Project report for the course Multisensory Interactive Systems Academic Year: 2024-2025

Project Report: A Multisensory Interactive System for Real-Time Bodyweight Exercise Feedback

Author: Samuele Trainotti

Email: samuele.trainotti@studenti.unitn.it

Matricula: 247428

Abstract

This report presents the design, implementation, and evaluation of a wearable multisensory system for real-time feedback on bodyweight exercises, stretching, and yoga poses. The present work has the aim of helping amateur and professional athletes to verify the correct execution of exercises without a personal coach, which can lead to reduced effectiveness and potential injuries. The proposed solution is a modular prototype composed of Inertial Measurement Units (IMUs) attached to the user's wrist and ankle, a microcontroller for data acquisition, and a Raspberry Pi for local processing and web-based feedback. The system tracks the user's movements, calculating angles and counting repetitions to provide both visual and auditory feedback through a web dashboard. A key aspect of this work is the comparative analysis of a basic accelerometer (ADXL337) and an advanced sensor with an integrated fusion algorithm (BNO055), highlighting the trade-offs between implementation complexity/data quality and cost of the equipment. The system's architecture, featuring a portable, wireless-accessible interface, distinguishes it from cable-dependent solutions. The main evaluation findings are expected to show that real-time, multisensory feedback can improve exercise accuracy and user motivation.

1. Introduction

This project is about the development of a wearable interactive system designed to assist users in performing bodyweight exercises correctly. As an amateur athlete training over 10 hours per week, the motivation for this project stems from a personal need for a reliable method to ensure proper form during crucial strength and stretching routines, which are vital for performance and injury prevention. The goal is to create a system that acts as a virtual coach, providing immediate and intuitive feedback on body posture and movement synchronization.

The context is the growing field of personal fitness technology, where sensors and interactive systems are increasingly used to monitor and enhance physical activity. This area is particularly interesting due to its potential to make personalized coaching more accessible and to explore scientific questions about the effectiveness of real-time biofeedback on motor learning and body awareness.

Our primary hypotheses are:

1. Real-time, multisensory feedback (visual and auditory) on joint angles and movement repetition can significantly improve the accuracy of a user's exercise execution compared to no feedback.
2. A system providing immediate, clear feedback will increase user engagement and confidence during unsupervised training sessions.

This project builds upon theoretical perspectives of multisensory perception by integrating auditory and visual cues to reinforce correct movements, potentially enhancing the user's proprioceptive awareness.

2. Related Work

Several projects and commercial products have explored the use of wearable sensors for fitness and rehabilitation. The objective is often to track activity, count repetitions, or provide feedback on form.

Systems like the **Vi "Hearable"**[1] fitness tracker use voice coaching to guide runners, but they lack the granular, real-time feedback on body posture that our system provides. Similarly, commercial products from companies like **Moov** and **Wahoo** offer wearable sensors that track metrics such as cadence, impact, and repetition count for various exercises. While effective, these systems often rely on proprietary algorithms and closed ecosystems, limiting customization and transparency.

In the academic sphere, researchers have extensively used IMUs for posture and activity recognition. For instance, a study[2] demonstrated the use of multiple IMUs to accurately classify different yoga poses. Their approach often involves complex machine learning models that require significant training data. Our project, in contrast, focuses on a more direct, physics-based approach by calculating Euler angles, which is computationally less intensive and does not require pre-trained models, making it more adaptable for custom or new exercises.

Another relevant area is the use of sonification for motor learning. Other research[3] has shown that auditory feedback linked to movement parameters can help individuals correct their motions more effectively. Our system incorporates this by using simple sound cues (e.g., a "bell" sound) to signal the completion of a repetition or the achievement of a target angle, providing an intuitive and non-distracting feedback channel.

My goal is to:

- **Directly comparing** the implementation complexity and data output of a basic accelerometer (ADXL337[4]) versus a sensor with on-board processing (BNO055[5]).
- Implementing a **portable, self-contained, and network-accessible architecture** (Teensy + Raspberry Pi) that frees the user from being tethered to a computer.
- Providing a **simple, open, and extensible web-based interface** for feedback, which can be accessed from any device on the local network.

3. Architecture Design

The system is designed as a modular and wearable unit that captures, processes, and presents motion data to the user in real-time.

Key Assumptions:

- The user will wear the sensors securely on the designated wrist and ankle. Movement of the sensor relative to the limb is assumed to be minimal.
- The present prototype is designed to evaluate the correct execution of one exercise: the bird dog pose[6]. In order to have the maximum precision of data outage, I decided that the BNO055 sensor, which is more accurate, is positioned on one ankle and the ADXL337 on the opposite wrist. This assumption is crucial because the ADXL337 provides instantaneous acceleration readings, and so, needs a clearer and wider movement to operate well.
- The user will be within the local Wi-Fi network range to access the feedback dashboard.

These assumptions are reasonable for typical bodyweight workout scenarios. Secure straps can minimize sensor shift, and most common exercises (superman pose, squats, lunges, bird dog pose) involve clear limb orientation changes.

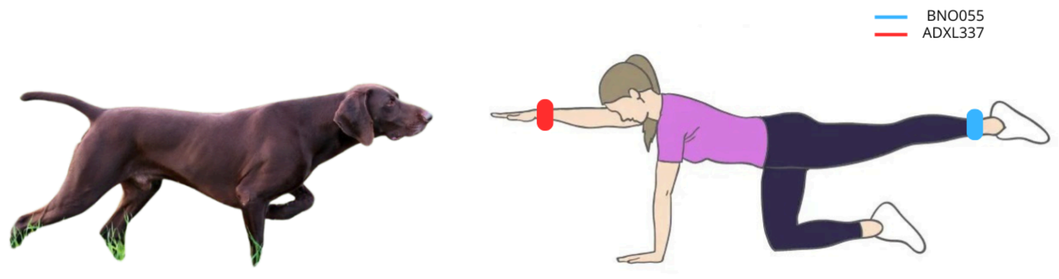

Fig. 1: Hunting dogs are trained to indicate the location of prey to their handler. That inspired the exercise taken into exam shown in the image above. Moreover, the position of the sensors is represented in order to better understand the configuration of the architecture.

System Components:
The architecture consists of three main hardware components and a software layer:

1. **Sensing Unit:** Comprises two IMUs: the ADXL337 (analog accelerometer, represented in red) and the BNO055 (9-DOF absolute orientation sensor, represented in blue). These are attached to the user's wrist and ankle.
2. **Processing Unit:** A **Teensy** microcontroller is used for data acquisition from the sensors. It performs initial data filtering and processing and communicates the results via a serial connection. Since the current system is just a prototype involving few sensors, it also

does the exercise recognition task (detecting movements).

3. **Server & Hosting Unit:** A **Raspberry Pi 4** receives the processed data from the Teensy. It runs a Python-based web server (built on Flask framework[7]) that hosts the user interface and serves the motion data. In a more complex scenario, with several sensors and/or MCUs, the Raspberry should be elected as a centralized hub for interconnecting each MCU and fusing data coming from those devices.
   Based on datasheets and common usage, with the current setup I expect that the instantaneous consumption for the Raspberry Pi and Teensy could be not more than 1000mA. This setup could be easily powered by a portable power bank (usually at least 5000mAh or more), making the entire system wearable.
4. **User Interface:** A web-based dashboard accessible via any device on the local network. It displays real-time data, repetition counts, and provides visual and auditory feedback.

Tools Used:

- **Hardware:** Arduino UNO (for initial prototyping), Teensy 3.6, Raspberry Pi 4, ADXL337, BNO055, a powerbank, a tp-link mobile router[8], and some electrical components for the circuitry (cables, breadboard, button).
- **Software:** Arduino IDE (with C++), Python with Flask framework, Processing (for initial visualization), HTML/CSS/JavaScript (for the dashboard).

3.1 Usage Model

From a user's perspective, the system is straightforward to use.

1. The users first strap the sensor modules to one wrist and the opposite ankle.
2. To create a wireless network, it should power the router.
3. They power on the system (e.g., by connecting the power bank to the Raspberry Pi). It's already configured for connecting to the router.
4. On a smartphone or laptop, they connect to the local Wi-Fi and navigate to the web address provided by the Raspberry Pi's server. (http://raspberryIP:5000)
5. The web dashboard appears, showing the status of the sensors. Visual and auditory feedback are available at the page http://raspberryIP:5000/feedback.
6. The system waits for the user to position comfortably. When ready, the calibration is started by pressing the button on the breadboard.
   In the bottom-right corner there is an icon indicating whether the audio is enabled. If not, users have to click on the icon a couple of times to tell the browser to enable the audio interface.
7. Since this is a prototype, we implemented only one exercise (bird dog) for demonstration. Therefore, the interface will directly show the target number of repetitions.
8. As the user performs the exercise, the dashboard provides real-time status information, while the feedback page provides both auditory signals and visual effects. There are three main phases with the respective effect:
   - Hold phase: when the user reaches the target position, it's prompted to hold for a default time (in our case 500ms). The screen background changes to dark

    yellow and an ascending sound is played.
    - Release phase: the user is prompted to go back to the starting position. The background changes to green and a descending sound is played.
    - Completed phase: when the target count of repetitions is reached, a “good job” text is displayed with a purple background and a 4-note fanfare is played.
9. The system does not reset its state by itself. The user has to reboot the Raspberry (e.g. by disconnecting and reconnecting the power supply) to start over. Refreshing the webpage would avoid cache corruptions.

The user does not need to know about the underlying data processing, how repetitions are counted and feedback generated, only how to interpret the visual and auditory cues.

4. Implementation

The implementation was carried out in distinct phases, evolving from initial sensor testing to the final integrated system.

4.1 Sensor Data Processing (ADXL337)

The raw data from the ADXL337 was noisy and required significant pre-processing. The steps implemented on the Teensy are as follows:

1. **Calibration:** Upon startup, 100 samples are read while the sensor is static to establish a "zero" baseline. This baseline is subtracted from subsequent readings to measure relative changes with respect to the starting position, and not just absolute values of orientation.
2. **Dead-Zone Filter:** To prevent minor jitters from being registered as movement, a threshold was implemented. If the change in acceleration is below this threshold (0.02g), the value is ignored.
3. **Exponential Moving Average (EMA) Filter:** A low-pass EMA filter was applied to the raw acceleration data to smooth it out (with an alpha constant of 0.2). A second EMA filter with a different alpha parameter (with value 0.2) was applied to the calculated pitch and roll angles for further stabilization.
4. **Normalization:** The x/y/z acceleration values were normalized by the magnitude vector. This ensures consistent values of the acceleration.
5. **Angle Calculation:** Pitch and roll angles were calculated from the processed acceleration data, and were processed with a EMA filter as well, but with a different alpha.

4.2 Sensor Data Processing (BNO055):

In contrast, the BNO055 was significantly simpler to implement. Using the Adafruit BNO055 library[8], the sensor was initialized, and its fused Euler angle outputs were read directly. The library handles the complex sensor fusion algorithms internally.

4.3 Sensor Data Processing (main):

The code for both sensors was integrated into a single C++ style project on Teensy, with separate header (.h) and source (.cpp) files for clean organization. The main .ino file handles the main loop, calling functions to read from each sensor. The logic for counting repetitions was implemented by detecting when the wrist reaches a target angle and then returns to a resting position. The ankle sensor is only used to check if the shin moves parallel to the ground, meaning that the rotation with respect to the starting position must remain small (for example, at most 20°).
For the proposed exercise, the user starts with ankles and wrists on the ground. Both shins and palms lie on the ground. When the user starts the movement he raises one arm and the

opposite leg simultaneously. So the target angle is 90° degrees up from the ground, since the starting position is marked as 0° orientation.
When the target position is reached a timer tracks the hold time. The repetition is marked as valid only if the hold time is greater than a threshold (500ms).
Upon successfully maintaining the position, the return phase is triggered and the code checks if the user returns back to the starting position.
This allows the correct tracking of a complete repetition and does not count small or incomplete movements.

4.4 Web Backend and Frontend

The Raspberry Pi runs a Flask web server that serves a webpage built with HTML and JavaScript. A Python script continuously reads the CSV data from Teensy's serial port. This data (e.g., angles, rep count) is then made available via a simple API endpoint. Initially the webpage was pooling updates on regular intervals (100ms), but this configuration did not allow feedback to be in real time.
This client-server architecture was an interesting implementation challenge, because ensuring a low latency between the physical movement and the feedback would enable a more natural and pleasant experience to the user.
A robust improvement was to enable the use of WebSockets, allowing the reception of data immediately as soon as ready on the serial connection.
Based on the received messages, the webpage updates its state and triggers the relevant effects through javascript animations and sound effects.

5. Proposed Evaluation

Given the proof-of-concept nature of this work, a two-stage evaluation approach was considered. The first stage involved a preliminary pilot study to gather initial qualitative feedback and identify any major usability issues. The second stage, proposed as future work, consists of a formal experimental design to quantitatively measure the system's effectiveness.

5.1 Pilot Study and Qualitative Feedback

A preliminary pilot study was conducted with a small group of non-expert users to gather initial impressions of the system's design and functionality. The feedback, while not scientifically rigorous, provided valuable validation and constructive criticism for future iterations.

The overall concept was reported to be promising. Users found the visual and auditory cues to be intuitive for understanding when a repetition was successfully completed, describing the real-time feedback as "engaging." Two key design aspects received specific comments:

1. **Device Independence**: The web-based feedback dashboard was praised for its flexibility, allowing it to be used on any browser-enabled device (phone, tablet, Smart TV) on the local network.
2. **System Portability**: Removing the tethered connection to a host computer was seen as a major advantage. However, users noted that the central unit itself was still somewhat bulky and uncomfortable to wear, and that the physical wires connecting it to the limb-worn sensors remained an obstacle to completely free movement.

These findings validate the core feedback mechanism and highlight specific hardware challenges to address in future designs.

5.2 Proposed Formal User Study

Based on the promising results of the pilot study, a more rigorous, formal evaluation is proposed to quantitatively assess the system's impact on exercise performance and user experience.

- **Hypotheses**:
  - Users performing an exercise with the system's feedback will achieve more accurate and consistent joint angles than users without feedback.
  - Users will report higher levels of engagement, confidence, and perceived accuracy when using the system.
- **Procedure**: A group of participants with varying fitness levels would perform a set number of repetitions of a specific exercise (e.g., bird dog or other implemented in the

future like squats) in two conditions: with and without the multisensory feedback. To mitigate learning effects, the order of conditions would be counterbalanced.

- **Variables**:
    - **Independent Variable**: The feedback condition (with vs. without the system).
    - **Dependent Variables**:
        - **Behavioral**: Movement accuracy, measured by the deviation from target angles.
        - **Subjective**: User-reported ratings on engagement, confidence, and usability via a post-session questionnaire.

This formal study would provide the empirical data needed to scientifically validate the system's benefits.

6. Discussion and Conclusions

The development of this prototype successfully demonstrated the feasibility of a portable, multisensory system for real-time exercise feedback. The qualitative feedback gathered during the pilot study suggests that the core concept is both useful and engaging for users. The architectural choice to serve feedback via a local web server proved to be highly effective, offering excellent device flexibility. This evidence, while not scientifically rigorous, provided valuable validation and constructive criticism for future iterations.

6.1 Limitations

This project, as a prototype, has several limitations:

- **Sensor Coverage:** The system currently uses only two sensors, on one wrist and one ankle. This limits its ability to provide feedback on more complex movements involving the torso, hips, or both sides of the body simultaneously.
- **IMU Drift and Sensor Capabilities:** The two sensors used have vastly different capabilities, which introduces specific limitations:
    - The **BNO055**, while robust, is still susceptible to the slow accumulation of orientation error (drift) common to all IMUs, particularly over long workout sessions. The current implementation does not include advanced software-level drift correction.
    - The **ADXL337**, being a 3-DOF accelerometer, presents a more fundamental problem. It only provides instantaneous acceleration readings. This makes its orientation data highly unreliable during dynamic movements, as it's impossible to distinguish the constant acceleration of gravity from the user's own motion. This makes the "drift" or error far more pronounced and immediate compared to the BNO055. Furthermore, **yaw (rotation around the vertical axis) could not be calculated at all**. An accelerometer can only determine tilt (pitch and roll) relative to the direction of gravity; it has no reference for horizontal orientation.
- **Generic Movement Model:** The system relies on pre-defined target angles for exercises. It does not adapt to a user's individual biomechanics, flexibility, or fitness level.

6.2 Lessons Learned

A key lesson was the stark contrast in complexity between processing raw sensor data (ADXL337) and using a sensor with integrated processing (BNO055). The significant time spent on filtering and stabilizing the ADXL337 data underscored the value of investing in more advanced hardware to simplify software development and focus on application-level logic. Furthermore, the iterative prototyping process—starting with a simple cabled Arduino setup and evolving to a more complex, user-friendly wireless architecture (Teensy + Raspberry Pi)—proved to be an effective strategy for managing complexity and achieving the final project goals.

6.3 Future Work

Building on this proof-of-concept, it is possible to enhance several features:

- **Expand Sensor Network:** Incorporate additional IMU sensors (such as the MPU9250s initially considered) to create a full-body motion capture system in order to have feedback on complex exercises like yoga poses.
- **Implement Machine Learning:** A machine learning model could be trained to automatically recognize different exercises, eliminating the need for manual selection. This would also allow for more nuanced and personalized feedback.
- **Develop User Profiles:** A future version could allow users to create profiles to track their progress over time, set personal goals, and customize the feedback to their specific needs and body types.

6.4 Conclusions

This project successfully demonstrated the feasibility of creating a wearable, portable, and accessible multisensory system for real-time exercise feedback. The implemented prototype effectively integrates different sensor technologies, a microcontroller, and a web-based interface to deliver intuitive guidance. The informal feedback gathered was positive, indicating that the system has the potential to improve exercise form and user motivation. The modular architecture provides a solid and scalable foundation for the future work required to develop this prototype into a fully-featured virtual coaching tool.

7. Group Members Contributions

Since the project was carried out on an individual basis, I was the only responsible for all aspects of the project, including initial ideation, hardware selection and sourcing, circuit prototyping on Arduino and Teensy, data processing and filtering algorithm implementation in C++, development of the Python Flask web server, and design of the HTML/JavaScript frontend user interface.

8. References

1. Vi "Hearable" fitness tracker. link to forum post
2. Wu, Ze & Zhang, Jiwen & Chen, Ken & Fu, Chenglong. (2019). Yoga Posture Recognition and Quantitative Evaluation with Wearable Sensors Based on Two-Stage Classifier and Prior Bayesian Network. Sensors. 19. 5129. 10.3390/s19235129 - link to paper.
3. Front. Neurosci., 27 May 2016, Sec. Neuroprosthetics, Volume 10 - 2016 , link to paper
4. SparkFun ADXL337 3-axis accelerometer. link to reference
5. Adafruit BNO055 Absolute Orientation Sensor. link to reference
6. Bird dog exercise. link to explanation
7. Flask Web Framework. website
8. TP-link travel router model TL-WR902AC. link to website
9. Adafruit BNO055 library. arduino docs

9. Code Appendix

Example 1: ADXL337 EMA Filtering on Teensy

```
static inline float exponentialFilter(float previous, float current,
float alpha) {
    return previous * (1.0f - alpha) + current * alpha;
}

// 1. Rilevamento movimento per filtraggio adattivo
isMoving = detectMovement(accelX, accelY, accelZ);
float dynamicAlpha = isMoving ? (ALPHA_ACCEL * 2.0f) : ALPHA_ACCEL;
```

```
filteredX = exponentialFilter(filteredX, accelX, dynamicAlpha);
filteredY = exponentialFilter(filteredY, accelY, dynamicAlpha);
filteredZ = exponentialFilter(filteredZ, accelZ, dynamicAlpha);
```

Example 2: Repetition count inside main file on Teensy

```
if (inTargetEffettivo && !faseAndata && tempoPosizione >= minHoldTime) {
    faseAndata = true; // Fase di andata raggiunta e mantenuta
    DEBUG_PRINTLN("FASE ANDATA completata - target raggiunto e
mantenuto");
}

if (faseAndata && attualeInStart) {
  // Ripetizione completata: andata + ritorno
  conteggioRipetizioni++;
  faseAndata = false;

  // Reset timers per la prossima ripetizione
  tempoInizioPosizione = 0;
  tempoPosizione = 0;
  tempoUscitaTarget = 0;
....
```

Example 3: Flask Server Endpoint on Raspberry Pi

```
# SocketIO event handlers
@socketio.on('connect')
def handle_connect():
   logging.info(f"Client connected: {request.sid}")
   active_connections.add(request.sid)
   # Send current data to new client
   emit('sensor_update', sensor_data)
   emit('system_status', {'status': 'connected', 'active_connections':
len(active_connections)})

@socketio.on('disconnect')
def handle_disconnect():
   logging.info(f"Client disconnected: {request.sid}")
   active_connections.discard(request.sid)
```

```
# Routes
@app.route('/')
def index():
    return render_template('dashboard.html')

@app.route('/feedback')
def feedback():
    return render_template('feedback.html')
```