Android Controlled Landmine Detector Robotic Vehicle

|  |  |  |
| --- | --- | --- |
| Akash Kumar (200907476) | Mrityunjay Badoni (200907288) | Solomon Sandeep K (200907284) |
| Department of Electronics & Communication Engineering | Department of Electronics & Communication Engineering | Department of Electronics & Communication Engineering |
| Manipal Institute of Technology | Manipal Institute of Technology | Manipal Institute of Technology |
| Manipal Karnataka | Manipal Karnataka | Manipal Karnataka |
| akash.kumar16@learner.manipal.edu | mrityunjay.badoni@learner.manipal.edu | kondangi.sandeep@learner.manipal.edu |

*I. OBJECTIVE*

*A. PROBLEM STATEMENT*

Since 1947, our nation has been at war. Thanks to the warriors from India who fought for us, we were able to win the war, but we also lost a great number of priceless lives. Although saving human lives and ensuring its security are two distinct tasks, both are equally important and required. This ground-breaking system is designed for operations where it is dangerous for people to enter, particularly in combat zones where there is a high likelihood that landmines will be buried beneath the ground.

*B. AIM / SOLUTION*

In order to identify landmines that have been buried in fields and are lethal when walked on, this project is focused on the development of a robotic vehicle with a metal detector sensor that can be controlled remotely by an android app, saving the lives of our soldiers and civilians.

The metal detector in the car should pick up on the metal object, which might be a landmine. A buzzer that alerts everyone to the presence of metal that might be a landmine should be activated upon the identification of a metal.

The goal of this project is to create a functional prototype of a robotic vehicle driven by an android that can locate landmines buried underground using a metal detector and sound an alarm to warn the user of their existence. The user ought to be able to manage the car using a smartphone thanks to the Raspberry Pi Pico microprocessor and the HC-05 Bluetooth module. This may be done by using a specially created app that talks to the RPi Pico.

The military troops may then utilise the vehicle to explore the battlefields, find land mines and other explosive devices that include heavy metal, and activate the buzzer when one is found, using the metal detector circuit that is linked to it.The use of a remote accessed vehicle can be an efficient way to detect under land landmine as there will not be any use of manned operation which can cause accidental triggering of landmine and serious harm.

The end result is an efficient and trustworthy system that can lessen the risks posed by landmines, particularly in border regions.

***II. INTRODUCTION***

Detection is done cautiously and farther away from the mine using a metal detector based on the Colpitts oscillator, as the name indicates. This robotic mine detection vehicle may be stationed on highways in danger zones in place of humans to locate mines and protect the lives of our soldiers. Using a "Landmine detection robotic vehicle" to discover land mines is a safe method. This automobile is controlled with a Bluetooth module, a Raspberry Pi Pico, and wireless connectivity. An Android application controls the entire system.

A remote control on an Android smartphone will be used to move the robot. The Bluetooth module will act as a bridge between the Raspberry Pi Pico and the smartphone. We'll use a Bluetooth module that can act as either a master or a slave in the system. The robot's movement will be determined by the controller. A metal detector may also detect bombs and mines. The android application employs commands like forward, backward, right, and left to steer the robotic automobile.

*III. Methodology*

*A. ANDROID CONTROL*

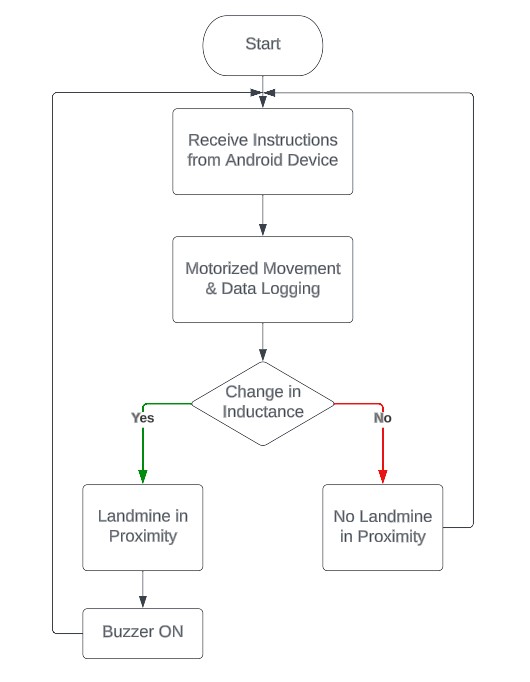
The user controls the automobile using an Android application that we developed particularly for this project. The Raspberry Pi Pico, the vehicle's brain, gets these commands remotely through Bluetooth. The Raspberry Pi Pico informs the motor driver to stop the automobile after getting a signal from the metal detector sensor. This prevents the automobile from accidentally detonating the landmine..

*B. LANDMINE DETECTION*

A built-in metal detector sensor might detect landmines hiding in the ground. The metal detector sensor informs the Raspberry Pi Pico when it detects a landmine. A metal detector in the vehicle warns the driver of the presence of a landmine. When a metal object contacts with the metal detector's coil, the buzzer used in this project's alert mechanism activates..

*C. METAL DETECTOR*

The Colpitts oscillator circuit is utilised to build a metal detector in this project. A Colpitts oscillator is an electrical oscillator circuit that produces sinusoidal output. It consists of an LC circuit that determines the frequency of the oscillator and a transistor amplifier that provides the oscillation-inducing feedback. The sensing component of the metal detector, an inductive coil, is linked to the Colpitts oscillator. When a metallic object is placed adjacent to the inductive coil, its inductance varies, causing the frequency of the oscillator to shift. When metal is present near the inductive coil, the frequency of the oscillator changes.This frequency change is detected using a buzzer.

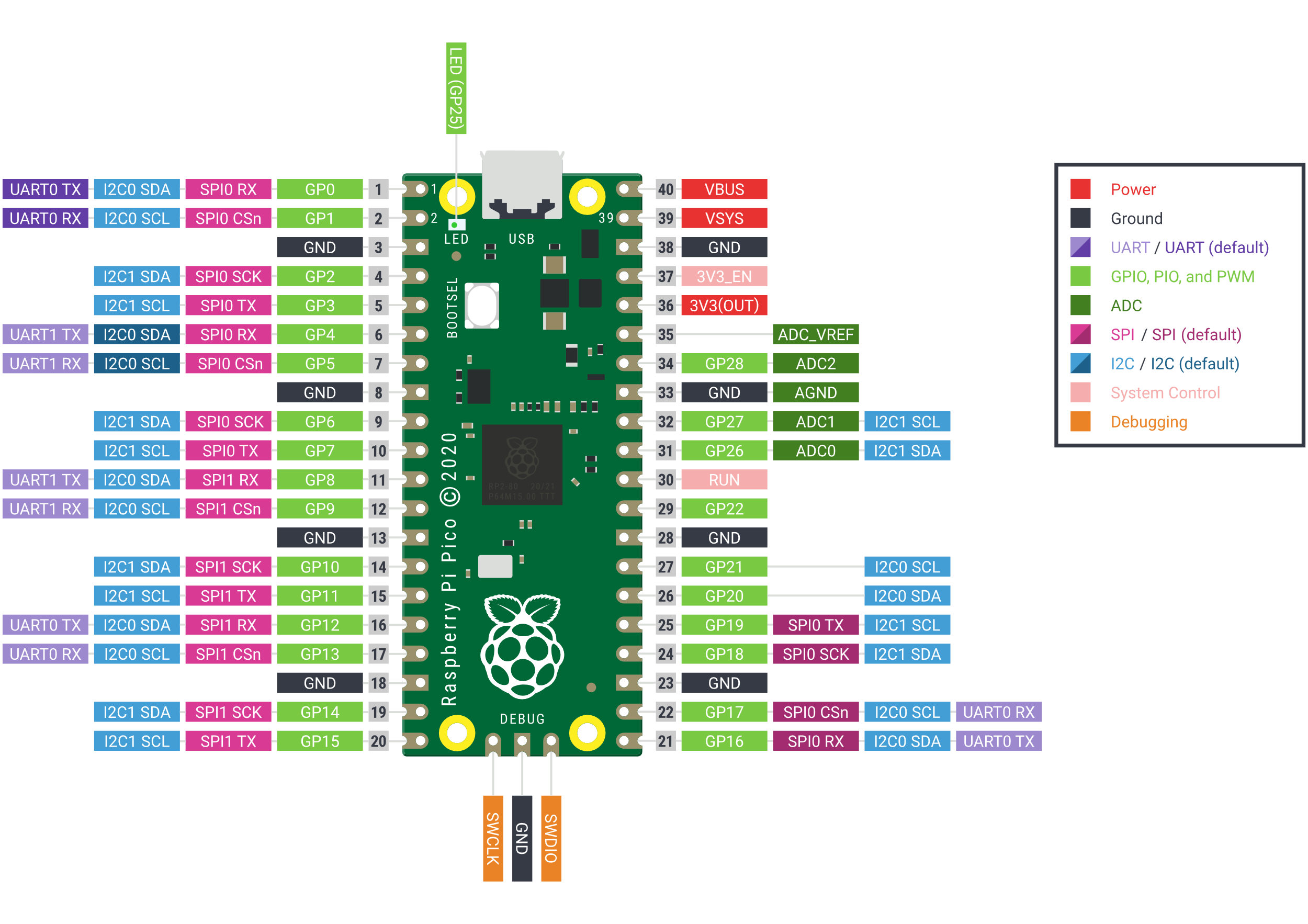


Diagram

Description automatically generated

*IV. COMPONENTS*

*A. RASPBERRY PI PICO*



There are 40 pins in all, including GND and VCC pins, according to the Raspberry Pi Pico pinout. Power, ground, UART, GPIO, PWM, ADC, SPI, I2C, system control, and debugging pins are some of the several types of pins.

26 of the 40 pins on the Raspberry Pi Pico are multipurpose GPIO pins. These 26 GPIO pins provide digital input and output on both the input and output sides. These digital pins have the designations GP0, GP1, and GP28.

Four 12-bit ADC analogue pins are included on the PCB. Consequently, we can read analogue inputs from a variety of sensors using these pins. However, the board does not include ADC4, one of these four pins, as a GPIO pin. A temperature sensor is internally attached to the fourth ADC pin.

*1. PWM Pins*

Each of the eight PWM blocks/slices on the Raspberry Pi Pico may provide up to two PWM outputs (a and b). Each block can therefore drive up to two PWM outputs. As a result, there are 16 PWM output channels accessible at once, and the Pico's GPIO pins may all provide PWM output.

Therefore, all GPIO pins can be configured to output PWM signals when necessary, but two GPIO pins with the same PWM designation cannot be used simultaneously.

*2. Programmable Input/Output Feature*

The PIO can be used to operate sensors or motors that are located outside the computer. Additionally, it can be used to connect to other digital systems. The PIO is a strong and adaptable tool that gives consumers extensive control over the actions of their Pico. Each RP2040 contains two PIO chips, which have similar capabilities to tiny CPUs. In order to accomplish this, they can use instructions that are kept in shared PIO memory.

The CAB BUS, the WS2812b driver, and other more complex peripherals that the chip doesn't directly support may be simulated using PIO instances. These emulated peripherals are able to interface with outside devices simultaneously with the main programme since the PIO state machines operate independently of the main CPU.

*3.Universal Asynchronous Receiver-Transmitter (UART) Pins*

There are two identical UART peripherals in the Raspberry Pi Pico. Asynchronous serial connection between the Pico and UART devices or other devices is accomplished via UART pins.

|  |  |
| --- | --- |
| UART Pins | GPIO Pins |
| UART0-TX | GP0/GP12/GP16 |
| UART0-RX | GP1/GP13/GP17 |
| UART1-TX | GP4/GP8 |
| UART1-RX | GP5/GP9 |

*3. GPIO Interrupts*

If one of the following GPIO pin state changes takes place, all GPIO pins on the board can be set up as an external interrupt:

* • High level (+3V)
* • Low level (GND)
* • Positive edge (change from a low to a high that is active)
* • Negative edge (change from a high to a low that is currently active)

*B. MOTOR DRIVER L298N*

A high-power motor driver module for driving DC and stepper motors is the L298N. An l298 motor driver IC and a 78M05 5M regulator make up this module. Up to four DC motors may be controlled by the L298N module, or two DC motors can be controlled in both directions.

|  |  |
| --- | --- |
| Pin Name | Description |
| Enable A and Enable B | These pins are used to enable and control the speed of the motor.  If the jumper is present, the speed will be maximum.  If the jumper is removed, we have to connect the PWM, which will vary the speed of the motor.  If it is grounded (GND) the motor is disabled. |
| IP1 and IP2 | These pins act as the input to motor A. |
| IP3 and IP4 | These pins act as the input to motor B. |
| 5V Pin | The 5V pin will act as an input or an output depending on the following conditions:  If the supply voltage is in the range of 5 -12V, the jumper should be connected, and the 5V pin will act as an output.  If the supply voltage is in the range of greater than 12V, the jumper should be disconnected, and the 5V pin will act as an input. |
| 12V Pin | The input voltage to the L298N motor is in the range of 5-35v and 2 amperes. |
| GND | Ground pin. |
| Out 1 & Out 2 | Output pins of motor A. |
| Out 3 & Out 4 | Output pins of motor B. |

*C. HC05 BLUETOOTH MODULE*

A Bluetooth module called the HC05 makes it possible for devices to communicate wirelessly. Being a Bluetooth serial port module, it communicates with other devices using a serial interface. The module may be set up as either a master or a slave device and supports Bluetooth 2.0. It can connect to other Bluetooth devices, including computers, tablets, and smartphones, as a master device. It can be linked to a master device, such a microcontroller or a computer, acting as a slave device. It can support a data rate of up to 2.1 Mbps and has a range of up to 10 metres.

|  |  |
| --- | --- |
| Pin Name | Description |
| VCC | This pin is used to provide power to the module in range of 3.3V to 5V DC. |
| GND | This pin is the ground pin of the module. |
| TXD | This pin is used to transmit data from the module to other devices. |
| RXD | This pin is used to receive data from another device |
| EN/Key | This pin is used to put the module in at mode. When this pin is set to HIGH, the module is in command mode When this pin is set to LOW, the module is in data mode. |
| State | The state pin is connected to on board LED, it can be used as feedback to check if Bluetooth is working properly. |

*D. MICROSD CARD MODULE*

*E. 5V MOTORS*

*F. WI-FI MODULE*

V. SOFTWARE USED

*A. THONNY IDE*

Python programmes for the Raspberry Pi Pico may be written using the well-liked integrated development environment (IDE) Thonny. Python programming on the Raspberry Pi Pico may be written, debugged, and tested using the Thonny IDE, which has an intuitive user interface and a straightforward, clean structure.

Micro Python is used by Thonny ide instead of regular Python. Micro Python is a compact and effective version of Python 3, which is designed to run on microcontrollers and in limited contexts. It provides a tiny part of the Python standard library.

*B. MIT APP INVENTOR*

We used MIT App Inventor to create the app for our project in accordance with its requirements. It offers a web-based editor for creating applications for smartphones that target the iOS and Android operating systems. It makes use of a programming language using blocks.

*VI. BLUETOOTH TECHNOLOGY*

*A. PROTOCOL STACK OF BLUETOOTH*

The actual protocols described in a standard are implemented in software or hardware by a protocol stack, allowing devices based on that standard to interact with one another.

*B. Radio Layer*

Like other wireless technologies, Bluetooth also functions. It uses radio frequency to transport data in the form of bits (ones and zeros). The radio layer stipulates how this function works. The Bluetooth radio frequency ranges from 2.402 GHz to 2.48 GHz in the ISM (Industrial, Scientific, and Medical) band.

*C. Baseband Layer*

Frequency hopping for interference reduction, medium access control, and data packetization are all handled by this layer. TDD (time division duplex) alternates the packets between transmission and reception. It also regulates channels, linkages, error correction, and flow management.

*D. Link Protocols*

The logical link control and adaptation protocol (l2cap) and the link management protocol (lmp) are two protocols that allow for the formation of links and the accompanying control of those links. These protocols are necessary to control, among other things, encryption, synchronisation, and device pairing.

*E. Service Discovery Protocol*

The SDP identifies the services that are accessible in the RF vicinity and establishes their characteristics. Devices need to support the same services in order to connect to one another. SDP merely offers ways to find services; it does not offer ways to access them.

*F. RFComm Protocol*

Applications like Bluetooth connections to printers and scanners are made possible by this protocol. For RFComm to guarantee trustworthy in-sequence transmission of byte streams, it depends on the Bluetooth baseband.

*G. Bluetooth Networking*

Establishing a master device and one or more slave devices enables Bluetooth connection. Any gadget could be either a master or a slave. This characteristic makes Bluetooth suitable for setting up ad-hoc networks. Any Bluetooth device can communicate with other devices within range by designating one of them as the master and the rest as slaves, which is one of the technology's most crucial features. Based on its address, the master device chooses the frequency hopping pattern.

*VII. CODE*

# importing PIN, RTC, UART, SPI from machine module

from machine import Pin, UART, RTC, SPI, PWM

# importing time, sdcard, uos module

import utime

import sdcard

import uos

# defining UART channel and Baud Rate

baudrate = 9600

uart = UART(0,baudrate)

# defining Pin for onboard\_led

led\_onboard = Pin(25, Pin.OUT)

led\_onboard.value(1)

# external pin connections to invernal variables for SPI

csPin = 9

sckPin = 10

mosiPin = 11

misoPin = 8

spiChannel = 1

# configuring CS as output to send data to SD Card

CS = Pin(csPin, Pin.OUT)

# initializing SPI interface

spi = SPI(spiChannel, baudrate=1000000, polarity=0, phase=0, bits=8, firstbit=SPI.MSB, sck=Pin(sckPin), mosi=Pin(mosiPin), miso=Pin(misoPin))

# initialise microSD card and start SPI communication

sd = sdcard.SDCard(spi,CS)

# mounting filesystem

vfs = uos.VfsFat(sd)

uos.mount(vfs, "/sd")

# real time clock

rtc = machine.RTC()

# initialising motor variables

EN\_A = PWM(Pin(7))

In1 = Pin(6,Pin.OUT)

In2 = Pin(5,Pin.OUT)

In3 = Pin(4,Pin.OUT)

In4 = Pin(3,Pin.OUT)

EN\_B = PWM(Pin(2))

# Setting the frequency of PWM signal

EN\_A.freq(1000)

EN\_B.freq(1000)

# Forward

def move\_forward():

In1.high()

In2.low()

In3.high()

In4.low()

# Backward

def move\_backward():

In1.low()

In2.high()

In3.low()

In4.high()

# Turn Right

def turn\_right():

In1.low()

In2.high()

In3.high()

In4.low()

# Turn Left

def turn\_left():

In1.high()

In2.low()

In3.low()

In4.high()

# Stop

def stop():

In1.low()

In2.low()

In3.low()

In4.low()

EN\_A.duty\_u16(65535)

EN\_B.duty\_u16(65535)

print("Starting of 'While' loop\n")

while True:

# checking if data is available

if uart.any():

# getting data and converting it into string

data = uart.read()

data = str(data)

print(data)

# getting time in tuple and converting it into string

timestamp = rtc.datetime()

timestring = "%04d-%02d-%02d %02d:%02d:%02d"%(timestamp[0:3] + timestamp[4:7])

insert = timestring + "," + data + "\n"

# create a file and write something to it

with open("/sd/data.txt", "a") as file:

file.write(insert)

print("Inserting")

if 'forward' in data:

move\_forward()

elif 'backward' in data:

move\_backward()

elif 'left' in data:

turn\_left()

elif 'right' in data:

turn\_right()

elif 'stop' in data:

stop()

elif 'speed' in data:

speed = data.split('|')

speed = int(speed[-2])

duty = int(speed \* 65535 / 100)

EN\_A.duty\_u16(duty)

EN\_B.duty\_u16(duty)

*VIII. RESULTS*

We were able to create a working prototype of an android-controlled robotic vehicle that can find underground landmines using a metal detector and warn the user of their presence by sounding an alarm that can be activated by an android device via Bluetooth using a specially developed android application. The HC-05 Bluetooth module and the Raspberry Pi Pico microprocessor were used to do this. The development of an application to use a smart device to interface with RPi for vehicle control was successful.

The precise movements of the robotic vehicle, including forward, backward, right, and left, are controllable. Overall, this project's objective has been successfully attained.

*IX. RESULT ANALYSIS*

The project's main goals were to create wireless connection between the truck and an Android handset and landmine detection. The constructed prototype is performing as expected, and we are now able to control the robotic car using an Android device and an application created specifically for that purpose. The vehicle can also identify heavy metals, which is a technique for locating landmines.

As the vehicle also detects metals with false positives, there is room for improvement. In order to boost efficiency, additional, more effective techniques might be utilised for landmine detection. The vehicle may be equipped with a wifi module and a camera module to increase its functionality and efficiency.

In general, the project's goal of offering a solution to the issue detailed in the problem description was successful.