Sharp Right
High	Low	Low	High	Vss	0	0	Vss	Reverse	Reverse	Backward

Figure 6:Table 1

Left Motor	**Right Motor**	**Robot Movement**
Straight	Straight	Straight
Stop	Straight	Left
Reverse	Straight	Sharp left
Straight	Stop	Right
Straight	Reverse	Sharp Right
Reverse	Reverse	Reverse

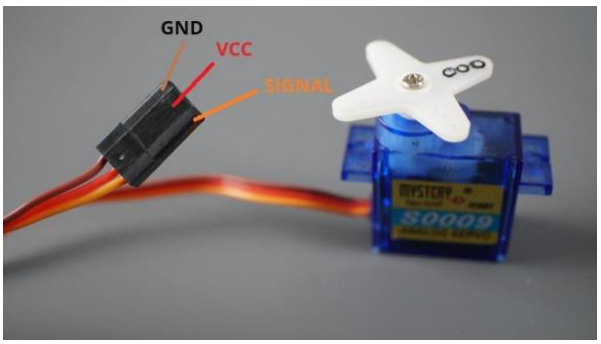
Figure 7:Table 2

WORKING OF SERVO MOTOR:

1. **Pins**:
    - **VCC**: Connects to a 5V power supply.
    - **GND**: Connects to the ground of the circuit.
    - **Signal**: Receives PWM signal from the microcontroller to control the angle.

Figure 8:servo motor

2. **Working**:
    - **PWM Signal**: The microcontroller sends a PWM signal to the signal pin, where the pulse width determines the servo's position.
        - A pulse width of ~1 ms rotates the servo to 0°.
        - ~1.5 ms sets it to 90° (middle position).



        - ~2 ms rotates it to 180°.
    - **Internal Mechanism**: The servo's internal control circuit adjusts the motor to match the position specified by the PWM signal.
    - **Continuous Adjustment**: The servo continuously monitors and maintains its position until the PWM signal changes.

This allows precise angular control, making servos ideal for tasks like steering or positioning.

PULSE WITH MODULATION:

PWM is a technique used to control the amount of power delivered to a device by varying the width of a digital signal's "ON" time (pulse width) while keeping the signal's frequency constant. It is widely used for motor control, LED dimming, and other applications requiring variable power.

Key Terms:

1. **Duty Cycle**:
    - The percentage of one cycle for which the signal is HIGH (ON).
    - Duty Cycle=ON TimeTotal Cycle Time×100%\text{Duty Cycle} = \frac{\text{ON Time}}{\text{Total Cycle Time}} \times 100 \%Duty Cycle=Total Cycle TimeON Time×100%.
    - Example: 50% duty cycle means the signal is ON half the time.
2. **Frequency**:
    - How often the PWM signal repeats in one second, measured in Hertz (Hz).

PWM Graph Description:

1. **X-Axis**: Time
2. **Y-Axis**: Voltage (HIGH/LOW states)
3. **Waveforms**:
    - **Low Duty Cycle**: Narrow HIGH pulses (low average power).
    - **Medium Duty Cycle**: Wider HIGH pulses (moderate power).
    - **High Duty Cycle**: Very wide HIGH pulses (high power).

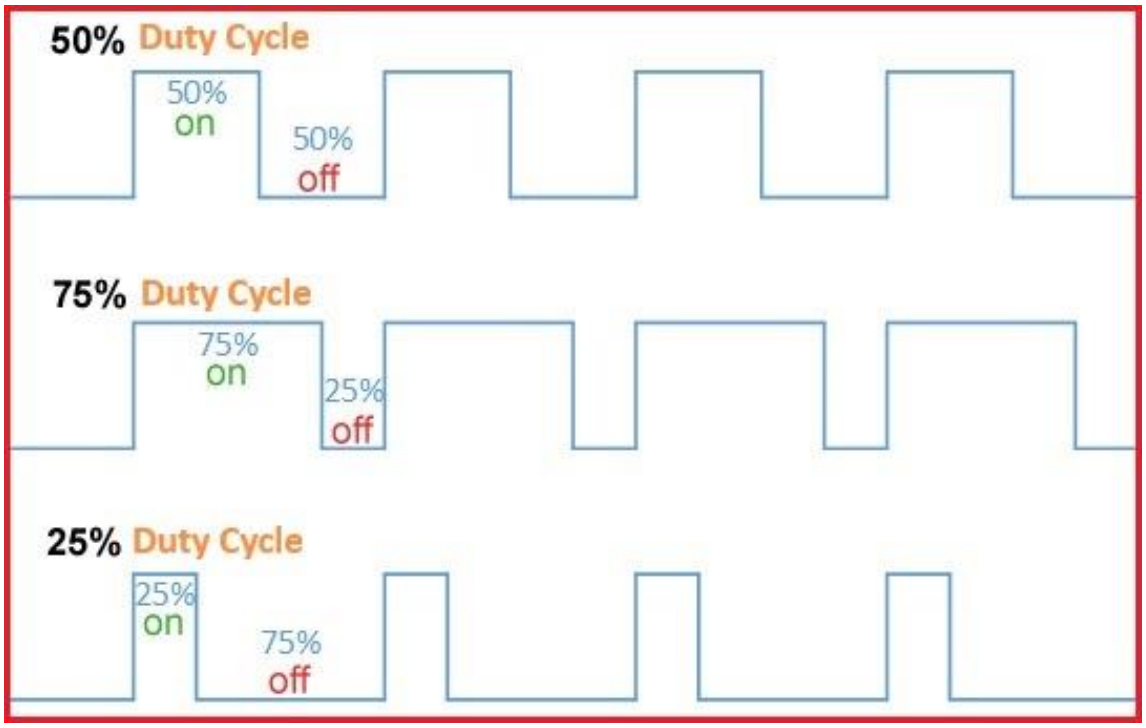
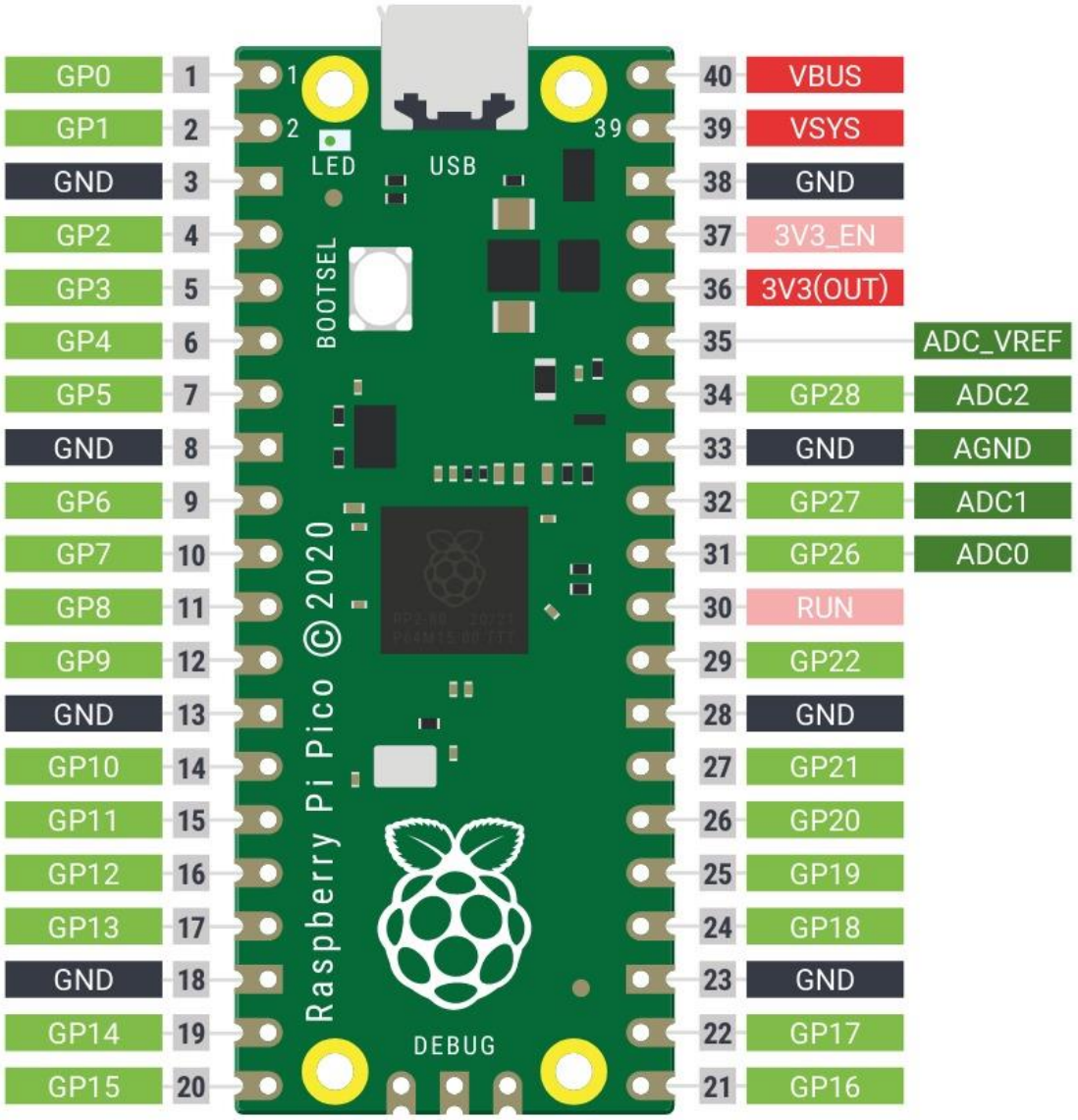


Figure 9:PWM signals

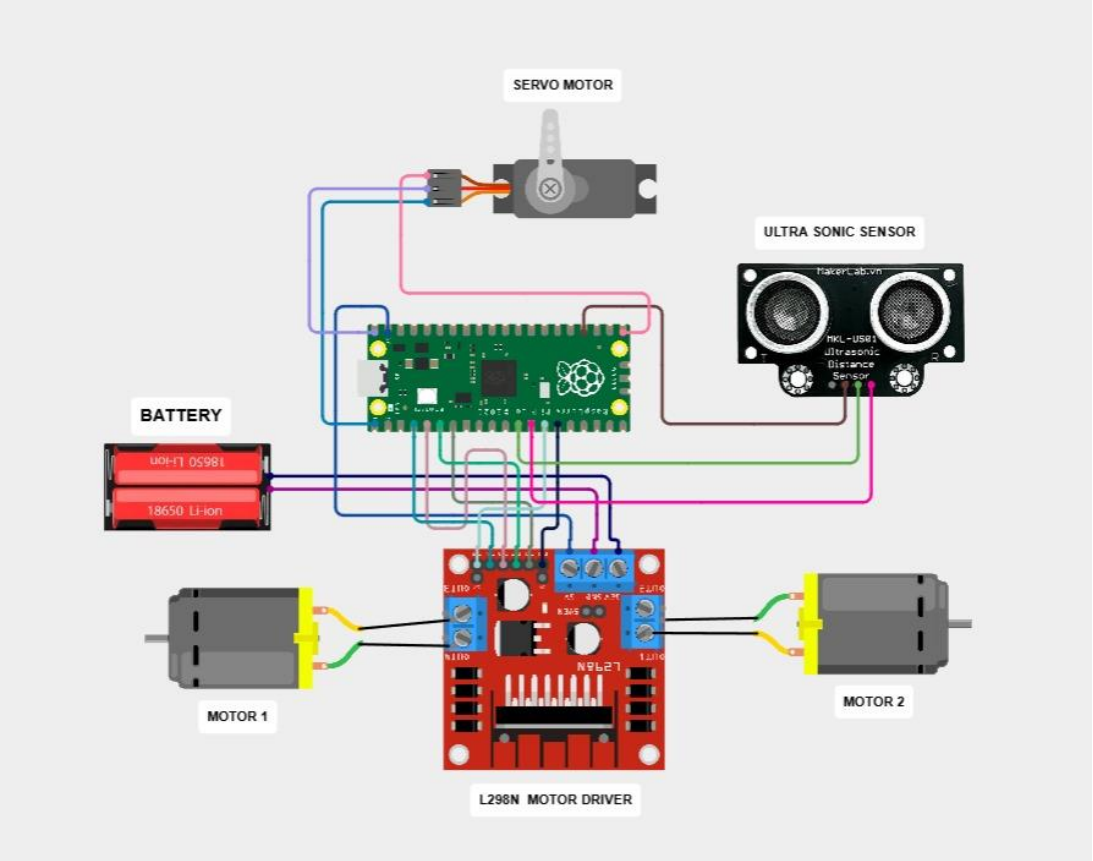
The graph above illustrates PWM signals with varying duty cycles:

1. **25% Duty Cycle**: The signal is HIGH for only 25% of the time, delivering low power.
2. **50% Duty Cycle**: The signal is HIGH for half the time, delivering moderate power.
3. **75% Duty Cycle**: The signal is HIGH for most of the time, delivering high power.

PIN DIAGRAM OF RPI2040:



BLOCK DIAGRAM:



ALGORITHM:

Step 1: **Initialization**

1. Initialize the ultrasonic sensor, servo motor, and motor driver.
2. Set the servo motor to the forward-facing position (90°).
3. Start the robot's motors for forward movement.

Step 2: **Obstacle Detection**

1. Continuously read the distance from the ultrasonic sensor.

2. If the distance is greater than the threshold (e.g., 20 cm), continue forward.
3. If the distance is less than the threshold:
    - Stop the motors.

Step 3: **Reverse**

1. Reverse the robot for a predefined time (e.g., 1 second) to move away from the obstacle.

Step 4: **Scan for Clear Path**

1. Rotate the servo to the left (e.g., 45°) and measure the distance using the ultrasonic sensor.
2. Rotate the servo to the right (e.g., 135°) and measure the distance again.

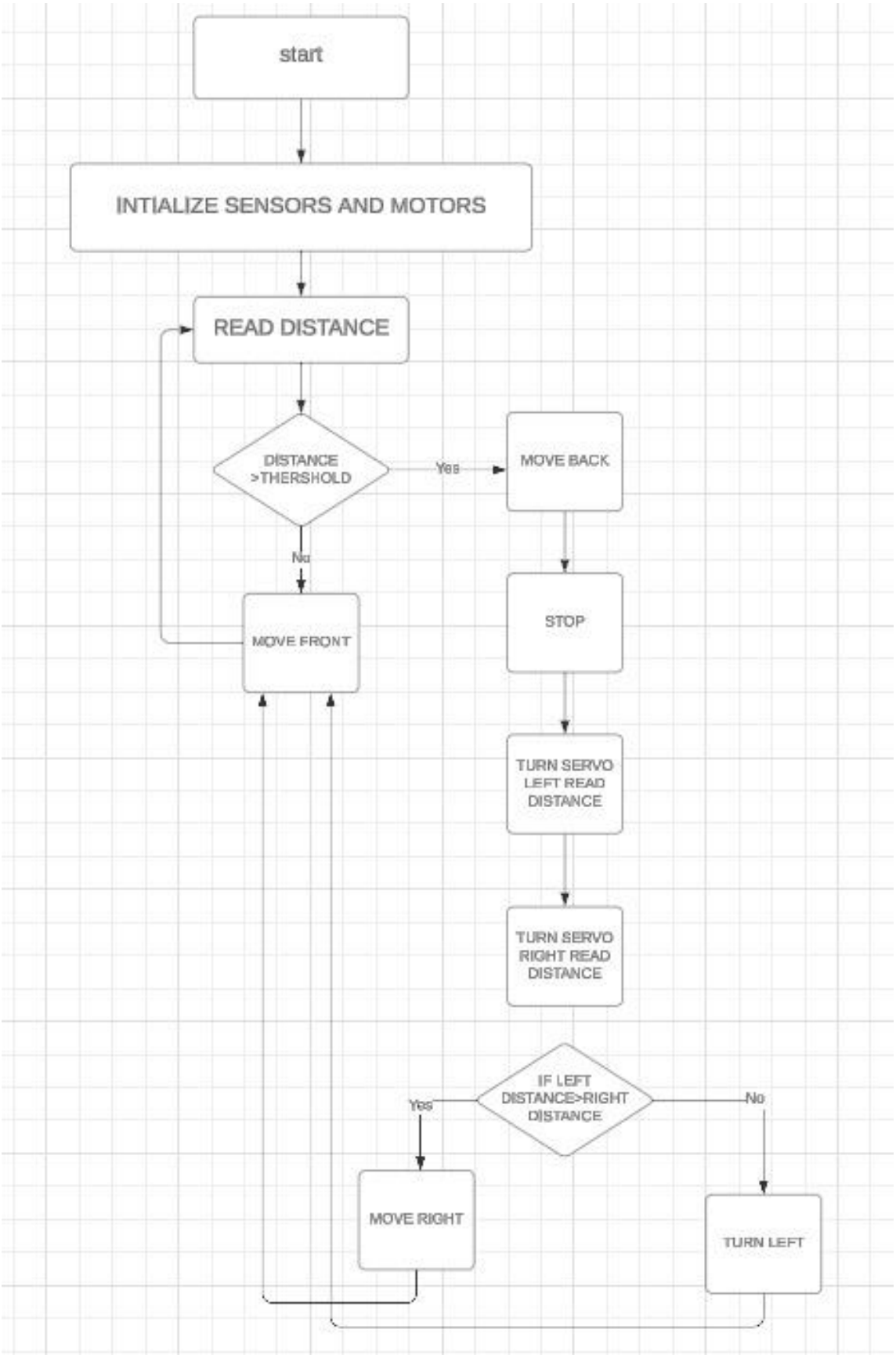
Step 5: **Decision Making**

1. Compare the left and right distances:
    - If the left distance is greater, turn the robot left and move forward.
    - If the right distance is greater, turn the robot right and move forward.
    - If both are similar, move forward in the original direction.

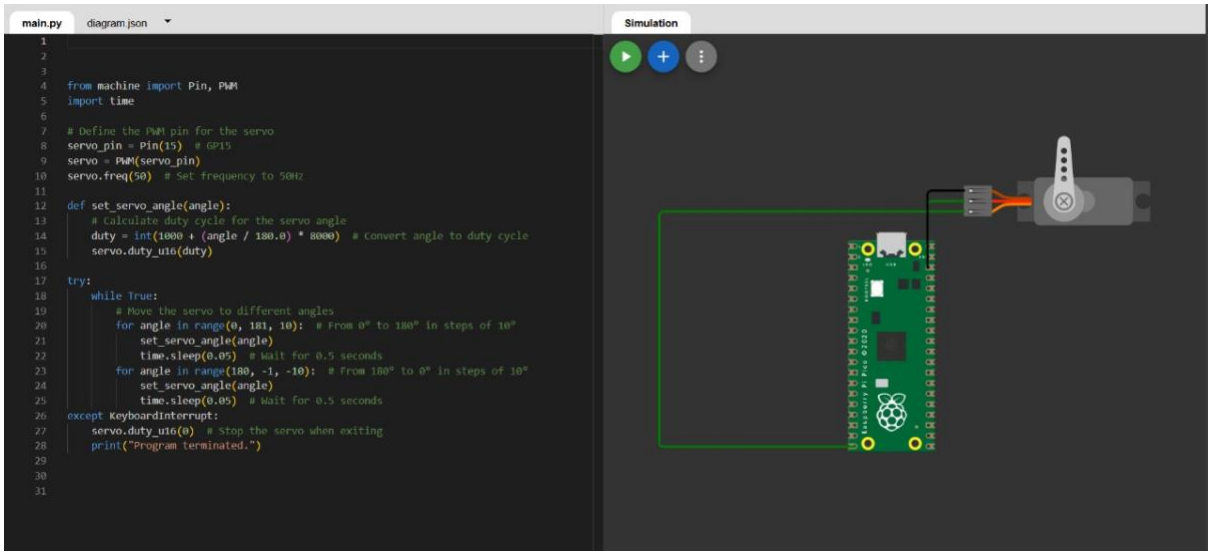
Step 6: **Repeat**

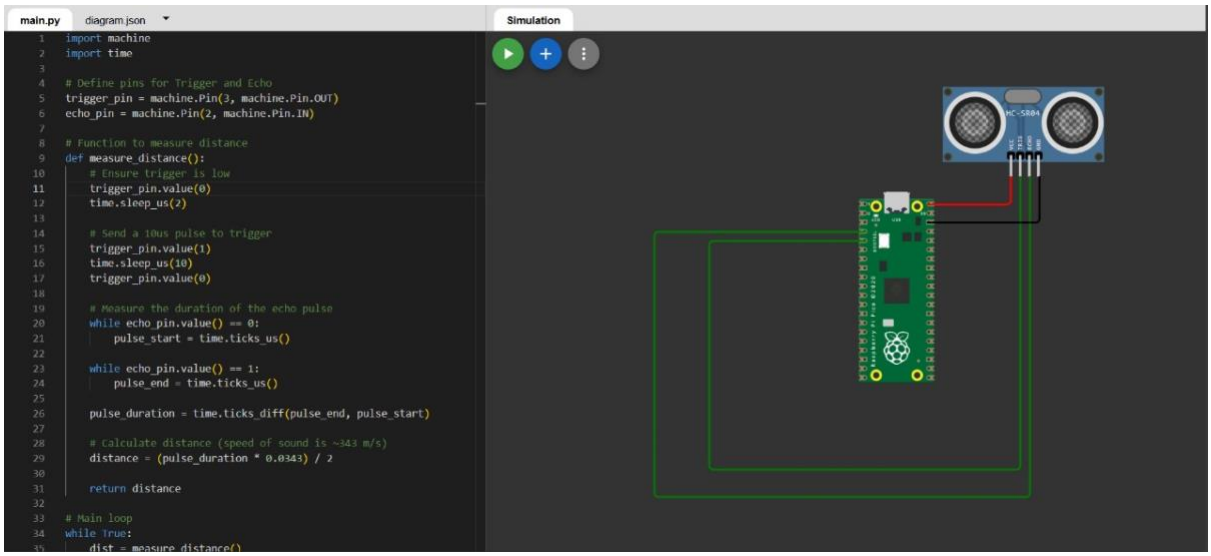
1. Reset the servo to the forward position (90°).
2. Continue obstacle detection and follow the same logic.

FLOW CHART:


start
INTIALIZE SENSORS AND MOTORS
READ DISTANCE
DISTANCE >THERSHOLD
Yes
MOVE BACK
No
MOVE FRONT
STOP
TURN SERVO LEFT READ DISTANCE
TURN SERVO RIGHT READ DISTANCE
IF LEFT DISTANCE>RIGHT DISTANCE
Yes
No
MOVE RIGHT
TURN LEFT

WokWI SIMULATION SCREENSHOT:





SIMULATION LINK:

SERVO:

https://wokwi.com/projects/412654455138816001

ULTRASONIC:
https://wokwi.com/projects/412653461367523329

YOUTUBE LINK:

https://youtu.be/u6CxA5TPE3o?si=Mne4DemZS34Q5azg

LINKDIN LINK:

PYTHON CODE:

```
from machine import Pin, PWM
import time
# Initialize the servo motor
servo = PWM(Pin(16))  # Servo pin set to 12
servo.freq(50)
# Initialize the ultrasonic sensor pins
Trig = Pin(12, Pin.OUT)  # Trig pin set to 10
Echo = Pin(13, Pin.IN)   # Echo pin set to 11
# Motor driver pins (L298N connections)
ENA = PWM(Pin(15))  # Enable pin A set to 6
IN1 = Pin(5, Pin.OUT)  # Input pin 1 set to 2
IN2 = Pin(4, Pin.OUT)  # Input pin 2 set to 3
IN3 = Pin(3, Pin.OUT)  # Input pin 3 set to 4
```

```
IN4 = Pin(2, Pin.OUT)  # Input pin 4 set to 5
ENB = PWM(Pin(14))  # Enable pin B set to 7


ENA.freq(500)
ENB.freq(500)
# Adjusted speed for lower power consumption and better response
speed = 100  # Speed of the robot (lower for safer operation)
def forward():
    ENA.duty_u16(speed)
    IN1.value(0)
    IN2.value(1)
    ENB.duty_u16(speed)
    IN3.value(1)
    IN4.value(0)
    print("Moving Forward")
def backward():
    ENA.duty_u16(speed)
    IN1.value(1)
    IN2.value(0)
    ENB.duty_u16(speed)
    IN3.value(0)
    IN4.value(1)
    print("Moving Backward")
def left():
    ENA.duty_u16(speed)
```

```
    IN1.value(1)
    IN2.value(0)
    ENB.duty_u16(speed)
    IN3.value(1)
    IN4.value(0)
    print("Turning Left")
def right():
    ENA.duty_u16(speed)
    IN1.value(0)
    IN2.value(1)
    ENB.duty_u16(speed)
    IN3.value(0)
    IN4.value(1)
    print("Turning Right")
def stop():
    ENA.duty_u16(0)
    IN1.value(0)
    IN2.value(0)
    ENB.duty_u16(0)
    IN3.value(0)
    IN4.value(0)
    print("Stopped")
# Function to get the distance from the ultrasonic sensor
def distance():
    Trig.value(0)
```

```
    time.sleep_us(2)
    Trig.value(1)
    time.sleep_us(10)
    Trig.value(0)

    while Echo.value() == 0:
        low = time.ticks_us()
    while Echo.value() == 1:
        high = time.ticks_us()
    t = high - low
    cm = t / 29 / 2  # Convert time to cm
    print(f"Measured distance: {cm:.2f} cm")
    return cm
def servo_left():
    servo.duty_u16(7500)  # Move servo to the left
    print("Servo moved left")
def servo_right():
    servo.duty_u16(1800)  # Move servo to the right
    print("Servo moved right")
def servo_start():
    servo.duty_u16(4800)  # Center the servo
    print("Servo centered")
while True:
    dis = distance()
    if dis < 15:  # Obstacle detected within 15 cm
```

```
stop()
time.sleep(1)
# Backward for a short time before turning
backward()
time.sleep(0.5)  # Reverse for half a second
stop()
time.sleep(1)
# Scan to the left
servo_left()
time.sleep(1)  # Wait for the servo to move
left_dis = distance()  # Measure left distance
print("Left Distance:", left_dis)
# Center the servo
servo_start()
time.sleep(1)  # Centering time
# Scan to the right
servo_right()
time.sleep(1)  # Wait for the servo to move
right_dis = distance()  # Measure right distance
print("Right Distance:", right_dis)
# Center the servo back
servo_start()
time.sleep(1)  # Centering time
# Decision making based on distance readings
if left_dis > right_dis:
```

```
            print("Decision: Turn Left")
            left()  # Turn left
            time.sleep(0.5)
        else:
            print("Decision: Turn Right")
            right()  # Turn right
            time.sleep(0.5)
        stop()  # Stop after turning
        time.sleep(1)  # Wait before moving again
    else:
        forward()  # No obstacles, move forward
        time.sleep(0.1)  # Short delay for forward movement
```

CONCLUSION:

The obstacle-avoiding robot successfully demonstrates the integration of sensors, actuators, and control algorithms to achieve autonomous navigation. By utilizing an ultrasonic sensor for real-time obstacle detection, a servo motor for directional scanning, and an L298N motor driver for movement control, the robot efficiently identifies and avoids obstacles. The implemented algorithm ensures smooth operation, allowing the robot to reverse and select the optimal path when faced with obstacles. This project highlights the potential of robotics in automation and sets a foundation for further enhancements, such as improved obstacle detection range, advanced path planning, and multi-sensor integration for complex environments.

REFERENCES:

Ultra sonic sensor-HC-SR04 (sparkfun.com)

Servo motor -SERVO MOTOR SG90 DATA SHEET (ic.ac.uk)

L298N Motor Driver -L298N Motor Driver.pdf (handsontec.com)

Raspberry Pi Pico- Microsoft Word - Document5 (digikey.com)