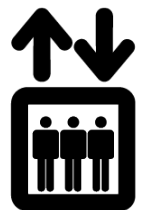


# Lift Projekt






Christian Meyer, Michael Fankhauser



# Vorgabe

- ☞ Die Beschleunigung und den zurückgelegten Weg einer Liftfahrt aufzeichnen und ausgeben.

# 1. Versuch

-  Höhe/Weg mittels GPS messen
  -  Kein Empfang in Gebäuden (insb. Liftschacht)
  -  Höhenmessung mit GPS extrem ungenau
-  Höhe/Weg mittels Barometer messen
  -  Barometer hat relativ starke Schwankungen

# 2. Versuch

📊 Beschleunigung mittels Beschleunigungssensor messen und Zeit der Liftfahrt → zurückgelegter Weg

📊 Doppeltes Integral: für jede Messung:

$$d'(v) = d(v) + v t + a t t$$

$$v'(t) = v(t) + a t$$

📊 Rauschen → Drift

📊 Messfehler innert kürzester Zeit riesig (in 10s ca. 10-20m)

# Probleme/Erkenntnisse (1)

- ↕ Erdbeschleunigung
- ↕ Nur die Z-Koordinate der Beschleunigung verwenden
- ↕ Rauschen mittels Schwellwerten filtern
  - ↕ Ausreisser führten zu Messfehler → Messung zu ungenau
- ↕ Durchschnitt aus mehreren Messungen verwenden

# 3. und bester Versuch

- 📊 Sensor-Fusion zwischen Barometer, Beschleunigungs- und Lagesensor
  - 📄 Python-Script als Grundlage



# Probleme/Erkenntnisse (2)

📊 Interpretation des Codes schwierig

📊 Grosser Aufwand bei Fehlersuche

📊 Falsche Zeiteinheit verwendet

📊 Millisekunden anstatt Sekunden



# Demo





```

private float computeAltitude(float compensated_acceleration, float altitude) {
    long current_time = System.currentTimeMillis();

    // Initialization
    if (!initialized) {
        initialized = true;
        first_altitude = estimated_altitude = altitude;
        estimated_velocity = 0;
        altitude_error_i = 0;
        first_time = current_time;
    }

    // Estimation Error
    float altitude_error = altitude - estimated_altitude;
    altitude_error_i += altitude_error;
    if (altitude_error_i < -2500f)
        altitude_error_i = -2500f;
    if (altitude_error_i > 2500f)
        altitude_error_i = 2500f;

    float inst_acceleration = compensated_acceleration * 9.80665f
        + altitude_error_i * KI;
    // time must be in s, not ms
    float dt = (current_time - last_time) / 1000f;

    // Integrators
    float delta = (inst_acceleration + KP1 * altitude_error) * dt;
    estimated_altitude += (estimated_velocity / 5.0f + delta) * (dt / 2)
        + (KP2 * dt) * altitude_error;
    estimated_velocity += delta * 10.0f;

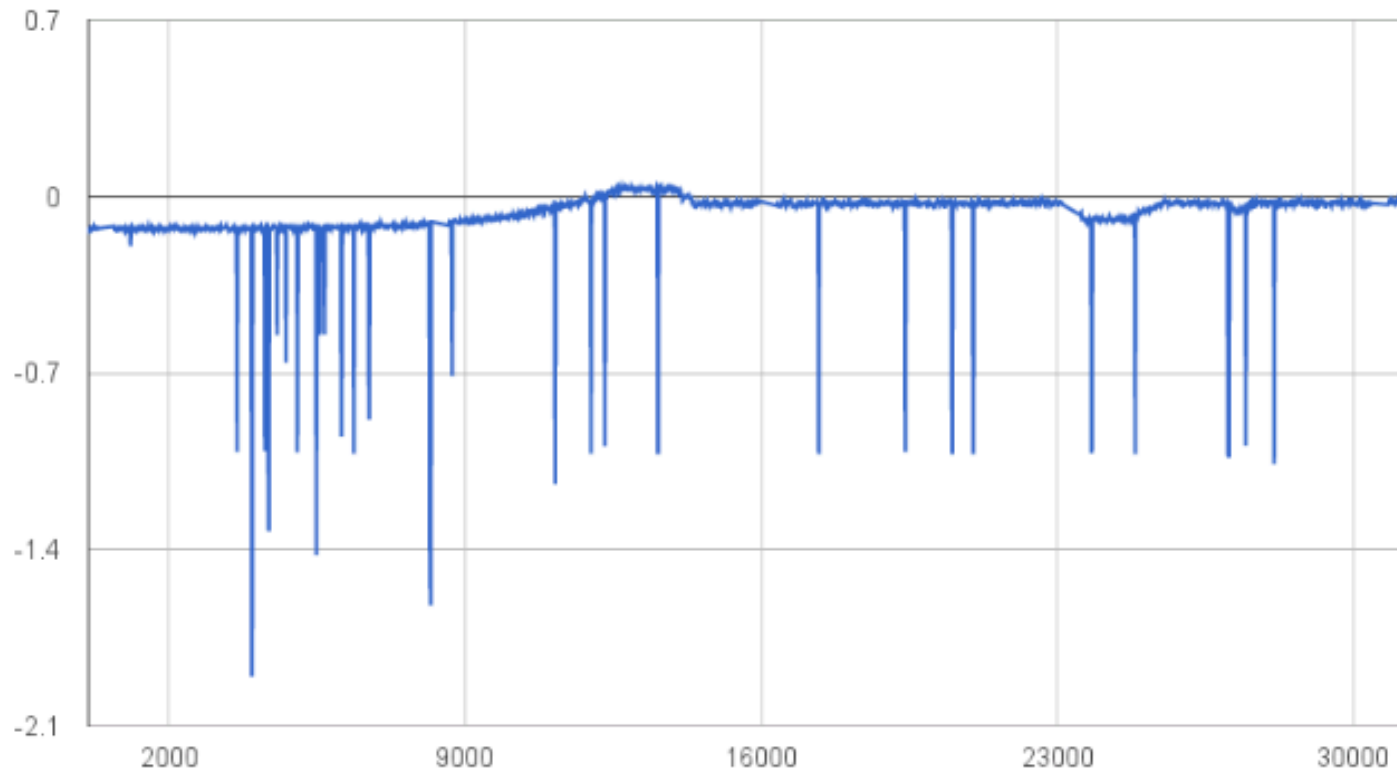
    last_time = current_time;

    return estimated_altitude;
}

```

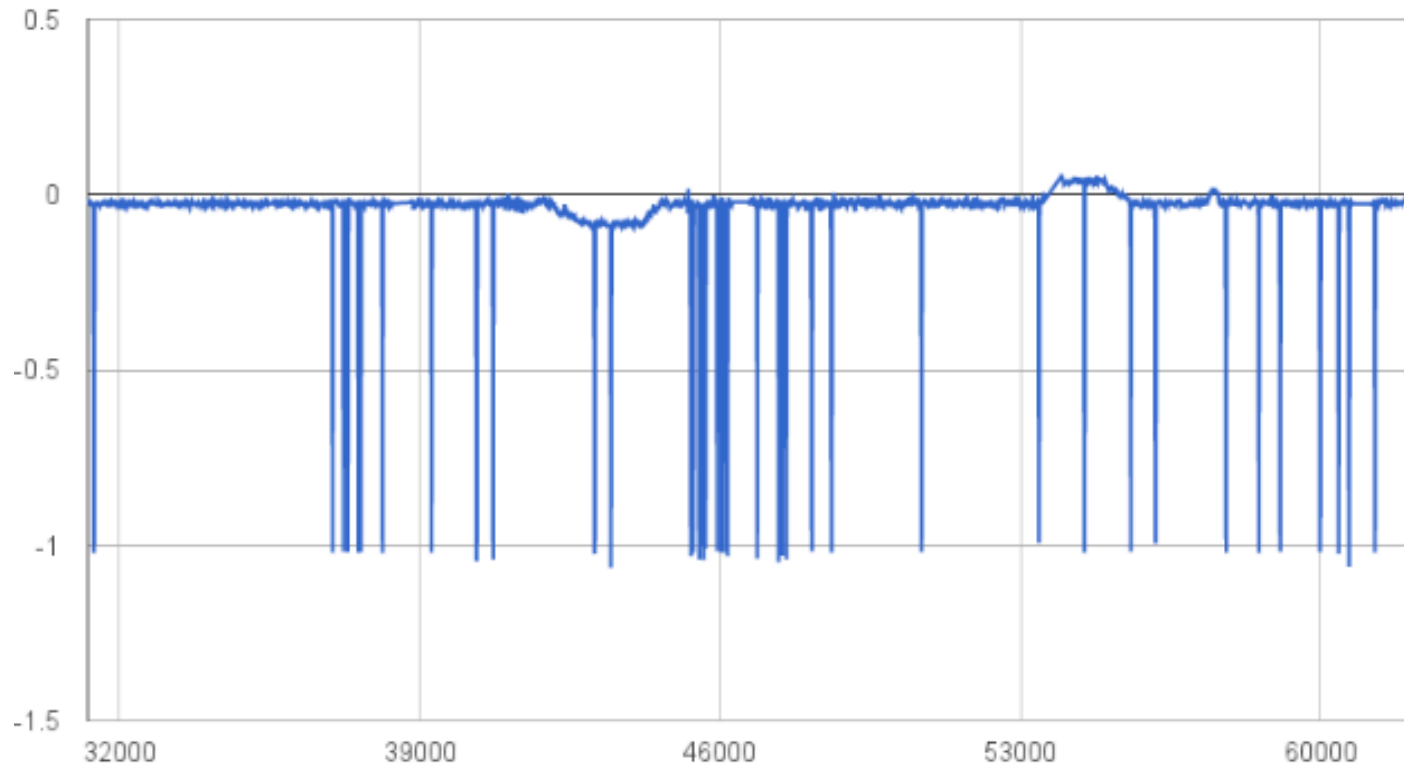
# Beschleunigungskurven (1)

 Hochfahren



# Beschleunigungskurven (2)

📊 Hinunterfahren



# GitHub Repository

[https://github.com/f4nky/Mobile\\_Lift](https://github.com/f4nky/Mobile_Lift)