Wheelchair Navigation using EOG Signals

PROBLEM STATEMENT AND OBJECTIVES

Conventional wheelchair control methods, such as joysticks or sip-and-puff systems, may not be suitable for everyone, particularly those with minimal physical movement capabilities. This limitation significantly impacts their quality of life, making them heavily reliant on caregivers for mobility and reducing their ability to perform daily activities independently. The problem, therefore, is the lack of an effective and accessible wheelchair navigation system for individuals with profound motor disabilities.

PROBLEM STATEMENT: To provide a better and effective navigation wheel chair mechanism using EOG signals

OBJECTIVE: To develop and implement a user comfort wheelchair navigation system using EOG signals.

EXPECTED OUTCOMES

- This prototype will demonstrate the feasibility and effectiveness of using EOG for wheelchair control.
- Improved mobility and independence for individuals with severe motor disabilities.
- The system will enable users to navigate their environment with greater ease and autonomy.
- An intuitive and accessible user interface that allows seamless interaction between the user and the wheelchair.

MATERIALS REQUIRED

- **Gel Electrodes:**

Use Case: Captures the electrical signals generated by eye movements.
Placement: Two electrodes are placed at the outer canthi (the outer corners) of the eyes to detect horizontal eye movements.
Two electrodes are placed above and below one of the eyes to detect vertical eye movements.
Ensure that the skin is clean and dry before applying the electrodes to obtain good signal quality.

- **BioAmp EXG Pill:**

Use Case: Amplify and filter the raw EOG signals captured by the gel electrodes.
Placement: The BioAmp EXG Pill is connected to the gel electrodes using appropriate connectors.
It can be placed on the user's head using a headband or attached to a cap to keep it secure and close to the electrodes for minimal signal loss.
Ensure secure and stable connections to maintain signal integrity.

- **RF Transmitter and Receiver:**

Use Case: The RF Transmitter-Receiver pair is used to implement a wireless connection between Bioamp EXG Pill Sensor and Arduino.
Placement: The RF Transmitter section is placed along with Bioamp EXG Pill Sensor and the RF receiver section is placed as the input to the Arduino.

- **Arduino Board:**

Use Case: Process the amplified and filtered EOG signals.
Implement signal processing algorithms to detect specific eye movements.
Generate control commands for the wheelchair based on detected eye movements.
Placement: The Arduino board can be placed in a small enclosure mounted on the wheelchair or carried in a small bag/pouch by the user.
Ensure that it is easily accessible for programming and adjustments.

- **Controlled Vehicle (Motorized Wheelchair):**

Use Case: Respond to control commands generated by the Arduino and move accordingly.
Placement: The controlled vehicle is the motorized wheelchair that the user sits in.

Ensure the wheelchair is equipped with a motor control unit that can interface with the Arduino for receiving commands.

- **Ultrasonic Sensor:**

Use Case: If an obstacle is detected within a predefined safety distance (e.g., less than 25 cm), the system automatically stops the wheelchair to prevent collisions.
Placement: The ultrasonic sensor is mounted at the front of the controlled vehicle to detect obstacles in the path of travel. It should be positioned at a height that optimally detects potential obstacles.

PROCEDURE

In this project, a wheelchair control system was developed using Electrooculography (EOG) signals. The system detects eye movements via electrodes and translates them into control commands for a wheelchair prototype. This system is implemented using an Arduino board, which processed the EOG signals and controlled the wheelchair's motors. The project was divided into five main steps: EOG Signal Acquisition, Signal Processing, Movement Classification, Motor Control, and Obstacle Avoidance.

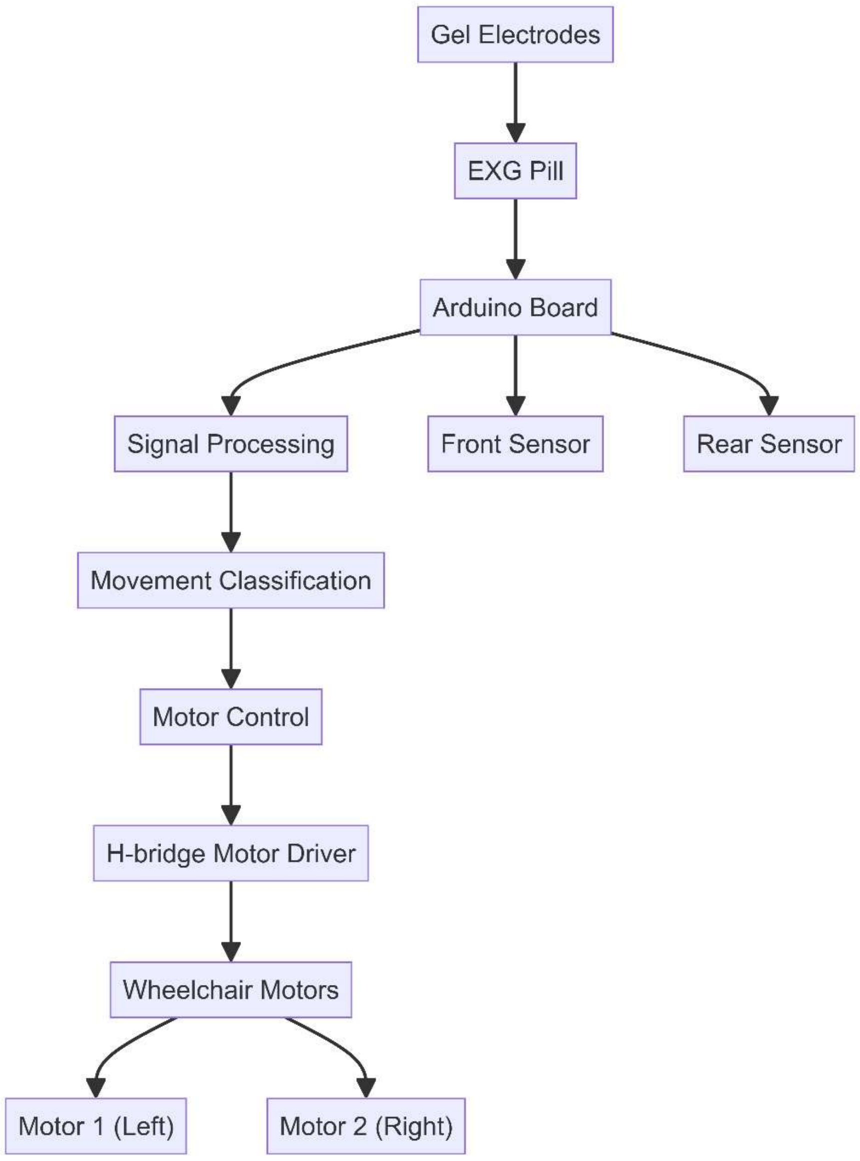
The core of the system is a main Arduino script that handled all aspects of the control process. The Arduino IDE is used for programming and its serial plotter for real-time signal visualization. EOG signals were acquired using gel electrodes connected to an EXG pill, which fed into the Arduino's analog inputs. Two channels are used: one for vertical eye movements (up/down) and another for horizontal movements (left/right).

The signal processing approach involved comparing raw EOG values against predefined thresholds to classify movements. A state machine is developed to manage different movement states, including stopping, moving forward/backward, and turning left/right. For motor control, PWM outputs were used to drive the wheelchair's motors through an H-bridge motor driver, with different PWM values for straight movement and turning to ensure smooth control.

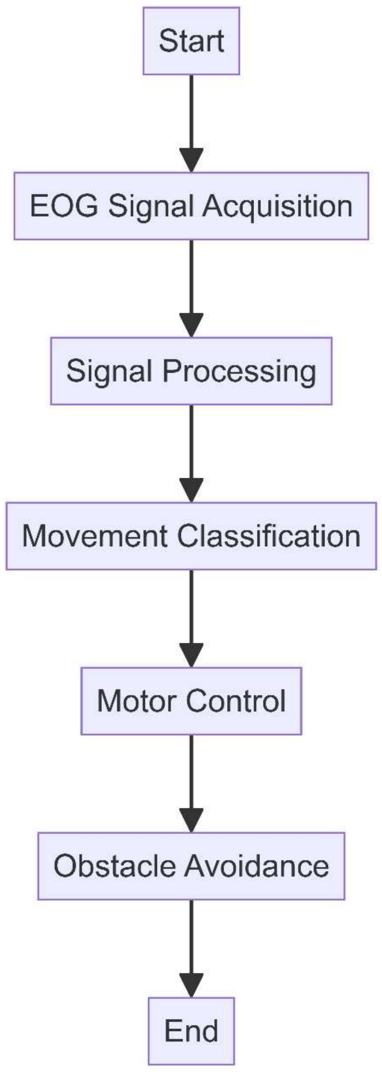
To enhance safety, obstacle avoidance is implemented using ultrasonic sensors at the front and back of the wheelchair. If an obstacle was detected within a set distance, the system automatically stops the wheelchair. A calibration process is included where users could test their eye movements and adjust thresholds, ensuring the system could be adapted to individual users' EOG signal characteristics.

To evaluate the system's performance, controlled tests were conducted assessing the accuracy of movement detection, system responsiveness, and the effectiveness of obstacle avoidance. Throughout the development and testing process, the Arduino IDE's serial plotter was used for real-time visualization of EOG signals and system states, allowing to fine-tune the algorithms and improve overall system performance.

Block Diagram:



Flow Diagram:



Code:

```
// EOG signal input pins
const int eogPinFB = A0;  // Front/Back control
const int eogPinLR = A1;  // Left/Right control

const unsigned long samplingInterval = 100; // Sampling interval in milliseconds (10 Hz)
const unsigned long debugInterval = 1000; // Debug print interval in milliseconds (1
second)
unsigned long lastSampleTime = 0;
unsigned long lastDebugTime = 0;

// Motor driver pins
const int motorPin1 = 10; // IN1 (Right wheels forward)
const int motorPin2 = 11; // IN2 (Right wheels backward)
const int motorPin3 = 5;  // IN3 (Left wheels forward)
const int motorPin4 = 6;  // IN4 (Left wheels backward)

// Ultrasonic sensor pins
const int trigPinFront = 2;
const int echoPinFront = 3;
const int trigPinBack = 7;
const int echoPinBack = 8;

// Speed (PWM range is 0-255)
const int speed = 75;
const int turnSpeed = 110;

// EOG thresholds for front/back movement
const int blinkThreshold = 700;
const int upThreshold = 500;
const int downThreshold = 200;
const int bufferLow = 300;
const int bufferHigh = 500;

// EOG thresholds for left/right movement
const int leftThreshold = 300;
const int rightThreshold = 590;
const int neutralLow = 300;
const int neutralHigh = 500;

// State variables
bool isMoving = false;
String currentMovement = "Stop";

void setup() {
  Serial.begin(9600);
  pinMode(eogPinFB, INPUT);
  pinMode(eogPinLR, INPUT);

  // Set motor pins as output
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);

  // Set ultrasonic sensor pins
  pinMode(trigPinFront, OUTPUT);
  pinMode(echoPinFront, INPUT);
  pinMode(trigPinBack, OUTPUT);
  pinMode(echoPinBack, INPUT);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB
  }
  Serial.println("RawFB,RawLR,Movement,State");
}

void loop() {
  unsigned long currentTime = millis();

  // Check if it's time to take a new sample
  if (currentTime - lastSampleTime >= samplingInterval) {
    lastSampleTime = currentTime;

    // Read the EOG signals
    int rawValueFB = analogRead(eogPinFB);
    int rawValueLR = analogRead(eogPinLR);

    // Determine movement based on EOG signals
    if (rawValueFB > blinkThreshold) {
      // Blink detected, stop the car
      isMoving = false;
      stopCar();
      delay(1000);
      currentMovement = "Blink";
    } else if (!isMoving) {
      // Car is not moving, check for start signal
      if (rawValueFB > upThreshold) {
        isMoving = true;
        currentMovement = "Forward";
        moveForward();
      } else if (rawValueFB < downThreshold) {
        isMoving = true;
        currentMovement = "Backward";
        moveBackward();
      } else if (rawValueLR < leftThreshold) {
        turnLeft();
        currentMovement = "Left";
      } else if (rawValueLR > rightThreshold) {
        turnRight();
        currentMovement = "Right";
      } else if (rawValueLR >= neutralLow && rawValueLR <= neutralHigh) {
        stopCar();
        currentMovement = "Stop";
      }
    } else {
      // Car is already moving, continue until blink or obstacle detected
    }

    // Print values for plotting
    Serial.print(rawValueFB);
    Serial.print(",");
    Serial.print(rawValueLR);
    Serial.print(",");
    Serial.print(currentMovement);
    Serial.print(",");
    Serial.println(isMoving ? "Moving" : "Stopped");

    // Debug print every second
    if (currentTime - lastDebugTime >= debugInterval) {
      lastDebugTime = currentTime;
      Serial.print("Debug - RawFB: ");
      Serial.print(rawValueFB);
      Serial.print(", RawLR: ");
      Serial.print(rawValueLR);
      Serial.print(", Movement: ");
      Serial.print(currentMovement);
      Serial.print(", State: ");
      Serial.println(isMoving ? "Moving" : "Stopped");
    }
  }


  // Check obstacles
  if (isMoving) {
    if (currentMovement == "Forward" && getDistance(trigPinFront, echoPinFront) < 30)
{
      stopCar();
      isMoving = false;
      currentMovement = "Stop";
    } else if (currentMovement == "Backward" && getDistance(trigPinBack,
echoPinBack) < 30) {
      stopCar();
      isMoving = false;
      currentMovement = "Stop";
    }
  }
}

void moveForward() {
  analogWrite(motorPin1, speed);
  analogWrite(motorPin2, 0);
  analogWrite(motorPin3, speed);
  analogWrite(motorPin4, 0);
}

void moveBackward() {
  analogWrite(motorPin1, 0);
  analogWrite(motorPin2, speed);
  analogWrite(motorPin3, 0);
  analogWrite(motorPin4, speed);
}

void turnLeft() {
  analogWrite(motorPin1, turnSpeed);
  analogWrite(motorPin2, 0);
  analogWrite(motorPin3, 0);
  analogWrite(motorPin4, turnSpeed);
  delay(410);
  stopCar();
}

void turnRight() {
  analogWrite(motorPin1, 0);
  analogWrite(motorPin2, turnSpeed);
  analogWrite(motorPin3, turnSpeed);
  analogWrite(motorPin4, 0);
  delay(410);
  stopCar();
}

void stopCar() {
  analogWrite(motorPin1, 0);
  analogWrite(motorPin2, 0);
  analogWrite(motorPin3, 0);
  analogWrite(motorPin4, 0);
 }

long getDistance(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  long distance = duration * 0.034 / 2;
  return distance;
}
```

RESULT

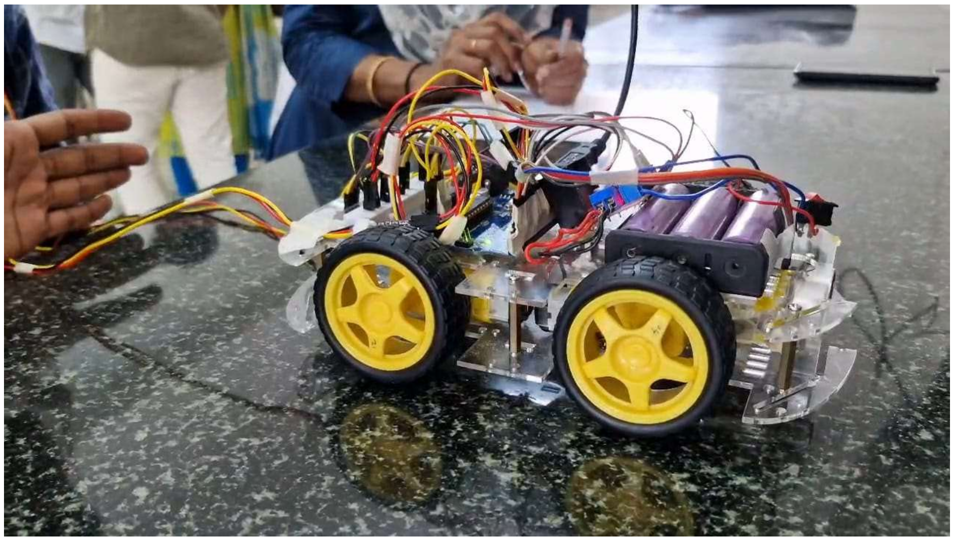


Fig : Controlled Vehicle

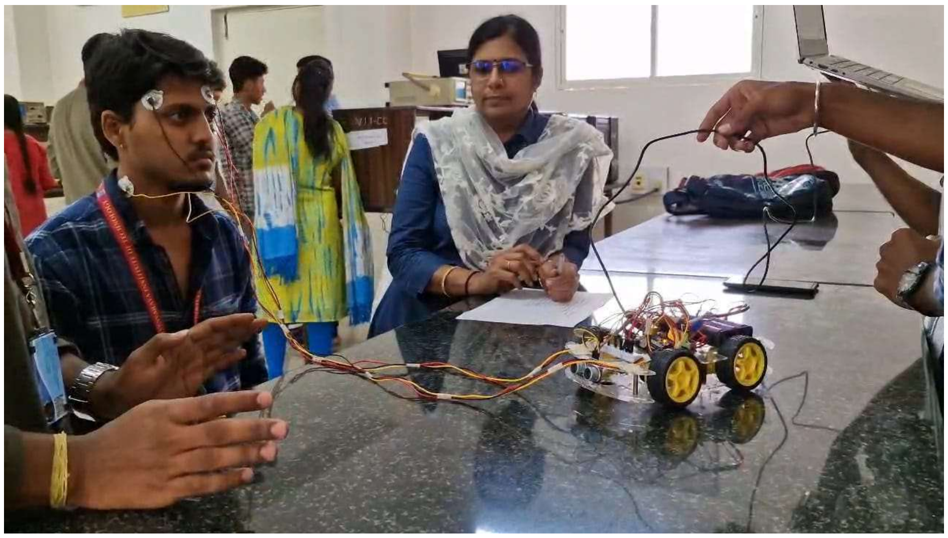


Fig : Wheelchair Navigation System Using EOG Signals