**Project Title**

**BLUE-DRIVE ROVER**

**BLUETOOTH CONTROLLED CAR**

***TABLE OF CONTENTS***

**S.NO NAME PG.NO**

**1. ABSTRACT 3**

**2. INTRODUCTION 3-4**

**3. COMPONENTS 5-6**

**4. DESIGN 6-7**

**5. SYSTEM SPECIFICATION 7**

**6. WORKING 7-9**

**7. CODE 9-18**

**8. RESULT 18**

**9. APPLICATIONS 19**

**10. CONCLUSION 19**

**11. FUTURE SCOPE 20**

***ABSTRACT***

Robots are indispensable in many manufacturing industries.

Once programmed, robots repeatedly perform functions

with a high accuracy that surpasses that of the most

experienced human operator. Human operators, however,

remain more versatile as reprogramming and parts

replacement may be required in order to alter the nature of

operation of robots when tasks change.

Robots were introduced due to development in the

technological world to help humans perform tasks or

assignments accurately, continuously and tirelessly in a

dangerous environment.

***INTRODUCTION***

This project presents the design and construction of a Bluetooth talking vehicle by the students of Computer Science Department. It describes the working and execution of the project extending the student’s education in electronics, programming and physics and, at the same time, provides resources for experimental purposes.

In this project we have planned a Bluetooth controlled vehicle which can assume voice command and it is named as BLUE DRIVE ROVER. The proposed framework will be able to enable the movement as indicated by the orders supplied through the application and supply them to the vehicle through a wireless system. Identification of speech is also called as "automatic speech recognition (ASR)". The Bluetooth module will get the order and given it to microcontroller(which is arduino in this case), where everything the handling work of the voice is finished also, permit vehicle to move as per voice order; i.e., move forward, move backward, turn left, turn right, and stop.

MOTOR 2

MOTOR 1

MOTOR 3

MOTOR 4

BATTERY

MOTOR DRIVER SHEILD

ARDUINO UNO

BLUETOOTH MODULE

This paper showes how to control robot controlled vehicle using Bluetooth module through android application of an android Smart Phone.

This project is fit for perceiving the voice order given by the client and play out specific activity such as movement in the required direction, or to process the orders via voice and complete the tasks or give visuals through the camera installed in the vehicle.

It proposes the plan of a robot that can be controlled utilizing an application running on an android phone. It sends control request through Bluetooth which has specific features like controlling the speed of the motor, identifying and offering the information to mobile about the course.

This was a sort of robot can be useful for versatility help for older and Incapacitated individuals.

HARDWARE:

Arduino UNO

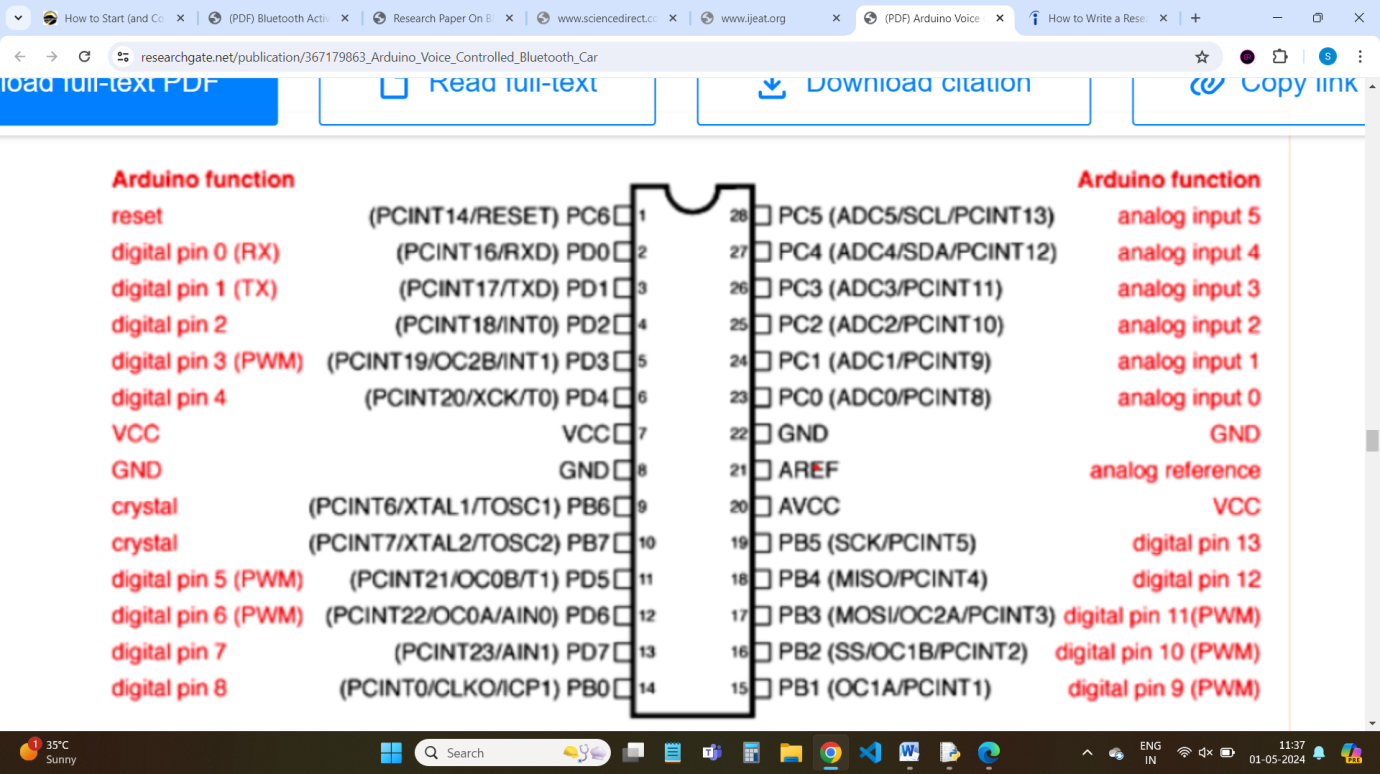
The Arduino Uno is an open-source microcontroller Consisting of 14 advanced I/O pins ,6 Analog I/O sticks, and is programmable with the Arduino IDE (Integrated Development Environment), through a B-type USB cable. It can be fueled by the USB link or by an outside 9-volt battery

Arduino Uno Pin Mapping:

Of the 32 pins accessible on the UNO board, 22 pins are related with information and result. In that, 14 pins (D0 to D13) are valid computerized I/O pins, which can be arranged according to our application. Every one of these Digital I/O pins are fit for obtaining or sinking 20mA of current (most extreme 40mA is permitted.

A significant point about Analog Input pins is that they can be arranged as Digital I/O pins, whenever required. Advanced I/O pins 3, 5, 6, 9, 10 and 11 are equipped for creating 8-bit PWM

(Pulse Width Modulation ) Signals

ARDUINO UNO PIN MAPPING

|  |  |
| --- | --- |
| Micro controller | Arduino UNO |
| Voltage | 3.7 volts |
| Pins used | 3.3, 5,6,10,11,GND |
|  |  |

Motor Driver:

Motor Driver is a powerful engine ideal for driving DC Motors and Stepper Motors. It has some control over up to 4 DC engines, or 2 DC engines with directional and speed control. This engine driver is ideal for mechanical technology and mechatronics ventures and ideal for controlling engines from microcontrollers, switches, transfers, and so forth and be constrained by Pulse Width Modulation (PWM).

Bluetooth sequential modules permit all sequential

empowered gadgets to speak with each other utilizing Bluetooth. It has six pins and two modes -

Data Mode and Command Mode.

**Software**:

Bluetooth Voice Control for Arduino:

Android Meets Robots: Voice Recognition

Utilizes android mobiles inside voice acknowledgment to pass voice orders to our robot.

It matches with Bluetooth Serial Modules and sends in the perceived voice as a string. It can be utilized with any miniature regulator which can deal with strings.

Design:

Transmitter Side:

The mobile application will get to the voice orders through the Bluetooth module and changes over the voice orders to computerized signals and moves the signs to the Robot.

Receiver (or) Robot Side:

The computerized signals which are communicated from the transmitter are gotten by the Bluetooth module present in the Robot will get the signal and direct it to the microchip which is

Available in the Arduino .It will check with the orders and during execution the DC motors will move in the direction as indicated by the provided order and the motor rotations are constrained

by motor driver which is associated with the Arduino.

System Specification:

Hardware Specification

Arduino UNO

• L298 Motor Driver IC

• Bluetooth Device Module

• 4 x 5V Geared Motors

• Connecting Wires

• Robot Chassis

• Battery Holder

• Power Supply

• Battery

* Arduino UNO
* Motor Driver shield
* Bluetooth module
* 4 DC motors
* Connecting wires
* Battery
* Battery holder

Software Specification

* Arduino IDE software
* Android phone
* Bluetooth Controller App

Working

Voice orders are handled by smartphone, and speech to-text recognition is done inside the application. Message is then shipped off the beneficiary side through Bluetooth. Text got by means of Bluetooth is sent to Arduino utilizing serial communication protocol. Arduino code checks the text received. At the point when the text is a matching string, Arduino controls the developments of the robot likewise in forward, in reverse, Turning Right, Turning Left and Stop.

ARDUINO FOR CAR

BLUETOOTH RC CONTROLER

Controller

BLUETOOTH

START

CHECKING OF SIGNALS

MOTOR SHEILD

END

CAR MOVEMENT

RIGHT

LEFT

BACKWARD

FORWARD

Steps to control the automated car:

* Download the application named “Bluetooth RC Controller“from Google play store.
* Switch on the Bluetooth on the smart phone and also on the Bluetooth module
* Pair your phone with the Bluetooth module.
* Now the application and the robot is ready to perform the activity.
* Once the application is connected, give orders to the robot.

As indicated in the arduino programming, the robot will move in the directed direction. For instance when we express forward in the application, the order is given to the Bluetooth module of the arduino which is associated with the arduino.

* According to the given command, robot will perform operations of moving forward, moving backward, turning left and turning right.
* In addition to these it can also respond to voice commands like playing a song, telling a joke, weather forecast and so on.

CODE:

#include<AFMotor.h>

AF\_DCMotor motor1(1,MOTOR12\_1KHZ)

AF\_DCMotor motor2(2,MOTOR12\_1KHZ)

AF\_DCMotor motor3(3,MOTOR12\_1KHZ)

AF\_DCMotor motor4(4,MOTOR12\_1KHZ)

char command;

void setup()

{

Serial.begin(9600);

}

void loop()

{

If (Serial.available()>0)

{

command = Serial.read();

stop();

switch(command)

{

Case ‘F’:

Forward();

Break;

Case ‘B’:

back();

Break;

Case ‘L’:

left();

Break;

Case ‘R’:

right();

break;

}

}

}

Void forward()

{

motor1.setSpeed (255);

motor1.run (FORWARD)

motor2.setSpeed (255);

motor2.run (FORWARD)

motor3.setSpeed (255);

motor3.run (FORWARD)

motor4.setSpeed (255);

motor4.run (FORWARD)

}

Void back()

{

motor1.setSpeed (255);

motor1.run (BACKWARD)

motor2.setSpeed (255);

motor2.run (BACKWARD)

motor3.setSpeed (255);

motor3.run (BACKWARD)

motor4.setSpeed (255);

motor4.run (BACKWARD)

}

Void right()

{

motor1.setSpeed (255);

motor1.run (FORWARD)

motor2.setSpeed (255);

motor2.run (FORWARD)

motor3.setSpeed (255);

motor3.run (BACKWARD)

motor4.setSpeed (255);

motor4.run (BACKWARD)

}

Void left()

{

motor1.setSpeed (255);

motor1.run (BACKWARD)

motor2.setSpeed (255);

motor2.run (BACKWARD)

motor3.setSpeed (255);

motor3.run (FORWARD)

motor4.setSpeed (255);

motor4.run (FORWARD)

}

Void stop()

{

motor1.setSpeed (255);

motor1.run (RELEASE)

motor2.setSpeed (255);

motor2.run (RELEASE)

motor3.setSpeed (255);

motor3.run (RELEASE)

motor4.setSpeed (255);

motor4.run (RELEASE)

}

CAMERA CODE

import android.os.Bundle

import android.view.View

import android.widget.EditText

import android.widget.TextView

import androidx.appcompat.app.AppCompatActivity

import com.cscorner.kvfpv.R

class MainActivity : AppCompatActivity() {

private val data = mutableMapOf<String, String>()

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

}

fun saveData(view: View) {

val keyInput = findViewById<EditText>(R.id.keyInput)

val valueInput = findViewById<EditText>(R.id.valueInput)

val outputLabel = findViewById<TextView>(R.id.outputLabel)

val key = keyInput.text.toString()

val value = valueInput.text.toString()

if (key.isNotEmpty () && value.isNotEmpty()) {

data[key] = value

outputLabel.text = "Saved: $key => $value"

} else {

outputLabel.text = "Please enter both key and value."

}

}

fun retrieveData(view: View) {

val keyInput = findViewById<EditText>(R.id.keyInput)

val outputLabel = findViewById<TextView>(R.id.outputLabel)

val key = keyInput.text.toString()

if (data.containsKey(key)) {

val value = data[key]

outputLabel.text = "Retrieved: $key => $value"

} else {

outputLabel.text = "Key not found."

       }

    }

}

CODE FOR FRONT PERSON VIEW (CAMERA)

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<EditText

android:id="@+id/keyInput"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:hint="Key"

android:layout\_marginTop="50dp"/>

<EditText

android:id="@+id/valueInput"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:hint="Value"

android:layout\_below="@id/keyInput"

android:layout\_marginTop="10dp"/>

<Button

android:id="@+id/saveButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Save"

android:layout\_below="@id/valueInput"

android:layout\_marginTop="10dp"

android:onClick="saveData"/>

<Button

android:id="@+id/retrieveButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Retrieve"

android:layout\_below="@id/saveButton"

android:layout\_marginTop="10dp"

android:onClick="retrieveData"/>

<TextView

android:id="@+id/outputLabel"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text=""

android:layout\_below="@id/retrieveButton"

android:layout\_marginTop="20dp"/>

</RelativeLayout>

RC CONTROLLER CODE

import android.Manifest

import android.bluetooth.BluetoothAdapter

import android.bluetooth.BluetoothDevice

import android.bluetooth.BluetoothSocket

import android.content.pm.PackageManager

import android.os.Bundle

import android.widget.Toast

import androidx.appcompat.app.AppCompatActivity

import androidx.core.app.ActivityCompat

import com.cscorner.bluetoothrccontroller.R

import kotlinx.android.synthetic.main.activity\_main.\*

import java.io.IOException

import java.io.OutputStream

import java.util.\*

class MainActivity : AppCompatActivity()

{

private lateinit var bluetoothAdapter: BluetoothAdapter

private lateinit var bluetoothSocket: BluetoothSocket

private lateinit var outputStream: OutputStream

private val DEVICE\_ADDRESS = "00:00:00:00:00:00" // Replace with your Arduino's Bluetooth address

private val PORT\_UUID = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb") // Standard UUID

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()

if (bluetoothAdapter == null) {

Toast.makeText(this, "Bluetooth not supported", Toast.LENGTH\_SHORT).show()

finish()

}

val btnConnect = null

btnConnect.setOnClickListener { connect() }

val btnForward = null

btnForward.setOnClickListener { forward() }

val btnBackward = null

btnBackward.setOnClickListener { backward() }

}

private fun connect() {

if (!bluetoothAdapter.isEnabled) {

Toast.makeText(this, "Bluetooth is disabled", Toast.LENGTH\_SHORT).show()

return

}

val device: BluetoothDevice = bluetoothAdapter.getRemoteDevice(DEVICE\_ADDRESS)

try {

if (ActivityCompat.checkSelfPermission(

this,

Manifest.permission.BLUETOOTH\_CONNECT

) != PackageManager.PERMISSION\_GRANTED

) {

// TODO: Consider calling

// ActivityCompat#requestPermissions

// here to request the missing permissions, and then overriding

// public void onRequestPermissionsResult(int requestCode, String[] permissions,

// int[] grantResults)

// to handle the case where the user grants the permission. See the documentation

// for ActivityCompat#requestPermissions for more details.

return

}

bluetoothSocket = device.createRfcommSocketToServiceRecord(PORT\_UUID)

bluetoothSocket.connect()

outputStream = bluetoothSocket.outputStream

Toast.makeText(this, "Connected to Bluetooth device", Toast.LENGTH\_SHORT).show()

} catch (e: IOException) {

Toast.makeText(this, "Connection failed", Toast.LENGTH\_SHORT).show()

e.printStackTrace()

}

}

private fun forward() {

sendData("F")

}

private fun backward() {

sendData("B")

}

private fun sendData(data: String) {

if (::outputStream.isInitialized) {

try {

outputStream.write(data.toByteArray())

} catch (e: IOException) {

Toast.makeText(this, "Failed to send data", Toast.LENGTH\_SHORT).show()

e.printStackTrace()

}

} else {

Toast.makeText(this, "Bluetooth not connected", Toast.LENGTH\_SHORT).show()

}

}

override fun onDestroy() {

super.onDestroy()

if (::bluetoothSocket.isInitialized) {

try {

bluetoothSocket.close()

} catch (e: IOException) {

e.printStackTrace()

}

       }

    }

}

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<Button

android:id="@+id/btnConnect"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="50dp"

android:text="Connect" />

<Button

android:id="@+id/btnForward"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/btnConnect"

android:layout\_marginTop="20dp"

android:layout\_centerHorizontal="true"

android:text="Forward" />

<Button

android:id="@+id/btnBackward"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/btnForward"

android:layout\_marginTop="20dp"

android:layout\_centerHorizontal="true"

android:text="Backward" />

<Button

android:id="@+id/btnRight"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/btnBackward"

android:layout\_marginTop="20dp"

android:layout\_marginRight="20dp"

android:layout\_alignParentEnd="true"

android:text="Right" />

<Button

android:id="@+id/btnLeft"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@id/btnBackward"

android:layout\_marginTop="20dp"

android:layout\_marginLeft="20dp"

android:layout\_alignParentStart="true"

android:text="Left" />

</RelativeLayout>

RESULTS AND DISCUSSIONS

Through our project, we accomplished the following tasks:

1. Robot is controlled through voice orders given by the client
2. Voice recognition is done through android application then a separate order is given to the voice controlled car
3. Microcontroller that is fitted, translates these orders and provides a suitable order that is associated with the motors of the car
4. Voice recognition is done inside the android application and afterward a separate order is

APPLICATIONS:

1. Education: Used as a learning tool to understand concepts of robotics, electronics, and programming.

2. Entertainment: Can be used for leisure activities, such as racing competitions or simply as a fun hobby.

3. Research and Development: Used by researchers and engineers to explore new technologies and improve existing ones.

4. Prototyping: Provides a platform for testing and prototyping new ideas in the field of robotics and automation.

5. Remote Control: Can be employed for tasks in hazardous environments or where human presence is impractical or risky, like exploring rough terrains or disaster zones.

67. Military and Security: Used for surveillance, reconnaissance (military observation of a region to locate an enemy or ascertain strategic features.), and other security-related purposes.

These applications demonstrate the versatility and usefulness of Bluetooth-controlled cars in various fields.

CONCLUSION

This paper proposed a Bluetooth activated robotic car using efficient components. The voice-controlled robotic arm vehicle using android combines a robotic controlled by voice commands received via a synchronized Bluetooth android phone providing restricted access.

FUTURE SCOPE

* In this module we have enabled communication through Bluetooth because of this there is a limitation of range.
* In future we can use Wi-Fi module to overcome this limitation.
* Another improvement can be using Internet of Things commonly known as IoT. This will enable the client to control the device from any distance.
* We know that Artificial intelligence or AI is booming right now in every field. Using AI in this model will permit the vehicle to move or advance without human interference.
* We can also use Global Positioning System (GPS) or Global Monitoring System(GMS) so that we can track our automated devices from home
* Industrial Automation: In factories or warehouses, Bluetooth-controlled vehicles can transport goods or perform specific tasks.