DOCUMENTATION

Gesture-controlled Robot Car Using MPU6050 and Arduino Microcontrollers

By
Khyle Matthew P. Gopole

September - October 2024

TABLE OF CONTENTS

TABLE OF CONTENTS ii

LIST OF TABLES iii

LIST OF FIGURES iv

PROJECT DESCRIPTION 1

PROJECT BACKGROUND 1

SCOPE AND DELIMITATIONS 4

CONCEPTUAL FRAMEWORK 5

HARDWARE DEVELOPMENT 6

SOFTWARE DEVELOPMENT 11

SYSTEM LOGIC 15

CASING DESIGN 18

PROTOTYPE DEVELOPMENT 20

PROTOTYPE SETUP 24

TESTING 25

CONCLUSION 25

RECOMMENDATION 26

REFERENCES 27

LIST OF TABLES

Table 1: Controller Circuit Logic 15

Table 2: Speed Indicators Logic 16

Table 3: Motors and Robot Circuit Logic 16

Table 4: Test Results 25

LIST OF FIGURES

Figure 1: Graphical Representation of Euler Angles [**1**] 2

Figure 2: Conceptual Framework 5

Figure 3: Hardware Block Diagram 6

Figure 4: Robot Circuit Schematic Diagram 7

Figure 5: Robot Circuit Diagram 8

Figure 6: Controller Circuit Schematic Diagram 9

Figure 7: Controller Circuit Diagram 10

Figure 8: Main Flowchart 11

Figure 9: Read Control Function 12

Figure 10: Set Control Function 13

Figure 11: Operate Robot Function 14

Figure 12: Dimensions of Controller Casing 18

Figure 13: Isometric View of Controller Casing 19

Figure 14: Lid of Controller Casing 19

Figure 15: Erobot Chassis by Erovoutika 20

Figure 16: Controller Setup 20

Figure 17: Controller PCB 21

Figure 18: Robot Car Setup (Top View) 22

Figure 19: Robot Car Setup (Bottom View) 23

Figure 20: Prototype Setup 24

PROJECT DESCRIPTION

Robotics and automation are popular and prominent fields nowadays. Most devices are implemented with automation for convenience and practicality. It covers a wide range of usage as it can be integrated in different applications. One simple application of robotics and automation is robot cars. These robot cars can be implemented in different ways as various sensors and actuators are easily accessible in the market. Common examples are line-following robot, maze-solver robot, robotic arm, etc. In terms of control, robot cars are usually controlled via remote using infrared. However, control operations are not only limited to remote as other sensors can be utilized to implement it in a unique way.

Hence, sensors like gyroscope and accelerometer can be integrated to a robot car to maneuver its control operations easily. By utilizing these sensors, a robot car can be controlled without using a physical remote but through hand gestures. This project explores the implementation of a simple robot car that makes use of an accelerometer and gyroscope to facilitate movement and direction of the car wirelessly. This is done by strapping a separate microcontroller system with a sensor at the top of the hand then communicating with another microcontroller system in controlling the robot based on the orientation of the hand relative to gravity.

Specifically, the accelerometer measures the change in velocity of a certain object while the gyroscope can measure angle of rotations in three axes (pitch, roll, yaw) with the influence of gravity. By complementing these two, the angles can be accurately extracted and calculated for more precise gesture controls. A buzzer to simulate horn feature can be integrated. A speed variation (Low, Medium, High) can also be added for flexibility. Lastly, speed indicators using LEDs can be added for a better user experience.

PROJECT BACKGROUND

The controller circuit operates using the accelerometer and gyroscope sensor module through tilt angles. Specifically, the controller gets the Euler angles which deals with rotational angles along the x, y, and z-axes which are also called roll (φ), pitch (θ), and yaw (ψ), respectively as represented by the figure 1 [1]. However, for convenience, the system would only consider and use tilt angles from the roll and pitch since the hand can be easily rotated along the x and y-axes. These angles are calculated by program based on some formulas.

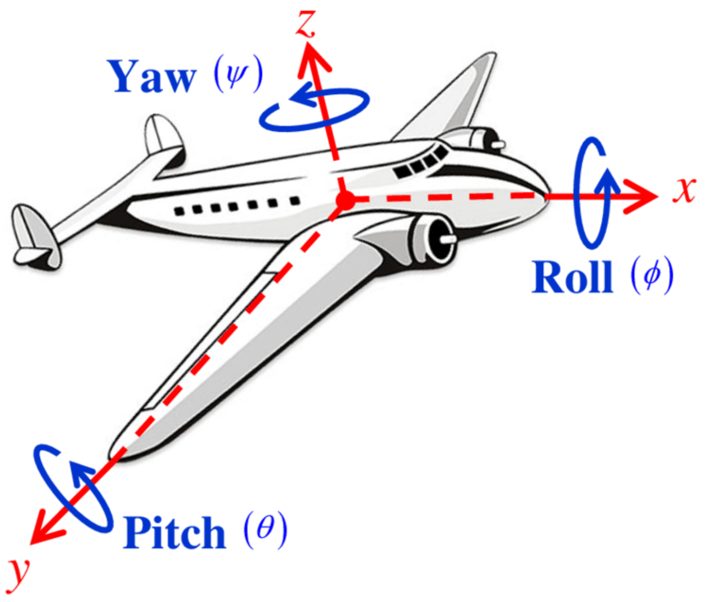

Figure 1: Graphical Representation of Euler Angles [**1**]

Since the MPU6050 module is used as the three-axis IMU (Inertial Measurement Unit), it supports gyroscope and accelerometer data extraction along the three axes. However, since the developer is only interested in measuring the roll angle (x-axis) and pitch angle (y-axis), the calculation will only include these Euler angles. When using this sensor, it has drawbacks since gyroscope and accelerometer are separate entities when operating the MPU6050. Hence, pitch and roll angles are calculated separately for gyroscope and accelerometer. The roll and pitch angles for accelerometer can be calculated by using the following equations below based on [2]. Multiplying these equations into 180/π is required since the inverse tangent equations result to radians unit which must be converted into degrees unit for the angles.

$$\varphi_A = \tan^{-1}\left(\frac{a_y}{\sqrt{(a_x)^2 + (a_z)^2}}\right) \times \frac{180}{\pi} \tag{1}$$

$$\theta_A = \tan^{-1}\left(\frac{a_x}{\sqrt{\left(a_y\right)^2 + (a_z)^2}}\right) \times \frac{180}{\pi} \tag{2}$$

Where φ_A represents the roll angle from accelerometer, θ_A represents the pitch angle from accelerometer, a_x represents the acceleration along x-axis, a_y represents the acceleration along y-axis, and a_z represents the acceleration along z-axis.

On the other hand, roll and pitch angles from the gyroscope can also be calculated. The intern is aware that gyroscope measures the angular velocity in a certain axis. Since the intern is only interested with the angle itself from that angular velocity, this angle can be calculated by multiplying the angular velocity to the elapsed time. This is outlined in equations 3 and 4 below.

$$\varphi_G = \omega_x \times t \tag{3}$$

$$\theta_G = \omega_y \times t \tag{4}$$

Where φ_G represents the roll angle from gyroscope, θ_G represents the pitch angle from gyroscope, ω_x represents the angular velocity along x-axis, ω_y represents the angular velocity along y-axis, and *t* represents the elapsed time.

When dealing with IMU sensors, the gyroscope and accelerometer has each of their own drawbacks. To compensate for these drawbacks, it is recommended to use complementary filter when calculating for the roll and pitch angles [2]. Basically, complementary filter is an equation used to combine similar values from separate sources. In other words, a certain percentage from the accelerometer values are added to a certain percentage from the gyroscope values. This increases the accuracy and reliability of the IMU sensor when doing so. Commonly, large portion of the angle would come from the gyroscope while the remaining percentage would come from the accelerometer. The intern tried researching about this then found out that 96% of the gyroscope angle is commonly considered while the remaining 4% comes from the accelerometer angle. Hence, the intern applied the same values as observed in equations 5 and 6.

$$\varphi = (0.96 \times \varphi_G) + (0.04 \times \varphi_A) \tag{5}$$

$$\theta = (0.96 \times \theta_G) + (0.04 \times \theta_A) \tag{6}$$

Where φ represents the final roll angle, θ represents the final pitch angle, φ_G represents the calculated roll angle from the gyroscope, φ_A represents the calculated roll angle from the accelerometer, θ_G represents the calculated pitch angle from the accelerometer and θ_A represents the calculated pitch angle from the accelerometer.

All these equations are applied in the extraction of the pitch and roll angles using the MPU6050 module. This is programmed using C++ programming language in the Arduino Nano microcontroller program in the controller.

SCOPE AND DELIMITATIONS

The gesture-controlled robot car project is implemented using Arduino Nano microcontrollers which are programmable using C++ programming language. The project's operation in terms of robot car operation only supports five states which include forward, backward, turning left, turning right, and stop. The horn button also mimics a real horn button in an actual car setup. Thus, the horn feature can only be triggered when holding the button. Releasing the button would immediately stop the horn of the robot car. Aside from that, the project supports three variations of speed which were configured during the testing within the range of 0-255 PWM level. This includes low, which is the default level, medium, and high for maximum speed output. Changing the speed can be triggered by pressing the speed button. It transitions from low to medium then high. From a high-speed level, pressing the speed button would just cycle the transition, returning it to low-speed level.

Furthermore, in terms of compatibility in terrain, the project can only run in a flat surface without large bumps due to the small size of the robot car prototype. The project does not support obstacle detection thus, the user must control it properly to avoid the robot car bumping to another surface that may damage the prototype. As for connectivity, the controller can only transmit signals to the robot car within a 100m distance. The transceiver module can only send one signal at a time hence, the user must only activate other features when the robot car is in a stable or idle state to prevent unintended movements from the robot car. In terms of controller, the accelerometer and gyroscope sensor highly rely on tilt angles hence, the user must start operating it while it is level or positioned flat while attached on the top of the user's hand. Lastly, the robot car's casing is not that secured due to the requirement of the company hence, components and wirings are exposed. This is because the company will use it for demonstration purposes.

CONCEPTUAL FRAMEWORK

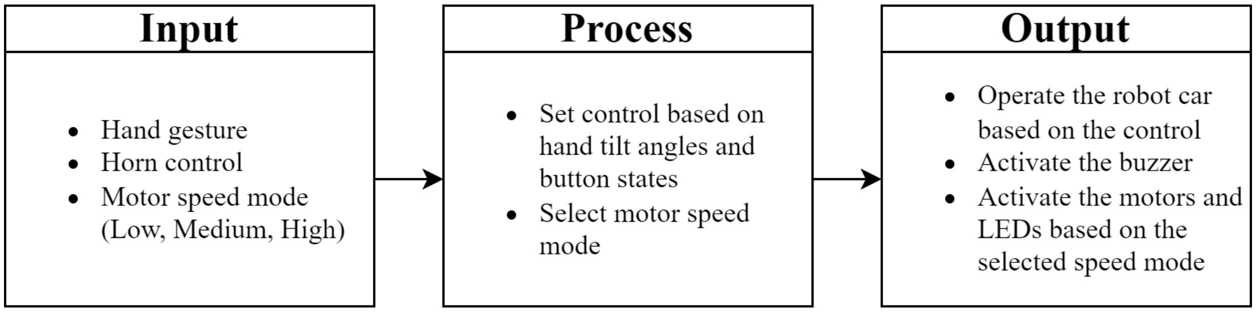
Input
Hand gesture
Horn control
Motor speed mode (Low, Medium, High)
Process
Set control based on hand tilt angles and button states
Select motor speed mode
Output
Operate the robot car based on the control
Activate the buzzer
Activate the motors and LEDs based on the selected speed mode

Figure 2: Conceptual Framework

Figure 2 outlines the input, process, and output framework of the project. The overall system takes the hand gesture through the gyroscope and accelerometer sensor and the horn control and motor speed modes via the buttons as inputs. The sensed hand tilt gesture would then be processed by the Arduino Nano microcontroller and set for controls based on the tilt or Euler angles of the hand on x and y-axes. The selection of motor speed mode would also be processed based on the number of times the speed button gets pressed. These controls would then be sent to the robot car circuit which then operates it based on the signal and activate the motors and LEDs depending on the speed mode (low, medium, high) as the overall output. This also includes the activation of buzzer if the horn button is pressed.

HARDWARE DEVELOPMENT

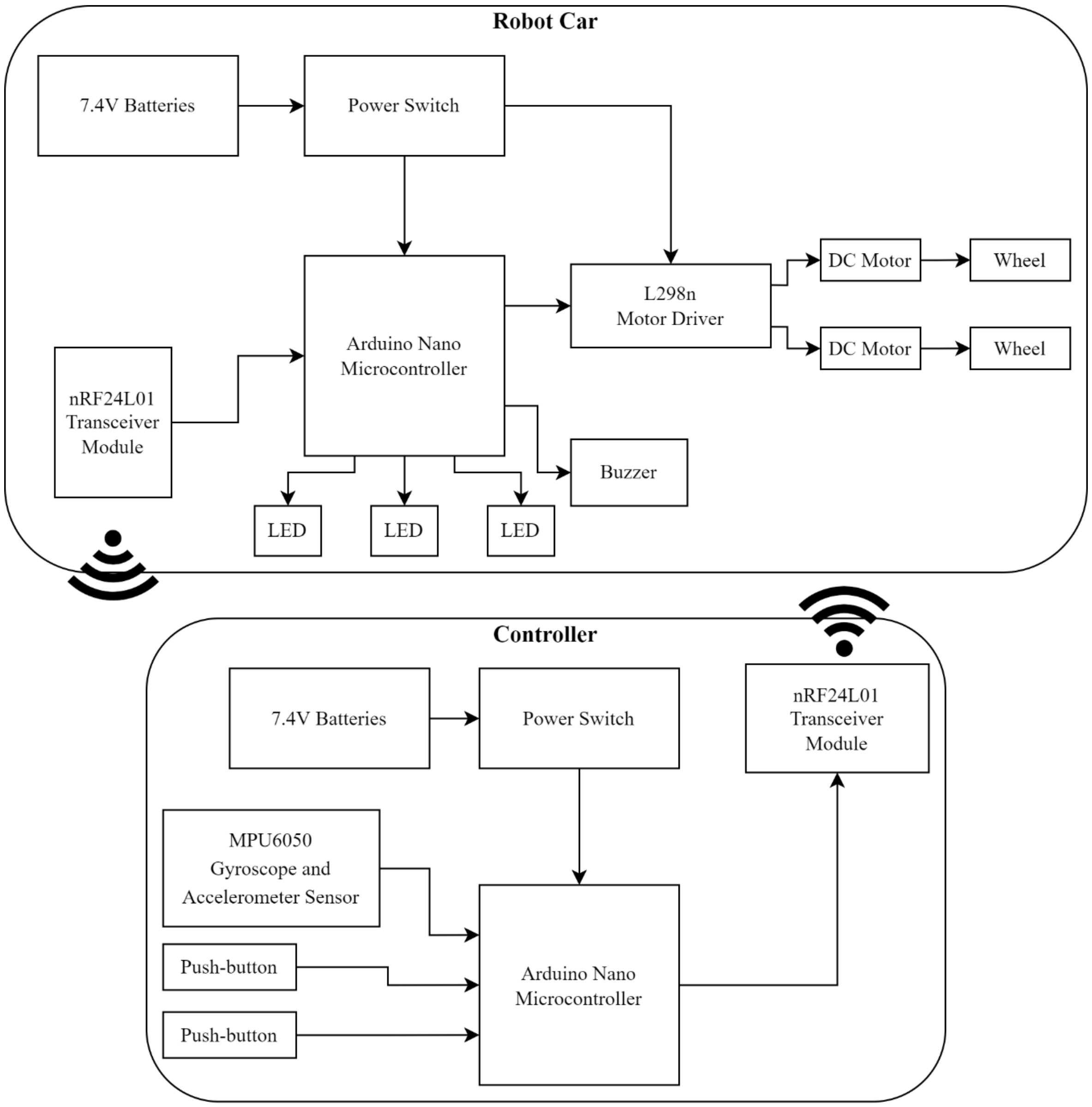
Robot Car
7.4V Batteries
Power Switch
Arduino Nano Microcontroller
L298n Motor Driver
DC Motor
Wheel
DC Motor
Wheel
nRF24L01 Transceiver Module
Buzzer
LED
LED
LED
Controller
7.4V Batteries
Power Switch
nRF24L01 Transceiver Module
MPU6050 Gyroscope and Accelerometer Sensor
Push-button
Push-button
Arduino Nano Microcontroller

Figure 3: Hardware Block Diagram

Figure 3 presents the hardware block diagram of the system which mainly consists of two circuits: Robot Car, and Controller. The robot car is powered by 7.4V batteries using two 18650 lithium batteries with a power switch. It consists of a L298n motor driver, two DC motors connected to wheels, three LEDs, a buzzer, an Arduino Nano microcontroller, and a nRF24L01 transceiver module. On the other hand, the controller has the same power supply with a power switch and mainly consists of a MPU6050 sensor for gyroscope and accelerometer, two push-buttons, an Arduino Nano microcontroller, and nRF24L01 transceiver module. Arduino Nano microcontroller is an inexpensive, efficient, and small microcontroller used in the company which is based on the ATmega328P chip thus, it is used as the central point of the gesture-controlled robot car development.

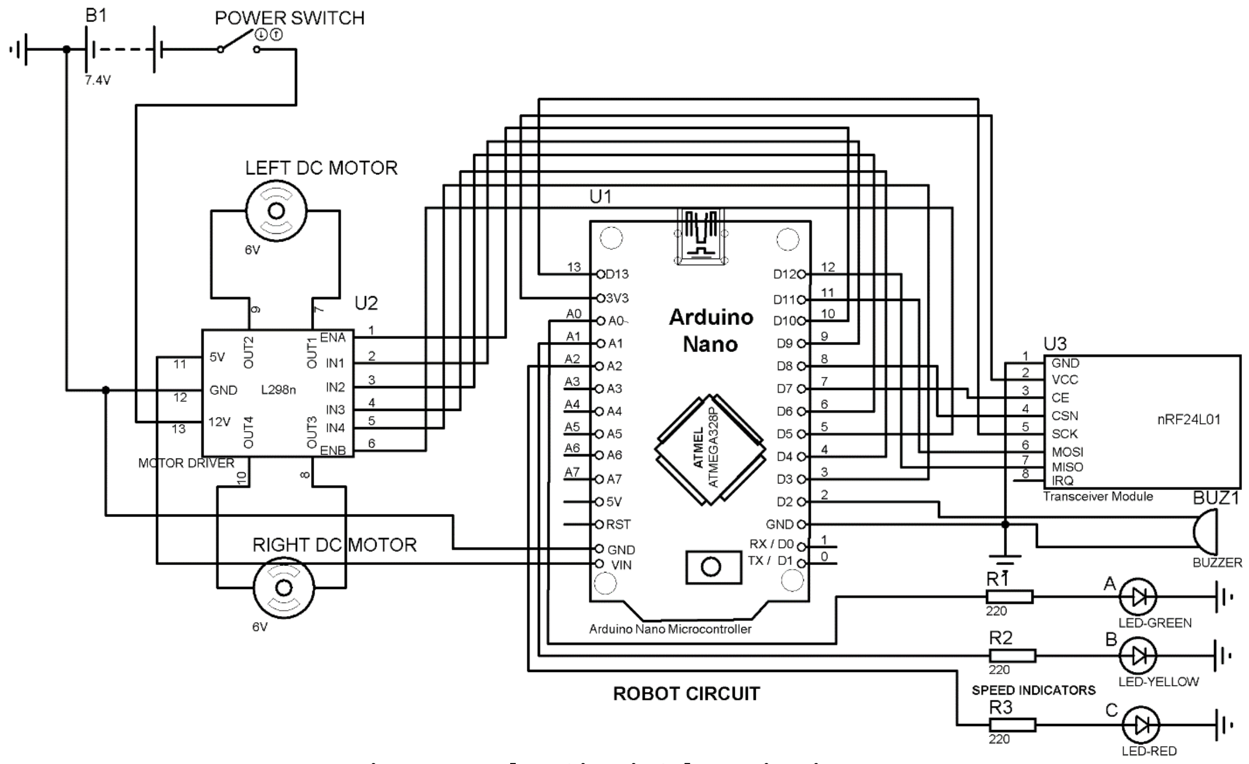

Figure 4: Robot Circuit Schematic Diagram

Figure 4 outlines the schematic diagram of the robot car circuit for wiring connections reference. As observed, the robot circuit is powered using a 7.4V battery by connecting two 18650 batteries in series. A power switch is used to only supply power to the circuit during operation. A L298n motor driver is necessary to handle the voltage requirement and enable control operations to the two DC motors which can be operated using 6V. These two DC motors are connected to the paired output terminals of the driver (OUT1, OUT2, OUT3, and OUT4). On the other hand, the 12V pin is connected to the positive side of the battery after the switch to supply power to it. The GND pin is then connected to the common ground of the whole circuit. On the other hand, the 5V pin of the driver is then connected to the VIN pin of the Arduino Nano microcontroller to serve as its power source. As for the input and enable pins of the driver, these are all connected to the digital pins of the Arduino Nano microcontroller for the control signals. The enable pins ENA and ENB are connected to digital pins D10 and D5 of the Arduino Nano microcontroller, respectively for motor speed control and activations of the left DC motor and right DC motor. Moreover, the input pins IN1, IN2, IN3, and IN4 are connected to digital pins D9, D6, D4, and D3 of the Arduino Nano microcontroller for motor direction and activation. Aside from that, a nRF24L01 transceiver module is also used for wireless connectivity between the robot car and controller circuits. This is powered by connecting its VCC pin to the 3V3 pin of the Arduin Nano microcontroller for its 3.3V power requirement. Its GND pin is connected to the common ground while its other pins CE, CSN, SCK, MOSI, and MISO are connected to digital pins D7, D8, D13, D11, and D12 of the Arduino Nano microcontroller. These connections are required to operate and facilitate wireless connections when using this transceiver module. Lastly, three LEDs are also included in the circuit as speed indicators. This enables the user to know which speed level is currently activated when operating the robot car. A resistance of 220Ω (R1, R2, and R3) is applied to LEDs A, B, and C for minimizing current based on the appropriate resistor value in common LEDs and its availability in the market. These are connected to digital pins A0, A1, and A2 of the Arduino Nano microcontroller. When observed, these pins are analog but can also be used as digital by specifying the pins D14, D15, and D16 to the program uploaded in the microcontroller.

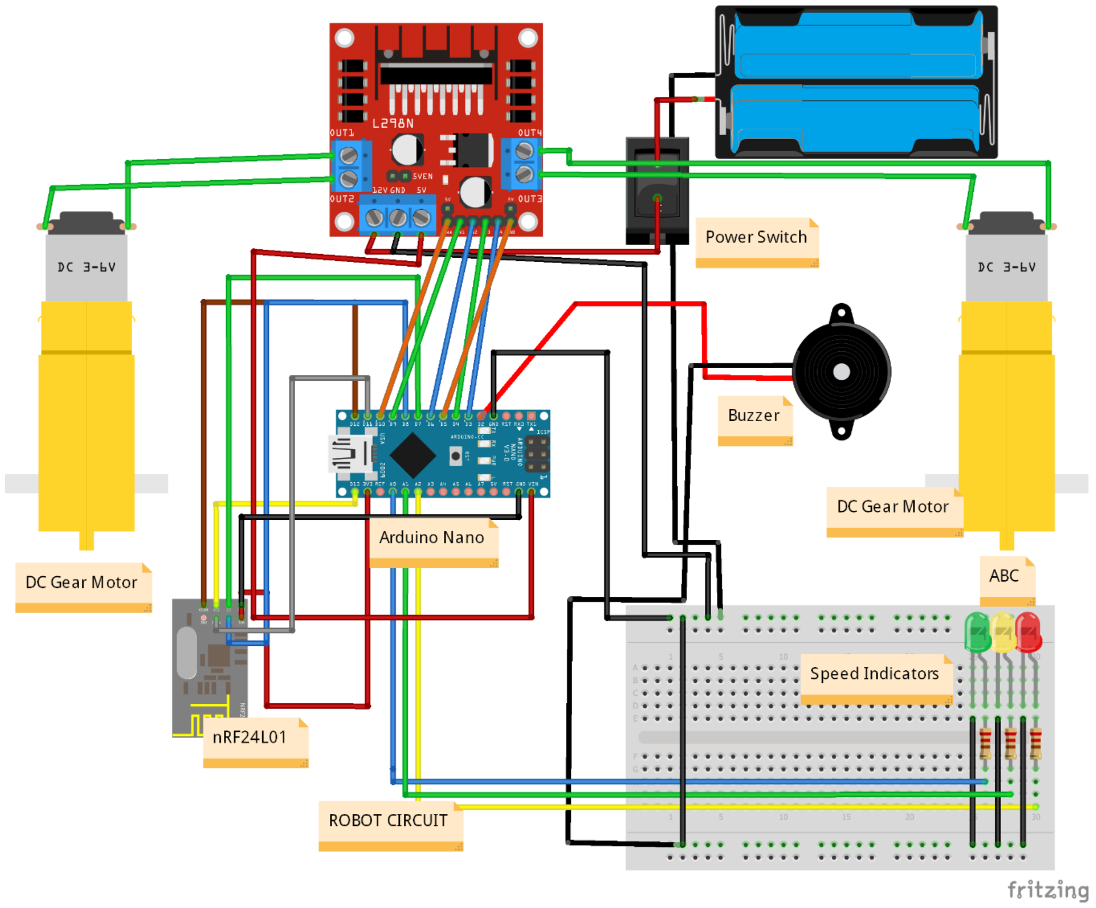

Figure 5: Robot Circuit Diagram

In addition, figure 5 presents the circuit diagram of the robot in which the actual components and connections are shown. This includes the Arduino Nano microcontroller, nRF24L01 transceiver module, two DC gear motors, L298n motor driver, batteries, power switch, buzzer, and three LEDs.

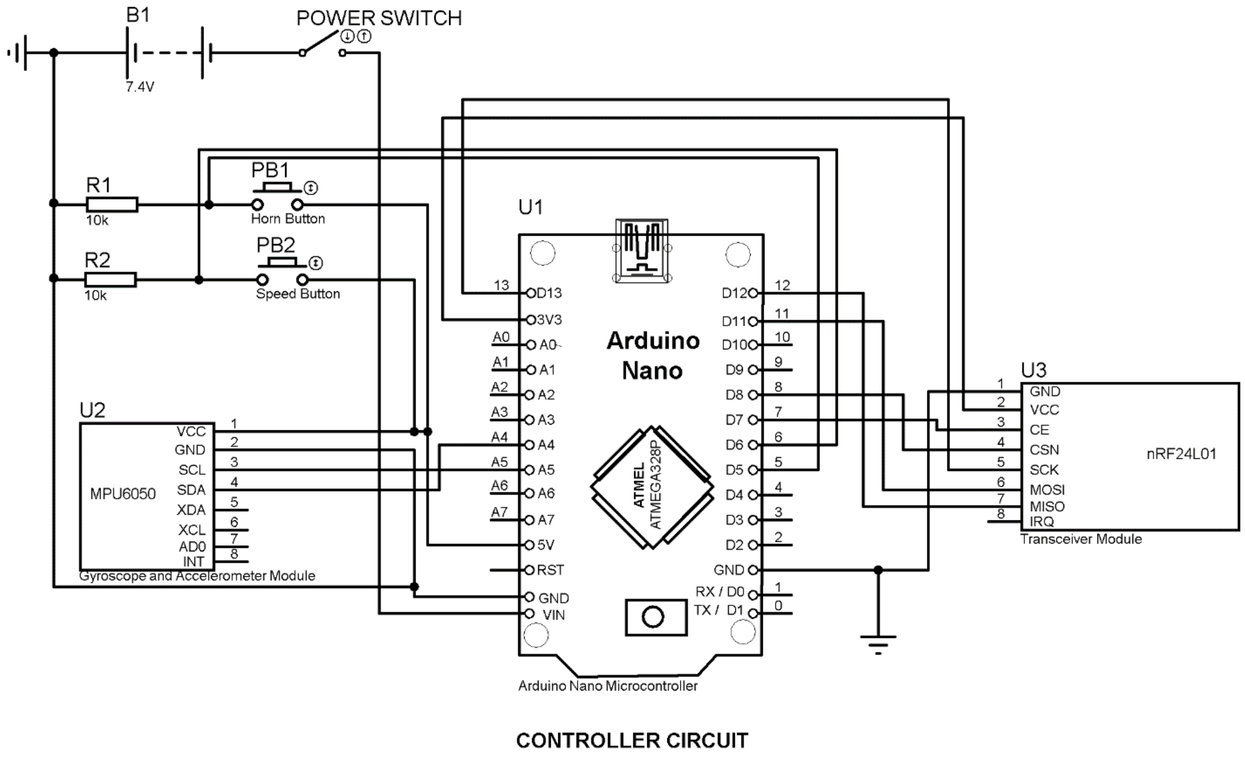

Figure 6: Controller Circuit Schematic Diagram

On the other hand, figure 6 outlines the schematic diagram for the connections of the controller circuit. Similarly, it is powered by a 7.4V battery B1 by connecting two 18650 batteries in series along with a power switch for efficient energy usage. Arduino Nano is also used as the microcontroller of the circuit which handles the control operations and processing. A MPU6050 module for the gyroscope and accelerometer is used which is responsible for the extraction of tilt angles in x and y-axes. These tilt angles are essential in maneuvering the controls of the robot car circuit depending on the thresholds of pitch and roll angles. These controls include forward, backward, left, right, and stop by default. Whichever threshold gets satisfied, a corresponding signal will be set and sent to the robot car circuit wirelessly. In terms of connection, its 5V and GND pins are connected to the 5V pin of the microcontroller and common ground, respectively. Its SCL and SDA pins are connected to pins A5 and A4 of the microcontroller, respectively. These connections are necessary to operate the I2C feature of the module, making it easier to calibrate and control. Moreover, two push-buttons PB1 and PB2 are used by connecting them to the 5V pin of the microcontroller for its power source. A 10kΩ resistor is used as an external pull-down resistor for each button which is also connected to the common ground. PB1 facilitates the activation of horn of the robot car while PB2 handles the speed transitioning of the robot car wirelessly. Wireless connection is made possible by integrating a nRF24L01 transceiver module. Its GND and VCC pins are connected to the common ground and 3V3 pin of the microcontroller for 3.3V power source, respectively. Its CE, CSN, SCK, MOSI, and MISO pins are connected to digital pins D7, D8, D13, D12, and D11 of the microcontroller, respectively. These connections are essential to operate the transceiver module, enabling wireless communication with the robot car circuit.

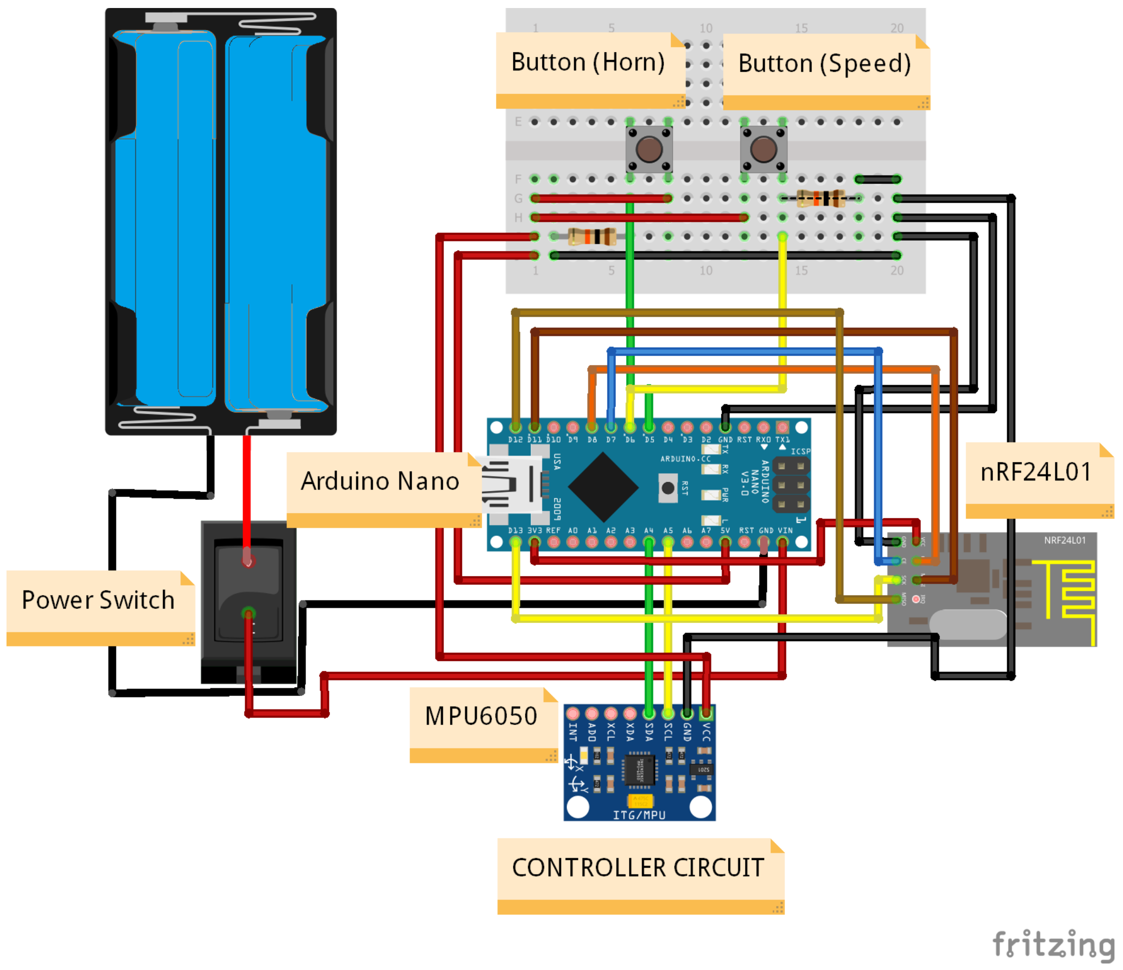

Figure 7: Controller Circuit Diagram

Similarly, figure 7 presents the circuit diagram of the controller that presents the actual components used with their connections. This includes the Arduino Nano microcontroller, nRF24L01 transceiver module, batteries, power switch, two push-button for speed and horn, and MPU6050 sensor.

SOFTWARE DEVELOPMENT

The software development part of the project mainly deals with the programming of the controller and robot car circuits using C++ programming language and Arduino IDE. Essential functions were specifically made for the gesture-controlled robot car project.

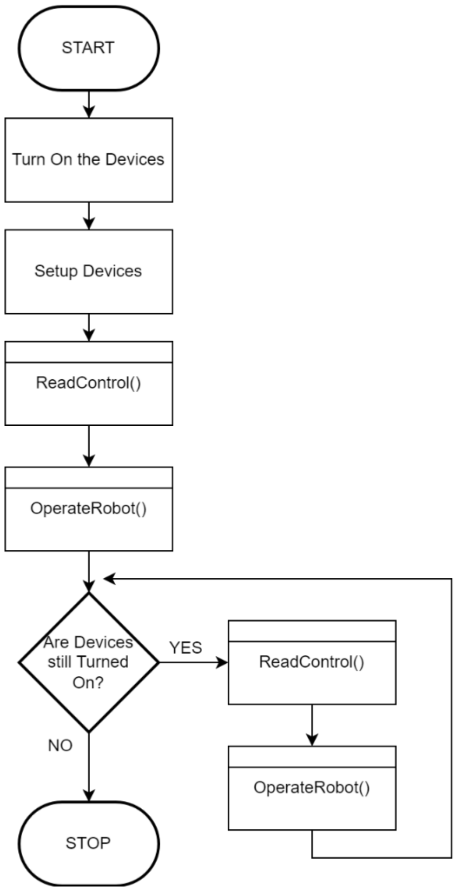

Figure 8: Main Flowchart

Figure 8 presents the main flow of the system. The user must first turn on the controller and robot car circuits by pressing their respective power switches. Once powered, both circuits would initially execute their setup thereby initializing all essential variables and values to prepare for execution. The ReadControl function is then executed which handles the setting of control operations for the controller. The control signal is then sent to the robot car circuit in which it executes the OperateRobot function based on the signal sent from the controller. If both circuits are still powered on, these two functions are executed continuously. Turning off their switches would then stop the program execution.

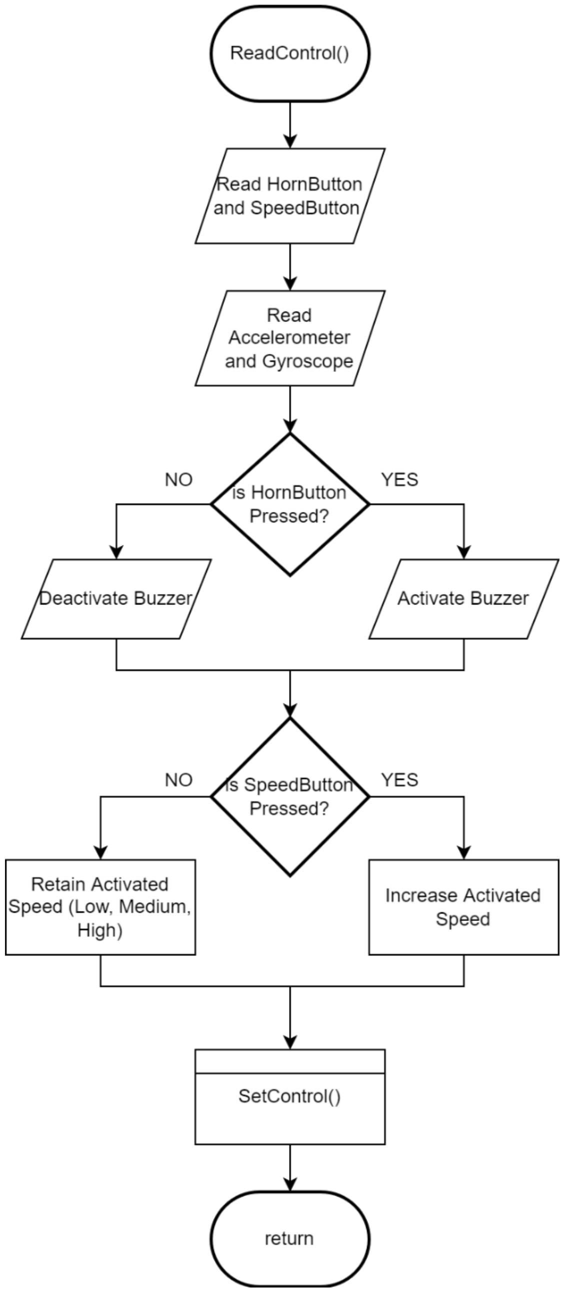

Figure 9: Read Control Function

The ReadControl function is used to read inputs from the buttons and sensor. It first reads a signal from the horn and speed buttons for activation. It then reads data from accelerometer and gyroscope. Once these inputs are read and set, an if-else condition is used to check whether either the horn button or speed button is pressed. If the horn button is pressed, it will activate the buzzer and deactivating it otherwise. On the other hand, if the speed button is pressed, it will transition the current speed level to the next level which can either be low, medium or high. Not pressing it would just retain whichever speed is currently activated. After reading from these two buttons, the SetControl function is executed.

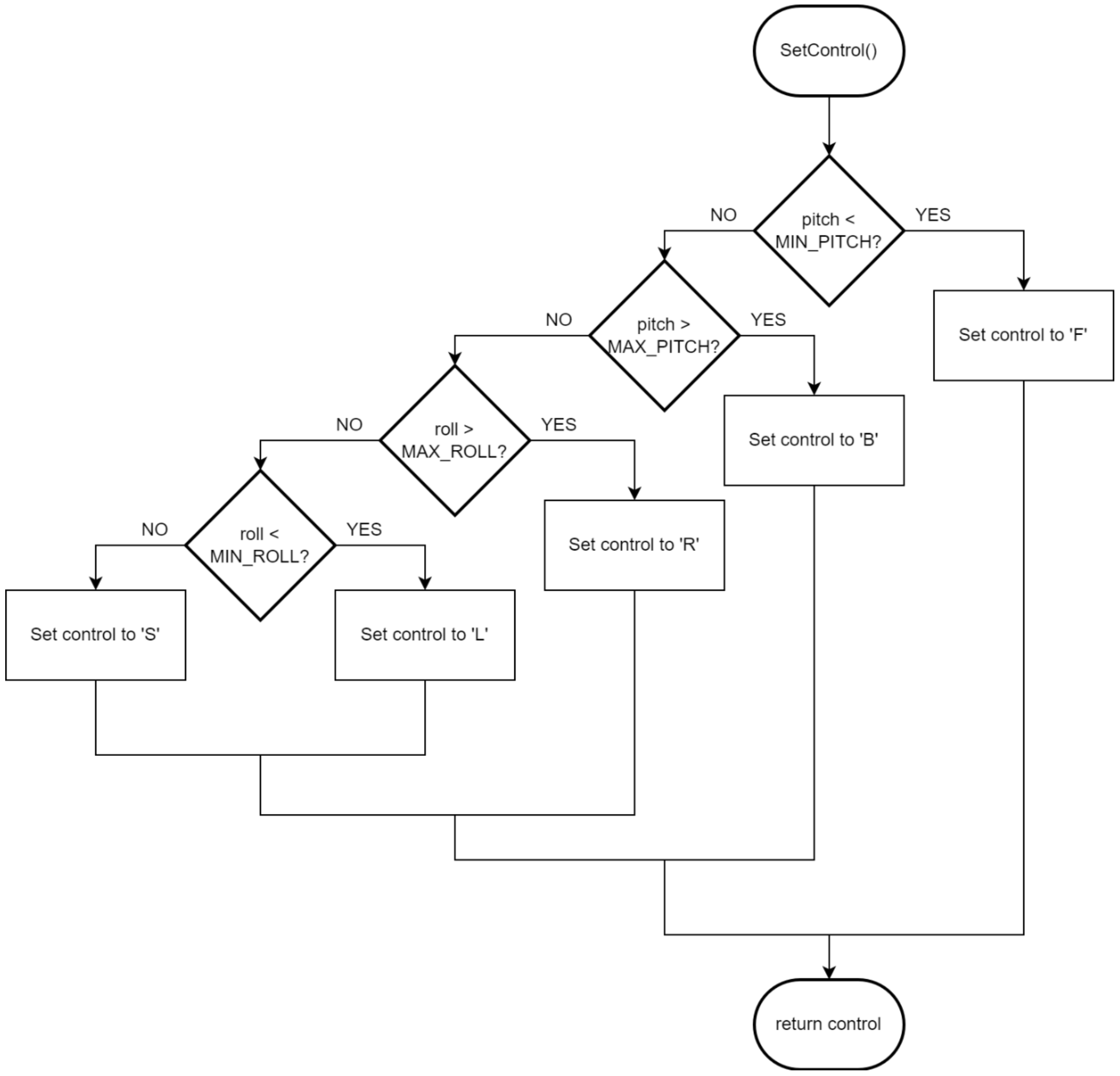

Figure 10: Set Control Function

The SetControl function uses the retrieved and computed data from the gyroscope and accelerometer. It makes use of two Euler angles from roll and pitch axes. Thresholding for both variables are used by applying them in a series of if-else conditional statements. The pitch and roll angles are used as independent variables while the control is dependent to it whichever condition is satisfied. The if-else condition checks first if the pitch angle is less than the minimum pitch angle threshold. Satisfying this condition would set the value of control to character “F”, representing the control operation of “Forward.” Otherwise, it will go to the next conditional statement which checks whether the pitch angle is greater than the maximum pitch angle threshold. Satisfying this condition would set the control value to “B” signifying “Backward” control. If not satisfied, it will go to the next conditional statement which checks if the roll angle is greater than the maximum roll angle threshold. If yes, the control value will be set to “R” for the “Right” control direction. Otherwise, it will go to the last condition which checks whether the roll angle is less than the minimum roll angle threshold. If this condition gets satisfied, the control would be set to “L” for “Left” direction control. Otherwise, it will set the control value to the default “S” character representing a “Stop” control. This function then returns the control value whichever condition was satisfied. Realistically speaking, this control value is transmitted to the other circuit via a transceiver module.

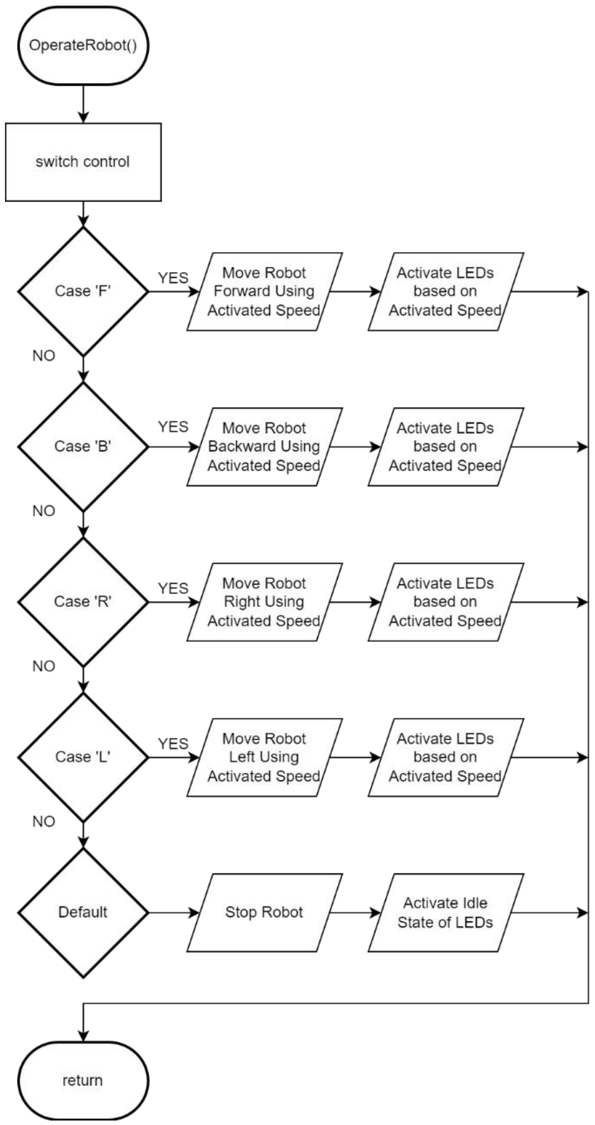
OperateRobot()
switch control
Case 'F'
YES
Move Robot Forward Using Activated Speed
Activate LEDs based on Activated Speed
NO
Case 'B'
YES
Move Robot Backward Using Activated Speed
Activate LEDs based on Activated Speed
NO
Case 'R'
YES
Move Robot Right Using Activated Speed
Activate LEDs based on Activated Speed
NO
Case 'L'
YES
Move Robot Left Using Activated Speed
Activate LEDs based on Activated Speed
NO
Default
Stop Robot
Activate Idle State of LEDs
return

Figure 11: Operate Robot Function

Lastly, the OperateRobot function facilitates the operation of the robot car circuit based on the control value signal sent from the controller circuit. A switch-case statement is used to check the value of the control variable then output the corresponding outcome. If the control value has a value of "F", it will move the robot forward then activate the LEDs based on the activated speed. A value of "B" will move the robot car backward then activating the LEDs pertaining to the activated speed. On the other hand, a value of "R" would move the robot car in a right direction which also activates the LEDs based on the activated speed. A value of "L" would then move the robot car in a left direction while also indicating the activated speed on its LEDs. Otherwise, the default output would carry out which stops the robot car from moving and activating the idle mode of the LEDs. Either of these statements satisfied would return the program to its main flow.

SYSTEM LOGIC

Parameter	Condition or State	Outcome
Horn Button	LOW	Deactivate Buzzer
	HIGH	Activate Buzzer
Speed Button	LOW	Retain Speed
	HIGH	Transition Speed
Pitch Angle	Less than -30° (Minimum Pitch)	Set control to 'F'
	Greater than 30° (Maximum Pitch)	Set control to 'B'
Roll Angle	Less than -30° (Minimum Roll)	Set control to 'L'
	Greater than 30° (Maximum Roll)	Set control to 'R'
Default	Default Case	Set control to 'S'

Table 1: Controller Circuit Logic

Table 1 outlines the functionality of the controller circuit. Four parameters or inputs are being observed from the controller. This includes the horn button, speed button, and tilt angles (pitch and roll). Pressing the horn button sends a HIGH signal which then activates the buzzer while not pressing it will return a LOW signal for buzzer deactivation. Similarly, pressing the speed button will trigger its HIGH signal to enable speed transitioning which can either be low, medium, or high speed. Not pressing it will retain its currently activated speed. On the other hand, tilt angles are important parameters since these are responsible for the signal to be sent to the robot circuit via the gyroscope and accelerometer. These angles include the pitch which is the angle along the y-axis and roll angle along the x-axis. Minimum and maximum thresholds for these angles are set to facilitate the control signal to be sent in the robot circuit. Tilting the sensor or the user's hand enough in a sinking position would get a pitch angle that is less than the minimum pitch of -30°. This would set the control to character *‘F’*. On the other hand, tilting the sensor in a rising position will result in a pitch angle reading that is greater than the maximum pitch of 30°. This would set the control to character *‘B’*. In terms of the roll angle, this can be maneuvered by tilting the user’s hand or sensor on either left or right side. Tilting it on the left side will get a roll angle reading that is less than the minimum roll of -30° thereby, setting the control to character *‘L’*. Alternately, tilting the sensor to the right side will result to a roll angle that is greater than the maximum roll of 30° which sets the control to character *‘R’*. Lastly, not satisfying either of these thresholds indicate that the sensor or user’s hand is in level. This will set the control to character *‘S’* which is the default control signal to be sent in the robot circuit.

Speed	LED A (Green)	LED B (Yellow)	LED C (Red)
Idle	HIGH	LOW	HIGH
Low	HIGH	LOW	LOW
Medium	LOW	HIGH	LOW
High	LOW	LOW	HIGH

Table 2: Speed Indicators Logic

Moreover, table 2 outlines the speed indicator states depending on the speed state of the robot circuit. A speed of idle indicates a stop state of the robot car which turns on the green (LED A) and red LEDs (LED C). This functionality mimics the headlights of a car since headlights of an actual car are positioned on the left and right corner which is then applied to the speed indicator logic. On the other hand, when the robot car is moving in any direction, its headlight will change based on the speed level currently activated. By default, the speed is set to low which then turns on the green LED (LED A) while the rest are turned off. Once speed is transitioned to medium level, the yellow LED (LED B) will turn on while the rest are turned off. Lastly, having a high speed level will turn on the red LED (LED C) while the rest are turned off. These indicators are essential for the user to know on what speed is currently operated in the robot car circuit.

Control	IN1	IN2	Left Wheel	IN3	IN4	Right Wheel	Operation
F	HIGH	LOW	Forward	HIGH	LOW	Forward	Move Forward
B	LOW	HIGH	Backward	LOW	HIGH	Backward	Move Backward
L	LOW	HIGH	Backward	HIGH	LOW	Forward	Turn Left
R	HIGH	LOW	Forward	LOW	HIGH	Backward	Turn Right
S	LOW	LOW	Stop	LOW	LOW	Stop	Stop or Idle

Table 3: Motors and Robot Circuit Logic

Furthermore, table 3 presents the motor logic and the robot circuit's operation based on the control signal sent from the controller circuit. Since the motors are controlled via its motor driver, input pins signals of the motor driver are manipulated by program to control its states. If the control signal has a value of 'F', both wheels must rotate forward. To enable this, IN1 and IN2 are set with HIGH and LOW signals, respectively which rotates the left wheel in a forward direction. Similarly, IN3 and IN4 are set with HIGH and LOW signals, respectively to rotate in a forward direction as well. Both wheels rotating in a forward direction would move the robot circuit forward as its operation. On the other hand, a control signal value of 'B' would rotate both wheels in a backward direction. This is done by setting IN1 and IN2 with LOW and HIGH signals, respectively which rotates the left wheel in a backward direction. IN3 and IN4 are set with LOW and HIGH signals, respectively to rotate the right wheel in a backward direction. Both wheels rotating in a backward direction would move the robot car circuit backward. As for a control value of 'L', the left wheel must rotate backward while the right wheel must rotate forward to turn the robot circuit left. This is done by setting the IN1 and IN2 to LOW and HIGH signals, respectively, rotating the left wheel backward while also setting IN3 and IN4 to HIGH and LOW signals, respectively to rotate the right wheel forward. Doing so would turn the robot car circuit in a left direction. Alternatively, a control value of 'R' would rotate the left motor in a forward direction while the right motor in a backward direction. To enable this, IN1 and IN2 must be set to HIGH and LOW signals, respectively turning the left motor forward while IN3 and IN4 must be set to LOW and HIGH signals, respectively to turn the right motor backward. With the contradicting direction, it turns the robot car circuit in the right direction. Lastly, a control value of 'S' sets all inputs IN1, IN2, IN3, and IN4 to LOW signal which stops and turns off the motors. Thus, stopping the robot car circuit or putting it in idle state.

CASING DESIGN

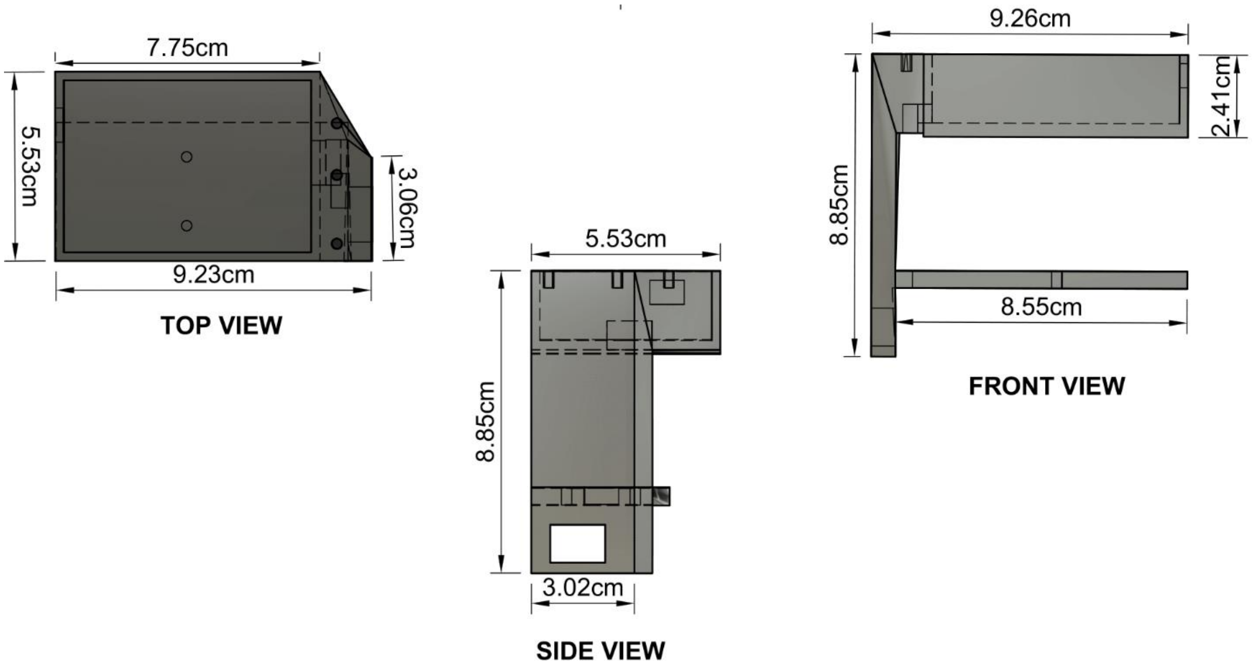

Figure 12: Dimensions of Controller Casing

Figure 12 presents the dimensions of the controller casing in different views. This casing will be used to enclose the controller circuit PCB and to provide stable platform when using the sensor. These dimensions were based on the sizing of the user's hand to easily facilitate the tilting control of the controller.

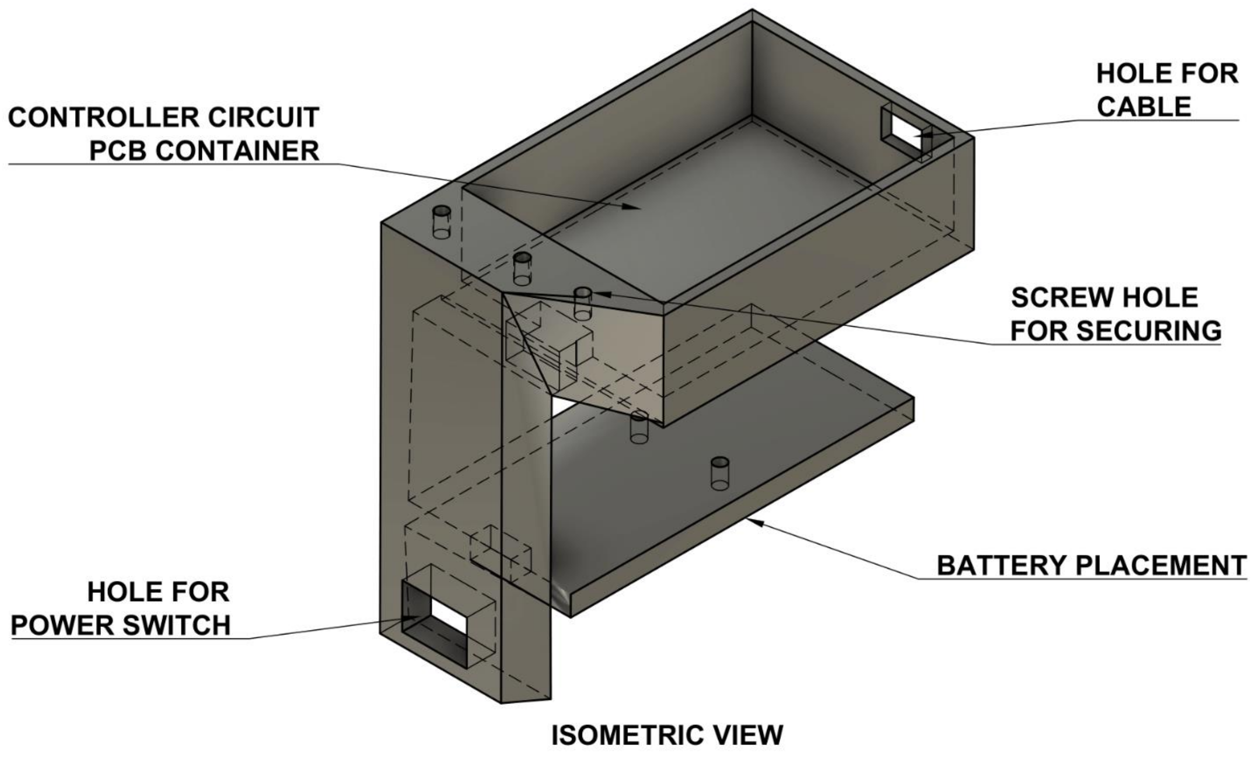

Figure 13: Isometric View of Controller Casing

Furthermore, figure 13 shows the isometric view of the controller casing to provide a detailed overview of its parts. A container platform for the controller circuit PCB was made to provide a flat surface for optimal performance since the sensor solely relies on its origin position when performing control operations. A hole for the microcontroller cable was also added to ensure easiness when debugging it or modifying its code when necessary. Aside from that, a hole for a power switch was included to attach it securely. The lower portion of the casing has a space provided for the battery placement where it will be attached using screws hence, several screw holes are used for component securing. This applies to the top of the casing too as three holes are included for the lid securement.

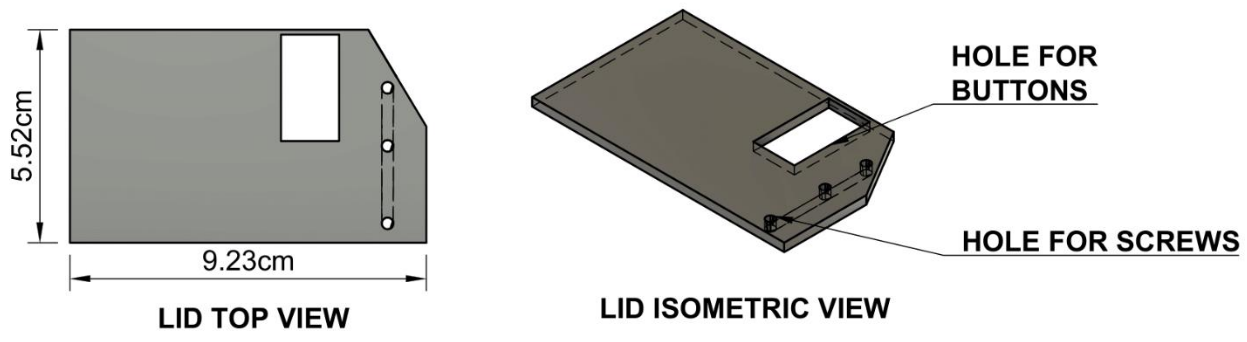

Figure 14: Lid of Controller Casing

As observed in figure 14, a lid is also implemented to complete the overall controller casing. This is essential to protect the PCB during operation. A hole for the buttons is also engraved to the lid for easy access when activating speed transition and horn features. As mentioned, three holes are included in the lid that is parallel to the top portion of the controller casing for attachment.

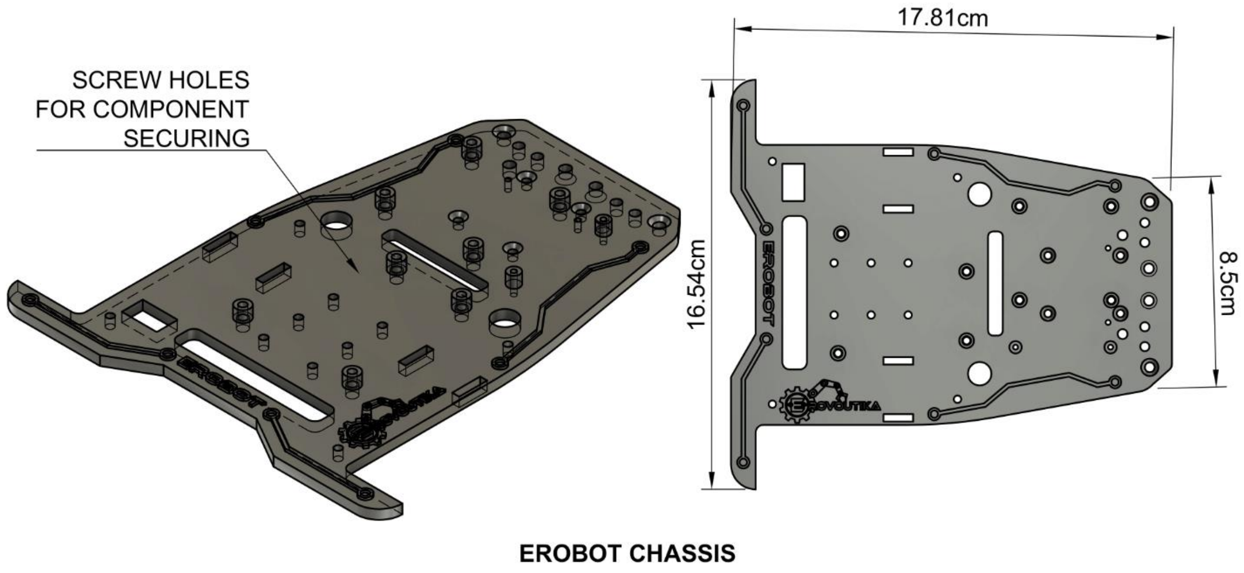

Figure 15: Erobot Chassis by Erovoutika

Figure 15 presents the Erobot Chassis and its dimensions which is a property of the company Erovoutika. This is also the chassis used for the development of the project's prototype, ensuring consistency and optimal robot car chassis as proven by several products of Erovoutika. As observed, several screw holes are applied to the chassis, providing effective assembly when incorporating different components such as motors, driver, etc.

PROTOTYPE DEVELOPMENT

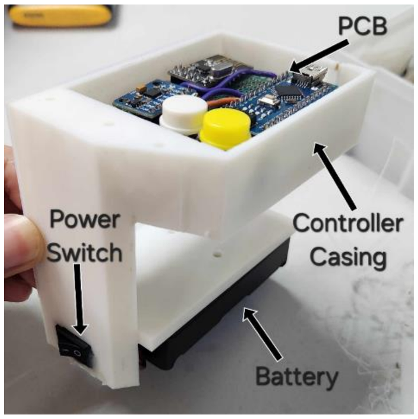

Figure 16: Controller Setup

For the actual prototype and setup, figure 16 presents the controller casing along with its parts. The casing was designed in this particular way to allow stable and secure usage when holding it. A power switch is placed on its left for easy access when pressing it using the right-hand thumb. The battery is attached to the lower part of the controller to minimize the space required at the top of the casing where the PCB is contained.

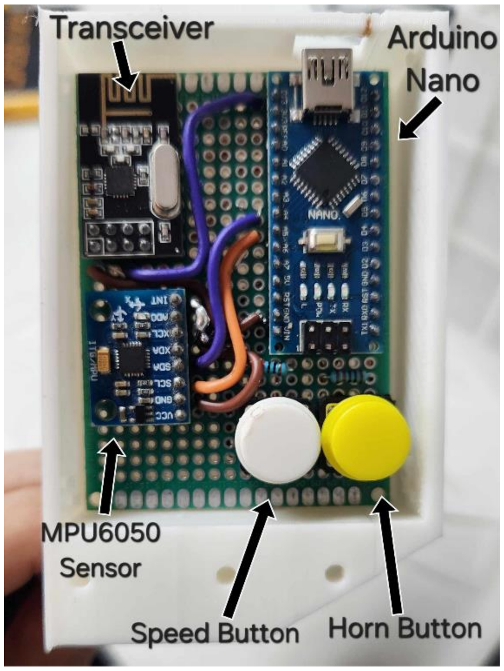

Figure 17: Controller PCB

Figure 17 presents the components soldered into the PCB. A universal PCB was used to implement the controller circuit since it does not have complex connections and several components which do not require a custom PCB and to reduce company project expenses. The components presented in the controller circuit schematic diagram are all visible in the PCB except for the battery and power switch which are placed in a different part of the casing. This includes the transceiver module, Arduino Nano, two push-buttons, and MPU6050 sensor. These were soldered by following the connections in its schematic diagram.

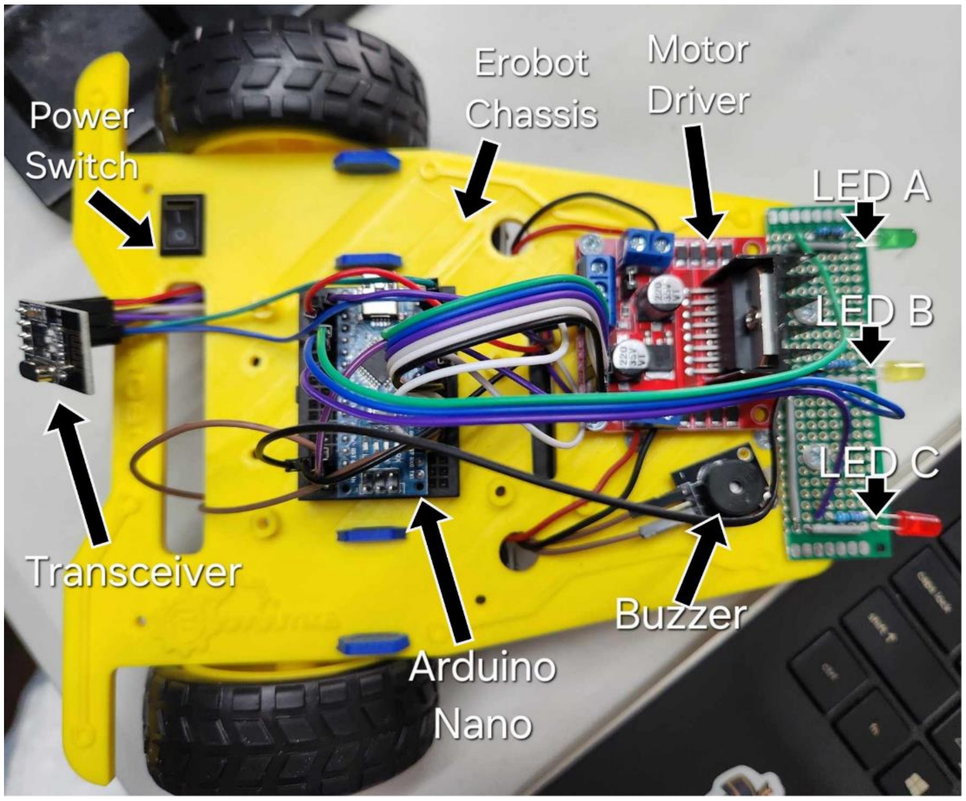

Figure 18: Robot Car Setup (Top View)

Furthermore, figure 18 presents the actual prototype setup of the robot car. This shows the components involved in the circuit that are visible at the top. This includes the power switch, transceiver module, Arduino Nano, motor driver, buzzer, and the three LEDs (LED A, LED B, and LED C). The LEDs were soldered in the universal PCB for consistent connection and attached to the front of the robot chassis. The robot car was built by using the Erobot Chassis provided by Erovoutika, ensuring consistent design for their robot projects.

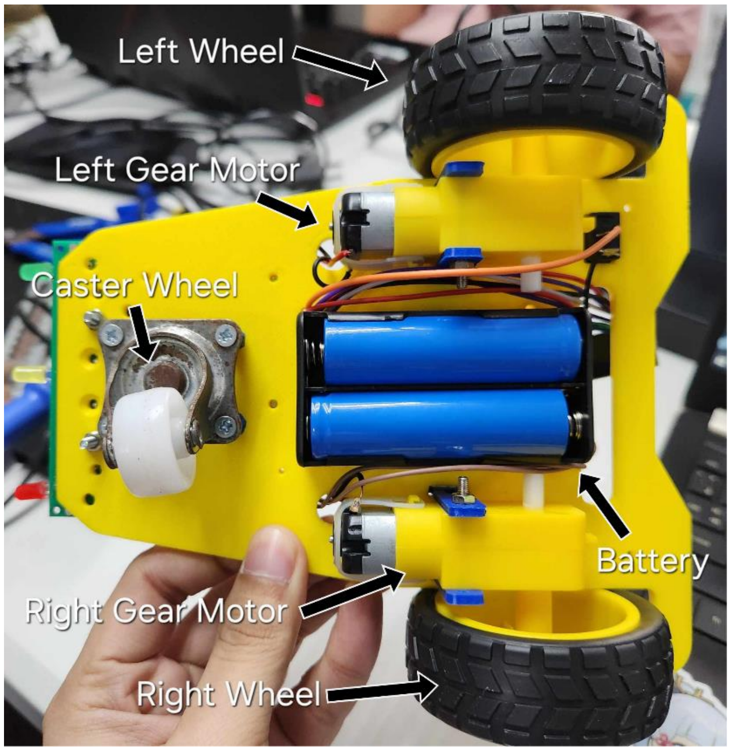

Figure 19: Robot Car Setup (Bottom View)

Lastly, figure 19 shows the bottom view of the robot car setup. This includes the parts and components visible under it which includes the left gear motor with the left wheel, a caster wheel, the right gear motor with the right wheel, and the battery. These are placed by following the common setup of robot cars in Erovoutika.

PROTOTYPE SETUP

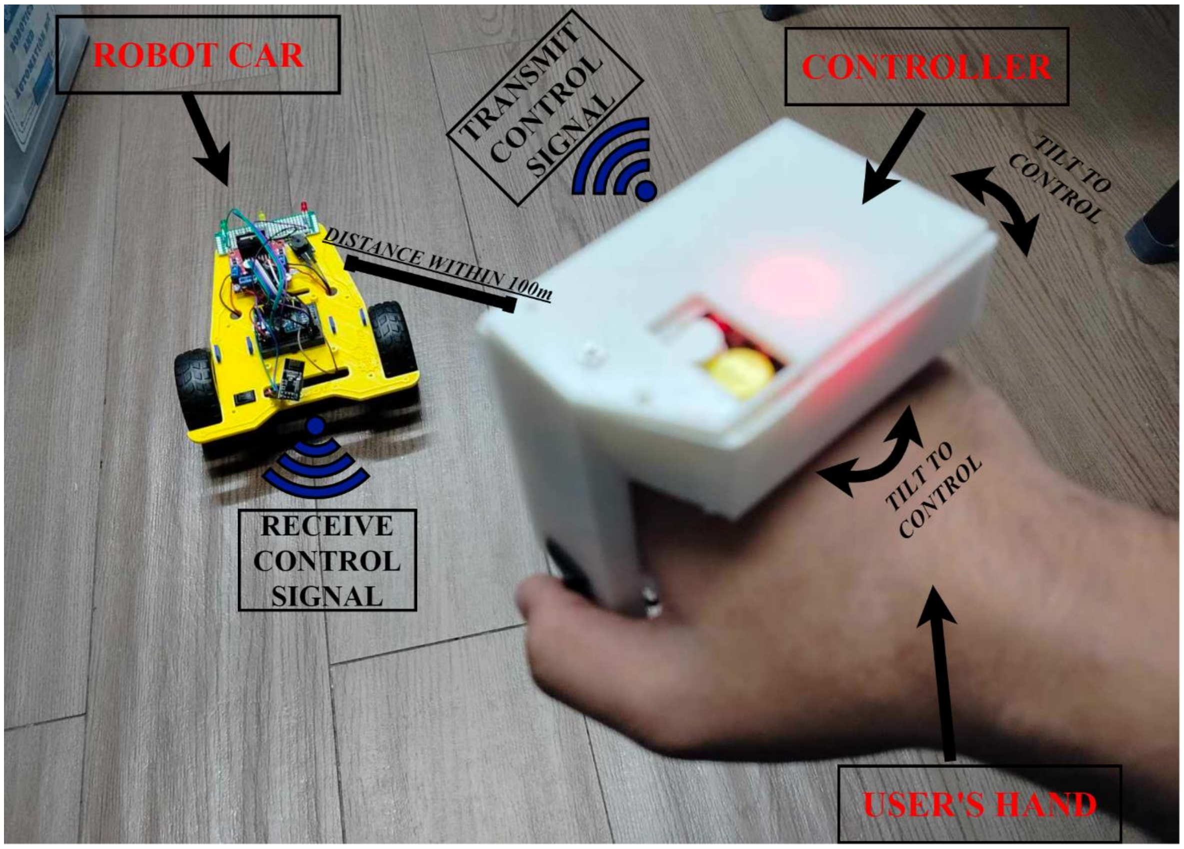

Figure 20: Prototype Setup

Figure 20 presents the setup of the prototype in which the intern tested the project. This was done by inserting the user’s hand into the gap of the controller casing then holding it in the same manner as shown in the figure. As observed, the position or angle of the user’s hand with the controller must be level or flat relative to the axes since this will serve as the origin of the gyroscope and accelerometer. Hence, the user must ensure that the user’s hand and controller are level before turning on the devices. The robot car must be placed on a flat surface for seamless operation. Tilting the controller through the user’s hand would modify the control signal being sent from the controller to the robot car thus, changing the direction of the robot car’s operation. The distance between the controller and robot car must be within 100m since this is the limitation of the transceiver module being used. Tilting the controller to the left would turn the robot car to the left while tilting the controller to the right would turn the robot car to the right. On the other hand, tilting the controller downward would forward the robot car while tilting it upward would backward the robot car.

TESTING

The intern tested the project by wearing and holding the controller on the right hand then placing the robot car on a flat surface. After holding the controller properly, the intern turned on the switch first of the robot car. The robot car must be placed within 100m radius between the user and the controller due to the distance limitation of the transceiver modules. After powering the robot car, the intern must turn on the controller while the position of the controller is level or flat relative to the hand. Once turned on, the intern can now tilt angles in different directions to test whether the actual outcome of the robot car movement is consistent with the expected outcome. The table below presents the results of the testing.

Action	Expected Outcome	Actual Result
Level the Controller or Flat	Robot Car Stops	Robot Car Stops
Tilt the Controller Below	Robot Car Moves Forward	Robot Car Moves Forward
Tilt the Controller Above	Robot Car Moves Backward	Robot Car Moves Backward
Tilt the Controller Left	Robot Car Turns Left	Robot Car Turns Left
Tilt the Controller Right	Robot Car Turns Right	Robot Car Turns Right
Press the Horn Button	Robot Car Beeps	Robot Car Beeps
Press the Speed Button	Robot Car Changes Speed	Robot Car Changes Speed

Table 4: Test Results

As observed in table 4, all the available actions in the project via the controller are tested. This includes the sensor operations and the buttons. The actual results are all consistent with the expected outcomes. From this, the intern was able to determine that the project is successful, and functions as intended.

CONCLUSION

In conclusion, the intern was able to design and develop a gesture-controlled robot project. Specifically, the intern was able to design a controller circuit that makes use of a gyroscope and accelerometer sensor module to facilitate the movement of the robot and the intern was able to successfully develop a robot car that is capable of moving based on the gesture controller circuit. The intern was also successful in integrating transceiver modules to allow wireless connectivity between the controller and robot circuits. Lastly, the intern successfully implemented a prototype for the whole project and tested them by tabulating all possible controls that can be triggered in the gesture-controlled robot project.

RECOMMENDATION

In terms of the gesture-controlled robot car project, certain improvements can be made. Most importantly, the casing of the robot car circuit. The company required the intern to only use the company robot chassis and to just let the wirings exposed for demonstration purposes. With the quality of the project, it can be improved by developing a more durable and secured robot car casing that does not expose the wirings and components while also protecting the components. Furthermore, if longer time is provided for the project development and its use is not restricted to new products, the intern can also improve it by applying specific features to the gesture-controlled robot car project such as pick and place feature and object avoidance. By adding the pick and place feature, the robot can be controlled using tilt angles while also being able to pick an object and place it to a different location. Also, the object avoidance feature can serve as precautionary measures to the robot car to not bump to objects that can potentially damage and affect its casing and connections.

REFERENCES

[1] H. Hashim, "Special Orthogonal Group SO(3), Euler Angles, Angle-axis, Rodriguez Vector and Unit-Quaternion: Overview, Mapping and Challenges," 2019. doi: 10.48550/arXiv.1909.06669.

[2] "Attitude Estimation Using IMU Sensors," Karooza. [Online]. Available: https://karooza.net/attitude-estimation-using-imu-sensors.