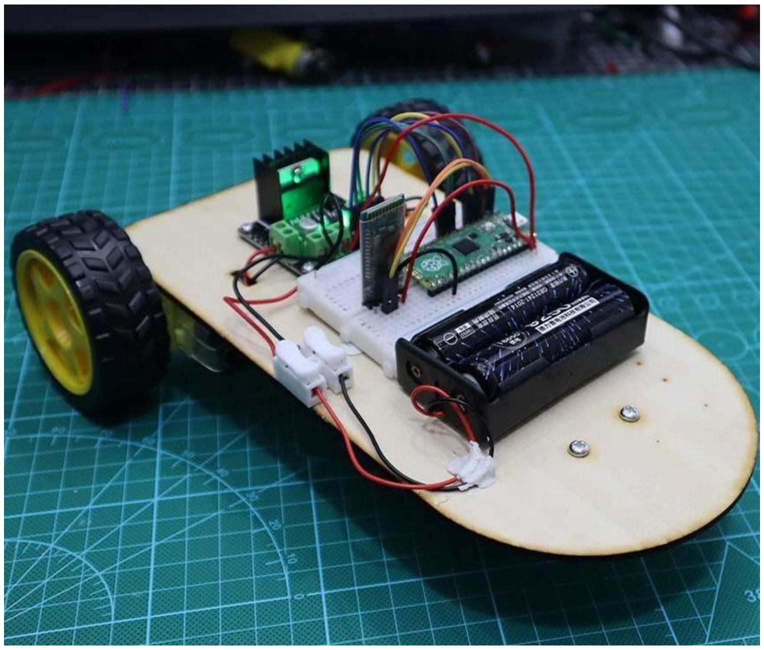
VIT-AP UNIVERSITY

Final Project Report:

Bluetooth Control Car Using Raspberry Pi Pico



UNDER THE GUIDANCE OF:

Prof. Eswaraiah Rayachoti

Submitted by:

S.No	STUDENT NAME	REG NUMBER
1.	Adithya Venkat Kumar	21MIC7037
2.	Janardhan Dutta	21MIC7042
3.	Shaik Sahir	21MIS7007
4.	Tasneem Farhana	21MIS7033
5.	Rahul	21BCE9953
6.	Mohan Kalyan	21BCE9959

INDEX

S.NO	NAME
1.	Abstract and Introduction
2.	Components Used
3.	Circuit Diagram
4.	Block Diagram
5.	Mobile Application Flow Chart
6.	Background and Problem Definition
7.	Objectives and Problems
8.	Procedure and Results
9	Discussions and Conclusions
10.	Code
11.	References

Abstract

In this project, we demonstrate how a computer, through efficient electronic programming, can control a robot. A robot can be considered an electromechanical machine with essential characteristics like sensing, movement, energy, and intelligence. It performs tasks using control systems, various power supplies, and software working together. We developed an Android application to guide an RC car's motion using remote buttons. The mobile device harboring the Android application acts as the car's remote control, with Bluetooth facilitating communication between the controller and the Android application.

Introduction

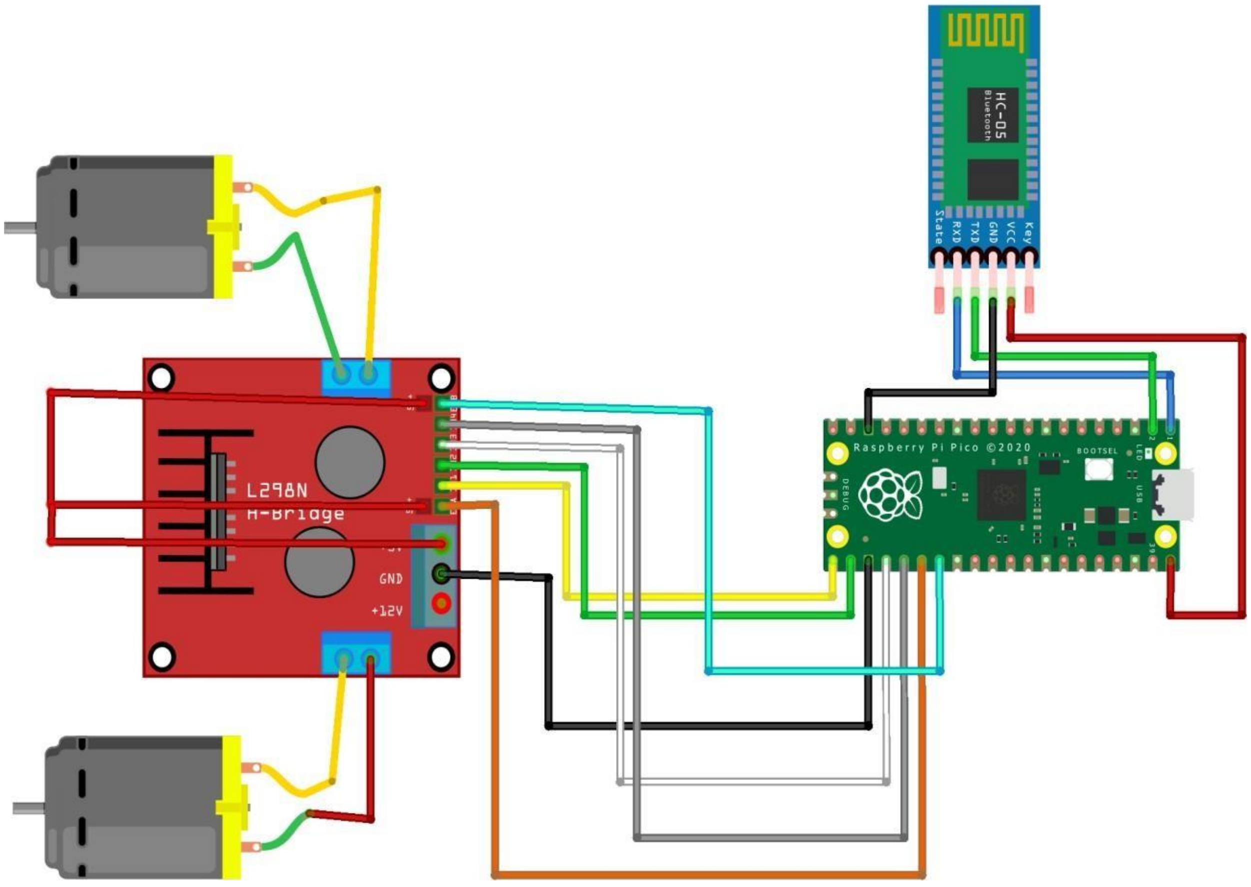
Robots have always fascinated students, hobbyists, and DIY enthusiasts. For beginners, building a robot (like a car or an arm) is a crucial project after learning the basics. This project involves implementing a Bluetooth Controlled Robot using a Raspberry Pi Pico and a few other components to build a simple robotic car controlled using an Android Phone over Bluetooth Communication. The robotic car can be controlled wirelessly via a Smartphone with an Android app, allowing the user to send commands directly to the robot, which can move forward, backward, left, and right, and can also be stopped.

The Raspberry Pi Pico's Bluetooth-controlled robot car is interfaced with a Bluetooth module (HC-05 or HC-06). A Bluetooth transceiver module at the receiving end receives commands and forwards them to the Raspberry Pi Pico, thus controlling the robotic car.

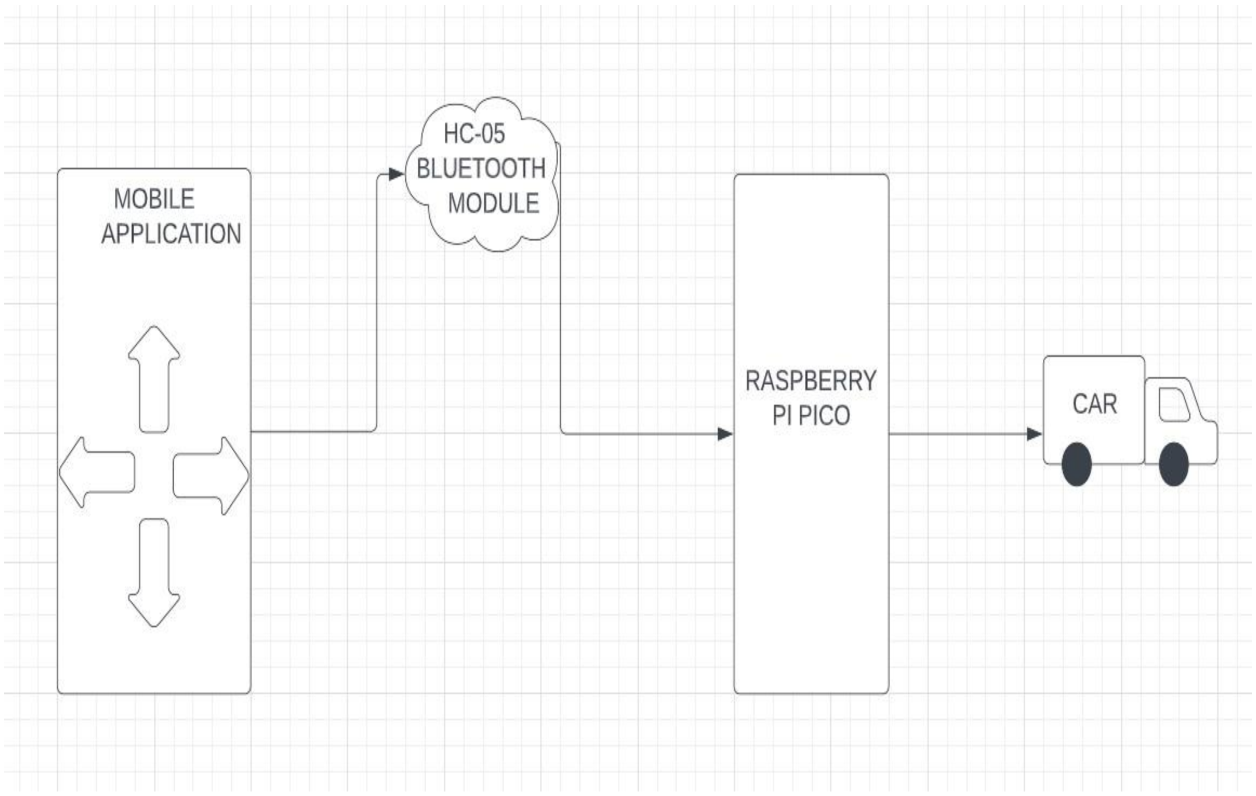
Components Used in the Project

1. Raspberry Pi Pico (124 MB RAM)
2. L298N Motor Driver Controller Board
3. DC Electric Motor along with Plastic TT MotoTire Wheel
4. Male-Female, Male-Male Jumpers
5. Single Wheel Light Duty Swivel Plate
6. Nuts and Bolts Kit
7. Electrical Cable Connectors
8. 1 * 2m - 20 AWG Single Core Wire

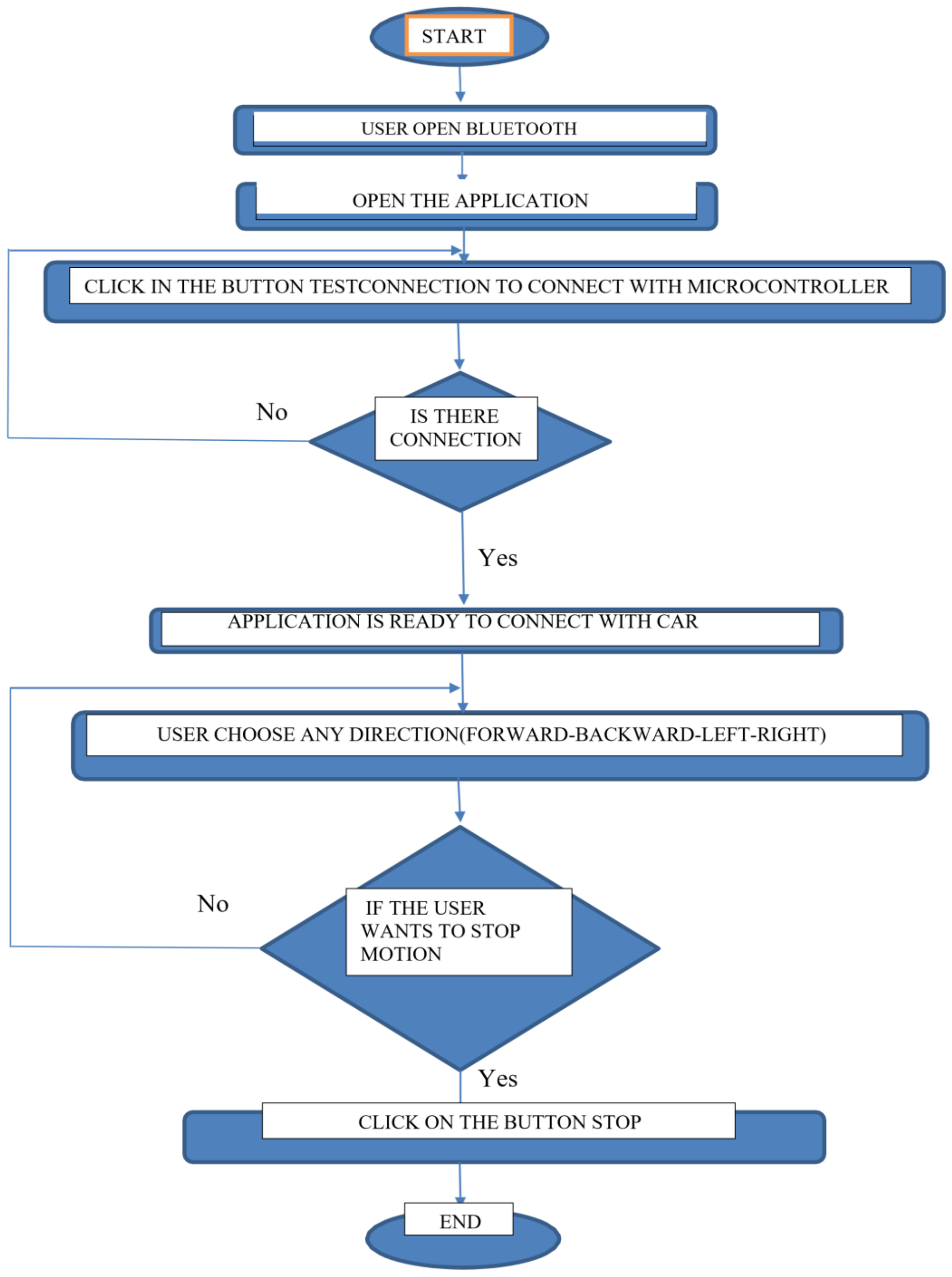
Circuit Diagram



Block Diagram



Mobile Application Flow Chart


START
USER OPEN BLUETOOTH
OPEN THE APPLICATION
CLICK IN THE BUTTON TESTCONNECTION TO CONNECT WITH MICROCONTROLLER
No
IS THERE CONNECTION
Yes
APPLICATION IS READY TO CONNECT WITH CAR
USER CHOOSE ANY DIRECTION(FORWARD-BACKWARD-LEFT-RIGHT)
No
IF THE USER WANTS TO STOP MOTION
Yes
CLICK ON THE BUTTON STOP
END

Background

Smartphones have become a sensation worldwide, boasting the largest margins in the tech sector. Bluetooth, one of its critical functions, is used in this project to control a three-wheeler car. This Bluetooth-controlled car can assist in various day-to-day tasks, acting as a helpful tool.

Problem Definition

A Bluetooth Controlled Car is operated via a wireless medium. Here, we utilize Bluetooth from any Bluetooth-enabled device around us, using both computers and mobile phones to run the mechanically constructed vehicle.

Objectives

- Control a robot car using a smartphone.
- Design a system that performs specific functions.
- Develop an Android application.
- Interface a module with a microcontroller.

Problems

- Car front wheels have a balance problem, with one wheel higher than the other.
- The Android application is slow and occasionally stops working, requiring a restart.

Procedure

The steps for creating this project are as follows:

1. Soldering the Raspberry Pi Pico board with Hydra pins.
2. Inserting code into the board using a USB cable.
3. Making all the connections of the Bluetooth module, Motor control board, and Raspberry Pi Pico on the breadboard using jumper wires (Male to Male, Male to Female, and Female to Female).
4. Fixing the car's body with nuts and bolts.
5. Attaching the breadboard with connections to the body.
6. Providing the DC motors with a power supply using batteries.
7. Running the car by switching on the power supply and controlling it with a mobile remote designed using MIT App Inventor.

Results

The project's results are encouraging. Commands are received from the application by the car's Bluetooth module and performed instantly.

Positive aspects of this project include:

- Bluetooth control mode offers the operator freedom to move the car anywhere within its range.
- The simple Android application can be installed on any Android-supported smartphone, making it accessible to anyone.
- This wireless control vehicle can be useful in daily tasks by adding extensions, like a robotic arm, to assist workers.

Discussions

- Bluetooth is effective for short-range communication (<100m) but not suitable for longer ranges.
- Bluetooth technology consumes significant power, quickly draining the battery.
- When using more than 12V of power, motors rotated due to high current. This was controlled by using a voltmeter and supplying 11.5 volts.

Conclusions

This project has successfully demonstrated a Bluetooth-controlled car using a Raspberry Pi Pico. It has various practical applications, such as:

- Children can play with this prototype.
- It can help carry small objects with an attached carrier.
- In offices, it can transport large amounts of files.
- It can be used in hardware games controlled by software.
- In shopping malls, customers can use it as a smart trolley.
- In specific scenarios, it can serve as a model savior kit.

Code

```
from machine import Pin, PWM, UART  # Importing PIN, PWM, and UART modules

import time  # Importing time module


# Defining UART channel and Baud Rate

uart = UART(1, 9600)


# Defining motor pins

motor1 = Pin(11, Pin.OUT)

motor2 = Pin(12, Pin.OUT)

motor3 = Pin(13, Pin.OUT)

motor4 = Pin(14, Pin.OUT)


# Defining enable pins and PWM object

enable1 = PWM(Pin(10))

enable2 = PWM(Pin(15))


# Defining frequency for enable pins

enable1.freq(1000)
```

```
enable2.freq(1000)


# Setting maximum duty cycle for maximum speed (0 to 65025)

enable1.duty_u16(65025)

enable2.duty_u16(65025)


# Forward

def move_forward():

    motor1.high()

    motor2.low()

    motor3.low()

    motor4.high()


# Backward

def move_backward():

    motor1.low()

    motor2.high()

    motor3.high()

    motor4.low()
```

```
# Turn Right

def turn_right():

    motor1.low()

    motor2.high()

    motor3.low()

    motor4.high()


# Turn Left

def turn_left():

    motor1.high()

    motor2.low()

    motor3.high()

    motor4.low()


# Stop

def stop():

    motor1.low()

    motor2.low()

    motor3.low()

    motor4.low()
```

```
while True:
    if uart.any():  # Checking if data is available
        data = uart.read()  # Getting data
        data = str(data)  # Converting bytes to string
        print(data)
        if 'F' in data:
            move_forward()  # Forward
        elif 'B' in data:
            move_backward()  # Backward
        elif 'R' in data:
            turn_right()  # Turn Right
        elif 'L' in data:
            turn_left()  # Turn Left
        elif 'S' in data:
            stop()  # Stop
        elif 'E' in data:
            speed = data.split("|")
            print(speed[1])
            enable1.duty_u16(int(speed[1]))  # Setting Duty Cycle
            enable2.duty_u16(int(speed[1]))  # Setting Duty Cycle
        else:
            stop()  # Stop
```

References

1. YouTube
2. Stack Overflow

The End