PROJECT REPORT

Measurement and Instrumentation

Final Project · Part 2

Speed Estimation Using the MPU6050 Accelerometer and Gyroscope

STUDENT NAME	**Asem Ameen Mahyoub Qasem Qaid**
NIM	**24524038**
STUDENT NAME	**Fawaz Yousif Mohamed Abbaker**
NIM	**24524040**

Measurement and Instrumentation
Undergraduate Program of Electrical Engineering
Faculty of Industrial Technology
Islamic University of Indonesia

2026

ABSTRACT

This report describes the design, construction and evaluation of a motion measurement system based on the MPU6050 accelerometer and gyroscope, combined with an MQ135 gas sensor on a single Arduino Nano. Readings are filtered on the sensor and in software, the system calibrates itself automatically at start-up, and the data are displayed in real time on a custom web dashboard connected through the Web Serial API.

Two independent methods of estimating speed from the MPU6050 were implemented and benchmarked against a reference speed obtained from a measured distance and a stopwatch time, with 15 trials for each method. Tilt-compensated integration of acceleration, tested on a campus cleaning vehicle, gave a root-mean-square error (RMSE) of 1.106 m/s and a mean absolute percentage error of 36.0%. Step counting, tested while walking, gave an RMSE of 0.161 m/s and a mean error of 9.8%. The roughly seven-fold difference is explained by integration drift, which grows with time, whereas discrete step detection does not accumulate error.

Keywords: MPU6050, accelerometer, gyroscope, complementary filter, speed estimation, step counting, RMSE, Web Serial API

CONTENTS

1 Electronic System Design

2 Results of the Constructed Device

3 Performance Analysis

4 Conclusion

References

1 Electronic System Design

1.1 System Overview

The monitoring system consists of three stages (Figure 1): the sensor stage (MPU6050 accelerometer/gyroscope and MQ135 gas sensor), the interface hardware stage (Arduino Nano) and the user interface stage (a PC or laptop running a custom HTML dashboard). Both sensors are connected to a single Arduino Nano, so only one USB serial connection is required.

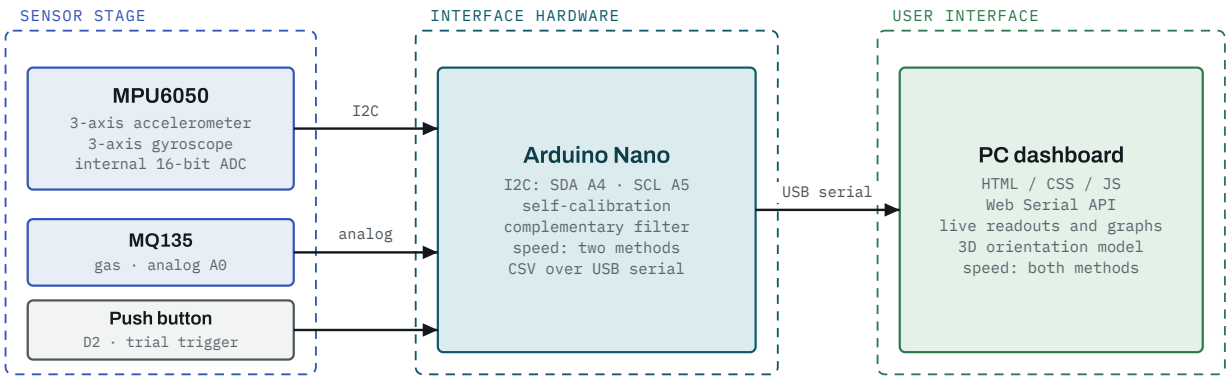


Figure 1. System block diagram: sensors, Arduino Nano and PC dashboard.

1.2 Sensor Interfacing

The MPU6050 communicates with the Arduino over the I2C bus (SDA to A4, SCL to A5). Because the MPU6050 contains its own 16-bit analog-to-digital converter and outputs fully digital data, no external amplifier is required. The MQ135 outputs an analog voltage proportional to gas concentration, connected directly to analog input A0; this voltage already lies within the 0–5 V input range of the Arduino ADC, so no amplification is needed here either. A push button on D2 (internal pull-up) is used to trigger benchmark trials.

Table 1. Pin connections

Component	Pin on Arduino Nano
MPU6050 SDA	A4
MPU6050 SCL	A5
MPU6050 VCC / GND	5V / GND
MQ135 analog output	A0
Push button	D2 → GND (internal pull-up)

1.3 Filter Design

Two layers of filtering reduce noise in the motion data (Figure 2). The MPU6050's internal digital low-pass filter (DLPF) is configured to a bandwidth of 21 Hz (`MPU6050_BAND_21_HZ`), which attenuates high-frequency vibration before the data are read by the Arduino. A first-order complementary filter then combines the short-term accuracy of the gyroscope with the long-term stability of the accelerometer to estimate the roll and pitch angles:

$$\theta = \alpha \cdot (\theta_{prev} + \omega \cdot \Delta t) + (1 - \alpha) \cdot \theta_{acc}, \quad \alpha = 0.98 \tag{1}$$

The filtered angle is also used to remove the component of gravity that leaks onto the acceleration axes whenever the sensor tilts, a necessary correction for Method 1. A fixed dead zone of 0.20 m/s² suppresses residual noise before integration.

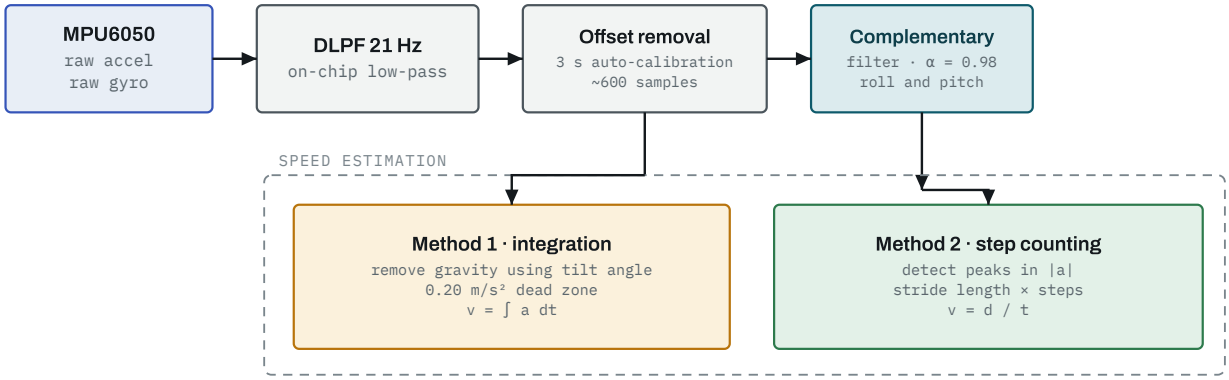


Figure 2. Signal-processing chain from raw sensor data to the two speed estimates.

1.4 Calibration Procedure

Instead of fixed, manually measured offsets, the system calibrates itself automatically on every power-up or reset (Figure 3). After initialisation, the Arduino prints a 3-second countdown and then samples the sensor for 3 seconds (about 600 samples) while it is assumed to be flat and stationary. The average of these samples becomes the offset of each of the six accelerometer and gyroscope axes. The Z-axis accelerometer offset additionally subtracts standard gravity (9.80665 m/s²) rather than zero, because a level, stationary sensor should read the force of gravity on that axis.



Figure 3. Automatic self-calibration sequence at start-up.

1.5 Monitoring Interface Software Design

As required by the assignment, an interface outside the standard Arduino Serial Monitor was built to display the sensor data. The interface is a web-based dashboard (HTML, CSS and JavaScript) that connects to the Arduino through the Web Serial API, parses the comma-separated values sent by the Arduino and renders them as live numeric readouts, real-time graphs and an animated 3D model that rotates with the sensor's roll and pitch.

The dashboard was developed with the assistance of generative AI (Claude), as permitted by the assignment guideline. The prompt used was:

> "make me a web dashboard that consists of two main sensors, the gas sensor given this code [gas sensor code] and the MPU given these codes, for the first method [first method code] and the second method [second method code]. Separate the two sensors in two different tabs. For the MPU you can show a 3D view rotating and changing its orientation according to the sensor change, and the values of the axes as well as the speed readings using both methods."

The interface shows 3-axis acceleration, 3-axis angular rate, temperature, speed (m/s), distance (m), step count and elapsed walking time. The two speed-estimation methods are displayed side by side: Method 1 (tilt-compensated integration of acceleration) and Method 2 (step count × average stride length ÷ time).

2 Results of the Constructed Device

2.1 Hardware Implementation

The MPU6050, the MQ135 and the push button were wired to a single Arduino Nano on a breadboard (Figure 4).

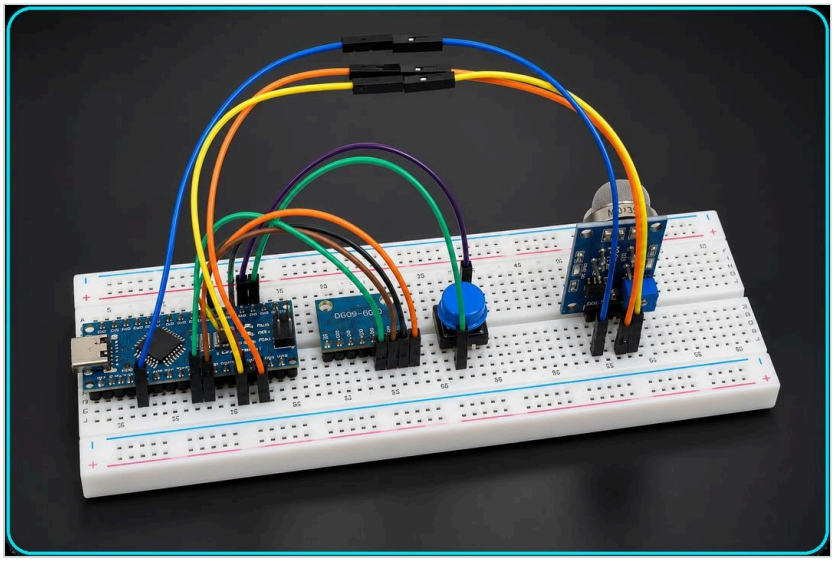


Figure 4. Assembled measurement system (MPU6050, MQ135 and Arduino Nano).

2.2 PC Dashboard Interface

The dashboard was implemented and tested with live sensor data. Figure 5 shows the Motion and Orientation tab, with the 3D orientation model, roll and pitch angles, raw acceleration and angular-rate values, and both speed estimates. A second tab hosts the MQ135 gas channel, showing that the interface can serve several sensor channels; for this course only the MPU6050 channel is required and analysed.

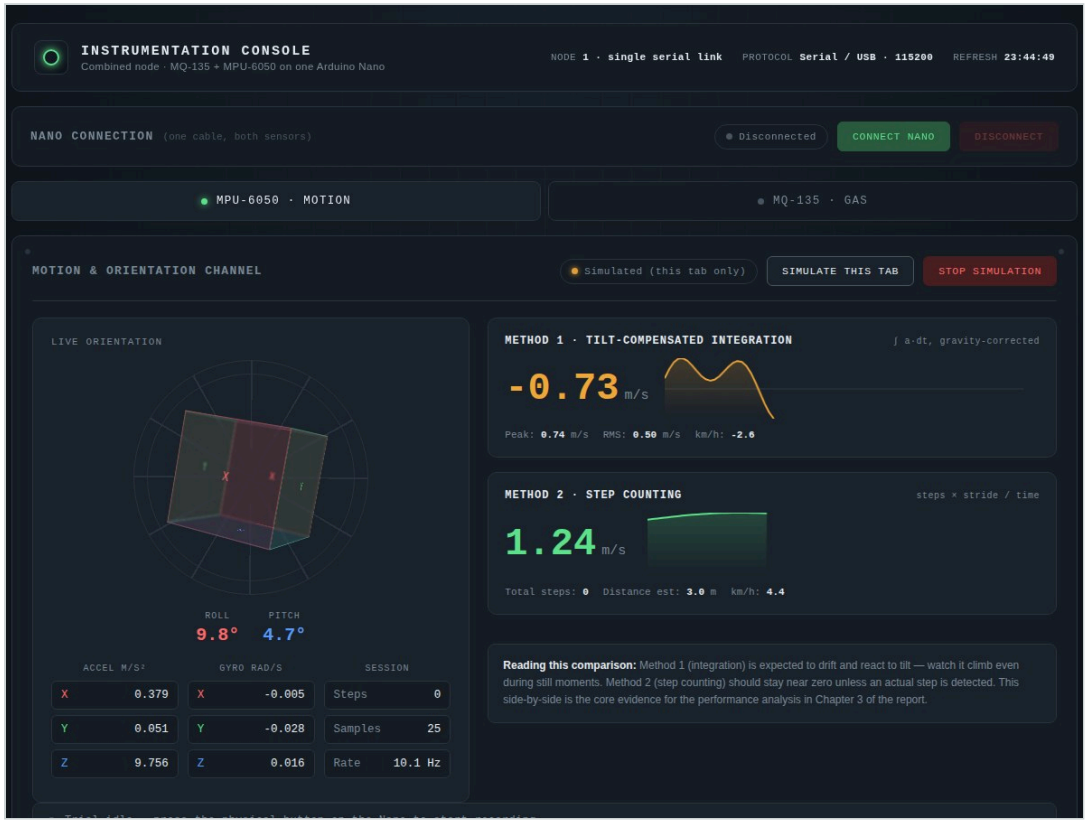


Figure 5. Dashboard, Motion and Orientation tab.

2.3 Experimental Set-up

Two experimental set-ups were used (Figure 6): the MPU6050 mounted on a campus cleaning vehicle for the vehicle-speed experiment, and the sensor held by hand during the walking-speed experiment along a corridor.



(a) Vehicle experiment (b) Walking experiment

Figure 6. Experimental set-ups.

2.4 Device Performance Overview

After calibration, the MPU6050 correctly reports acceleration and angular rate on all three axes. The dashboard updates in real time at about 10 Hz with no noticeable lag over the USB serial connection, and the 3D model follows the physical orientation of the sensor during roll and pitch. Both speed-estimation methods run simultaneously and are displayed side by side for comparison.

3 Performance Analysis

3.1 Benchmark Methodology

The two speed-estimation methods were evaluated against a manually measured reference:

- **Method 1, tilt-compensated integration:** linear acceleration, with gravity removed using the complementary-filter angle, is integrated numerically over time to obtain velocity.
- **Method 2, step counting:** footstep impacts are detected as spikes in the total acceleration magnitude, and speed is computed as (steps × stride length) ÷ elapsed time.

For each trial a known distance was measured, the elapsed time was recorded with a stopwatch, and the reference (calculated) speed was obtained as distance ÷ time. Accuracy was quantified with the root-mean-square error between the measured and calculated speeds:

$$RMSE = \sqrt{\left(\Sigma \left(v_{measured} - v_{calculated} \right)^2 / n \right)} \tag{2}$$

Fifteen trials were collected for each experiment, satisfying the minimum number of data points required by the worksheet.

3.2 Experiment 1: Vehicle Speed (Method 1)

The MPU6050 was mounted on a campus cleaning vehicle, which travelled known distances while the elapsed time was measured with a stopwatch (Table 2, Figure 7).

Table 2. Vehicle experiment results (Method 1)

No.	Distance (m)	Time (s)	Measured (m/s)	Calculated (m/s)	Error
1	5	2.98	1.02	1.68	39.3%
2	5	2.98	1.50	1.68	10.7%
3	5	2.75	1.74	1.82	4.4%
4	5	2.63	1.73	1.90	8.9%
5	10	4.72	1.45	2.12	31.6%
6	10	3.91	1.27	2.56	50.4%
7	10	4.21	1.06	2.38	55.5%
8	10	3.68	2.06	2.72	24.3%
9	15	5.55	0.71	2.70	73.7%
10	15	4.70	1.58	3.19	50.5%
11	15	5.18	1.13	2.90	61.0%
12	15	4.78	2.73	3.14	13.1%
13	20	6.75	2.24	2.96	24.3%
14	20	7.22	1.65	2.77	40.4%
15	20	7.48	1.28	2.67	52.1%

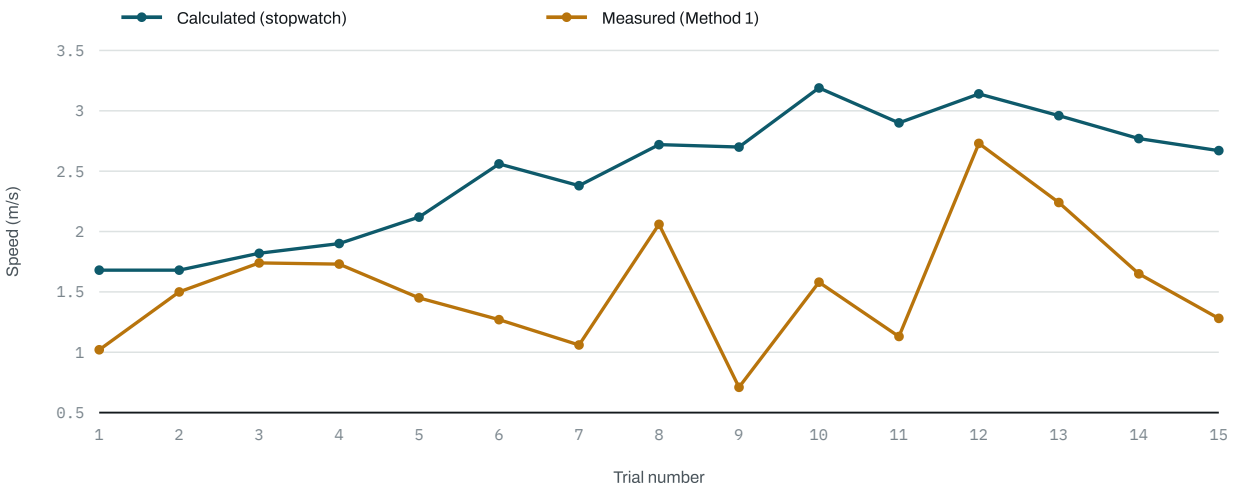


Figure 7. Vehicle experiment: measured (Method 1) and calculated speed for each trial.

For Experiment 1, RMSE = 1.106 m/s and the mean absolute percentage error is 36.0%. The sensor consistently under-reported speed relative to the stopwatch reference, and the error tended to grow for longer trials, consistent with integration drift, in which accumulated bias grows with elapsed time rather than with the distance travelled.

3.3 Experiment 2: Walking Speed (Method 2)

The MPU6050 was carried by a person walking a known distance. Method 2 detected the number of steps; distance was estimated as steps × stride length and speed as distance ÷ elapsed time (Table 3, Figure 8).

Table 3. Walking experiment results (Method 2)

No.	Distance (m)	Steps	Time (s)	Measured (m/s)	Calculated (m/s)	Error
1	5	8	5.47	1.02	0.91	12.1%
2	5	7	4.25	1.15	1.18	2.5%
3	5	8	4.49	1.25	1.11	12.6%
4	5	8	4.55	1.23	1.10	11.8%
5	10	12	7.48	1.12	1.34	16.4%
6	10	14	6.71	1.46	1.49	2.0%
7	10	14	9.08	1.08	1.10	1.8%
8	10	15	8.58	1.22	1.17	4.3%
9	15	19	11.22	1.18	1.34	11.9%
10	15	19	9.45	1.41	1.59	11.3%
11	15	20	9.50	1.47	1.58	7.0%
12	15	20	11.02	1.27	1.36	6.6%
13	20	25	12.82	1.37	1.56	12.2%
14	20	24	11.51	1.46	1.74	16.1%
15	20	24	11.69	1.40	1.71	18.1%



Figure 8. Walking experiment: measured (Method 2) and calculated speed for each trial.

For Experiment 2, RMSE = 0.161 m/s and the mean absolute percentage error is 9.8%, substantially more accurate than Method 1. The remaining error is attributable mainly to natural variation in stride length between trials rather than to sensor drift.

3.4 Comparative Discussion

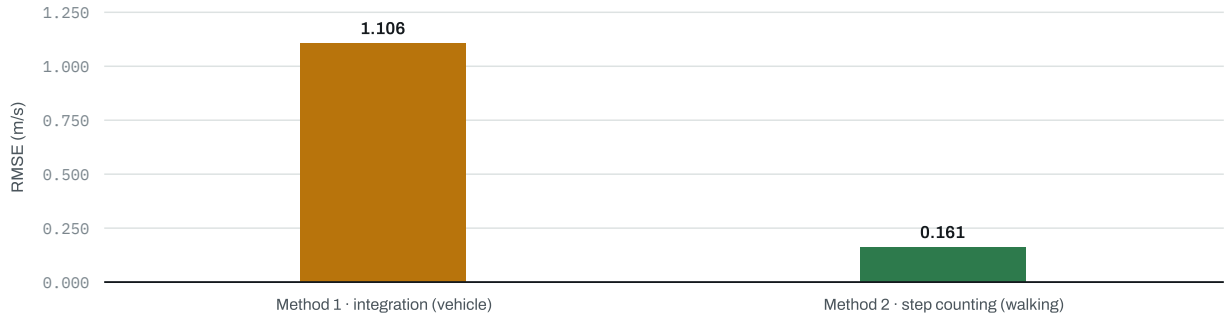


Figure 9. RMS error of the two methods.



Figure 10. Absolute error of each trial, Method 1 and Method 2.

The roughly seven-fold difference in RMSE between the two methods (1.106 m/s against 0.161 m/s) has a physical rather than incidental explanation. Method 1 obtains velocity by integrating acceleration, so any small residual bias ε left after calibration accumulates as a velocity error of approximately ε·t, which grows without bound as the trial gets longer. This is visible in the data: the vehicle trials with greater distance and time (trials 5, 6, 9, 10 and 11) show some of the largest errors. Vehicle motion also involves vibration and small unavoidable tilts, which project gravity onto the acceleration axes where it is misread as linear motion, further increasing the integration error.

Method 2 does not integrate a continuous signal. It counts discrete, large-amplitude footstep events, so its accuracy is bounded by how consistent the stride length is between trials rather than by trial duration. This structural difference, event detection against continuous integration, explains why Method 2 remained accurate and stable across all 15 trials while the error of Method 1 grew increasingly unpredictable in longer trials.

These results indicate that raw accelerometer integration is not reliable for general-purpose vehicle speed measurement without additional correction, such as fusion with GPS or a wheel encoder, whereas step-based estimation is well suited to its intended use of measuring pedestrian walking speed.

4 Conclusion

A combined accelerometer, gyroscope and gas monitoring system was designed, built and evaluated using the MPU6050 and MQ135 sensors on a single Arduino Nano, with a custom web dashboard as the PC interface. The system meets the core requirements of the project: it displays calibrated, filtered accelerometer, gyroscope and gas readings in real time outside the standard Arduino Serial Monitor, with automatic self-calibration at start-up to minimise manual set-up error.

Two independent methods of estimating speed were implemented and benchmarked against manually measured ground truth over 15 trials each. Step counting achieved a much lower RMS error (0.161 m/s) than tilt-compensated integration (1.106 m/s), showing that the choice of estimation method matters as much as sensor calibration itself, as a direct consequence of integration drift compared with discrete event detection.

Overall, the MPU6050 proved effective for motion sensing and pedestrian speed estimation, although integration alone is not accurate enough for general vehicle speed measurement. Recommended improvements include full sensor fusion, a magnetometer for an absolute heading reference, and adjusting the stride length dynamically according to walking cadence to reduce the residual error of Method 2.

References

1. Universitas Islam Indonesia, *Measurement and Instrumentation: Final Project Guidelines, Part 2*, even semester 2025/2026.
2. Generative AI prompt reference: "How to make the MPU6050 a speed sensor?"
3. TDK InvenSense, *MPU-6050 Datasheet*. Available: alldatasheet.com
4. Mouser Electronics, *MPU-6050 6-DOF Accelerometer and Gyro Datasheet*. Available: mouser.com