Department of Computer & Software Engineering

College of E&ME, NUST, Rawalpindi

Microcontroller and Microprocessor Based Design

Project Report

Title: Neo Octane by MADS

Group Members:

Saad Rasheed – Reg # 337909

Ammad Ali – Reg # 346904

Mashaal Ibne Masha Allah – Reg # 341923

Daniyal Zahid – Reg # 359080

Abstract— With the continuous increase in car ownership, the modern intelligent transportation system has increasingly regarded vehicle safety as its key factor. Between 2002 and 2012, the lack of proper obstacle avoidance contributed to the deaths of millions of people in traffic accidents around the world, and the economic cost of these accidents amounted to $0.48 trillion. More than 90% of these accidents are caused by human factors. Therefore, to avoid vehicle collisions and minimize the impact of accidents, a method of adding an increasing proportion of active safety systems is proposed. Ever since drive-by-wire technology (DBW) emerged, there has been an ongoing effort to push cars away from their mechanical features. The deployment of autonomous vehicles with collision avoidance is the most significant possibility of this feat.

Contents

👋 Introduction:

The main aim of this project is to construct a remote-controlled vehicle using a microcontroller. A smartphone will be used to control the movement of the vehicle. Tilting the phone will move the vehicle in the corresponding direction. This can be achieved using the smartphone's built-in gyroscope. The vehicle will be equipped with 4 motors and 4 fixed wheels, and a motor controller will be used to control the speeds of said motors. To provide an interface to the user, we'll be using an application called Blynk IoT. This will provide the user with ability to control the vehicle as well as receive feedback from it regarding any collision or possible collision. The vehicle will be equipped with proximity sensors to prevent the vehicle from any collisions from nearby objects. To power the automobile and the microcontroller, rechargeable batteries will be used.

🧰 Parts:

🧠 Arduino Uno

👮 L298N Motor Driver

🛞 Motors and Wheels (4)

🏠 Chassis

🔋 18650 rechargeable batteries (3)

🚨 Infrared Sensors (2)

📡 Bluetooth Module (HC-05)

🔊 Buzzer

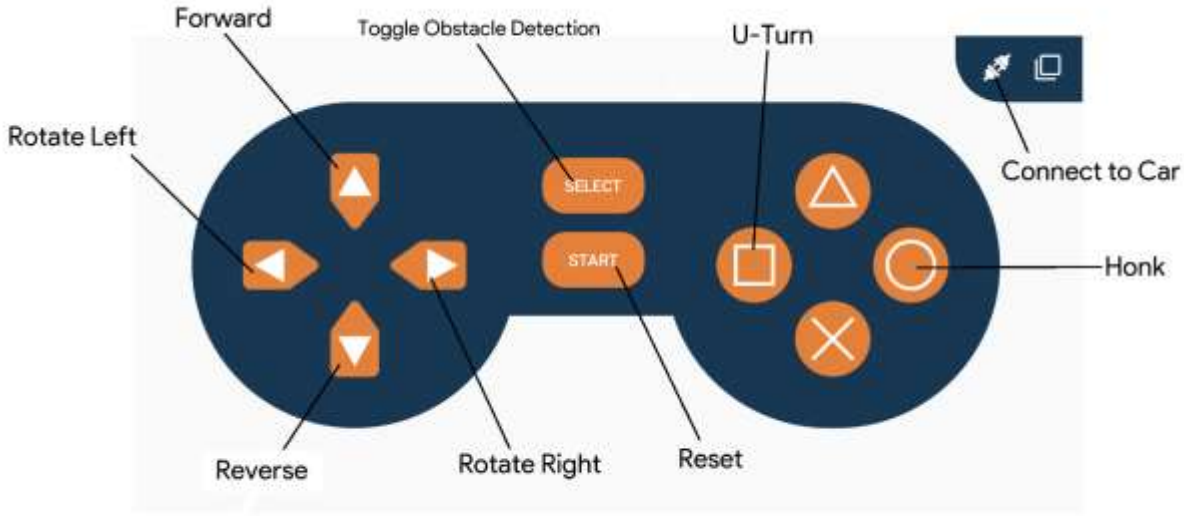
📺 OLED Screen

🔌 Jumper wires

📌 Breadboards

🧭 Navigation:

To navigate the car, built-in interface of Dabble app has been used. Each button on the interface has a separate function. These functions are shown along with the corresponding buttons on the screen below. To increase the speed of the car, tilt the phone forward. To decrease the speed, tilt the phone backward. To steer the car to the right, tilt the phone to the right. To steer the car to the left, tilt the phone to the left.



🚗 Locomotion:

L298N is the brain behind motor control. Our main power supply, 3 18650 batteries, are connected to this module. The 5-volt supply from the module is used to power ESP32.

OUTA and OUTB are used to control two motors each for each side. ENA (enable A) is used to control the speed of the OUTA motors. ENB (enable B) is used to control the speed of OUTB motors. IN1 and IN2 are used together to control the direction of OUTA motors. IN3 and IN4 are used together to control the direction of OUTB motors.

IN(1/3)	IN(2/4)	Motor(A/B)
LOW	LOW	Stop
LOW	HIGH	Forward
HIGH	LOW	Backward
HIGH	HIGH	Stop

⚠ Obstacle Avoidance:

To avoid crashing into obstacles, ultrasonic sensor and trigger switches work together. Ultrasonic sensor calculates the distance between the car and any obstacle in front of the car. If the distance is lesser than a threshold value, the car will not move forward, and the user is notified of the obstacle on the interface. In case the obstacle is of low profile and the ultrasonic sensor cannot detect it, the car will bump into the obstacle and the trigger switches will be triggered. This will send an interrupt to the ESP which will in turn send notification to the user.

Code:

```
// v1.6
// added OLED code.

#define CUSTOM_SETTINGS
#define INCLUDE_SENSOR_MODULE
#define INCLUDE_GAMEPAD_MODULE
#define INCLUDE_NOTIFICATION_MODULE

#include <Dabble.h>
#include <L298NX2.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// pins
#define ENA 5
#define ENB 6
#define IN1 4
#define IN2 7
#define IN3 8
#define IN4 12
#define forwardIRPin 11
#define backwardIRPin 10
#define buzzer A0
#define ResetPin A1
```

```
// thresholds
#define maxSpeed 255
#define minSpeed 80
#define upperThresh 9.5
#define lowerThresh 0

// motor controllers
L298N leftMotors(ENA, IN1, IN2);
L298N rightMotors(ENB, IN3, IN4);

// flags
bool upButton = false;
bool downButton = false;
bool leftButton = false;
bool rightButton = false;
bool squareButton = false;
bool circleButton = false;
bool selectButton = false;
bool IROff = true;
bool forwardStop = false;
bool backwardStop = false;

int speed = 0;
double yaxis = 0;
double xaxis = 0;
double zaxis = 0;
int rightSpeed = 0;
int leftSpeed = 0;
L298N::Direction leftDirection;
L298N::Direction rightDirection;

// reset the arduino
void reset() {pinMode(ResetPin, OUTPUT);}

// display a single character on the OLED
void singleCharDisplay(char c)
{
  if (leftButton || rightButton) {return;}
  display.clearDisplay();
  display.setCursor(56, 5);
  display.setTextColor(WHITE);
  display.setTextSize(4);
  display.print(c);
}
```

```
// display an arrow on the OLED
void arrowDisplay(bool right)
{
  display.clearDisplay();
  display.setCursor(32, 5);
  display.setTextColor(WHITE);
  display.setTextSize(4);
  right? display.print("<--"): display.print("-->");
}

// get button states and accelerometer values from Dabble app
void communications()
{
  Dabble.processInput();
  yaxis = Sensor.getAccelerometerYaxis();
  xaxis = Sensor.getAccelerometerXaxis();
  zaxis = Sensor.getAccelerometerZaxis();
  upButton = GamePad.isUpPressed();
  downButton = GamePad.isDownPressed();
  rightButton = GamePad.isRightPressed();
  leftButton = GamePad.isLeftPressed();
  squareButton = GamePad.isSquarePressed();
  circleButton = GamePad.isCirclePressed();

  if (GamePad.isStartPressed()) {reset();}

  if (GamePad.isSelectPressed()) {
    IROff = !IROff;
    delay(200);
    Notification.clear();
    Notification.notifyPhone(String("Toggled"));
  }
}

// calculate speed from phone's accelerometer values
void speedControl()
{
  double diff = zaxis - xaxis;
  speed = 30 * diff;
  if (speed < minSpeed) {speed = 0;}
  if (speed > maxSpeed) {speed = maxSpeed;}
}

// calculate individual speeds of left and right motors depending on the phone's
accelerometer values
```

```
void directionControl()
{
  double offset = 0;
  leftSpeed = speed;
  rightSpeed = speed;
  double y = yaxis;

  // rotate the car to left if left button is pressed
  if (leftButton) {
    leftSpeed = maxSpeed;
    rightSpeed = maxSpeed;
    leftDirection = L298N::BACKWARD;
    rightDirection = L298N::FORWARD;
  }
  // rotate  the car to the right if right button is pressed
  else if (rightButton) {
    leftSpeed = maxSpeed;
    rightSpeed = maxSpeed;
    leftDirection = L298N::FORWARD;
    rightDirection = L298N::BACKWARD;
  }
  else if (y > lowerThresh && y < upperThresh)
  {
    offset = 30 * (y - lowerThresh);
    leftSpeed += offset;
    rightSpeed -= offset;
    leftDirection = L298N::FORWARD;
    rightDirection = L298N::FORWARD;
  }
  else if (y > -upperThresh && y < -lowerThresh)
  {
    offset = 30 * (y + lowerThresh);
    leftSpeed += offset;
    rightSpeed -= offset;
    leftDirection = L298N::FORWARD;
    rightDirection = L298N::FORWARD;
  }
  else if (y > upperThresh)
  {
    leftSpeed = maxSpeed;
    rightSpeed = maxSpeed / 2;
    leftDirection = L298N::FORWARD;
    rightDirection = L298N::BACKWARD;
  }
  else if (y < -upperThresh)
```

```
  {
    leftSpeed = maxSpeed / 2;
    rightSpeed = maxSpeed;
    leftDirection = L298N::BACKWARD;
    rightDirection = L298N::FORWARD;
  }

  // check if leftSpeed and rightSpeed are not out of range
  if (leftSpeed < minSpeed) {leftSpeed = minSpeed;}
  else if (leftSpeed > maxSpeed) {leftSpeed = maxSpeed;}
  if (rightSpeed < minSpeed){rightSpeed = minSpeed;}
  else if (rightSpeed > maxSpeed) {rightSpeed = maxSpeed;}

  // if car has to go backwards, reverse the left and right directions
  if (downButton && !upButton)
  {
    if (leftDirection == L298N::BACKWARD) {leftDirection = L298N::FORWARD;} else
{leftDirection = L298N::BACKWARD;}
    if (rightDirection == L298N::BACKWARD) {rightDirection = L298N::FORWARD;}
else {rightDirection = L298N::BACKWARD;}
  }
}

// run the motors
void runMotors()
{
  leftMotors.setSpeed(leftSpeed);
  rightMotors.setSpeed(rightSpeed);
  leftMotors.run(leftDirection);
  rightMotors.run(rightDirection);
}

// stop the motors
void stopMotors()
{
  leftMotors.stop();
  rightMotors.stop();
}

// turn the car to a certain position using a delay value
void turn(int delayTime)
{
  runMotors();
  delay(delayTime);
  stopMotors();
```

```
}

// manages all the manoeuvres that our car can perform
void manageManoeuvre()
{
  if (squareButton) {
    leftSpeed = maxSpeed;
    rightSpeed = maxSpeed;
    leftDirection = L298N::BACKWARD;
    rightDirection = L298N::FORWARD;
    if (upButton && !downButton) {turn(1000);}
    else {turn(700);}
  }
}

// manages movements of car if without the contraints of IR sensor
void motorsControlWithoutIR()
{
  if (rightButton) {
    arrowDisplay(true);
    runMotors();
  }
  else if (leftButton) {
    arrowDisplay(false);
    runMotors();
  }
  else if (upButton && !downButton) {
    runMotors();
    singleCharDisplay('F');
  }
  else if (!upButton && downButton) {
    runMotors();
    singleCharDisplay('R');
  }
  else {
    stopMotors();
    singleCharDisplay('N');
  }
}

// manages movement of car with obstacle detection turned on
void motorsControlWithIR()
{
  if (rightButton) {
    arrowDisplay(true);
```

```
    leftSpeed = maxSpeed;
    rightSpeed = maxSpeed;
    leftDirection = L298N::FORWARD;
    rightDirection = L298N::BACKWARD;
    runMotors();
  }
  else if (leftButton) {
    arrowDisplay(false);
    leftSpeed = maxSpeed;
    rightSpeed = maxSpeed;
    leftDirection = L298N::BACKWARD;
    rightDirection = L298N::FORWARD;
    runMotors();
  }
  else if (upButton && !downButton)
  {
    if (!forwardStop) {
      runMotors();
      singleCharDisplay('F');
    }
    else if (forwardStop)
    {
      stopMotors();
      Notification.notifyPhone(String("Obstruction infront."));
    }
  }
  else if (!upButton && downButton) {
    if (!backwardStop) {
      runMotors();
      singleCharDisplay('R');
    }
    else if (backwardStop)
    {
      stopMotors();
      Notification.notifyPhone(String("Obstruction at the back."));
    }
  }
  else {
    stopMotors();
    singleCharDisplay('N');
  }
}

void setup()
{
```

```
  Serial.begin(9600);

  // initialize the display
  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }
  display.display();
  delay(100);
  singleCharDisplay('N');
  display.display();

  Dabble.begin(9600); // initialize dabble
  pinMode(forwardIRPin, INPUT);
  pinMode(backwardIRPin, INPUT);
  pinMode(buzzer, OUTPUT);

  Notification.clear();
  Notification.setTitle("Obstacle Detection");
}

void loop()
{
  squareButton = false;
  digitalRead(forwardIRPin) == HIGH? forwardStop = false: forwardStop = true;
  digitalRead(backwardIRPin) == HIGH? backwardStop = false: backwardStop = true;
  communications();
  circleButton? digitalWrite(buzzer, HIGH): digitalWrite(buzzer, LOW); // turn
buzzer on if circle button is pressed on phone
  directionControl();
  speedControl();
  manageManoeuvre();
  IROff? motorsControlWithoutIR(): motorsControlWithIR();
  display.display();
}
```

🤔 Challenges and Solutions:

- We initially used ESP32 because it has built-in Wi-Fi and Bluetooth module, but we had trouble interfacing it with ultrasonic sensors. The ESP microcontroller works on 3.3V logic but the ultrasonic sensor works on 5V logic.
- We used Blynk IOT to interface the car, but a noticeable delay was present in it. Blynk IOT works on Wi-Fi and all the inputs from the app were first uploaded to the Blynk cloud and from there they were transmitted to the ESP.
- The ultrasonic sensors we were using were very inaccurate due to which obstacle avoidance was also inaccurate. We also used trigger switches to stop the car as a back if a collision did occur but the issue with them was that the interrupt that it produced crashed the entire ESP controller. To solve both problems we used Infrared sensors instead. They were very accurate and after using them, there was also no need for trigger switches.
- Most of our challenges were simply avoided by using Arduino Uno as our microcontroller. Instead of using Wi-Fi in the ESP32 we switched to Bluetooth and connected it's a module to Arduino Uno.

🔮 Conclusion:

By using the Arduino microcontroller, we successfully built a Remote-Control car and interfaced it with the Dabble App. The car runs smoothly and is equipped with obstacle avoidance which successfully detects any object in its vicinity. In the future we can create a separate app to navigate the car and provide a more intuitive interface to the user. We can also add a camera at the front of the car and show the feed to the user.