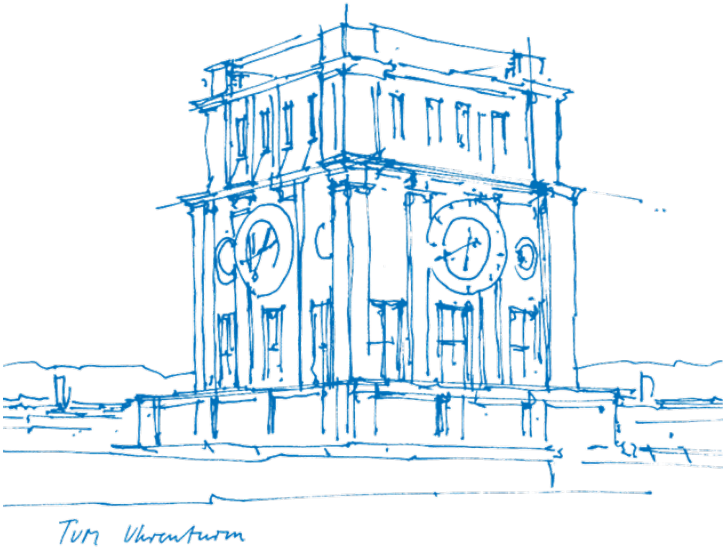
Altius
TUM School of Computation, Information and Technology
Technical University of Munich

Altius - Climbing Performance Evaluation System Project Report

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



Contents

1 Abstract

This report details the development of a wearable cyber-physical system designed for real-time evaluation of climbing and bouldering performance. Utilizing Arduino Nano 33 BLE Rev2-based sensor modules mounted on each limb, the system captures full-body motion data, which is then streamed live to a central processor. This data undergoes real-time analysis to generate a comprehensive performance score at the conclusion of each climbing session. The system integrates hardware and software components, employing Bluetooth Low Energy (BLE) for efficient data communication and a Python script for concurrent data collection and processing. Key performance metrics implemented include fall detection, rhythm and flow analysis, smoothness evaluation via jerk analysis, stability assessment during holds, and grip count tracking. Through rigorous testing and iterative parameter tuning, the system demonstrates objective, data-driven assessment of climbing movements, offering a foundation for enhanced athletic performance evaluation and further research in human motion tracking.

2 Introduction

Climbing, unlike mainstream sports such as running or cycling, remains significantly underrepresented in consumer wearable technology. Despite climbing's growing popularity, few smartwatches provide performance analytics for climbers. Metrics such as limb movement fluency or consistency remain largely inaccessible, which could provide valuable insights for improving technique and training. During our literature review, our team was inspired by the work of Kosmalla et al., who introduced ClimbSense, a system that leverages wrist-worn IMUs to automatically recognize climbed routes with high accuracy [1] and the article of Ladha et. al. [2] about the climbing analysis system for amateur climbers.

We developed a wearable system for real-time evaluation of climbing performance using Arduino-based sensor modules. Each module is mounted on a different limb to capture motion data throughout the climb. This data is streamed live to a central processor, where it is analyzed to produce a performance score at the end of each session. The system integrates hardware and software components in a cyber-physical architecture, combining real-time sensing, data communication, and computational analysis. This approach enables objective, data-driven assessment of full-body movement in climbing scenarios and opens up opportunities for further research in human motion tracking and athletic performance evaluation.

3 System Design and Hardware

We designed a smartwatch-like sensing system and deployed four identical units, one on each wrist and ankle, to capture full-body motion data during climbing. Each unit was built around the Arduino Nano 33 BLE Rev2, chosen for its compact size and integrated IMU capabilities. The board features a 9-axis Inertial Measurement Unit (IMU) with a 3D accelerometer, gyroscope and magnetometer. The readings from the IMUs were later used to recognize the movements of the climber and calculate the score.
Our IMUs support Bluetooth Low Energy (BLE), a wireless communication protocol designed for short-range,

low-power data exchange. BLE operates by broadcasting small packets of data in an advertising mode and establishing short connections for more data-intensive interactions. It achieves this by maintaining devices in a low-energy idle state and waking them only when necessary to transmit or receive data [3].
Even though the board's operating voltage is 3.3V, the input voltage for the Vin pin is 4.5 - 21V, therefore 7.4V LiPo batteries were chosen to power the boards.

To house the electronics, we custom designed an enclosure in Fusion 360 as shown in Figure 5.1, precisely modeled to fit the Arduino boards and a 7.4V LiPo battery, such that no shifting of the sensor could occur inside the box, leading to more reliable data. The case was 3D printed and featured screw holes for secure assembly, that prevented components from falling out during use. For added stability during climbing sessions, the enclosure was mounted using Velcro straps on the bottom, that ensured a firm and slip-resistant attachment to the climber.

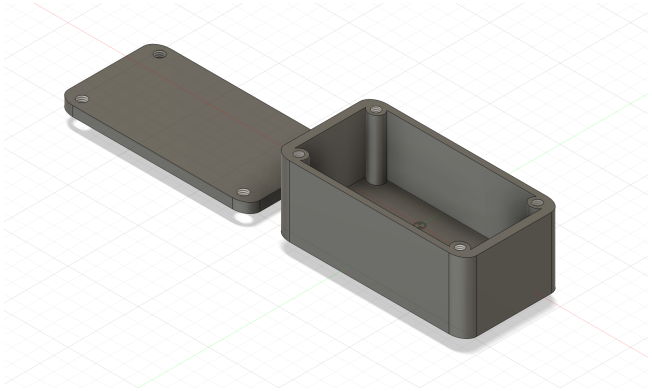


Figure 3.1 3D printed model design

4 Methodology

4.1 Sensor Setup

To transmit data from the sensors, IMU and Bluetooth Low Energy (BLE) functionalities were initialized using `Arduino_BMI270_BMM150` and `ArduinoBLE` libraries respectively. Once initialized, each Arduino advertises itself over BLE as either `IMU_LeftLeg`, `IMU_RightLeg`, `IMU_LeftHand`, or `IMU_RightHand`. A custom BLE service was defined instead of using standard BLE services, since our use case did not match any predefined BLE Services. Additionally, using a unique UUID helped avoid potential misidentification during testing in environments with many active BLE devices, such as the climbing gym. This ensured that each Arduino was reliably identified and communicated only with our intended BLE client. Once Arduino connects to BLE central device, it transmits the data at a rate of 30Hz. To support this, a custom characteristic was defined, capable of holding 36 bytes of data. To minimize communication overhead, all 9 sensor values were transmitted together in a single characteristic update (4 bytes for each value).

4.2 Data Collection

The python script (`logger.py`) was used to connect to multiple Arduinos via BLE, receive sensor data from each and log it to .csv files in real time. It uses the `bleak` library for Bluetooth Low Energy communication and `asyncio` for running multiple tasks concurrently. The script first scans the environment for BLE devices, filters

for the expected names, and establishes connections to the corresponding Arduino boards. Once connected, it subscribes to incoming sensor data. These values are timestamped and written to device-specific CSV files for later analysis.

Our implementation uses asynchronous programming via Python's `asyncio` library. Because BLE connections are inherently event-driven and can block execution, handling multiple IMUs simultaneously requires a non-blocking architecture. The script uses asynchronous coroutines (`async def`) and `asyncio.gather()` to concurrently run multiple data collection tasks. This allows all sensors to stream data in parallel without interfering with each other.

The IMU data is sent in binary format as a 36-byte BLE characteristic update, which must be unpacked using the `struct` module. The script unpacks the data as nine 32-bit floating-point values in little-endian format, validates the length of incoming packets, and handles edge cases such as malformed data or unexpected disconnections. The data was captured in CSV files in the following format: `timestamp, accX, accY, accZ, gyroX, gyroY, gyroZ, magX, magY, magZ`. A separate CSV file was created for each limb.

During development, we encountered challenges with the BLE connection setup, particularly on macOS. Although we initially assigned unique names to each Arduino device to distinguish the four units, macOS would override these names in the cache after the initial connection and generically rename them as "Arduino", although the correct names would be displayed in the application nRF Connect [4]. This interfered with our scanning and connection process using the `bleak` library, which relies on device names for identification. To resolve this, we modified our scanning approach to access the raw advertising data broadcast by each Arduino, which allowed us to retrieve the correct locally set device name directly from the advertisement payload.

4.3 Scoring System

4.3.1 Rhythm and fall detection

To evaluate the climber's control and coordination, we implemented a metric that detects the falls and analyzes the rhythm and flow of the climbing process. For this, we utilized the IMU-based acceleration data collected from the sensors and processed it through our implemented code for data handling.

Fall Detection

We categorize falls in two types. First one is partial fall, where not all parts experiences a sudden and uncontrolled motion meaning that at least one part still hold de current position. Second one is the full-body falls, where all parts show a similar and high acceleration indicating the losing contact with the wall. Every data stream was analyzed to compute the acceleration vector for all parts.

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

If this magnitude exceeded the defined threshold (10 m/s^2), the event was recorded as a partial fall for the corresponding part. If all four parts crossed the threshold limit within the synchronization window (0.5 seconds) then it is considered as a full fall. This enables to detect the times, when the climber completely loses the control.

Rhythm and Flow analysis

To measure how consistently and rhythmically the climbing process goes, we evaluated the temporal structure of movement. A skilled climber shows and alternating pattern of movement and rest, with regular pauses for stabilization and planning the next move. In order to reduce noise and false transitions the acceleration magnitudes were smoothed. Movement states are detected as either active motion or rest. These states were segmented into movement-rest cycles to compute the flow score.

$$\text{Rhythm Score} = \frac{std_{\text{I}}}{mean_{\text{I}}}$$

Where:

- $\mathbf{std}_{\mathrm{I}}$ is the standard deviation of time intervals between movement actions across parts,
- $\mathbf{mean}_{\mathrm{I}}$ is the mean of time intervals between movement actions across parts.

A window of movements was considered as "in rhythm" if the duration fell within the range (1.0-3.0 seconds), showing that it is a natural climbing pace.The sudden and unexpected movements, as well as too long rests were penalized in the scoring phase. The lower the final score is better the rhythm and flow of the climber, which is calculated by the standard deviation of time intervals between movement actions across parts.

This joint analysis both using acceleration values provides a comprehensive analysis for critical fall moments (via fall detection) and overall climb dynamics (via rhythm-flow analysis). Together, they offer an overall expression of technical skills, timing and essential attributes that affects climbing process.

4.3.2 Arm vs Leg Usage

To assess the distribution of effort between upper and lower limbs, we implemented a metric that quantifies the relative movement activity of the climber's arms versus legs. Skilled climbing technique typically emphasizes efficient use of the legs, while minimizing arm fatigue. Therefore, this metric offers insight into movement economy and technical balance.

For each limb, we computed the magnitude of the acceleration signal by combining the three axes into a single scalar value:

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

We then tracked the change in acceleration magnitude over time. If the absolute difference between two consecutive measurements exceeded a defined movement threshold $\theta = 0.8$ m/s^2, the event was counted as a discrete motion. This process was repeated independently for each of the four limbs.

The movement counts were grouped by limb type, where the arms included the left and right hands, and the legs included the left and right legs. This grouping enabled us to calculate the total number of motion events per limb category.

The following ratios were then computed:

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

Where M_{arms} and M_{legs} are the total number of detected movements for arms and legs respectively.

To interpret the outcome, we defined three usage categories:

- *Leg usage < 0.3*: Dominant arm usage, potentially indicating inefficiency or poor technique.
- *Leg usage 0.3–0.6*: Balanced use of upper and lower limbs.
- *Leg usage > 0.6*: Effective leg engagement, associated with energy-efficient climbing.

This metric provides a compact yet informative view of climber technique, highlighting whether the movement pattern is economical or relies too heavily on upper-body strength.

4.3.3 Smoothness

For the evaluation of climbers' steadiness and calmness of movement, we developed a **smoothness metric** through jerk analysis, which measures the rate of change of acceleration over time. Smoothness is a key aspect of climbing technique, as more skilled climbers tend to maintain consistent, steady movements with minimal sudden, jerky movements. To process the collected IMU data:

- We first computed the magnitude of the acceleration and gyroscope vectors to combine the 3D signal into a single scalar value:

- A rolling window standard deviation (window size = 10 samples) was applied to detect whether the limb was still or moving. Thresholds were chosen empirically (0.8 m/s^2 for acceleration, 8.0 deg/s for gyroscope) to balance sensitivity and noise tolerance:
- Movement periods were then extracted by identifying transitions between stillness and movement using binary classification and timestamped segmentation.

Once movement periods were detected, we calculated jerk (change in acceleration over time) within each segment. The average absolute jerk from both acceleration and gyroscope data was combined into a single smoothness indicator. To make the results interpretable, the score was normalized using a predefined maximum expected jerk value. This resulted in a score between 0 (very jerky) and 100 (very smooth). The final scoring formula used was:

$$\text{Smoothness Score} = \left(1 - \frac{\text{Median Jerk}}{\text{Max Expected Jerk}}\right) \times 100$$

To interpret and analyse, we visualized both acceleration magnitude and jerk. The acceleration magnitude is shown in the top subplot (Figure 4.1), with detected movement periods highlighted. The corresponding jerk signal is displayed in the bottom subplot over time.

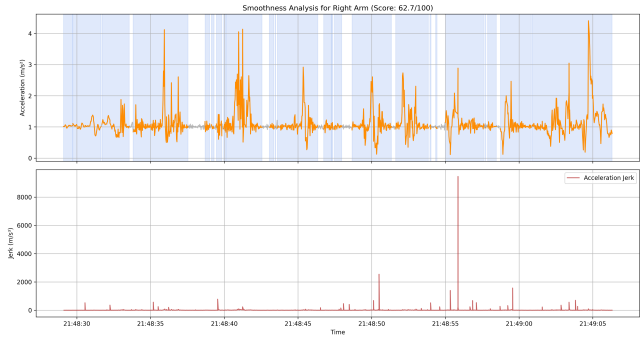


Figure 4.1 Detected movement periods and corresponding jerk used for smoothness evaluation.

A key challenge in developing this feature was selecting the appropriate thresholds to distinguish real movement from sensor noise. If the system is too strict, it may miss actual movement, if too sensitive, it may detect false positives. Through iterative tuning using recorded climbing data from testing, we established thresholds that generalized well across different cases, and resulted in a reasonable score.

4.3.4 Stability

We defined stability as a metric that evaluates a climber's steadiness during static positions. The algorithm processes raw accelerometer and gyroscope data by dividing it into short time windows, approximately 0.25 seconds in length (marked with dotted lines on Figure 4.2). Within each window, it calculates the magnitude of both acceleration and angular velocity to determine whether the limb was relatively still. A window is classified as "still" if the average deviation from resting values (1g for acceleration, and 1 for gyroscopic motion) remains below a specified tolerance.

Once all still windows are identified, the algorithm looks for groups of at least three consecutive still windows. This time period was chosen using trial and error - after data collection we realized that a hold usually lasts at least 0.75s. These clusters are interpreted as holds—periods where the climber is assumed to be gripping or stepping on a hold with minimal movement. The identified 'hold' periods are highlighted green on the graph (Figure 4.2). For each of these hold periods, the algorithm then calculates the standard deviation of the accelerometer and gyroscope signals. Lower variability indicates a steadier, more controlled hold. A hold window is marked as stable if the variation remains below a set threshold across all axes.

The final output is a stability score representing the proportion of hold windows that were stable. A higher score suggests better control and better technique, while lower scores may indicate hesitation, or poor grip mechanics.

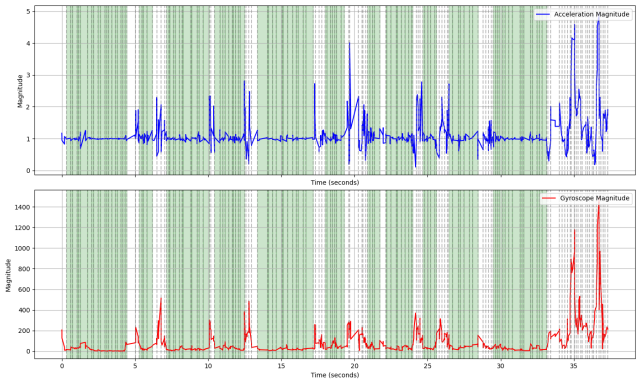


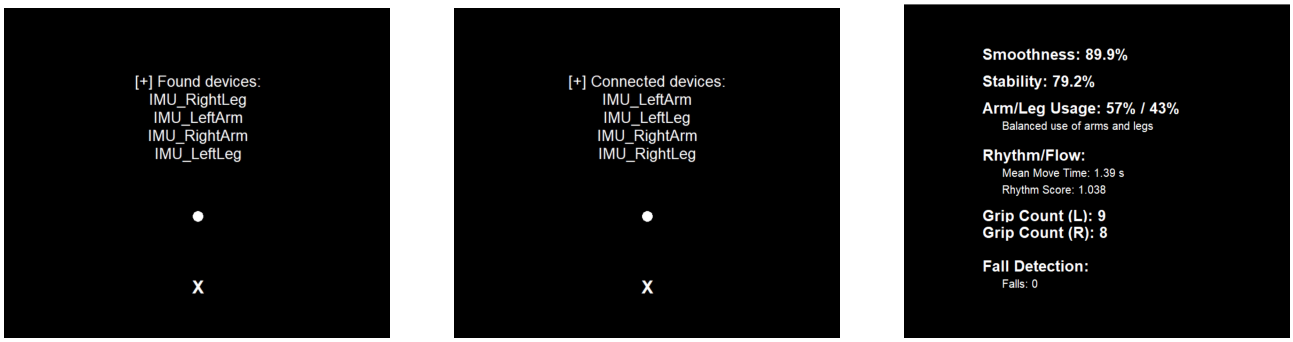
Figure 4.2 Graphs representing accelerometer and gyroscope example readings for 1 limb

4.3.5 Grip Count

We define a 'grip' as a period of time when the climber holds on to the hold. To calculate the number of grips for each arm, the algorithm described in the stability section was used. Our script analyzes the data and detects periods of stillness. Each period of stillness in between sharp changes in acceleration counts as one grip. Tracking how often a climber holds a grip with each arm during a climb can provide valuable insights. It may reveal arm dominance or indicate whether the climber is approaching the route in a balanced and symmetrical manner.

4.4 GUI

We created a simple GUI to improve the experience of using our climbing evaluation system. The following are the screenshots of the GUI:



(a) Loading screen **(b)** Data recording screen **(c)** Data display

Figure 4.3 Overview of GUI screens

5 Challenges

The bracelets were tested extensively throughout the development process. One of the earliest challenges we encountered was the low rate of data transmission. Initially, we implemented a polling approach, where the central device repeatedly requested data from the Arduinos. This method proved inefficient, yielding a data rate of only 2 Hz—far too low for capturing dynamic movement. Switching to BLE notifications, where the Arduinos

automatically pushed data to the central device as it became available, led to a significant improvement. This adjustment enabled a stable transmission rate of 30 Hz, which was essential for accurate and responsive motion tracking.
Beyond resolving communication issues, testing also played a critical role in tuning the parameters of the analysis functions. Parameter tuning was one of the biggest challenges. It was important to collect the data from many different kinds of climbs (fast, slow, smooth, quick etc) to be able to choose appropriate values. In total, we collected over 30 datasets during development to refine and validate our system.
Another big challenge was the design of the bracelets. They had to be compact, so that they would not interfere with the climb, while also being sturdy enough to keep the Arduinos intact. Before landing on our final design, we created 3 prototypes.
Finally, we had issues with the `bleak` library, when connecting to all 4 of the Arduinos. Specifically, the issues arose from the combination of `bleak`'s device discovery and connection routines with Python's `asyncio` event loop. Occasionally, when attempting to establish concurrent connections to multiple Arduinos, one or more of the devices would fail to respond or enter a blocked state. One contributing factor appears to be improper shutdown of BLE sessions. When the program was interrupted or terminated abruptly (e.g., via SIGINT or a manual stop), the Arduinos were sometimes left in a non-reset state, possibly due to incomplete disconnection handshakes. We attempted various strategies to mitigate this issue, including staggering the connection attempts and adding explicit disconnection logic on shutdown. While these steps reduced the frequency of connection failures, they did not fully eliminate the problem. In some cases, the Arduinos still required manual power cycling after the program exited unexpectedly or was terminated.

Figure 5.1 Assembled bracelet (bolts are not screwed in)

6 Conclusion

This project successfully developed and implemented a novel wearable system for objective, real-time climbing performance evaluation. By integrating Arduino-based IMU sensors with Bluetooth Low Energy communication and a robust Python-based data processing pipeline, we created a cyber-physical system capable of capturing and analyzing full-body motion during climbing sessions. The system's key achievements include the design of compact, 3D-printed sensor enclosures, the establishment of a reliable 100 Hz data transmission rate, and the development of sophisticated algorithms for assessing critical performance aspects such as fall detection, rhythm and flow, movement smoothness (via jerk analysis), static stability during holds, and grip count.

Challenges encountered, particularly with BLE device identification on macOS, were effectively overcome by adapting our scanning approach to parse raw advertising data, ensuring reliable connections. Extensive testing, involving over 30 datasets, was crucial for refining communication protocols and empirically tuning the parameters of our analytical functions, leading to a robust and accurate scoring system.

The developed system provides climbers with objective, data-driven insights into their performance, enabling them to identify areas for improvement and track progress. This work lays a strong foundation for future research in several directions. Potential enhancements include the integration of machine learning models for more nuanced movement pattern recognition, the development of a user-friendly mobile application for real-time feedback and visualization, and further validation of the scoring metrics against expert human assessment. Additionally, exploring the system's applicability to other sports or rehabilitation scenarios could broaden its impact.

References

[1] F. Kosmalla, F. Daiber, and A. Krüger, "Climbsense: Automatic climbing route recognition using wrist-worn inertia measurement units," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 2033–2042, 2015.

[2] C. Ladha, N. Y. Hammerla, P. Olivier, and T. Plötz, "Climbax: Skill assessment for climbing enthusiasts," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, (New York, NY, USA), pp. 235–244, Association for Computing Machinery, 2013.

[3] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *sensors*, vol. 12, no. 9, pp. 11734–11753, 2012.

[4] Nordic Semiconductor, "nRF Connect for Desktop." `https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-Desktop`, n.d. Accessed: 2025-07-18.