Automated Obstacle Avoidance Robot

MID-SEM PROJECT REPORT

Submitted

In the partial fulfillment of the requirements for the Mid-Term Project Assignment Evaluation of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

Aditya Shreeram [1941012071]

Soumesh Khuntia [1941012135]

UNDER THE GUIDANCE OF

Dr. Shubhendu Kumar Sarangi — **Dr. Farida A. Ali**

Assistant Professor — **Associate Professor**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH

SIKSHA 'O' ANUSANDHAN (DEEMED TO BE UNIVERSITY)

BHUBANESWAR-751003, ODISHA, INDIA.

DECEMBER-2022

DECLARATION

We hereby declare that the work described in this thesis "**Automated Obstacle Avoidance Robot**" which is being submitted in partial fulfillment for the award of **Bachelor of Technology** in the Department of **Computer Science and Engineering** affiliated to Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar (Odisha) is the result of investigations carried out by us under the Guidance of **Dr. Shubhendu Kumar Sarangi, Assistant Professor, Centre for IoT** and of **Dr. Farida A. Ali, Associate Professor**, **Department of Computer Science & Engineering**, **Institute of Technical Education and Research (ITER), Bhubaneswar.**

The work is original and has not been submitted for any Degree/Diploma of this or any other university.

Place: Bhubaneswar	**Aditya Shreeram** [1941012071]
Date: 23/12/2022	**Soumesh Khuntia** [1941012135]

CERTIFICATE

This is to certify that the project report entitled "**Automated Obstacle Avoidance Robot**" being submitted by **ADITYA SHREERAM (1941012071) & SOUMESH KHUNTIA (1941012135)** in partial fulfillment for Mid-Term Project Assignment Evaluation of **Bachelor of Technology** in the Department of **Computer Science and Engineering** affiliated to Siksha 'O' Anusandhan (Deemed to be University), Bhubaneswar (Odisha)**,** is a record of Bonafede work carried out by them during the academic year 2022-2023 under our guidance and supervision.

The results embodied in the report have not been submitted to any other University or Institution for the award of any degree or diploma.

(Dr. Shubhendu Kumar Sarangi)	**(Dr. Farida A. Ali)**
Project Guide	**Project Guide**

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Dr. Debahuti Mishra, Professor, Head of the department, Computer Science and Engineering, Institute of Technical Education and Research (ITER), Siksha ‘O’ Anusandhan (Deemed to be University), Bhubaneswar for his continued support and valuable guidance and encouragement extended to us during our project work. We thank him for his painstaking efforts to guide us throughout our work.

We thank Dr. Shubhendu Kumar Sarangi, Assistant Professor of Center for Internet of Things (CIoT), Institute of Technical Education and Research (ITER), Siksha ‘O’ Anusandhan (Deemed to be University), Bhubaneswar for his valuable advice, support and help during our project work.

We thank Dr. Farida A Ali, Associate Professor of Department of Computer Science & Engineering, Institute of Technical Education and Research (ITER), Siksha ‘O’ Anusandhan (Deemed to be University), Bhubaneswar for her valuable comments and suggestions that greatly helped in improving quality of report.

We thank Mr. Pradeep Ku. Bebrata, Mr. Bipin Ku. Karan, Ms. Mamata Sutar, Mrs Shradhanjali Nayak, Mr. Upendra Kumar Prusty, Institute of Technical Education and Research (ITER), Siksha ‘O’ Anusandhan (Deemed to be University), Bhubaneswar for their valuable comments and suggestions that greatly helped during circuit analysis and housing of the project.

We thank all our teachers and professors for their valuable comments after reviewing our project reports.

We wish to extend our special thanks to all our colleagues and friends who helped directly or indirectly to complete our project assignment work.

We extend our thanks to our parents and all our family members for their unceasing encouragement and support which helped us a lot in completing this work.

Signature of Students

ABSTRACT

An automated obstacle avoidance robot using a Raspberry Pi Pico is a small, portable device that is capable of navigating through a variety of environments while avoiding obstacles in its path. The robot is equipped with an ultrasonic sensor, which allows it to detect the presence of objects in its surroundings. The Raspberry Pi Pico, a small and powerful microcontroller, is used to process the sensor data and control the robot's movement and an L298N motor driver that powers the wheels.

The robot can move in any direction, including forward, backward, left, and right, using motors & wheels for propulsion. The robot can be programmed to follow a predetermined path or a pre-programmed sequence of actions.

Overall, the automated obstacle avoidance robot using a Raspberry Pi Pico is a versatile and cost-effective solution for a variety of applications, including search and rescue, surveillance, and environmental monitoring. It can operate in challenging environments and can be easily modified or expanded with additional sensors and capabilities.

INDEX

CHAPTER 1: INTRODUCTION

1.1 Introduction:

An automated obstacle avoidance robot is a device that can navigate through its environment while avoiding obstacles. This type of robot can be useful in a variety of applications, such as search and rescue, manufacturing, or home automation. One way to build an obstacle avoidance robot is by using a Raspberry Pi Pico, an L298N motor driver, and an ultrasonic sensor.

The Raspberry Pi Pico is a microcontroller that can be used to control the robot's movements and make decisions based on sensor input. The L298N motor driver is a device that allows the Raspberry Pi Pico to control the robot's motors, allowing it to move forward, backward, and turn. The ultrasonic sensor is used to detect obstacles in the robot's path.

To build the robot, the Raspberry Pi Pico and L298N motor driver are first connected to the robot's motors. The ultrasonic sensor is then mounted on the front of the robot and connected to the Raspberry Pi Pico. The Raspberry Pi Pico is programmed to continuously scan the environment in front of the robot and use the ultrasonic sensor to detect any obstacles. If an obstacle is detected, the Raspberry Pi Pico can then adjust the robot's movements to avoid the obstacle.

Overall, an automated obstacle avoidance robot using a Raspberry Pi Pico, L298N motor driver, and ultrasonic sensor is a simple and effective way to build a robot that can navigate through its environment while avoiding obstacles. In achieving the task, the Raspberry Pi Pico is loaded with a program written using MicroPython language.

The main objectives of the project are:

1. To design and build a robot that can autonomously navigate through a cluttered environment and avoid obstacles.
2. To integrate sensors, such as ultrasonic sensors or infrared sensors, to detect and avoid obstacles in the robot's path.
3. To use the Raspberry Pi Pico as the main controller for the robot, to process sensor data and make decisions on how to navigate through the environment.
4. To implement a control system that can accurately and smoothly steer the robot to avoid obstacles and follow a desired path.
5. To optimize the robot's performance and efficiency by fine-tuning the sensor settings and control algorithms.

1.2 Project Overview:

The "Automated Obstacle Avoidance Robot using Raspberry Pi Pico" project aims to design and build a robot that can autonomously navigate through an unknown environment while avoiding obstacles in its path.

The project explains the implementation of “**Automated Obstacle Avoidance Robot”** using Raspberry Pi Pico.

The organization of the thesis is explained here with:

Chapter 1	Presents introduction to the overall thesis and the overview of the project.
Chapter 2	Presents the topic Internet Of Things. It explains about what Internet Of Things is, Need for Internet Of Things and its applications.
Chapter 3	Presents the hardware description. It deals with the block diagram of the project and explains the purpose of each block. In the same chapter the explanation of all the components and modules used in the project.
Chapter 4	Presents the software description. It explains the implementation of the project using Thonny IDE and MicroPython.
Chapter 5	Presents the advantages, disadvantages, and applications of the project.
Chapter 6	Presents the results, conclusion, and prospects of the project.
Chapter 7	Presents how we both as a team divided the work, each of us performed and our reviews of each other.

CHAPTER 2: INTERNET OF THINGS (IOT)

2.1 Introduction:

The Internet of Things (IoT) refers to the growing network of physical devices, such as appliances, vehicles, and industrial equipment, that are connected to the internet and capable of exchanging data. These devices are equipped with sensors, software, and connectivity that allow them to collect and share data with other devices and systems.

2.2 Need:

The need for the Internet of Things arises from the increasing complexity of modern society and the desire to make our lives more convenient, efficient, and sustainable. With the IoT, we can better monitor and control the systems and devices that make up our world, from transportation and energy to healthcare and agriculture.

2.3 Applications:

- The Internet of Things has a wide range of applications across various industries and sectors. Some examples include:

- Smart homes: IoT devices can be used to automate and control various aspects of a home, such as lighting, temperature, and security.

- Transportation: IoT technologies can be used to improve the efficiency and safety of transportation systems, such as by monitoring traffic patterns and providing real-time updates to drivers.

- Healthcare: IoT devices can be used to monitor and track the health of individuals, such as by measuring vital signs and alerting caregivers to any issues.

- Agriculture: IoT sensors can be used to monitor and optimize crop growth, irrigation, and other aspects of farming.

- Manufacturing: IoT technologies can be used to improve the efficiency and productivity of manufacturing processes, such as by monitoring equipment and identifying maintenance needs.

CHAPTER 3: HARDWARE DESCRIPTION

3.1 Introduction:

In this chapter the block diagram of the project and design aspect of independent modules are considered. Block diagram is shown in fig: 3.1

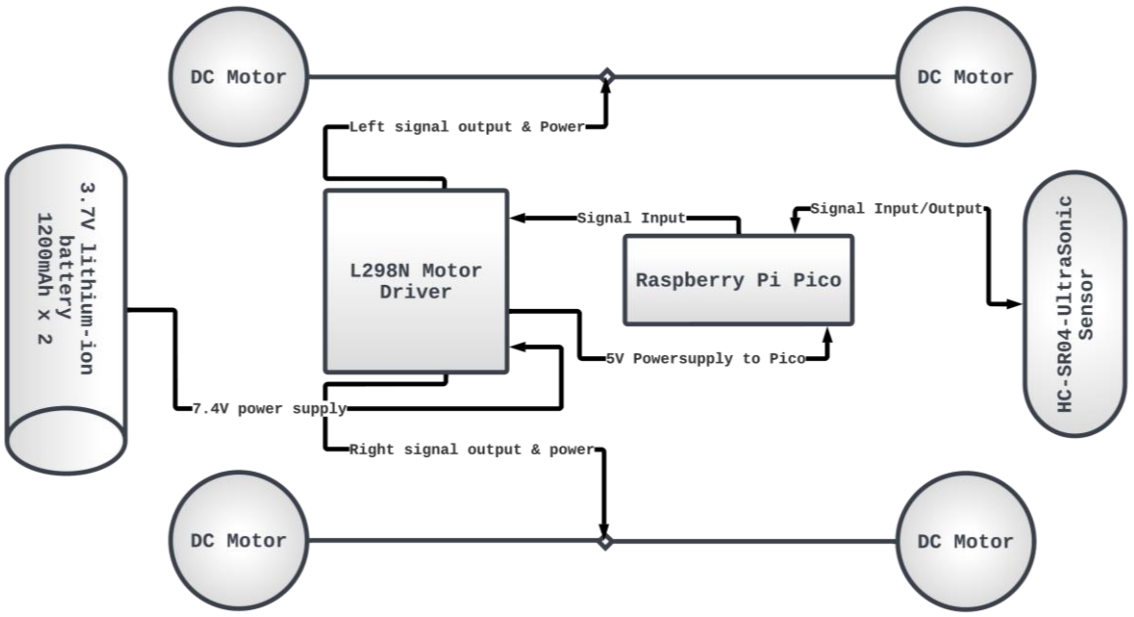


Figure 3.1: Block diagram

The main blocks of this project are:

- Raspberry Pi Pico
- HC-SR04 Ultrasonic Sensor
- L298N motor driver
- DC-motors with wheels
- Lithium-ion batteries

3.2 Raspberry Pi Pico:

Raspberry Pi Pico is a microcontroller board developed by the Raspberry Pi Foundation. It is a low-cost, high-performance device that is designed for use in a wide range of applications, including Internet of Things (IoT) projects, home automation, and educational projects. The Raspberry Pi Pico is based on the RP2040 microcontroller, which was developed by the Raspberry Pi Foundation specifically for use with the Pico. The Pico features a dual-core Arm Cortex-M0+ processor, 256KB of SRAM, and 2MB of on-board QSPI Flash memory, and can run at clock speeds of up to 133 MHz It also includes a range of connectivity options, including USB, I2C, and SPI, as well as support for various sensors and peripherals. The Raspberry Pi Pico is a powerful and versatile tool for makers, hobbyists, and educators, and is an ideal platform for learning about programming and electronics.

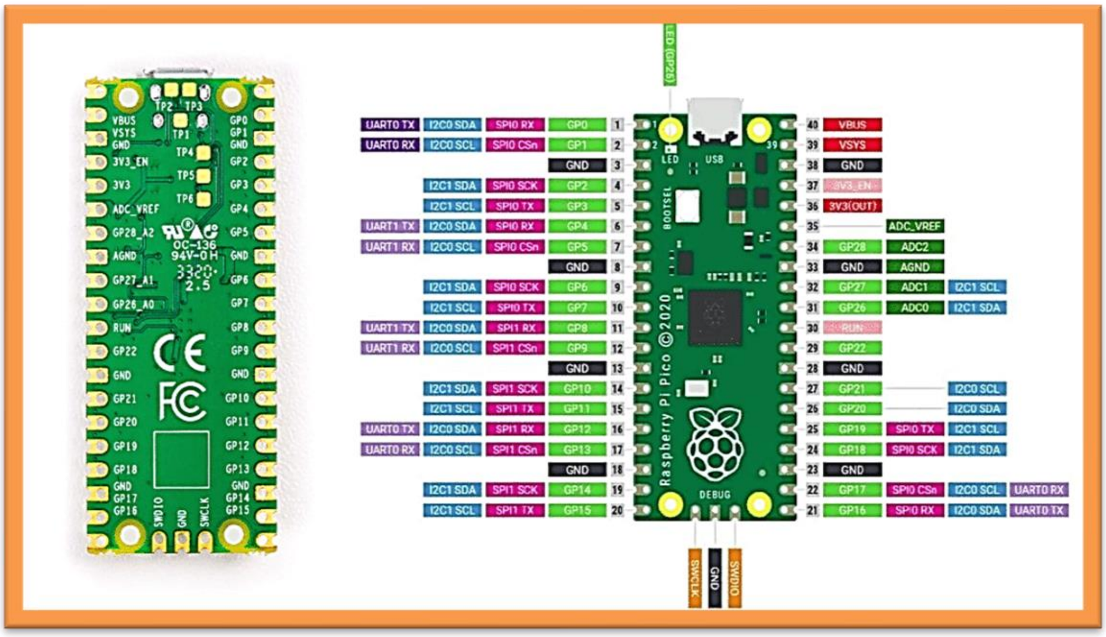


Figure 3.2: Raspberry Pi Pico & Pinout

3.3 HC-SR04 Ultrasonic Sensor:

The HC-SR04 ultrasonic sensor is a distance measurement device that uses ultrasonic waves to determine the distance to an object. It consists of a transmitter and a receiver, both of which are housed in a small plastic module. The transmitter emits an ultrasonic pulse, which is reflected off the object and received by the receiver. The time it takes for the pulse to be transmitted and received is used to calculate the distance to the object. The HC-SR04 is commonly used in robotics, automation systems, and other applications where accurate distance measurement is required. It is relatively inexpensive and easy to use, making it a popular choice for many projects.

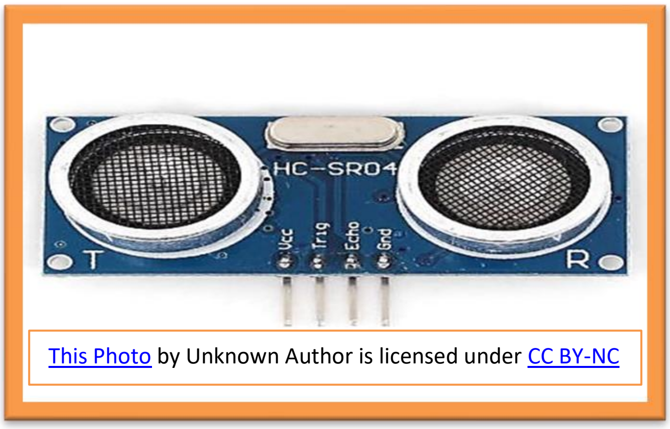


Figure 3.3:HC-SR04 Ultrasonic Sensor

3.4 L298N motor driver:

The L298N is a dual H-bridge motor driver integrated circuit (IC). It is commonly used to control the speed and direction of DC motors, as well as stepper motors and other types of actuators. The L298N can drive motors with up to 2A of current, making it suitable for use with a wide range of motors. It has a high voltage rating, with a maximum operating voltage of 46V, which makes it suitable for use in a variety of applications.

The L298N is controlled using pulse-width modulation (PWM) signals, which can be generated using a microcontroller or other digital device. The duty cycle of the PWM signals is used to control the speed of the motor, while the direction of the motor is controlled by the polarity of the applied voltage. The L298N also has built-in protection features, including overcurrent protection, thermal protection, and short-circuit protection, which help to prevent damage to the device and the motors it is driving.

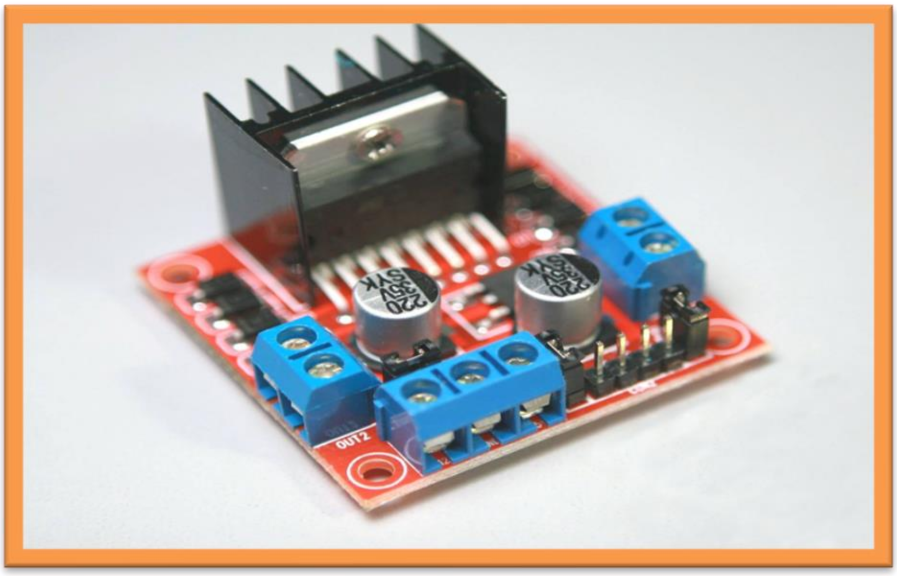


Figure 3.4: L298N motor driver

3.5 DC motors(200rpm) with wheels:

200 rpm DC motor is a type of electric motor that operates at a rotational speed of 200 revolutions per minute (rpm). DC motors are commonly used in a variety of applications, including robotics, automation systems, and portable devices, due to their versatility and relatively low cost.

200 rpm DC motors are typically small, making them well-suited for use in applications where space is limited. They are also available in a range of shapes and sizes, including round, square, and rectangular configurations, which allows them to be easily integrated into a variety of designs.

DC motors operate by using the interaction between a magnetic field and an electric current to produce motion. They are typically powered by a DC power source, such as a battery or a power supply, and can be controlled using a variety of techniques, including pulse-width modulation (PWM) and voltage control.

200 rpm DC motors are used in a wide range of applications, including conveyor systems, robotics, automation systems, and portable devices. They are particularly well-suited for use in applications where low speed and precise control are required, such as in robotic arms or precision positioning systems.

Figure 3.5: DC Motor & wheel

3.6 Lithium-ion batteries:

Lithium-ion batteries (Li-ion batteries) are a type of rechargeable battery that uses lithium ions as the main charge carrier. They are widely used in portable electronic devices, electric vehicles, and other applications due to their high energy density, low self-discharge rate, and ability to withstand many charge and discharge cycles.

Li-ion batteries consist of a cathode, an anode, and an electrolyte solution. The cathode is typically made of lithium cobalt oxide ($LiCoO2$) or lithium manganese oxide ($LiMn2O4$), while the anode is usually made of graphite. During charging, lithium ions are transferred from the cathode to the anode, and during discharge, the ions are transferred back to the cathode. This movement of ions generates a current that can be used to power electrical devices.

Li-ion batteries are known for their high energy density, which allows them to store a large amount of energy in a small package. They also have a low self-discharge rate, which means that they can hold their charge for long periods of time without significant losses. Additionally, Li-ion batteries can withstand many charge and discharge cycles, making them suitable for use in applications where the battery will be regularly charged and discharged.

Overall, Li-ion batteries are a popular choice for portable electronic devices, electric vehicles, and other applications due to their high energy density, low self-discharge rate, and ability to withstand many charge and discharge cycles.

Figure 3.6: Lithium-ion Battery

3.7 Circuit Diagram

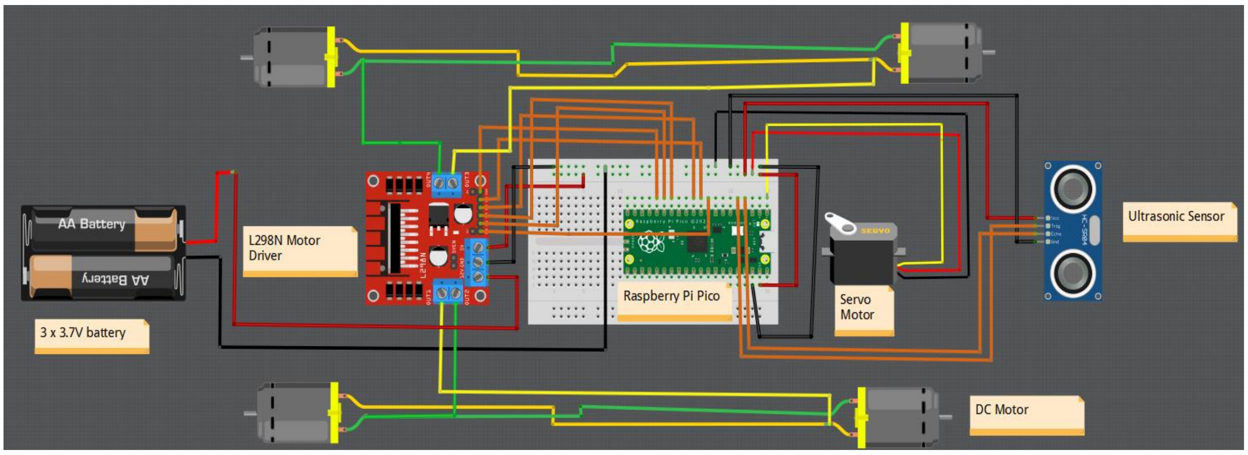


Figure 3.7

CHAPTER 4: SOFTWARE DESCRIPTION

4.1 Micro Python:

Micro Python is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimized to run on microcontrollers and in constrained environments.

MicroPython is packed full of advanced features such as an interactive prompt, arbitrary precision integers, closures, list comprehension, generators, exception handling and more. Yet it is compact enough to fit and run within just 256k of code space and 16k of RAM.

MicroPython aims to be as compatible with normal Python as possible to allow you to transfer code with ease from the desktop to a microcontroller or embedded system.

MicroPython is a full Python compiler and runtime that runs on bare metal. You get an interactive prompt (the REPL) to execute commands immediately, along with the ability to run and import scripts from the built-in filesystem. The REPL has history, tab completion, auto-indent and paste mode for a great user experience.

MicroPython strives to be as compatible as possible with normal Python (known as CPython) so that if you know Python you already know MicroPython. On the other hand, the more you learn about MicroPython the better you become at Python.

In addition to implementing a selection of core Python libraries, MicroPython includes modules such as "machine" for accessing low-level hardware.

4.2 Thonny:

Thonny is a new Python IDE for learning and teaching programming that can make program visualization a natural part of the beginners' workflow. Among its prominent features are different ways of stepping through the code, step-by-step expression evaluation, intuitive visualization of the call stack and mode for explaining the concepts of references and heap. It supports educational research by logging user actions for replaying or analyzing the programming process. It is free to use and open for extension.

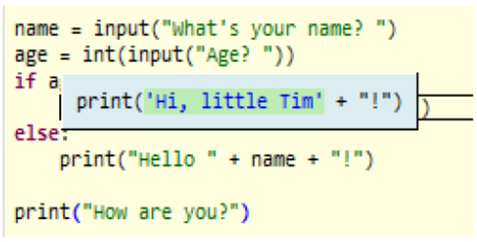
Features:

- **Easy to get started:** Thonny comes with Python 3.10 built in, so just one simple installer is needed and you're ready to learn programming. (You can also use a separate Python installation, if necessary.) The initial user interface is stripped of all features that may distract beginners.

- **No-hassle variables:** Once you're done with hello-worlds, select View → Variables and see how your programs and shell commands affect Python variables.

- **Simple debugger:** Just press Ctrl+F5 instead of F5 and you can run your programs step-by-step, no breakpoints needed. Press F6 for a big step and F7 for a small step. Steps follow program structure, not just code lines.

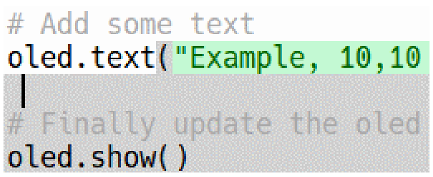
- **Step through expression evaluation:** If you use small steps, then you can even see how Python evaluates your expressions. You can think of this light-blue box as a piece of paper where Python replaces subexpressions with their values, piece-by-piece.



```
name = input("What's your name? ")
age = int(input("Age? "))
if a
    print('Hi, little Tim' + "!")
else:
    print("Hello " + name + "!")

print("How are you?")
```

- **Highlights syntax errors:** Unclosed quotes and parentheses are the most common beginners' syntax errors. Thonny' s editor makes these easy to spot.



```
# Add some text
oled.text("Example, 10,10

# Finally update the oled
oled.show()
```

- **Explains scopes:** Highlighting variable occurrences reminds you that the same name doesn't always mean the same variable and helps spotting typos. Local variables are visually distinguished from global variables.

- **Beginner friendly system shell:** Select Tools → Open system shell to install extra packages or learn handling Python on command line. PATH and conflicts with other Python interpreters are taken care of by Thonny.

- **Simple and clean pip GUI:** Select Tools → Manage packages for even easier installation of 3rd party packages

CHAPTER 5: ADVANTAGES, DISADVANTAGES & APPLICATIONS

5.1 Advantages:

- **Enhanced adaptability:** By using sensors and algorithms to detect and avoid obstacles, an automated obstacle avoidance robot can adapt to changes in its environment more easily than a robot without such capabilities. This can be especially useful in environments where the layout or objects may change frequently, as the robot can easily adapt to these changes and continue to operate effectively.

- **Reduced human intervention:** An automated obstacle avoidance robot can operate more independently than a manually operated robot, reducing the need for human intervention. This can be especially useful in situations where it is impractical or unsafe for a human operator to be present, such as in hazardous or remote environments.

- **Increased reliability:** By using sensors and algorithms to navigate its environment, an automated obstacle avoidance robot can be more reliable than a manually operated robot. This can be especially useful in applications where the robot needs to operate consistently and accurately, such as in manufacturing or inspection tasks.

- **Improved efficiency:** Automated obstacle avoidance can allow a robot to navigate its environment more quickly and efficiently, reducing the time and energy required to complete tasks. This can be especially useful in environments where the robot needs to navigate around many obstacles or in situations where manual operation would be slow or difficult.

- **Increased safety:** By using sensors and algorithms to avoid obstacles, an automated obstacle avoidance robot can navigate its environment more safely than a manually operated robot or a robot without obstacle avoidance capabilities. This can reduce the risk of collisions and injuries to both the robot and any humans or objects in its path.

5.2 Disadvantages:

- **Cost:** Implementing obstacle avoidance capabilities in a robot can be expensive, as it requires specialized sensors and algorithms. This can make automated obstacle avoidance robots more expensive to purchase and maintain than robots without such capabilities.

- **Complexity:** Implementing obstacle avoidance capabilities in a robot can be complex, requiring advanced programming and integration of sensors and algorithms. This can make it more difficult to design, build, and maintain automated obstacle avoidance robots, especially for those with limited programming experience.

- **Limited flexibility:** While automated obstacle avoidance can allow a robot to adapt to changes in its environment, it may also limit the robot's flexibility in certain situations. For example, a robot with obstacle avoidance capabilities may be unable to navigate through tight or cluttered spaces, as it may be unable to detect or avoid obstacles in these environments.

- **Dependence on sensors:** An automated obstacle avoidance robot relies on its sensors to detect and avoid obstacles, and these sensors can be prone to errors or malfunctions. This can lead to incorrect or incomplete data being fed to the robot's algorithms, resulting in incorrect or suboptimal behavior.

- **Limited ability to adapt to new environments:** While automated obstacle avoidance can allow a robot to adapt to changes in its known environment, it may not be able to adapt to completely new or unknown environments. In these situations, the robot may be unable to accurately detect or avoid obstacles, potentially leading to collisions or other problems.

5.3 Applications:

- There are many potential uses for an automated obstacle avoidance robot. Some possible applications include:
    - Industrial inspection: The robot could be used to inspect hard-to-reach or hazardous areas in a factory or other industrial setting.
    - Search and rescue: The robot could be deployed in disaster areas or other dangerous environments to search for survivors or collect data.
    - Military operations: The robot could be used to perform reconnaissance or other tasks in a military setting.
    - Educational demonstrations: The robot could be used to teach students about robotics and programming concepts.

CHAPTER 6: RESULTS

6.1 Results:

Upon completion of this project, we expect to have a fully functional automated obstacle avoidance robot that can navigate through a given environment while avoiding any obstacles that may be present. We will test the robot's performance in a variety of situations and environments to ensure that it is able to effectively avoid obstacles and reach its destination.

6.2 Conclusion:

In conclusion, the automated obstacle avoidance robot that we have designed and built using the Raspberry Pi Pico and micropython is a powerful and versatile device that is capable of navigating through a given environment while avoiding obstacles. While there are some limitations to the capabilities of the robot, it is a valuable tool that can be used in a variety of applications.

6.3 Prospects:

The performance of the system can be further improved in terms of the operating speed, more sensors can be interfaced by using advanced versions of controllers. This system can be widely used anywhere regarding the transportation of objects, exploration , and in other industries.

CHAPTER 7: TEAMWORK

7.1 SUMMARY OF TEAMWORK

7.1.1 ATTRIBUTES

1	Attends group meetings regularly and arrives on time.
2	Contributes meaningfully to group discussions.
3	Completes group assignments on time.
4	Prepares work in a quality manner.
5	Demonstrates a cooperative and supportive attitude.
6	Contributes significantly to the success of the project.

7.1.2 SCORE

1=strongly disagree; **2**=disagree; **3**=agree; **4**=strongly agree

Student 1: ____________________________________

Student 2: ____________________________________

	Evaluated by	
Student 1	Attributes	Student 2
	1	
	2	
	3	
	4	
	5	
	6	
	Grand Total	

	Evaluated by	
Student 2	Attributes	Student 1
	1	
	2	
	3	
	4	
	5	
	6	
	Grand Total	

Signature of

Student 1

Signature of

Student 2

References

The sites which were used while doing this project include:

i. www.robu.in
ii. www.google.com
iii. www.youtube.com
iv. www.allaboutcircuits.com
v. www.howstuffworks.com
vi. www.hackster.io
vii. www.instructables.com
viii. www.raspberrypi.org

Appendix

CODE:

```
import utime

from machine import Pin, time_pulse_us


# Constants

ECHO_TIMEOUT = 3000

DELAY = 0.01

TURN_DELAY = 0.1

MIN_DISTANCE = 15


# Motor control pins

IN1 = Pin(0, Pin.OUT)

IN2 = Pin(1, Pin.OUT)

IN3 = Pin(2, Pin.OUT)

IN4 = Pin(3, Pin.OUT)


# Ultrasound sensor pins

echo = Pin(14, mode=Pin.IN, pull=None)

trigger = Pin(15, mode=Pin.OUT, pull=None)


# Motor control functions

def forward():

    IN1.low()

    IN2.high()

    IN3.low()
    IN4.high()

def right():
    IN1.low()
    IN2.high()
    IN3.high()
    IN4.low()

def left():
    IN1.high()
    IN2.low()
    IN3.low()
    IN4.high()

def back():
    IN1.high()
    IN2.low()
    IN3.high()
    IN4.low()

# Ultrasound sensor function
def get_distance():
    trigger.value(0)
    utime.sleep(0.05)
    trigger.value(1)
    utime.sleep(0.05)
    trigger.value(0)
    pulse_time = time_pulse_us(echo, 1, ECHO_TIMEOUT)
    cms = (pulse_time / 2) / 29.1
    if cms > 0:
        return cms
    return -1

# State machine
state = "FORWARD"
while True:
    distance = get_distance()
    print(distance)

    if state == "FORWARD":
        if distance <= MIN_DISTANCE and distance != -1:
            state = "OBSTACLE_AHEAD"
        else:
            forward()
    elif state == "OBSTACLE_AHEAD":
        back()
        utime.sleep(TURN_DELAY)
        state = "TURN_RIGHT"
    elif state == "TURN_RIGHT":
        right()
        utime.sleep(TURN_DELAY)
        state = "FORWARD"

    utime.sleep(DELAY)
```