Altius
Climbing Performance Evaluation System

Daria Ilina, Deniz Isik, Emirhan Tepebaşı, Mahammad Azizov, Nadia Damianova, Oleksandra Sobchyshak, Raisa Rusal, Sofiia Chehrynets

Introduction

Wearable Sensor Integration

Altius uses Arduino Nano 33 BLE Rev2-based sensors mounted on each limb to track full body motion during climbing sessions, ensuring detailed data collection

Real-Time analysis and Scoring

Captured data is streamed live to a central device, where it is analysed to produce a performance score at the end of each session

Objective Metrics for Evaluation

Key metrics are fall detection, rhythm and flow, smoothness, arm vs leg usage, stability and grip count, offering a robust and data-driven assessment of climbing performance

System Overview & Components Used

System Overview:

- 4 wearable units (1 per limb)
- Each unit captures **9-axis motion data** (accelerometer, gyroscope, magnetometer)
- Data streamed via **Bluetooth** to a Python-based central
- Performance evaluated via **algorithmic metrics**
- Scores are displayed in **GUI**

Hardware:

- Arduino Nano 33 BLE Rev2
- 7.4V LiPo batteries
- 3D printed cases
- Velcro straps

Software:

- Arduino Firmware (IMU & BLE Communication)
- Python (Data Acquisition & Processing)
- CSV Data Format for Logging Sensor Data

Arduino setup

- Each Arduino was programmed to send data to the central device using Bluetooth Low Energy (BLE) connection
- Data was sent as a single 36 byte string (9 values, 4 bytes each)
- Data was collected by the central device using bleak and asyncio python libraries
- Data was then logged to 4 .csv files, corresponding to each limb

Scoring System Overview

Metrics:

- Rhythm and Fall Detection
- Arm vs Leg Usage
- Smoothness
- Stability
- Grip Count

Each metric scores 0–100

Fall Detection and Rhythm

Goal: Evaluate control, coordination, and timing by detecting falls and assessing movement rhythm throughout the climb

***Fall Detection* – How it works:**

- Acceleration magnitude is calculated for each limb
- Partial fall: only some limbs exceed 10 m/s²
- Full fall: all limbs exceed threshold within 0.5 s
- Identifies moments of lost control

***Rhythm & Flow* – How it works:**

- Smoothed signals → detect motion vs rest
- Analyze intervals between movements
- "In rhythm" = 1.0–3.0 s between motions

Algorithm 1 Fall & Rhythm Detection in *ClimbSense*

Input: acceleration data for limbs $\mathcal{L}$; thresholds t_{fall}, t_{move}; sync window δ; pause p

Output: partial falls $\mathcal{P}$, full falls $\mathcal{F}$, rhythm score $\mathcal{R}$

procedure AnalyzeClimb($\mathcal{L}, t_{fall}, t_{move}, \delta, p$)
  $\mathcal{P} \leftarrow \emptyset$, $\mathcal{F} \leftarrow \emptyset$, movementTimes $\leftarrow \emptyset$
  for all limbs $\ell \in \mathcal{L}$ **do**
    for all frames f **do**
      compute magnitude $m = \sqrt{a_x^2 + a_y^2 + a_z^2}$
      if $m > t_{fall}$ **then**
        add $(f.timestamp, \ell)$ to $\mathcal{P}$
      if $m > t_{move}$ **and** $f.timestamp - lastMoveTime > p$ **then**
        add $f.timestamp$ to $movementTimes$
        $lastMoveTime \leftarrow f.timestamp$
  Group $\mathcal{P}$ events within window δ
  if all limbs triggered within δ **then**
    add full fall time to $\mathcal{F}$
  sort $movementTimes$
  compute intervals Δ between moves
  $\mathcal{R} \leftarrow \frac{\text{std}(\Delta)}{\text{mean}(\Delta)}$
  return $\mathcal{P}, \mathcal{F}, \mathcal{R}$

Arm vs Leg Usage

Goal: Evaluate how evenly the climber distributes movement between arms and legs, as an indicator of technique and efficiency

How it works:

- For each limb, we compute acceleration magnitude from IMU data
- A movement is counted if the change in magnitude exceeds a threshold ($\theta = 0.8\ m/s^2$)
- Total motion events are grouped: Arms vs. Legs
- Ratios are calculated to measure usage balance:

$$\text{Arm Usage Ratio} = \frac{M_{\text{arms}}}{M_{\text{arms}} + M_{\text{legs}}}, \quad \text{Leg Usage Ratio} = \frac{M_{\text{legs}}}{M_{\text{arms}} + M_{\text{legs}}}$$

Threshold Interpretation:

< 0.3 — Indicates excessive reliance on upper limbs

0.3–0.6 — Suggests balanced limb usage

> 0.6 — Reflects effective use of lower limbs, associated with improved endurance

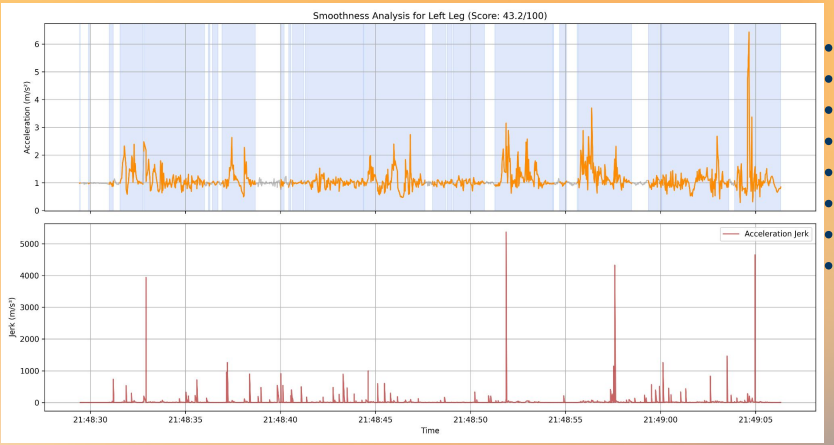
Smoothness

Goal: Measure how calm and steady the climber's movements are
We look at how much acceleration and rotation change - fewer sudden spikes = smoother movement

How it works:

- Compute magnitude of acceleration and gyroscope:
- A rolling standard deviation detects movement vs. stillness.
- Compute jerk: the rate of change of acceleration and rotation.
- Score Normalization:

$$smoothness\ score = \left(1 - \frac{median\ jerk}{\max expected\ jerk}\right) \cdot 100$$



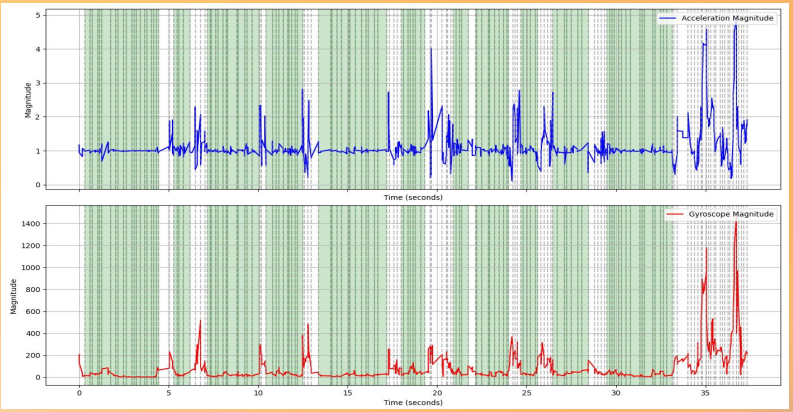
Score range: ***0 (jerky)*** *to* ***100 (very smooth)***

Stability

Goal: Evaluate how steady the climber is during static positions (holds), based on sensor stability

How it works:

- Sensor data is split into short time windows (0.25s)
- A window is "still" if values stay near resting levels
- Sequences of still windows (≥ 0.75s) are marked as holds
- Each hold is checked for signal stability (low standard deviation)
- A hold is stable if variation remains below threshold



Stability Score = proportion of hold windows classified as stable

High score → good control and precise movement

Low score → unstable holds, possible hesitation or grip issues

Grip Count

Goal: Count how often the climber holds a grip with each arm — to analyze symmetry and arm usage

How it works:

- Based on the same “still window” detection as in the Stability metric
- A grip is detected when a still period occurs between two motion phases
- Each arm is tracked independently
- The number of grips per arm is counted across the full climb

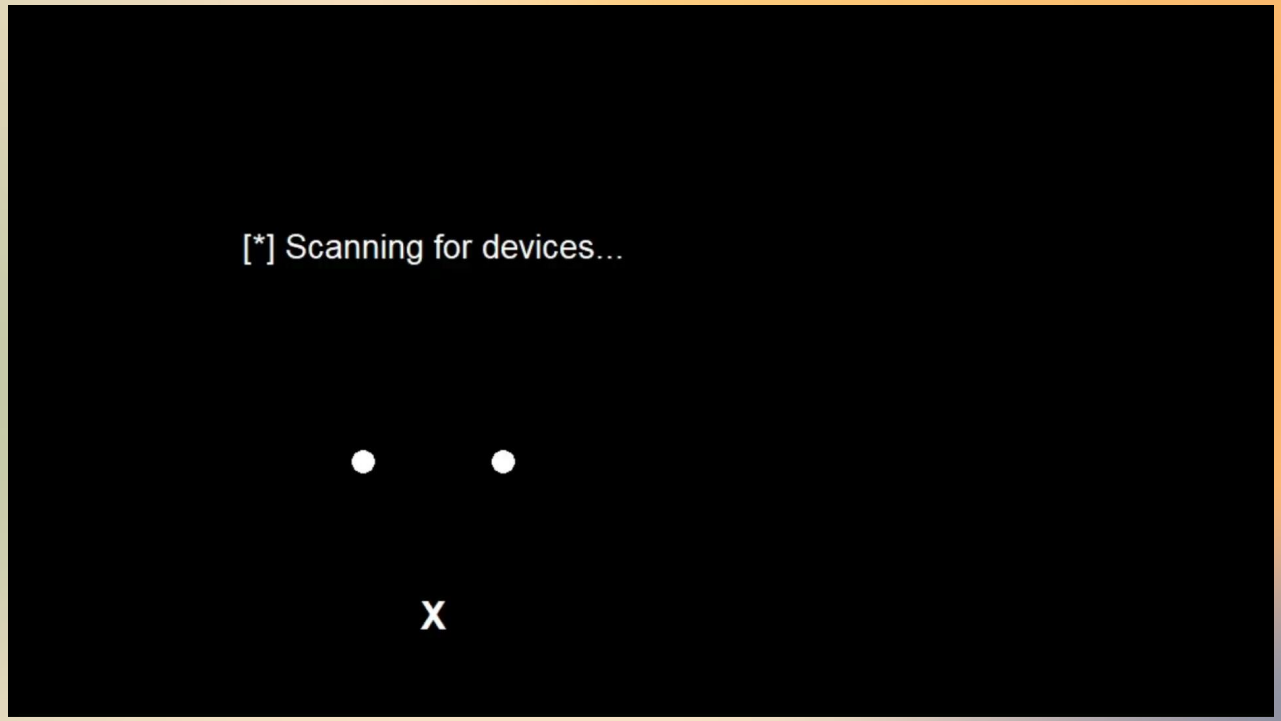
→ ***Helps identify arm dominance***

→ ***Shows if the climber uses both arms symmetrically***

→ ***Useful for technique assessment and movement balance***

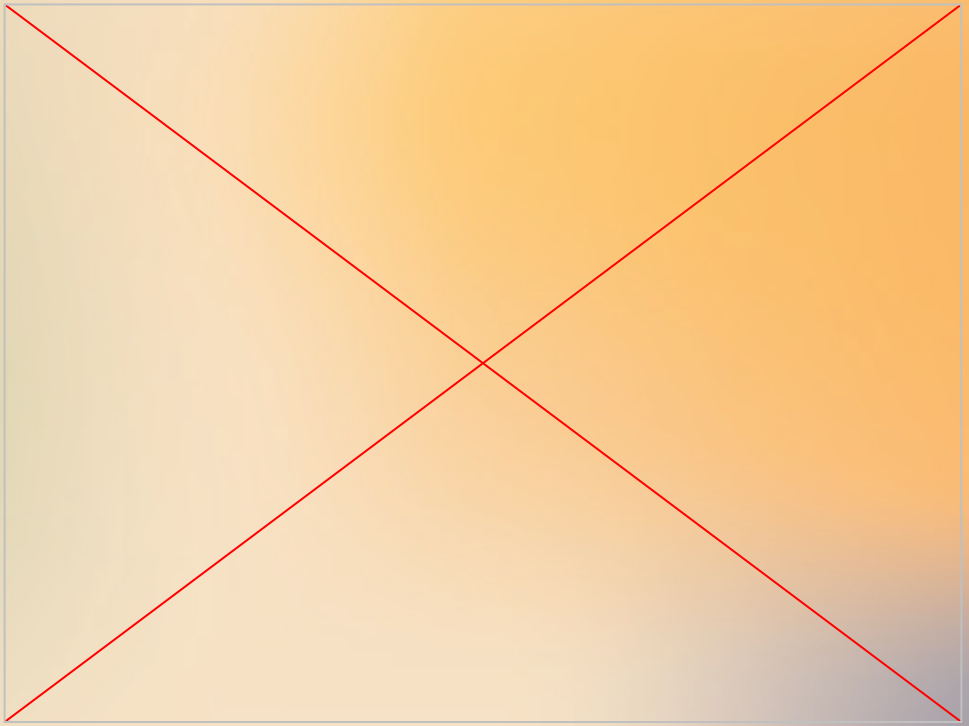
Not Smooth Climb

https://github.com/flaferty/altius/tree/main/videos



Arms vs Legs

https://github.com/flaferty/altius/tree/main/videos



Technical Challenges

1. Low data transmission rate
2. A lot of data for threshold tuning
3. Issues with concurrent connection

Challenges

1. Finding the project idea
2. Choosing the right materials
3. Time management

Thank you!

:)