Real-time Over-steering Detection of Vehicle using Machine Learning and Embedded System Integration

Devika Vijapur, Nidhi Desai, Aishwarya Naik, Smita Ganur, Supriya Katwe

Contributing authors:
devikavijapur0718@gmail.com, nidhidesai752@gmail.com, ashnaik2003@gmail.com, smitaig2003@gmail.com, supriyak@kletech.ac.in
;

Abstract

Road accidents are one of the global safety concerns leading to loss of millions of lives every year. One of the factor leading to this is over-steering. Oversteering is phenomenon that occurs when the rear wheels of the vehicle lose grip which causes the vehicle to turn more than expected. The detection of over-steering in real-time is crucial for the improvement of vehicle safety to prevent accidents as well as for advanced driving assistance systems(ADAS). This paper presents a holistic approach to over-steering detection using a decision tree algorithm. The proposed system analyzes various vehicle dynamics parameters such as lateral acceleration, yaw rate and steering angle to identify the patterns that cause over-steering. The system incorporates collection of real-time data from Inertial Measurement Unit (IMU) sensors that enhances reliability of oversteering detection under various conditions. The model is trained from the data obtained, using decision tree algorithm and obtained accuracy of 96.08%. The hardware implementation is done by placing ESP-32 integrated with MPU 6050 and Arduino Nano 33 BLE sense accordingly in the vehicle. Based on thresholds of the parameters mentioned in the paper, oversteering is detected.

Keywords: Oversteering detection, decision trees, machine learning, advance driver assistance systems(ADAS).

1 Introduction

Oversteering detection is an important aspect of advanced driver assistance systems(ADAS), particularly to prevent major road accidents that are caused due to the condition of oversteering which leads to the loss of control on the vehicle. As mentioned, oversteering is a phenomenon that occurs when the rear wheels of the vehicle lose friction causing the vehicle to rotate more than intended that further leads to causing fatal road accidents. The United States, in the year 2015, reported 35,000 road casualties, which was a 7.7% increase from 2014. The mortality rate per 100 million miles traveled also increased to 1.12 from 1.08. In 2014, the contributors to deadly accidents were 12% including speeding and oversteering was responsible for 4.1% of the fatalities [1]. Figure (1) shows the graph of the percentage of accident types that occurred during the period from 2007 to 2016. One of the factor of these accidents is related to oversteering.

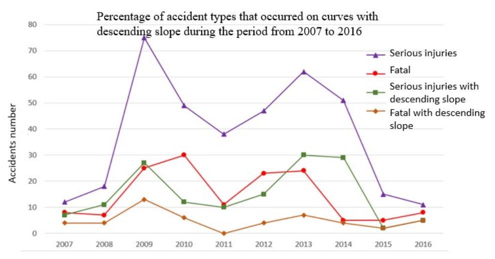


Fig. 1: Accident types that occurred during the period from 2007 to 2016 [2]

In addition to the eye-opening reports, oversteering is particularly dangerous because of its ability to cause uncontrollable skids in environments like wet or icy roads, rough or uneven pathways, sharp turns and even excessive tire wear can lead to the risk of oversteering. Besides these conditions, the inexperience of the driver or lack of awareness of how to control the vehicle during such conditions, may lead to fatalities. Figure 2, shows oversteering of a vehicle where the rear wheels lose traction, causing the vehicle to turn more than intended during turns, causing road accidents and fatalities. Detecting oversteering in real time allows the vehicle safety systems such as electronic stability control(ECU), to intervene and take necessary actions. Apart from everyday driving, oversteering detection is also crucial in motorsports where precision and control are most important during high speeds.

Oversteering detection plays a pivotal role in modern driver assistance systems such as autonomous driving where human intervention is minimal and considering only real-time data monitoring. The existing detection systems lack in identifying oversteering conditions in real time, which in turn is reducing its effectiveness in giving timely notifications. This gap is one of the major concerns in the detection systems. Therefore, it is essential to develop an oversteering detecting system that can precisely identify the oversteering conditions and provide timely notifications to reduce the risk of road accidents and fatalities.

Fig. 2: Oversteering [3]

2 Literature Survey

Several studies have evaluated the methods for detecting the conditions of oversteering. In [4], wheel speed, steering angle, and IMU data are given as input to the fuzzy, which is a high-level control system. The fuzzy output is given to the PID-AFC, which is a low-level control system to handle the speed of the left and right rear wheels. A recognition system has been developed based on the low-cost sensors such as gyroscope and accelerometer that are present in a smartphone [5]. In [6] oversteering is detected by analyzing slip angles (front vs. rear), lateral tire forces, yaw dynamics, tire saturation, handling diagrams, and deviations from the intended path. Sensors used include IMUs for yaw rate and acceleration, wheel speed sensors for slip detection, steering angle sensors for driver input, tire force sensors for direct force measurements, and GPS for tracking position and trajectory. Optical slip sensors are also utilized for precise slip angle estimation. The study [7] explores oversteering detection using factors like tire cornering stiffness, yaw rate, and slip angle. In [8] longitudinal velocity and the added weight bias are given as an input to the Bond Graph single-track model. This project developed a system to detect different vehicle trajectories, obstacle avoidance, and the detection of understeer and oversteer conditions using stability criteria. Using the Classification Learner app, they tried KNN classifiers, SVMs, quadratic discriminant analysis, and decision trees to detect oversteering. Another study [4] focuses on the use of Adaptive Neuro-Fuzzy Inference System (ANFIS), Fuzzy Logic Control, Subtractive Clustering, Co-simulation with MATLAB/Simulink and CarSim to develop a robust fuzzy inference system that can effectively detect Understeer (US) and Oversteer (OS) in vehicles, thereby enhancing vehicle stability without relying on model-based approaches. Multiple sensors were used to gather data on vehicle dynamics.

3 Methodology

Oversteering detection involves three main stages that include data preparation, model training, and model testing. Figure(3) represents the workflow of the oversteering detection process. The data preparation block includes data collection and the binary labeling. Data was collected in real time by navigating the vehicle (a car) for about 15 minutes. The data points were collected with the help of CoolTerm and arduino IDE. Details regarding the collection of data are further described in section 3.1. In the model training phase, firstly, the data was split into training and testing sets. The ratio of training to testing set was set as 70:30. The training dataset was used to train the Decision tree model. Lastly, in the model testing phase test dataset was used for model evaluation to know if the oversteering is happening correctly or not.

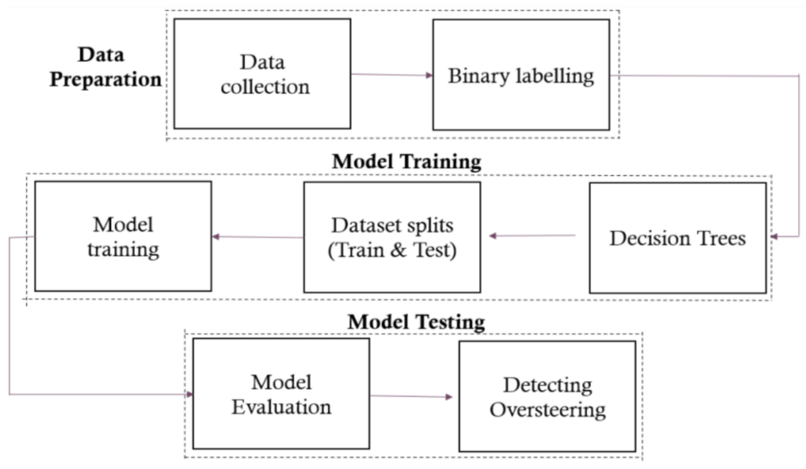


Fig. 3: Workflow diagram for oversteering detection

3.1 Data Preparation

The MPU-6050 sensor is an Inertial Measurement Unit(IMU) which consists of a 3-axis accelerometer and a 3-axis gyroscope, which helps to measure Lateral acceleration, Forward acceleration and Yaw Rate. This was interfaced with an ESP-32 microcontroller, which is designed to be compatible with the Arduino IDE framework, allowing us to program it using essential tools and libraries. The I2C protocol (Inter-Integrated Circuit) was used for the integration of MPU-6050 and ESP-32. This integrated setup was placed at the center of gravity of the vehicle (behind the handbrake) in real time as shown in Figure(4) to get the data values for acceleration in x-axis (forward acceleration), y-axis (lateral acceleration) and yaw rate (gyroscope's y-axis).Arduino Nano 33 BLE Sense, which is a 9-axis IMU (Accelerometer, Gyroscope and Magnetometer), was used to measure steering angle. The z-axis of the gyroscope was used to measure the steering angle. This was placed at the center of the steering wheel.The baud rate used for obtaining the data is 115200. Steering Angle, Yaw Rate, Lateral acceleration are critical to detect the circumstances of oversteering. Real-time data values were generated and collected by these sensors in a moving vehicle. 8,327 data values were generated by navigating the vehicle for about 15 minutes. The data values were collected via CoolTerm and arduino IDE, which is an adaptive terminal apt for serial communication. The data was transferred into Excel sheets, which helped in convenient labelling of data. The values were labelled "0", indicating a normal condition or "1," indicating oversteering condition, by comparing each value with a threshold that was predetermined. If the value at that instant exceeded its threshold value, it implied oversteering had taken place. Using this comprehensive dataset, we further proceeded with software and hardware implementation.

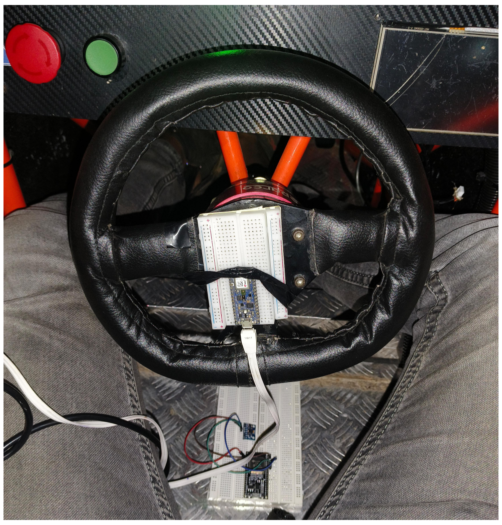


Fig. 4: Dataset Generation

3.2 Model training using Decision tree algorithm

The labelled dataset was uploaded and read into a Pandas DataFrame in Google Colab. It was split into a training set (70%) and testing set (30%), ensuring that the model learns from the major part of the data while keeping a portion for evaluation. The Decision Tree Classifier uses the decision rules recursively splits the dataset into subsets.
Gini Impurity:

$$Gini(D) = 1 - \sum_{i=1}^{C} p_i^2 \qquad [9] \tag{1}$$

In the training phase, the model learns necessary patterns from the 70% of the dataset and creates a tree-like structure. The model inspects the lateral acceleration, yaw-rate and steering angle data to determine the feature that best separates "Normal" and "Oversteering" conditions. Following the decision rule, the tree compares the sensor values to the set threshold, and if it exceeds, it is classified as oversteering. Decision Tree employs mathematical formulas for the best feature splitting at each node. Gini Impurity was considered to measure the probability of incorrect classification of a randomly chosen instance at a node. The key parameters used for oversteer detection include the yaw rate, lateral acceleration, and steering angle. Their interrelationships are described by the following equations.

3.3 Yaw Rate

The yaw rate r is defined as the rate of rotation about the vertical axis and can be computed using: From turn radius:

$$r = \frac{V}{L} \cdot \tan(\delta) \qquad [10] \tag{2}$$

3.4 Lateral Acceleration

The lateral acceleration a_y can be calculated as:

$$a_y = \frac{V^2}{R} \qquad [11] \tag{3}$$

3.5 Steering Angle

Steering angle can be calculated as:

From yaw rate,

$$\delta = \tan^{-1}\left(\frac{L \cdot r}{V}\right) \qquad [12] \tag{4}$$

where:

- r: Yaw rate (rad/s), V: Vehicle speed (m/s), L: Wheel base (m), a_y: Lateral acceleration (m/s^2), δ: Steering angle (radians)

3.6 Model Testing

The hardware implementation is done using two different controllers, namely Arduino Nano 33 BLE sense and ESP32 integrated with MPU 6050, which is a 6-axis motion tracking device. The integration of the sensors is done as shown in the Figure(5). The oversteering detection is done based on threshold values of three different parameters that include lateral acceleration, yaw rate and steering angle. Arduino Nano 33 BLE sense as mentioned above has 9-axis Inertial Measurement unit(IMU) which is placed at the center of steering wheel and based on the movement of steering it makes from center, the angle is measured by the in-built IMU sensor and provides us with data of steering angle from Gyroscope Z-axis. Using the software Arduino IDE, the Arduino Nano 33 BLE Sense is connected through UART, and data is transmitted through it. Arduino IDE (Integrated Development Environment) is an open-source software that is used for writing, compiling and uploading code onto the boards required. The program is written using the arduino IDE software to receive data. Data is received from the IMU sensor; if it is greater than threshold, then it is considered oversteering and therefore indicates to the driver that the vehicle is about to oversteer. If the value is below a threshold, then the vehicle is in normal condition and no indication is shown. Figure(6), shows the hardware implementation by combining ESP-32 with Arduino Nano 33 BLE sense and MPU-6050.

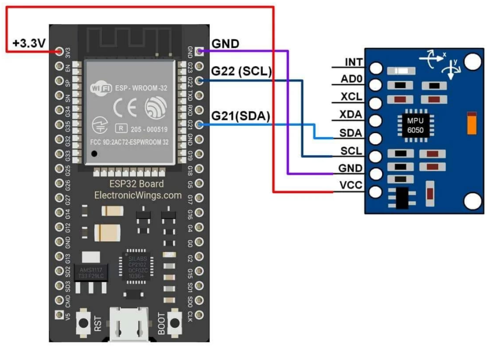


Fig. 5: Sensors Integration

[13]

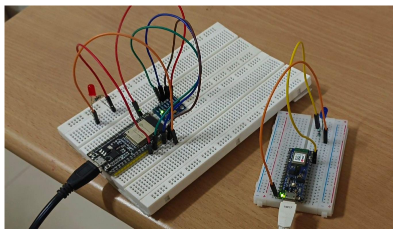


Fig. 6: Hardware Implementation using ESP-32 and Arduino Nano 33 BLE sense and MPU 6050

4 Procedural Approach of Oversteering Detection

The procedural approach for oversteering detection is as follows:

Algorithm 1 Oversteering Detection Algorithm

```
1: Input: Thresholds for Lateral acceleration, Yaw Rate, and Steering Angle
2: Read the Excel file and convert it into a pandas DataFrame (df)
3: Define a function to check thresholds for:
     • Lateral acceleration
     • Yaw Rate
     • Steering Angle
4: Label rows as:
     • oversteering (1) if any value exceeds the threshold
     • normal (0) otherwise
5: Define the feature set (x) comprising:
     • Lateral acceleration
     • Yaw Rate
     • Steering Angle
6: Set the target variable (y) as the Label
7: Split the dataset into training and testing subsets
8: Train a machine learning model using the training data
9: Evaluate the model by predicting labels for the test dataset and compute accuracy
10: Implement a function for interactive user input to predict whether the state is
    oversteering (1) or normal (0)=0
```

5 Results and Conclusion

The proposed system demonstrates outstanding competence in recognizing the challenge of real-time oversteer detection. The integration of ESP-32 , MPU-6050 and Arduino Nano 33 BLE sense aid in the accurate collection of data from different parameters. A real-time dataset was produced with the parameters important for detection. It was then labelled 0 or 1, indicating normal or oversteer condition, respectively. The ML-model trained with a real-time dataset using the Decision Tree algorithm Figure9 achieves 96.08% accuracy. The dataset was split into 70:30 ratio, 70% for data training and 30% for data testing.Figure8 represents the accuracy of the training and testing set. The model was capable of taking inputs from the user of parameters mentioned providing output as normal or over-steering. The model help to realize the parameter importance using the classification report Figure10 and the confusion matrix Figure11. The integrated sensors were then used for testing in a real-time vehicle by placing them at particular locations and comparing the data against their threshold values.When exceeded, the driver will be alerted using indicators like blinking of LEDs.

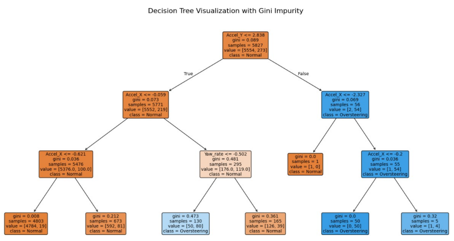


Fig. 7: Training and Testing Error

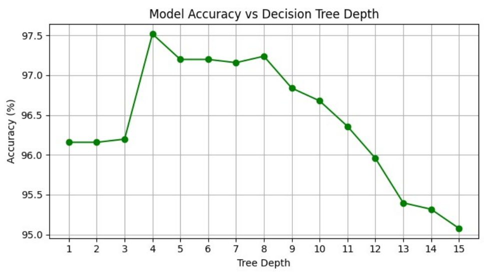
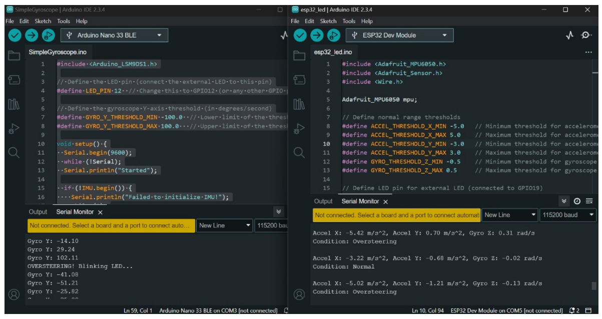


Fig. 8: Training and Testing Accuracy


SimpleGyroscope | Arduino IDE 2.3.4
File
Edit
Sketch
Tools
Help
Arduino Nano 33 BLE
SimpleGyroscope.ino
Output
Serial Monitor
Not connected. Select a board and a port to connect auto...
New Line
115200 bau
Ln 59, Col 1
Arduino Nano 33 BLE on COM3 [not connected]
esp32_led | Arduino IDE 2.3.4
ESP32 Dev Module
esp32_led.ino
Not connected. Select a board and a port to connect automati
115200 baud
Ln 10, Col 94
ESP32 Dev Module on COM5 [not connected]

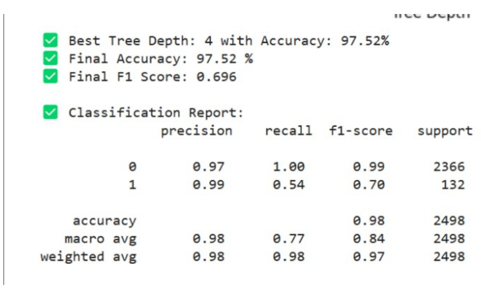
```
#include <Arduino_LSM9DS1.h>

// Define the LED pin (connect the external LED to this pin)
#define LED_PIN 12  // Change this to GPIO12 (or any other GPIO
 
// Define the gyroscope Y-axis threshold (in degrees/second)
#define GYRO_Y_THRESHOLD_MIN -100.0  // Lower limit of the thres
#define GYRO_Y_THRESHOLD_MAX 100.0   // Upper limit of the thres

void setup() {
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Started");

  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
```

```
Gyro Y: -14.10
Gyro Y: 29.24
Gyro Y: 102.11
OVERSTEERING! Blinking LED...
Gyro Y: -41.08
Gyro Y: -51.21
Gyro Y: -25.82
```

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;

// Define normal range thresholds
#define ACCEL_THRESHOLD_X_MIN -5.0   // Minimum threshold for accelerome
#define ACCEL_THRESHOLD_X_MAX 5.0    // Maximum threshold for accelerome
#define ACCEL_THRESHOLD_Y_MIN -3.0   // Minimum threshold for accelerome
#define ACCEL_THRESHOLD_Y_MAX 3.0    // Maximum threshold for accelerome
#define GYRO_THRESHOLD_Z_MIN -0.5    // Minimum threshold for gyroscope
#define GYRO_THRESHOLD_Z_MAX 0.5     // Maximum threshold for gyroscope

// Define LED pin for external LED (connected to GPIO19)
```

```
Accel X: -5.42 m/s^2, Accel Y: 0.70 m/s^2, Gyro Z: 0.31 rad/s
Condition: Oversteering

Accel X: -3.22 m/s^2, Accel Y: -0.68 m/s^2, Gyro Z: -0.02 rad/s
Condition: Normal

Accel X: -5.02 m/s^2, Accel Y: -1.21 m/s^2, Gyro Z: -0.13 rad/s
Condition: Oversteering
```

Fig. 9: Decision tree classification



```
✅ Best Tree Depth: 4 with Accuracy: 97.52%
✅ Final Accuracy: 97.52 %
✅ Final F1 Score: 0.696

✅ Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      2366
           1       0.99      0.54      0.70       132

    accuracy                           0.98      2498
   macro avg       0.98      0.77      0.84      2498
weighted avg       0.98      0.98      0.97      2498
```

Fig. 10: Classification Report

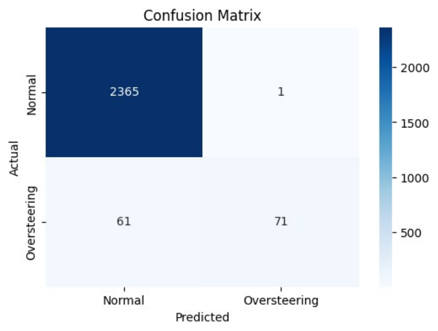


Fig. 11: Confusion Matrix

6 Future scope

There is exceptional potential for fusion and collection of data from ESP-32 and Arduino Nano 33 BLE sense concurrently. Recording of the parameters like forward and lateral acceleration, yaw rate and steering angle at once from fused sensors and being read at the terminal is to be explored. Transferal of the trained model onto the controllers is a course of action to be executed, where the data being collected from the sensors can directly be read by the model and help it to put out its decision.Alternate methods to alert the driver upon over-steering is beep sound.Other possibility involve using the data sensors that can be sent to electronic control unit(ECU) to control the over-steering and avoid further accidents.

References

[1] Administration, N.H.T.S., *et al.*: Early estimate of motor vehicle traffic fatalities in 2012. Publication DOT HS **811**, 741 (2013)

[2] Macedo, M., Maia, M., Kohlman Rabbani, E., Marinho, M.: Spatial analysis of the variables involved in the frequency and severity of traffic accidents on rural highways in pernambuco. American Scientific Research Journal for Engineering, Technology, and Sciences **78**, 226–246 (2021)

[3] Autos, V.: QA: What's the difference between oversteering and understeering? https://www.facebook.com/veteranautos.av/photos/qa-whats-the-difference-between-oversteering-and-understeeringoversteering-and-u/701059592247510/. Accessed: 2025-04-29 (2023). https://www.facebook.com/veteranautos.av/photos/qa-whats-the-difference-between-oversteering-and-understeeringoversteering-and-u/701059592247510/

[4] Hirche, B., Ayalew, B.: A fuzzy inference system for understeer/oversteer detection towards model-free stability control. SAE International Journal of Passenger

Cars-Mechanical Systems **9**(2016-01-1630), 831–838 (2016)

[5] Liu, X., Mei, H., Lu, H., Kuang, H., Ma, X.: A vehicle steering recognition system based on low-cost smartphone sensors. Sensors **17**(3), 633 (2017)

[6] Timings, J.P., Cole, D.J.: Efficient minimum manoeuvre time optimisation of an oversteering vehicle at constant forward speed. In: Proceedings of the 2011 American Control Conference, pp. 5267–5272 (2011). IEEE

[7] Habib, N., Aziz, N., Faris, D., Nacer, M.: Electrical kart dynamics analysis using bond graph single-track model. In: CCCA12, pp. 1–8 (2012). IEEE

[8] Freudling, T.: Detecting Oversteering in BMW Automobiles with Machine Learning. https://www.mathworks.com/company/technical-articles/detecting-oversteering-in-bmw-automobiles-with-machine-learning.html. Accessed: 2025-04-29 (2018)

[9] LearnDataSci: Gini Impurity - LearnDataSci. Accessed: 2025-05-05 (n.d.). https://www.learndatasci.com/glossary/gini-impurity/

[10] Gillespie, T.D.: Fundamentals of Vehicle Dynamics. Society of Automotive Engineers, Warrendale, PA (1992)

[11] Xu, J., Yang, K., Shao, Y., Lu, G.: An experimental study on lateral acceleration of cars in different environments in sichuan, southwest china. Discrete Dynamics in Nature and Society (2015) https://doi.org/10.1155/2015/494130

[12] Gillespie, T.D.: Fundamentals of Vehicle Dynamics. Society of Automotive Engineers, Warrendale, PA (1992)

[13] Electroniclinic: ESP32 with MPU6050 wiring diagram. https://www.electroniclinic.com. Accessed: 2025-04-29