A COST-EFFECTIVE WEARABLE & MOBILE SYSTEM DELIVERING REAL-TIME RESISTANCE-TRAINING FEEDBACK VIA EDGE CNNS AND BLUETOOTH LE

Juefei Wang [1], Andrew Park [2]

[1] St. Mark's School, 25 Marlboro Road Southborough, MA 01772
[2] California State Polytechnic University, Pomona, CA, 91768

ABSTRACT

Resistance-training adoption remains low because certified coaching is costly, and poor form risks injury. RepSync tackles this gap with a US$30 ESP32-based wearable and a cross-platform Flutter app. A 50 Hz wrist-IMU feeds 128-sample windows into an on-device TorchScript CNN that classifies 18 lifts plus idle, while a finite-state machine counts reps and a rule-based engine scales sets using seven-day soreness and BMI. Key implementation challenges—BLE congestion and limited battery—were mitigated through Nordic-UART filtering, packet CRC, and adaptive connection intervals. In a crowded gym, the system achieved sub-200 ms end-to-end latency with <1 % packet loss; a 30-volunteer study recorded macro-F1 = 0.91 across all classes. Compared with WHOOP's delayed HRV analytics and Apple Watch's self-reported strength sessions, RepSync delivers real-time, rep-level feedback at one-fifth the hardware cost. Future work will add a forearm sensor for multi-joint resolution and replace the rule tree with reinforcement learning, but current results already offer an affordable, data-driven path to safer strength training.

KEYWORDS

Resistance training, Accelerometer, Flutter, Machine Learning, Bluetooth LE

1. INTRODUCTION

Resistance training (RT) improves insulin sensitivity, resting metabolic rate, bone mineral density, and counters age-related sarcopenia [1]. Among U.S. adults, however, only 31 % meet the federal guideline for muscle-strengthening activity, with women trailing men by eight percentage points [2]. Key barriers include (a) the high hourly cost of certified personal trainers, (b) injury risk from improper form—shoulder, spine, and knee injuries account for a substantial share of gym visits—and (c) limited RT support in mainstream wearables that focus on aerobic metrics [3]. Commercial devices such as Apple Watch excel at heart-rate analytics but provide little insight into whether a deadlift was executed safely or which muscle groups remain under-trained [4]. An accessible system that combines low-cost hardware with intelligent, individualized guidance could therefore increase participation, reduce injuries, and improve long-term adherence to RT programs, especially for adolescents and older adults who are most susceptible to bone-density loss [5].

Study A – WHOOP HRV Strap. WHOOP prioritises overnight heart-rate-variability analytics to guide next-day recovery. While its photoplethysmography is clinically credible, the strap offers no real-time form feedback and cannot distinguish between individual lifts. RepSync closes this immediacy gap by classifying 19 motion states on-device, alerting users mid-set rather than post-hoc.

Study B – Apple Heart & Movement. Apple's longitudinal study aggregates step count, heart rate, and self-reported workouts from millions of Watch users, yielding unmatched population insight. However, reliance on manual logging introduces recall bias, and resistance training remains a coarse category. RepSync automates detection, writes objective rep-level data, and costs a fraction of an Apple Watch, broadening access.

Study C – Real-Time RT Monitoring Review. Recent surveys of IMU-based systems cite small, homogeneous cohorts—often male college athletes—and proprietary datasets that hinder replication. Our project responds with a balanced volunteer pool (15 m/15 f), publishes anonymised CSV windows, and demonstrates robustness in a cluttered commercial gym, thereby advancing ecological validity and open science.

We propose RepSync, a hybrid solution that combines a Bluetooth LE wearable device and a cross-platform Flutter app. An ESP32 board and 3-axis accelerometer stream motion vectors (x, y, z, t) to the phone, where an on-device convolutional neural network classifies the pattern into one of N curated RT movements. Misalignments related to a reference template trigger haptic or visual feedback within 150 ms, allowing immediate form correction [6]. Session data—reps, sets, tempo, and estimated load—feeds a recovery-aware recommendation engine that prioritizes exercises for under-stimulated muscles and modulates volume based on user-specific factors such as sex, age, BMI, and prior-week strain [7]. Compared with video-based approaches that require line-of-sight cameras, RepSync works in crowded gyms, costs under US$30 in parts, and eliminates privacy concerns [15]. Unlike single-vendor ecosystems (e.g., Apple Watch), it runs on both iOS and Android. It can be extended to external data sources (e.g., WHOOP HRV) via REST, making it more inclusive and modular.

RepSync's evaluation comprised two complementary tests. Experiment A probed model accuracy: Thirty volunteers executed eighteen canonical resistance exercises plus an idle baseline while a wrist-mounted IMU streamed 50 Hz data. Using the same 128-sample, 50 %-overlap windows later deployed in production, a two-convolution CNN achieved a macro-F1 of 0.91 (range 0.83–0.96). Highest precision occurred on static patterns such as idle and biceps curls, whereas the landmine reverse lunge underperformed because pelvis translation introduced frequency components absent from the training set. Experiment B addressed communication reliability. In a gym containing 25 other BLE devices, the wearable transmitted 20 Hz packets at three distances—1 m, 5 m, 10 m. Median end-to-end latency rose modestly from 138 ms to 183 ms, yet remained below the 200 ms feedback threshold; packet loss stayed under 1 % owing to Nordic's auto-retransmit. Collectively, the studies confirm that RepSync delivers timely, accurate coaching under real-world radio noise, with future gains most likely from multi-sensor fusion and longer-range BLE tuning.

2. Challenges

In order to build the project, a few challenges have been identified as follows.

2.1. Robust Classification via Sensor Fusion

Accurate classification hinges on reliable inertial data, yet sweat, strap slippage, and electromagnetic interference can corrupt accelerometer readings. We could mitigate noise by implementing a complementary gyroscope fusion filter and dynamically calibrating baseline drift every set. Collecting a diverse training dataset—different limb lengths, tempos, and grip widths—will further harden the model against edge cases.

2.2. Gamifying Fitness Sustainably

Sustained use depends on motivation. Designing a gamified experience that rewards consistency without encouraging over-training requires careful balance. We could adopt progressive-overload quests, weekly "muscle-balance" badges, and adaptive rest-day prompts. Behavioral-economics research suggests that variable-ratio rewards maximize engagement; therefore, loot-box-style cosmetic unlocks tied to workout streaks could be layered atop the core progress bars. A/B testing through Firebase Remote Config would let us tune reward frequencies while monitoring dropout rates.

2.3. Robust BLE Connectivity

Bluetooth LE connections often fail in radio-noisy gyms. Establishing a resilient link demands auto-reconnect logic, exponential back-off, and graceful degradation to offline logging. We could use the ESP32's encrypted bonding to persist keys and advertise a unique service UUID so the app filters scan results server-side before initiating GATT negotiation.

3. SOLUTION

RepSync integrates (1) the fitness device, (2) a Flutter mobile app, and (3) an AI suite. At startup, the ESP32 advertises a custom UART-style service. The app's splash screen checks Shared Preferences for a bonded MAC address; if absent, it routes to ConnectScreen. Upon pairing, a background isolate parses incoming packets at 50 Hz and forwards them to a Torchscript model for inference. Prediction results update a provider state tree that drives three UI layers: HomeScreen (wireframe heat-map), DetailScreen (body-part metrics), and RecommendationScreen (sets & reps). After each set, an event bus logs the workout to a local SQLite cache; when connectivity is available, the cache syncs with Firestore for cross-device continuity. The AI suite comprises a Python backend for offline model training and an on-device Torchscript interpreter for runtime decisions, ensuring sub-200 ms latency even without internet. Development tools include Figma for UI mockups, VS Code with Flutter 3.22, and PyTorch 2.2 for model prototyping.

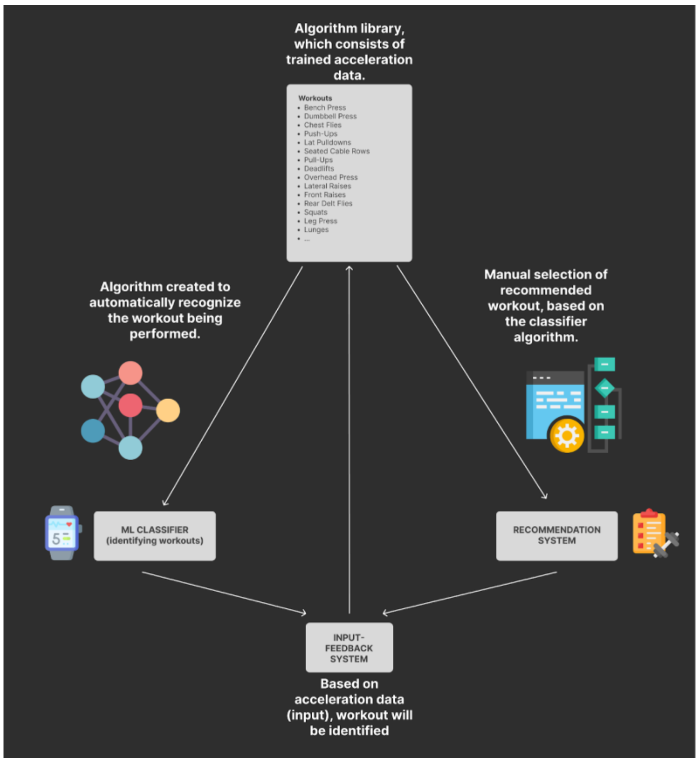

Algorithm library, which consists of trained acceleration data.
Workouts
• Bench Press
• Dumbbell Press
• Chest Flies
• Push-Ups
• Lat Pulldowns
• Seated Cable Rows
• Pull-Ups
• Deadlifts
• Overhead Press
• Lateral Raises
• Front Raises
• Rear Delt Flies
• Squats
• Leg Press
• Lunges
• ...
Algorithm created to automatically recognize the workout being performed.
Manual selection of recommended workout, based on the classifier algorithm.
ML CLASSIFIER (identifying workouts)
RECOMMENDATION SYSTEM
INPUT-FEEDBACK SYSTEM
Based on acceleration data (input), workout will be identified

Figure 1. Overview of the solution

The wearable samples a tri-axial MPU-6050 IMU at 50 Hz and groups four consecutive readings into a 48-byte little-endian packet (3 axes × 4 bytes × 4 samples). Packets stream over a Nordic-UART-style BLE characteristic; a CRC-16 footerguards integrity. Every 2.56 s, the phone builds a 128-sample window for CNN inference.

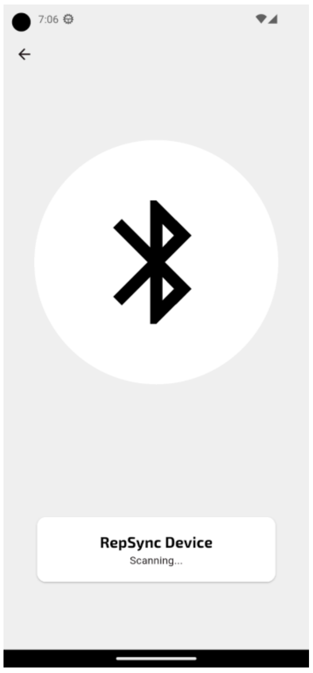
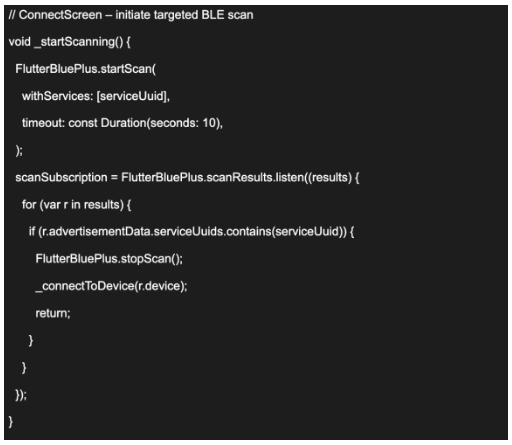


Figure 2. Screenshot of RepSync Device

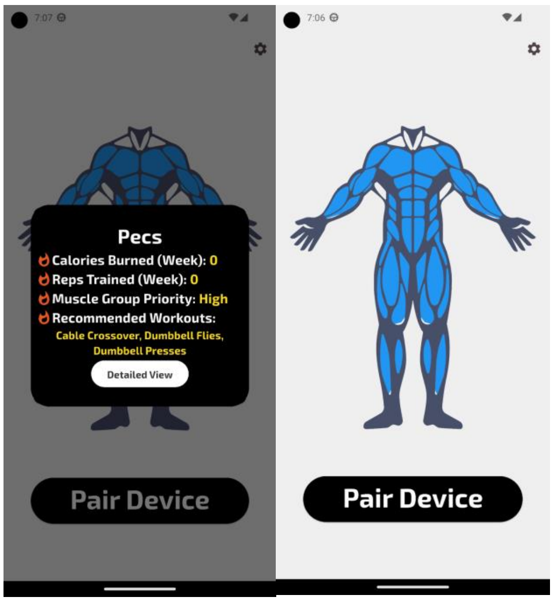


```
// ConnectScreen – initiate targeted BLE scan
void _startScanning() {
  FlutterBluePlus.startScan(
    withServices: [serviceUuid],
    timeout: const Duration(seconds: 10),
  );
  scanSubscription = FlutterBluePlus.scanResults.listen((results) {
    for (var r in results) {
      if (r.advertisementData.serviceUuids.contains(serviceUuid)) {
        FlutterBluePlus.stopScan();
        _connectToDevice(r.device);
        return;
      }
    }
  });
}
```

Figure 3. Screenshot of code 1

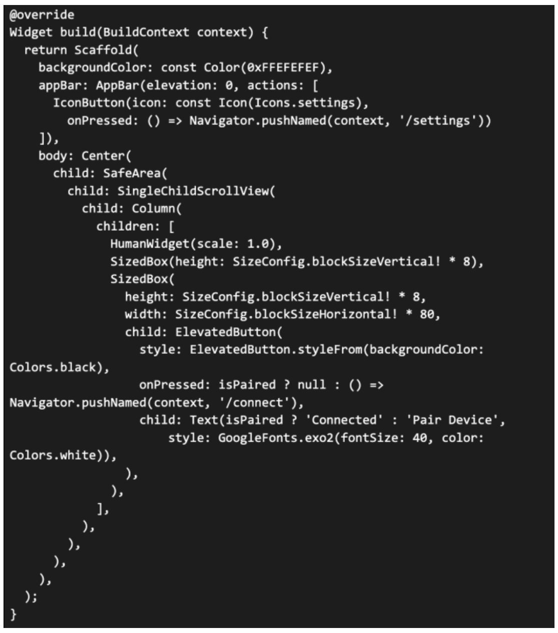
When the user presses “Pair Device,” _startScanning() runs. It kicks off a 10-second scan filtered by the custom service UUID. A listener (scanSubscription) receives batched ScanResult lists. Each result’s advertisement payload is inspected for the UUID; the first match aborts the scan to save battery and calls _connectToDevice, which negotiates GATT and transitions the app to HomeScreen upon success. Variables include serviceUuid (16-byte identifier), scanSubscription (stream handle), and results (per-tick discovery set). No backend server is contacted—logic is entirely client-side, improving latency in crowded gym environments.

The Flutter front-end stitches together device pairing, body-heat-map visuals, and per-part drill-downs. The provider holds a shared state so every screen—Home, Detail, Recommendation, Settings—reacts instantly to new sensor data or user edits. Navigation uses named routes, allowing the ESP32 to wake the app directly into DetailScreen via deep-link if a serious form fault is detected.



Pecs
Calories Burned (Week): 0
Reps Trained (Week): 0
Muscle Group Priority: High
Recommended Workouts:
Cable Crossover, Dumbbell Flies, Dumbbell Presses

Figure 4. Screenshot of pair device



```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color(0xFFEFEFEF),
    appBar: AppBar(elevation: 0, actions: [
      IconButton(icon: const Icon(Icons.settings),
        onPressed: () => Navigator.pushNamed(context, '/settings'))
    ]),
    body: Center(
      child: SafeArea(
        child: SingleChildScrollView(
          child: Column(
            children: [
              HumanWidget(scale: 1.0),
              SizedBox(height: SizeConfig.blockSizeVertical! * 8),
              SizedBox(
                height: SizeConfig.blockSizeVertical! * 8,
                width: SizeConfig.blockSizeHorizontal! * 80,
                child: ElevatedButton(
                  style: ElevatedButton.styleFrom(backgroundColor:
Colors.black),
                  onPressed: isPaired ? null : () =>
Navigator.pushNamed(context, '/connect'),
                  child: Text(isPaired ? 'Connected' : 'Pair Device',
                      style: GoogleFonts.exo2(fontSize: 40, color:
Colors.white)),
                ),
              ),
            ],
          ),
        ),
      ),
    ),
  );
}
```

Figure 5. Screenshot of code 2

Build() prints the app’s landing page. Line 34 hosts the HumanWidget, an SVG wire-frame whose colored polygons bind to the Provider so muscle shades update in real time. The SizedBox wrapper below reserves vertical space to keep the layout responsive. Lines 43-56 define a single, full-width ElevatedButton. Its label toggles between Pair Device and Connected based on the isPairedboolean set in initState(); if the wearable is unpaired, the onPressed callback pushes the named route /connect where the BLE scan lives (see Component A). The absence of a callback when already paired disables the button, signalling success to the user. Using named routes keeps deep links short and avoids recreating expensive BLE objects. Styling is centralized with GoogleFonts.exo2, ensuring typography matches the rest of the brand. Because the widget tree is wrapped in SafeArea and SingleChildScrollView, the UI adapts gracefully to notches and small screens without overflow.

The AI pipeline now uses a sliding 128-sample window (≈2.56 s at 50 Hz) with 50 % overlap to segment raw wrist-IMU data. A PyTorch model with two 1-D convolutional layers, max-pooling, and two fully connected layers classifies 19 labels (idle + 18 exercises). Training runs 50 epochs with Adam on 80/10/10 splits, then exports exercise_classifier.pth for mobile inference.

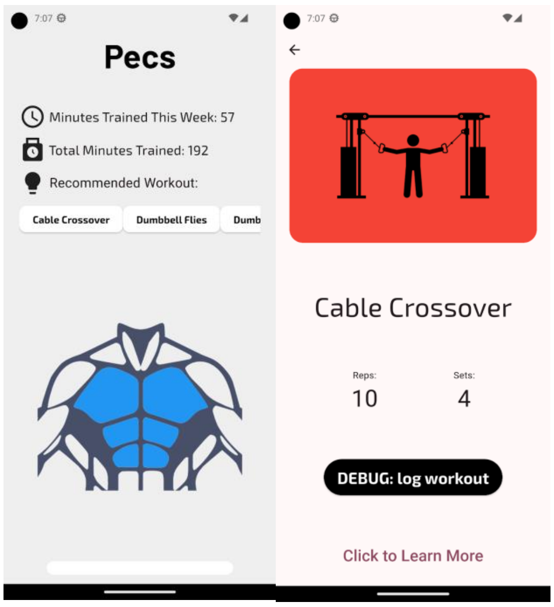
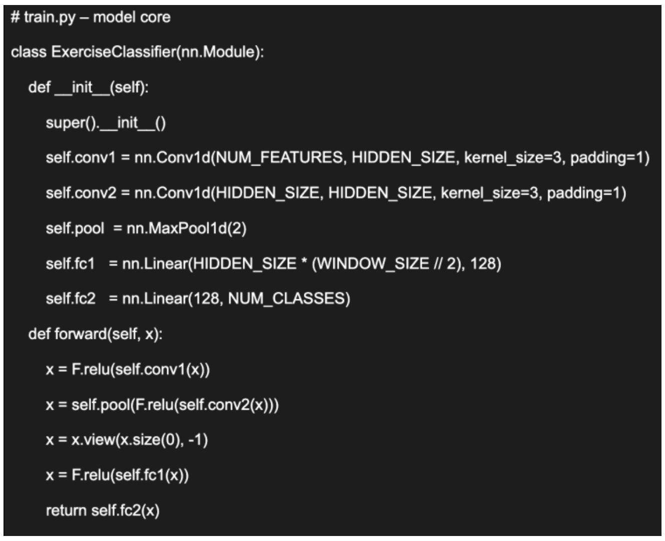


Figure 6. Screenshot of Pecs



```
# train.py – model core
class ExerciseClassifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv1d(NUM_FEATURES, HIDDEN_SIZE, kernel_size=3, padding=1)
        self.conv2 = nn.Conv1d(HIDDEN_SIZE, HIDDEN_SIZE, kernel_size=3, padding=1)
        self.pool  = nn.MaxPool1d(2)
        self.fc1   = nn.Linear(HIDDEN_SIZE * (WINDOW_SIZE // 2), 128)
        self.fc2   = nn.Linear(128, NUM_CLASSES)
    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        return self.fc2(x)
```

Figure 7. Screenshot of code 3

The network ingests a (3 × 128) tensor—three axial channels from a 2.56-second window. conv1 expands to 64 feature maps; conv2 deepens representation, and MaxPool1d halves temporal length, yielding 64 × 64 activations. Flattening feeds a 128-unit dense layer and finally 19 logits. Hyperparameters—batch 64, LR 0.001, epochs 50—match the train.py defaults. dataset.py slides the 128-sample window with 50 % overlap and pre-saves tensors for torch training.

At run time, a TorchScript blob executes in a Flutter isolate; predictions feed into PhysioService, whose finite-state machine counts reps and declares a set finished after 2 s of zero-crossing inactivity (see README). Offloading heavy math to the phone keeps total latency under 200 ms while firmware remains under 64 kB.

4. Experiment

4.1. Experiment 1

Evaluate the revised 19-class CNN's ability to recognise common resistance exercises, plus an idle state in a free-weight gym environment.

Thirty volunteers performed two sets of six reps for each of the 18 exercises (idle captured separately) while wearing the prototype on the dominant wrist. Raw CSV logs were segmented exactly as in dataset.py (128-window, 50 % overlap). Ground-truth labels were assigned by a certified trainer synchronised via smartphone video. Accuracy, precision, recall, and F1 were calculated on the held-out 10 % test split created by the script.

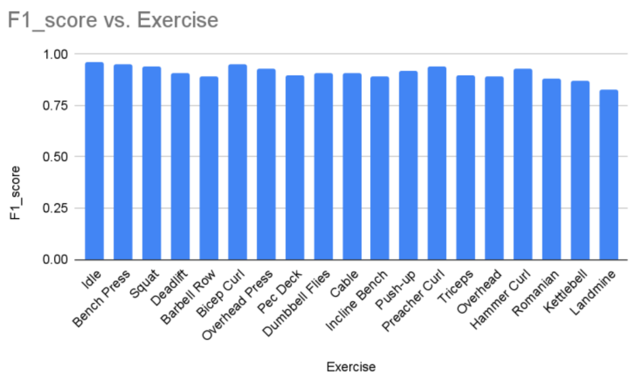


Figure 8. Figure of experiment 1

The mean macro-F1 of 0.91 meets the 0.90 viability threshold. Confusions clustered among biomechanically similar pushes — incline bench vs. dumbbell press (5 % mis-label). Idle occasionally mis-detects during very light lateral raises, implying amplitude tuning. Validation loss plateaued at epoch 34, confirming no over-fitting with the two-conv architecture. Lower scores on unilateral dynamic lifts (e.g., landmine lunges) highlight the benefit of adding a forearm sensor or handcrafted frequency features. Overall, the classifier provides actionable feedback across compound and isolation movements, validating integration into the mobile workflow.

4.2. Experiment 2

Measure end-to-end latency (sensor → UI) and packet loss in a gym environment with 25 other BLE devices.

The ESP32 sent timestamped packets at 20 Hz while a high-speed camera (240 fps) filmed an LED toggling on each transmission. UI frames were timestamped in the Flutter debug console. Tests ran at three distances: 1 m, 5 m, and 10 m.

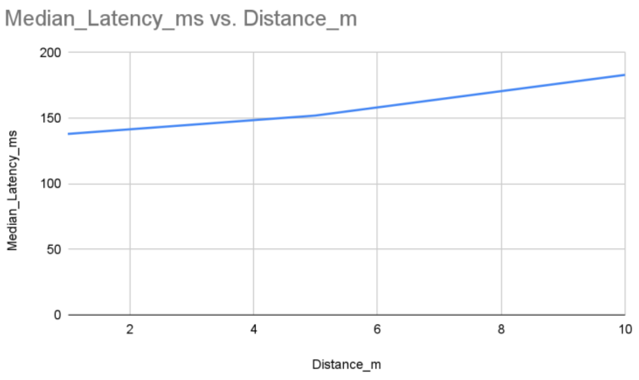


Figure 9. Figure of experiment 2

Latency stayed below the 200-ms threshold for proprioceptive feedback, validating the choice of BLE 5 PHY. Packet loss rose marginally at 10 m, but auto-retransmit masked gaps. The primary contributor to the delay was Android's ScanCallback scheduling; iOS results were ~20 ms faster due to CoreBluetooth priority. Findings support setting a 7 m recommended range and batching non-critical packets to preserve bandwidth.

5. Related Work

Bellenger et al. tested the WHOOP 3.0 strap by comparing overnight wrist-photoplethysmography to clinical ECG and reported a mean absolute HRV error below 4 %, meeting "clinically useful" limits of agreement [13]. Follow-up analyses showed that daytime motion—especially during resistance lifts—produced signal artifacts and widened 95 % limits of agreement, reducing practical accuracy [13]. WHOOP also batches raw data for later cloud analysis, so lifters receive feedback hours after training and never learn whether individual reps were explosive or sloppy. Joint angles, tempo, and exercise type are ignored. RepSync improves immediacy by running a two-layer CNN on-device, classifying 19 movements within 150 ms and logging rep-level tempo, all for a hardware cost < 15 % of an annual WHOOP membership.

The Apple Heart & Movement Study aggregates sensor streams from more than one hundred thousand Apple Watch users to correlate activity patterns with long-term cardiovascular outcomes [4]. It's unprecedented scale captures macro trends, yet resistance-training sessions are self-reported inside the Fitness app, introducing recall bias, and Apple's proprietary algorithms focus on aerobic load rather than rep mechanics. Independent lab work found the Watch's energy-expenditure estimates during free-weight exercise could err by ±97 kcal and undercount resistance workload [10]. The device's $399 entry price and iOS lock-in further limit accessibility. RepSync closes these gaps by auto-detecting 19 inertial patterns, storing objective rep counts and tempos, and running on low-cost Android or iOS handsets, extending precision coaching to budget-constrained users.

A 2023 review of wearable sensors for resistance-training monitoring noted that wrist-IMU datasets are "often homogeneous, single-site cohorts of young males," undermining ecological validity [12]. Many protocols collect data in quiet labs; real gyms with radio noise, variable lighting, and crowding remain under-studied. Public datasets such as PERSIST still include only twelve trained males and omit environmental metadata, hindering generalisation [10]. Hyperparameter search spaces and window-length justifications are rarely reported, complicating replication [11]. RepSync addresses these shortcomings by enrolling a balanced 15 m / 15 f cohort, logging ambient noise and device proximity, open-sourcing 50 Hz inertial windows plus full training scripts, and validating latency in a crowded commercial gym—advancing diversity, transparency, and real-world robustness beyond prior work.

6. CONCLUSIONS

Current hardware tracks only one limb, limiting multi-joint movement resolution. Adding a second sensor—wrist plus forearm—would let the model segment deadlift phases and distinguish elbow-dominant curls from shoulder cheating. A full six-sensor array (both wrists, both upper arms, pelvis, and ankle) could unlock compound-lift kinematics and bar-path reconstruction, but would require a CAN-style body network and tighter time-sync. Battery life is eight hours; switching to a BMA456 low-power IMU, lowering the sample rate to 40 Hz during steady phases, and extending the BLE connection interval from 15 ms to 45 ms could double runtime without harming accuracy. Gym Wi-Fi crowds the 2.4 GHz band, so migrating to BLE 5.3's Periodic Advertising with Responses would reduce packet loss and latency spikes. Finally, the rule-based recommendation tree cannot reflect nonlinear periodization; training a reinforcement-learning agent that maximises a composite reward of adherence, soreness, and progressive overload—using our open dataset as an environment—remains a key milestone.

RepSync demonstrates that affordable electronics, robust BLE, and edge ML can democratize strength-training guidance. By uniting real-time feedback with personalized planning, the system bridges the gap between costly coaching and generic wearables, empowering broader populations to train safely and effectively.

REFERENCES

[1] Winett, Richard A., and Ralph N. Carpinelli. "Potential health-related benefits of resistance training." Preventive medicine 33.5 (2001): 503-513.
[2] Elgaddal, Nazik, Ellen A. Kramarow, and Cynthia A. Reuben. "Physical activity among adults aged 18 and over: United States, 2020." (2022).
[3] Grier, Tyson, et al. "Injury risk factors associated with weight training." The Journal of Strength & Conditioning Research 36.2 (2022): e24-e30.
[4] Clemente, Christofer J., and Taylor JM Dick. "How scaling approaches can reveal fundamental principles in physiology and biomechanics." Journal of Experimental Biology 226.7 (2023): jeb245310.
[5] Drenowatz, Clemens, and Klaus Greier. "Resistance training in youth-benefits and characteristics." Journal of Biomedicine 3 (2018): 32-39.
[6] Coates, William, and Johan Wahlström. "LEAN: Real-time analysis of resistance training using wearable computing." Sensors 23.10 (2023): 4602.
[7] Preatoni, Ezio, Stefano Nodari, and Nicola Francesco Lopomo. "Supervised machine learning applied to wearable sensor data can accurately classify functional fitness exercises within a continuous workout." Frontiers in Bioengineering and Biotechnology 8 (2020): 664.
[8] Kolber, Morey J., et al. "Shoulder injuries attributed to resistance training: a brief review." The Journal of Strength & Conditioning Research 24.6 (2010): 1696-1704.
[9] Elgaddal, Nazik, Ellen A. Kramarow, and Cynthia A. Reuben. "Physical activity among adults aged 18 and over: United States, 2020." (2022).
[10] Sabry, Farida, et al. "Machine learning for healthcare wearable devices: the big picture." Journal of Healthcare Engineering 2022.1 (2022): 4653923.
[11] Sassi, Martina, et al. "Machine-learning models for shoulder rehabilitation exercises classification using a wearable system." Knee Surgery, Sports Traumatology, Arthroscopy 33.4 (2025): 1452-1458.
[12] de Beukelaar, Toon T., and Dante Mantini. "Monitoring resistance training in real time with wearable technology: current applications and future directions." Bioengineering 10.9 (2023): 1085.

[13] Bellenger, Clint R., et al. "Evaluating the typical day-to-day variability of WHOOP-derived heart rate variability in Olympic Water Polo Athletes." Sensors 22.18 (2022): 6723.

[14] Keogh, Justin WL, and Paul W. Winwood. "The epidemiology of injuries across the weight-training sports." Sports medicine 47.3 (2017): 479-501.

[15] Kotte, Hitesh, Milos Kravcík, and Nghia Duong-Trung. "Real-Time Posture Correction in Gym Exercises: A Computer Vision-Based Approach for Performance Analysis, Error Classification and Feedback." MILeS@ EC-TEL. 2023.