Resting Angle Baseline with Tilt Detection

Anti-Theft Sensing PoC – Technical Summary Document

---

1. Objective / Use Case

The objective of this Proof of Concept (PoC) is to design and validate a **tilt-based anti-theft detection mechanism** using a motion sensor on an embedded Linux board. The system continuously monitors the resting angle of a stationary object (e.g., a parked two-wheeler) and triggers an alert **only when an abnormal or sustained change in orientation is detected**, indicating possible unauthorized movement or theft.

This PoC focuses **exclusively on tilt detection logic**. Immobilizers, DTCs, or vehicle control features are **explicitly out of scope**.

---

2. Problem Understanding

When a vehicle is parked, its orientation remains nearly constant, apart from minor vibrations and sensor noise. A theft attempt typically introduces:

- A **significant angular deviation** from the resting position
- The deviation is **sustained over time** (not a momentary bump)

Key challenges:

- Avoid false positives due to noise
- Detect only *meaningful* movement
- Trigger alert **once per event**, not continuously
- Work within constraints of embedded Linux (Buildroot)

---

3. Hardware & Platform Used

3.1 Sensor

- **MPU-9250 (9-DOF IMU)**
- Accelerometer, Gyroscope, Magnetometer
- **Only accelerometer data is used** for this PoC
- Communication via **I2C**

Reason for selection:

- Widely available
- Stable accelerometer output
- Suitable for static angle measurement
- Low power

Gyroscope and magnetometer were intentionally not used to keep the PoC simple and robust.

3.2 Embedded Board

- **Buildroot Linux**
- Kernel: Linux 4.9
- Architecture: **aarch64 (ARM64)**
- I2C device nodes available (`/dev/i2c-*`)

Important constraint discovered:

- BusyBox `wget` supports **HTTP only**
- No native HTTPS / TLS support

---

4. High-Level Approach

The system follows a **baseline-and-deviation model**:

1. Capture the *resting angle* when the system is armed
2. Continuously read accelerometer data
3. Compute current tilt angle
4. Compare with baseline
5. If deviation exceeds threshold **continuously for a fixed duration**, trigger alert

---

5. Core Detection Logic

5.1 Baseline Locking

- Baseline angle is captured once at startup
- Represents the parked / locked position
- Example:

  ```
  Baseline angle = -14.72°
  ```

This baseline remains constant until system restart or re-arming.

5.2 Continuous Sampling

- Accelerometer X-axis is read periodically
- Sampling interval: **100 ms**
- Raw data converted to approximate tilt angle

5.3 Noise Filtering

Observed issues:

- Small fluctuations (±0.1–0.3°)
- Occasional 0.00 glitches due to I2C timing

Mitigations:

- Ignore minor variations
- Use absolute difference from baseline
- Logic only reacts beyond threshold

5.4 Threshold Detection

- **Tilt threshold:** configurable (e.g., 5–10°)
- Difference calculated as:

  ```
  diff = |current_angle - baseline_angle|
  ```

Only values exceeding threshold are considered suspicious.

5.5 Sustained Tilt Timer (Critical Feature)

To avoid false alerts from short bumps:

- A timer accumulates **only while diff > threshold**
- If diff falls below threshold, timer resets

Example:

```
Tilt sustained: 100 ms
Tilt sustained: 200 ms
...
Tilt sustained: 3000 ms
```

Alert is triggered **only if tilt persists for ≥ 3 seconds**.

5.6 One-Time Alert Trigger

- Once alert condition is met:
  - Alert is sent
  - Further alerts are suppressed
- Prevents message flooding

---

6. Alert Delivery Mechanism

6.1 Initial Challenge

Telegram Bot API requires **HTTPS**.

However:

- BusyBox `wget` on Buildroot → **no HTTPS support**
- Result: Silent failures despite correct logic and credentials

This issue was **not related to application code**, but platform limitations.

6.2 Final Solution

- Use **static aarch64 `curl` binary** with HTTPS support
- Deploy directly onto the board
- Use `-k` flag to bypass missing CA certificates (PoC-safe)

Example command used internally by the application:

```
/data/local/tmp/curl -k -X POST \
  -d "chat_id=<ID>" \
  -d "text=   BIKE THEFT ALERT!" \
  https://api.telegram.org/bot<TOKEN>/sendMessage
```

This approach:

- Requires no firmware rebuild
- Is common practice for embedded PoCs
- Fully validated end-to-end

---

7. Validation & Results

7.1 Normal Condition

- Board stationary

- Output shows:

  ```
  Current -13.62°, Diff 0.13°
  ```

- No alert triggered

7.2 Theft Simulation

- Board tilted manually and held ~3 seconds
- Output:

  ```
  Current -6.29°, Diff 7.20°
  Tilt sustained: 3000 ms
  🚨 THEFT DETECTED
  ```

- Telegram alert received successfully

---

8. Challenges Faced & Resolutions

Challenge	Resolution
Sensor noise	Threshold + sustained timer
False triggers	Baseline locking
Repeated alerts	One-shot trigger logic
HTTPS unsupported	Static curl binary
Android vs Linux mismatch	Correct toolchain & binaries

---

9. Final Outcome

- Reliable tilt-based anti-theft detection
- No false alerts during normal conditions
- Alert triggered **only on sustained abnormal movement**
- End-to-end validated on real hardware

This PoC successfully demonstrates a **practical, deployable sensing logic** for anti-theft scenarios using minimal hardware and software components.

---

10. Future Enhancements (Scale-Up)

- Adaptive threshold based on terrain
- Multi-axis tilt vector magnitude
- Sensor fusion (accelerometer + gyro)
- Power optimization (sleep modes)
- Secure certificate-based HTTPS

- Gateway-based alert aggregation

---

11. Conclusion

This PoC proves that **resting-angle baseline with sustained tilt detection** is an effective, low-cost, and robust approach for anti-theft sensing. The final implementation balances simplicity, reliability, and real-world constraints of embedded Linux systems.

End of Document

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <linux/i2c-dev.h>
#include <math.h>

/* ---------------- CONFIG ---------------- */
#define I2C_DEV        "/dev/i2c-4"
#define MPU_ADDR       0x68

#define SAMPLE_DELAY_MS   500
#define ALERT_TIME_MS     5000
#define TILT_THRESHOLD    7.0
#define AVG_SAMPLES       5

#define BOT_TOKEN "8561189280:AAHd7ruFtBHIs_lXcDhBh7Y30TmmxbFK_pI"
#define CHAT_ID   "8106006550"

/* --------------- I2C HELPERS --------------- */
int read_regs(int fd, uint8_t reg, uint8_t *buf, int len)
{
    if (write(fd, &reg, 1) != 1)
        return -1;
    return read(fd, buf, len);
}

double read_tilt(int fd)
{
    uint8_t d[6];

    if (read_regs(fd, 0x3B, d, 6) != 6)
        return 999.0;

    int16_t ax = (d[0] << 8) | d[1];
    int16_t ay = (d[2] << 8) | d[3];
    int16_t az = (d[4] << 8) | d[5];

    double Ax = ax / 16384.0;
    double Ay = ay / 16384.0;
    double Az = az / 16384.0;

    return atan2(Ax, sqrt(Ay * Ay + Az * Az)) * 180.0 / M_PI;
}

double read_tilt_avg(int fd)
{
    double sum = 0;
    int cnt = 0;

    for (int i = 0; i < AVG_SAMPLES; i++) {
        double v = read_tilt(fd);
        if (v != 999.0) {
            sum += v;
            cnt++;
        }
        usleep(2000);
    }

    return (cnt == 0) ? 999.0 : (sum / cnt);
}

/* --------------- TELEGRAM ALERT --------------- */
void send_telegram_alert(double cur, double diff)
{
    char cmd[512];

    snprintf(cmd, sizeof(cmd),
        "/data/local/tmp/curl -k -s -X POST "
        "-d \"chat_id=%s\" "
        "-d \"text=      BIKE THEFT ALERT! Current=%.2f°, Diff=%.2f°\" "
        "https://api.telegram.org/bot%s/sendMessage",
        CHAT_ID, cur, diff, BOT_TOKEN);

    system(cmd);
}

/* ---------------- MAIN ---------------- */
int main(void)
{
    int fd = open(I2C_DEV, O_RDWR);
```

```
if (fd < 0) {
    perror("I2C open failed");
    return 1;
}

if (ioctl(fd, I2C_SLAVE, MPU_ADDR) < 0) {
    perror("I2C ioctl failed");
    return 1;
}

uint8_t wake[2] = {0x6B, 0x00};
write(fd, wake, 2);
usleep(100000);

printf("      Bike LOCKED\n");

double baseline = read_tilt_avg(fd);
if (baseline == 999.0) {
    printf("Baseline failed\n");
    return 1;
}

printf("Baseline = %.2f°\n", baseline);

int tilt_timer = 0;

while (1) {
    double cur = read_tilt_avg(fd);
    if (cur == 999.0)
        continue;

    /* ===== ADD-1: INVALID ANGLE REJECTION ===== */
    if (cur < -85.0 || cur > 85.0) {
        printf("      Invalid angle %.2f°, ignoring\n", cur);
        tilt_timer = 0;
        usleep(SAMPLE_DELAY_MS * 1000);
        continue;
    }
    /* ========================================== */

    double diff = fabs(cur - baseline);
    printf("Current %.2f°, Diff %.2f°\n", cur, diff);

    if (diff > TILT_THRESHOLD) {
        tilt_timer += SAMPLE_DELAY_MS;
        printf("Tilt > threshold, timer = %d ms\n", tilt_timer);

        if (tilt_timer >= ALERT_TIME_MS) {
            printf("      THEFT DETECTED\n");
            send_telegram_alert(cur, diff);
            tilt_timer = 0;
            sleep(5);
            }
        } else {
            tilt_timer = 0;
        }

        usleep(SAMPLE_DELAY_MS * 1000);
    }
}
```