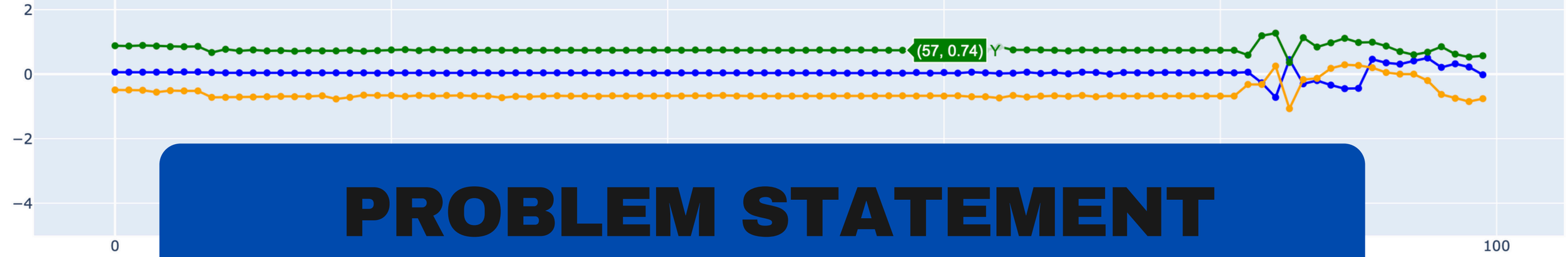
DEAKIN UNIVERSITY

REAL-TIME CRASH DETECTION WITH ARDUINO AND PYTHON DASH

ALERTING WITH ARDUINO LSM6DS3

AYUSH JAIN S225476349



PROBLEM STATEMENT

- Road crashes require fast detection for emergency response.
- Commercial crash detection exists (Apple, Google), but it is closed-source and proprietary.
- No open, low-cost, reproducible solution exists for students or researchers.
- Goal: Design a transparent crash detection pipeline from sensor → dashboard → alert.

EXISTING METHODS

- **Apple Crash Detection:** Uses accelerometer, gyroscope, GPS, barometer, microphone → reduces false positives.
- **Google Pixel Safety:** Accelerometer + gyroscope + location.
- **Academic work (WreckWatch):** Phone accelerometers → many false positives.
- **Gap:** No open educational example with correct units, live dashboard, and automated alerts

METHODOLOGY

REAL-TIME CRASH DETECTION WITH ARDUINO

01

1. Sensor (Arduino LSM6DS3)

- Arduino Nano 33 IoT with built-in LSM6DS3 IMU
- Collects 3-axis accelerometer data at ~100 Hz
- Calculates total g-force and detects sudden spikes

02

Serial Communication

- Data streamed over USB serial to laptop
- Format: ax, ay, az, crash_flag
- Low latency (<50 ms transfer)

.

03

Python + Dashboard

- Python script reads serial data
- Dash app buffers 100 samples for smooth graphs
- Real-time plots: XYZ graph + crash gauge

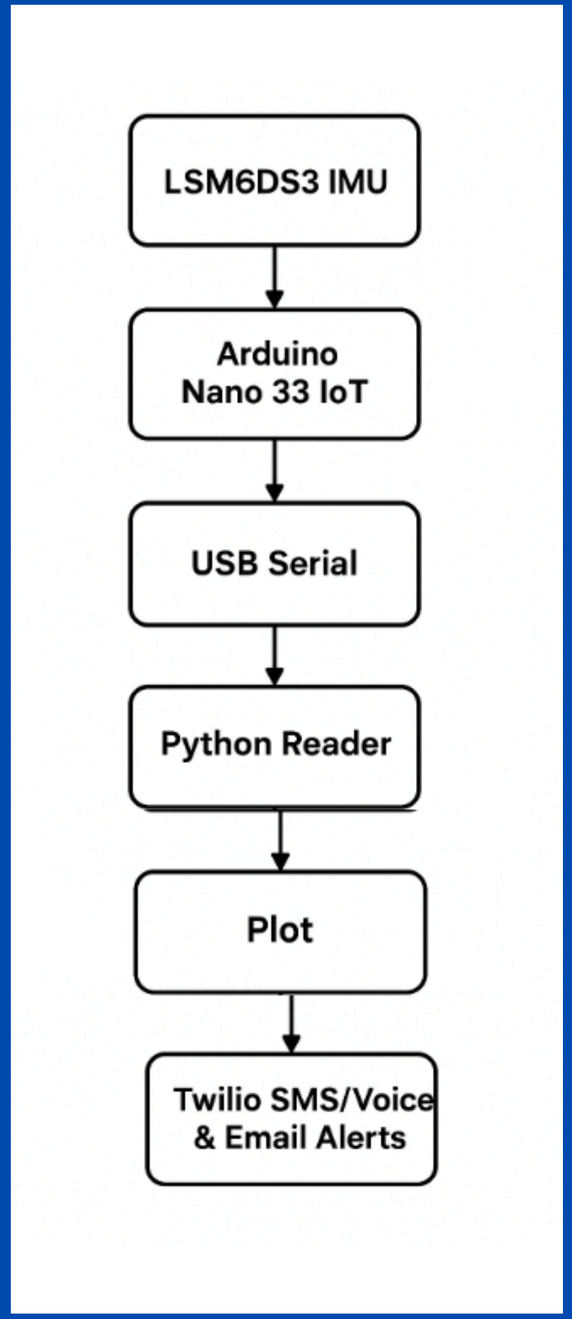
04

Alerts

- •If crash detected → SMS & email sent
- Twilio API for SMS, SMTP for email
- One-time notification per event (no spam)

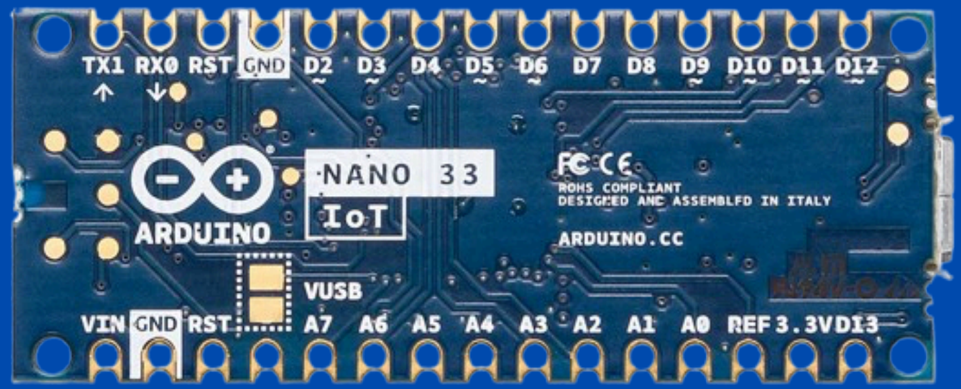
DATA FLOW

- **Sensing:** Arduino Nano 33 IoT (LSM6DS3 IMU) measures acceleration and calculates g-force in real time.
- **Streaming:** Data (ax, ay, az, crash flag) is transmitted via USB serial to the host computer.
- **Processing:** Python script reads the stream, filters noise, and feeds values into a rolling buffer.
- **Visualising:** Plotly Dash dashboard updates live with graphs and a crash indicator.
- **Alerting:** If a crash is detected, the system sends instant SMS and email notifications.



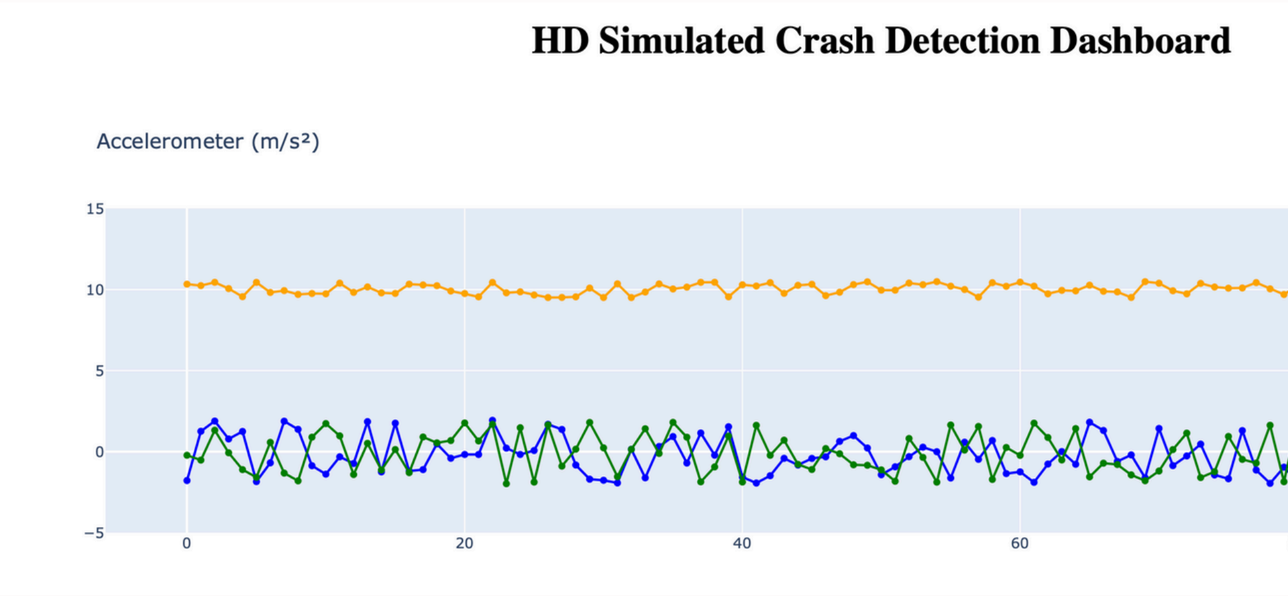
HARDWARE SETUP

- **Microcontroller:** Arduino Nano 33 IoT (built-in LSM6DS3).
- **Connection:** USB to laptop.
- **Test setup:** Safe drops and taps on foam pad.
- Low cost, portable, reproducible by students.



SOFTWARE SETUP

- Arduino sketch:
- Reads accelerometer at ~100 Hz.
- Computes g-force and crash flag.
- Streams data via serial.
- Python Dash app:
- Buffers data in real time.
- Displays accelerometer graphs & crash gauge.
- Triggers Twilio (SMS) and SMTP (email) alerts.



CRASH DETECTION LOGIC

- On Arduino (edge):
- Compute g_total = √(ax²+ay²+az²).
- Calculate excess-g (orientation independent).
- Apply jerk and latching logic for stability.
- On Python (host):
- Rolling buffer of 100 samples.
- Real-time graph & crash gauge.
- One-time SMS & email notification per crash.

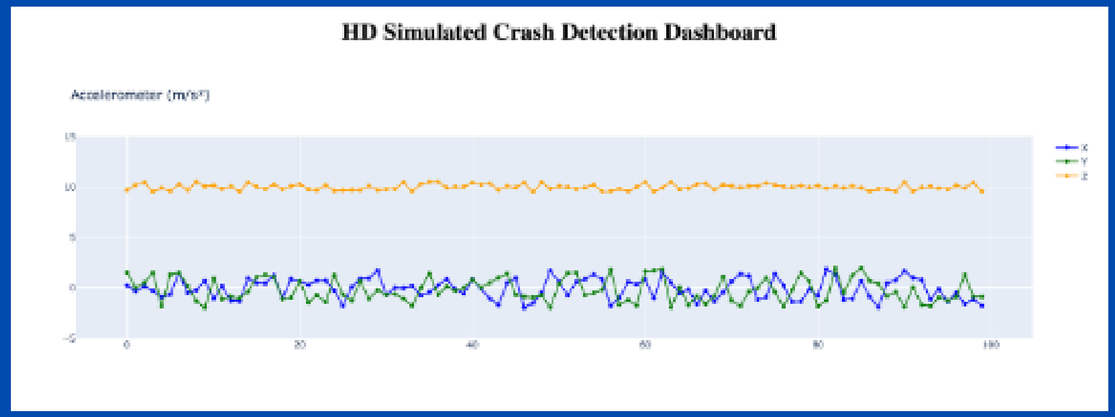
crash.ino.ino README.adoc secrets.h sketch.json thingProperties.h

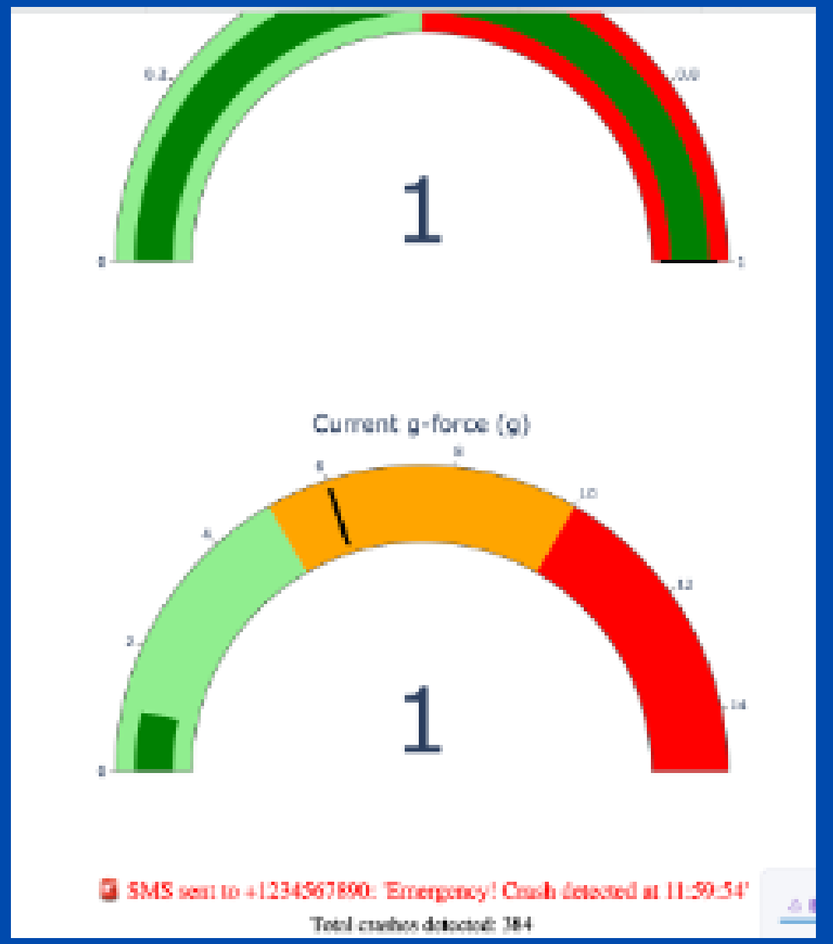
```
#include <Arduino_LSM6DS3.h>
const float CRASH_THRESHOLD = 0.8; // g, low for testing
const unsigned long CRASH_DURATION = 1000; // keep crash HIGH for 1 second
unsigned long crashStartTime = 0;
int crash = 0;
void setup() {
  Serial.begin(115200);
  while (!Serial);
  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }
  Serial.println("Arduino Crash Detection Started");
}
void loop() {
  float ax, ay, az;
```

RESULTS

When the crash flag latched, the Python backend sent notifications. I configured Twilio and SMTP so that within about one second of a detected event, my phone received an SMS and a test email arrived in my inbox. This confirmed the end-to-end path from sensor → detection → alert was working. Importantly, the system only sent one notification per crash, avoiding repeated alerts from a single impact.

Condition	Trial	Detected	*False Alarms*	Notes
Resting on desk	10	0	0	Stable at ~1 g
Gentle cable bump	5	0	0	Debounce suppressed
Strong finger tap	10	9	1	Gauge turned red, SMS received
Continuous vibration	5	0	0	Voting suppressed false alarms





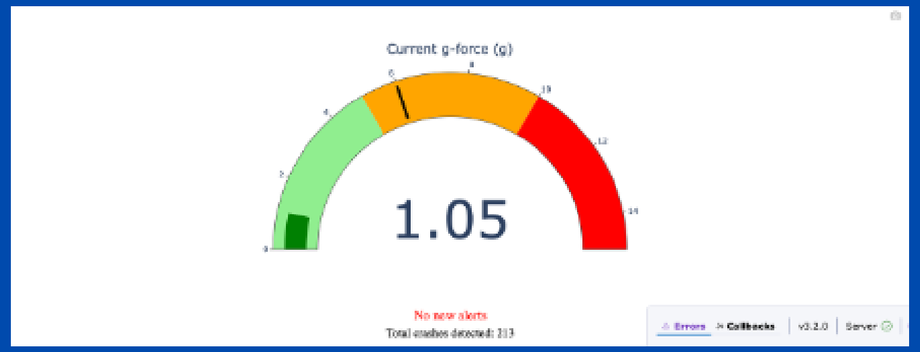
DISCUSSION & LIMITATIONS

Strengths:

- Transparent, reproducible, low cost.
- Reduced false positives compared to literature.
- Fast detection (<300 ms).
-

Limitations:

- ±4 g IMU range may miss severe crashes.
- Only accelerometer data (no GPS/gyro).
- Safe lab tests only, not real collisions.



CONCLUSION & FUTURE WORK

• **Performance**:
System must load in under 3 seconds and support high-volume traffic (1M bookings/hour).

- • Built an end-to-end crash detection system with Arduino + Python.
- Successfully detected crash-like events and sent alerts in real time.
- Demonstrates SIT225 skills: data capture, analysis, dashboard, notifications.

Future Work:

- Add GPS and gyroscope for more context.
- Use ML classifier to improve accuracy.
- Share dataset for educational use.
- Optimise for mobile/IoT deployment.

THANK YOU