SAFE Personal Trainer - Final Report

SAMUEL SHMIDMAN, Brandeis University, USA
ABBIE MURPHY, Brandeis University, USA
FACUNDO ROITMAN, Brandeis University, USA
ETHAN KONDEV, Brandeis University, USA

ACM Reference Format:
Samuel Shmidman, Abbie Murphy, Facundo Roitman, and Ethan Kondev. 2018. SAFE Personal Trainer - Final Report. *Proc. ACM Meas. Anal. Comput. Syst.* 37, 4, Article 111 (August 2018), 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Whether you are stepping into the gym for the first time or you're a seasoned athlete, proper form is crucial. It can mean the difference between progress and injury. While personal trainers provide valuable guidance, they can be costly and are not always available for every workout. This can leave beginners feeling lost and experienced lifters open to injury. To bridge this gap, we are proposing a pair of smart devices designed to ensure safe movement, track workouts, and detect potential dangers– automatically alerting support when needed: our SAFE Personal Trainer system is composed of two portable components, the SAFE Personal Trainer Hub and the SAFE Personal Trainer Armband.

The SAFE Personal Trainer system seeks to democratize access to personal fitness training by making it more affordable, accessible, and accurate than ever before. Our technology aims to empower individuals to exercise correctly and safely, whether in a gym environment or from the comfort of their homes. By leveraging state-of-the-art sensors and real-time data analytics, the SAFE Personal Trainer system not only guides users through their workouts but also monitors their progress and adjusts their exercise plans dynamically. This approach ensures that each workout is optimized for effectiveness and safety, making personal training accessible to a broader audience at a fraction of the cost of traditional services.

Ultimately, our project is about more than just physical fitness; it's about fostering a sense of confidence and well-being that comes from knowing you're working out correctly and safely. With the SAFE Personal Trainer, users can focus on what truly matters: their health and fitness goals.

Authors' addresses: Samuel Shmidman, Brandeis University, Waltham, Massachusetts, USA, samuelshmidman@brandeis.edu; Abbie Murphy, Brandeis University, Waltham, Massachusetts, USA, abigailmurphy@brandeis.edu; Facundo Roitman, Brandeis University, Waltham, Massachusetts, USA, facundoroitman@brandeis.edu; Ethan Kondev, Brandeis University, Waltham, Massachusetts, USA, ethankondev@brandeis.edu.

ACM 2476-1249/2018/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

2 DESCRIPTION

2.1 Purpose

When establishing a fitness goal, we find that the biggest barrier to success has to do with knowledge: How can one be sure to follow a correct and healthy exercise routine? Considering the muscular strain that weighted physical activity can bring to one's health–not to mention the risks associated with weightlifting accidents–it is our project's priority to develop a device that can guarantee users to train any muscle group successfully. The SAFE Personal Trainer Hub is where that improved gym experience begins. It is a pair of small devices with the ability to act as a hired personal trainer: observe a user's form and posture when they exercise, detect if said form is faulty in any way, and finally send signals with guidance as to how to correct it.

In addition, we recognize that a successful workout experience goes far beyond correctness and safety. That is why, in order to provide personalized descriptions of a user's performance at the gym, we include in our SAFE Personal Trainer system the SAFE Personal Trainer Armband, a sister device to The Hub. The Armband's purpose is to get further information from the user at the time of the workout, ensuring that the advice provided is customized to their needs. By measuring exertion and intensity, as well as counting repetitions to display progress over time, the Armband will allow the user to focus entirely on their exercise, ensuring that their progress is being tracked and recorded and their health is in optimal condition at all times.

2.2 Usage and Design

Both devices are designed to be portable and easy to set up. When the user powers on The Hub, they'll see options on the screen to either start a new workout or view previous workouts. Selecting "Start New Workout" brings them to a setup page where they can customize their session by choosing the exercise and desired workout duration. Once selections are made, the user hits "Start," prompting a short countdown. They then step back and begin their workout.

During the session, a live video window lets the user see themselves to ensure they're in frame. Next to this video, real-time stats such as heart rate and rep count are displayed, collected via the SAFE armband and relayed to the application using Bluetooth. As the workout nears completion, the countdown clock turns red as a visual warning. Users can choose to end the session early by tapping "End Workout," or simply wait for the time to run out.

After finishing, users are prompted to name their workout; if skipped, it will default to the date and time. A brief loading period follows as the pose estimation model analyzes the video. Once analysis is complete, feedback on form and stats from the SAFE armband are saved. At this point, users can choose to immediately watch their analyzed workout, which includes on-screen feedback and a pose-estimation skeleton overlay. If they prefer to wait, they can return to the home screen and access all saved sessions under "View Previous Workouts," where they can watch either the raw or analyzed video and review their stats.

2.3 Device List: SAFE Personal Trainer Hub

- **Raspberry Pi** - The device responsible for processing and maintaining memory for the SAFE Hub, collecting input from the camera, and outputting video onto the screen. The Raspberry Pi is also where the recorded video will be processed using the pose estimation model. It also connects to the Armband through Bluetooth to collect data from the accelerometer and heart-rate monitor. Finally, the Raspberry Pi is responsible for hosting and running the GUI (a python web app) that allows for a user-friendly reflection of this data and these videos.
- **Screen** - At boot, displays the GUI for the user to be guided through the workout process. Users can interact with the GUI through the touch-screen capabilities. Displays the live video and other workout

stats during a session, and allows the user to view previous workouts and their form analysis. Connected to the Raspberry Pi using:
  – HDMI cable - High quality visual data.
  – Micro-USB connection cable - Power to the screen.
  – GPIO Connction cable - Enable touch-screen and other data to be transfered back to Raspberry pi.
- **Camera** - Captures movement and image data to analyze form and detect risks. Implemented with PicamZero, the camera is started after the countdown step on the GUI. It simultaneously feeds the video stream to the GUI for the user to view during a workout, and saves the video to the file specific to the workout. This saved video is used for the pose analysis. Connected with a 15- pin CSI ribbon cable.
- **Enclosure** - A 3-D printed enclosure that can be placed on the floor or a lower surface. Tilts the screen upwards to be more visible by the user, and points the camera towards the user.

2.4 Device List: SAFE Personal Trainer Armband

- **Raspberry Pi Pico** - Collect data from the other devices on the armband and sending data to the Raspberry Pi and GUI. All communication with data collection devices is through wires, and data sent to the Raspberry Pi is transferred through Bluetooth.
- **Heart-Rate Monitor** - Monitors heart-rate to be sent to Raspberry Pi and displayed on the screen as well as saved as workout stats. connected to the pico with a 4-pin wire.
- **Accelerometer** - Used to detect movement and count repetitions during the workout. Also sent to the Raspberry Pi to be integrated into the GUI and workout data. Connected to the Pico with a 4-pin wire.
- **Button** - Allows you to start and stop the workout from far away (optimization for pose estimation model). Connected to the Pico with a 4-pin wire.

3 SCHEMATICS

3.1 The Hub

The SAFE Personal Trainer Hub is used to display the GUI to users as well as capture video for analysis. A Raspberry Pi is at the heart of the hub, visualized in figure 1. The camera is connecting using a 15-Pin CSI ribbon cable which is clipped into the Pi's camera connector (top device). The screen is connected via HDMI cable to carry the live video feed as well as GUI visuals to the screen. A micro-USB also plugs into the Pi's 5v power rail to run its backlight and logic. Finally, a 40 pin header is used for additional power and to relay touch-screen information.

3.2 Armband

The SAFE Personal Trainer Armband is responsible for collecting key biometric and motion data during workouts. It uses a Raspberry Pi Pico mounted on a Seeed Studio Grove Shield to simplify sensor connections. As shown in figure 2, the heart rate sensor is connected to analog port A1 using a 4-pin Grove wire, while the gyroscope and accelerometer module is connected to I2C0. A physical button for user input is wired to digital port D20. The Grove Shield standardizes power, ground, and data lines, enabling consistent communication across all connected modules. This configuration allows the Armband to reliably transmit data to the Hub via Bluetooth for real-time and post-workout analysis.

3.3 Form Analysis Model

The form analysis model is split into two main phases: the pose estimation, and form evaluation.

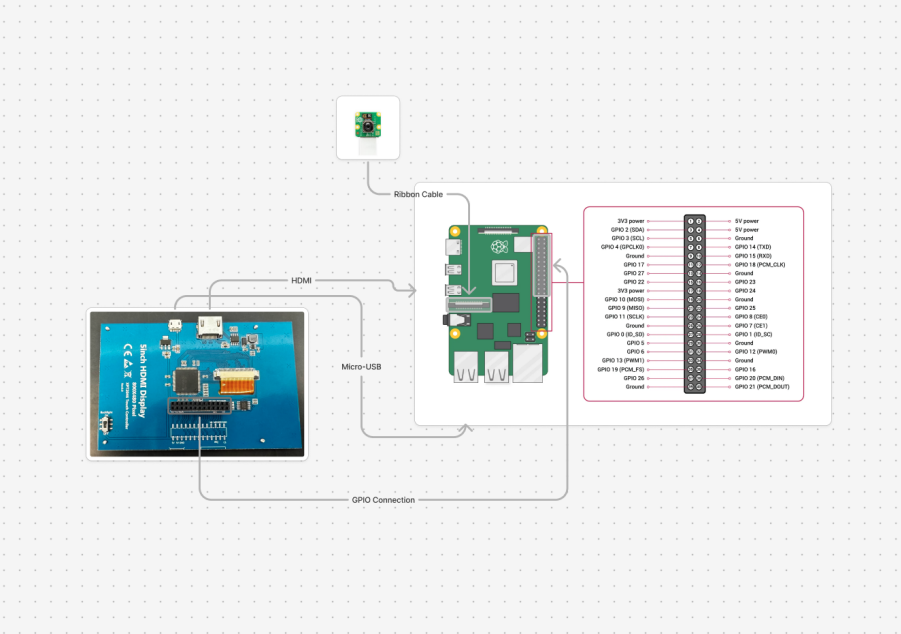

Fig. 1. Schematics for the SAFE Personal Trainer Hub.

3.3.1 Pose Estimation Model. We chose to use a pre-trained model from Tensorflow called the MoveNet "Lightning" model. This model is a 2-D, *center-out* pose estimator that maps 17 keypoints to an image of a person (see figure 3). The output of this model is normalized x-y coordinates as well as a confidence score for each keypoint. We chose the lightning model because of the high-volume of frames that would be processed as the lightning model is more lightweight and latency-efficient than its "thunder" counterpart.

3.3.2 Form Analysis. For each frame analyzed, the form analysis occurs directly after the pose estimation for that particular frame occurs. Both exercises obtain feedback based on the current frame, and overlay the given feedback as text onto the analyzed video. However, squats and push-ups both have different metrics as well as different optimizations.

Squat Form Analysis: Squat form was evaluated on three metrics: back straightness, head position, and squat depth. Back straitness was measured with the angle between edges knee-hip and the vertical axis. Our evaluator calculates this angle, and if the user is perceived to be bending over too much in any frame, the feedback will message to "keep back straight." Head position is measured based on the angle between the edges nose-shoulder and shoulder hip. If it seems that the user is looking down too much (angle is too small in any frame throughout the workout), the user will get a message to "keep eyes looking forward". Finally, squat-depth is measured using the edges of ankle-knee and knee-hip. If the user makes it down to proper depth, the feedback will say that they hit good depth for that rep. The challenge for squat depth is that not every frame can indicate if the correct depth is being met. We had to choose the correct frame to analyze for this. To do so, we kept additional variables to hold the previous squat angle, meaning if the angle previously is larger than before (with some buffer), it is assumed that the user is moving upwards in their repetition, and the previous angle should be used to determine if proper depth was hit. In order to smooth out the estimation and account further for variations between frames in the pose estimation, the

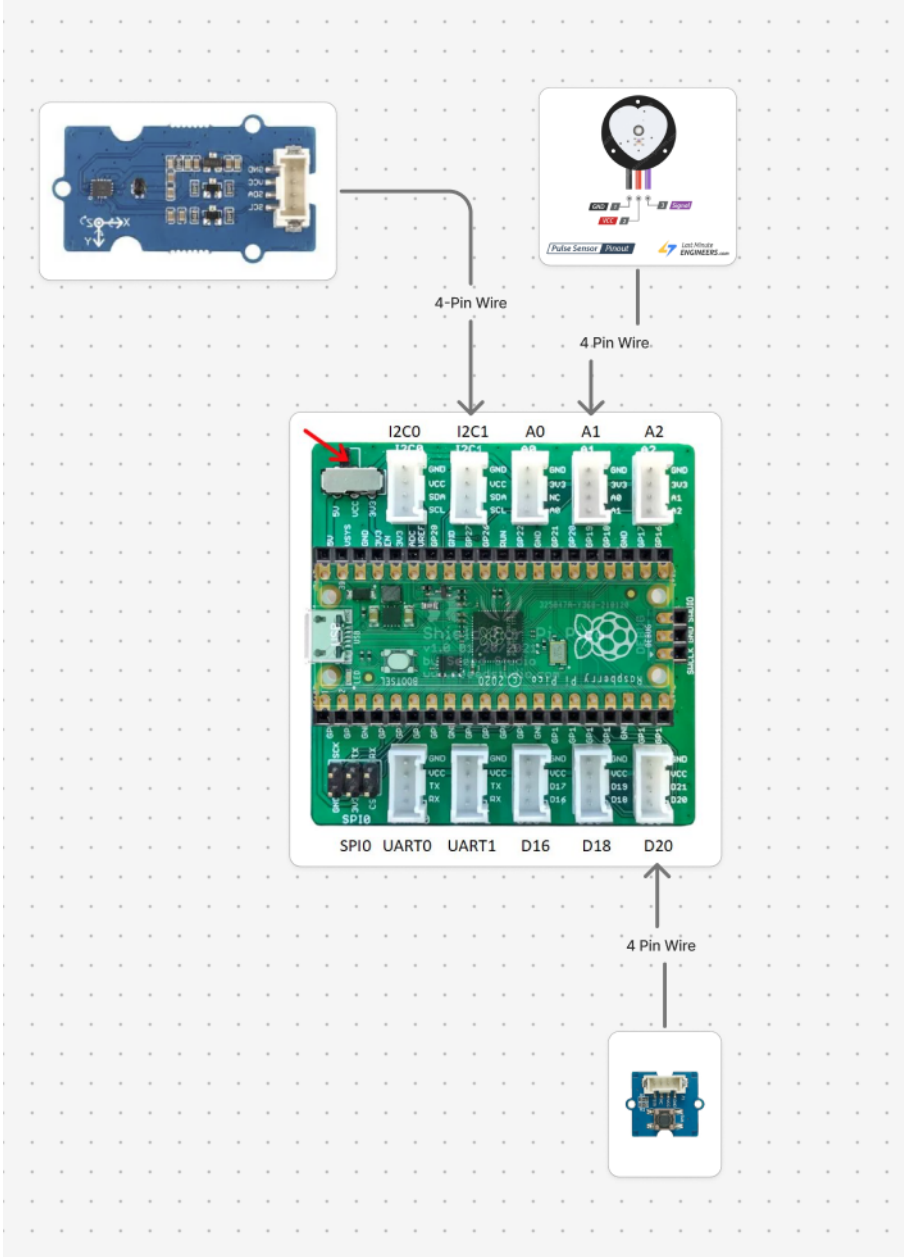

Fig. 2. SAFE Personal Trainer Armband Schematics

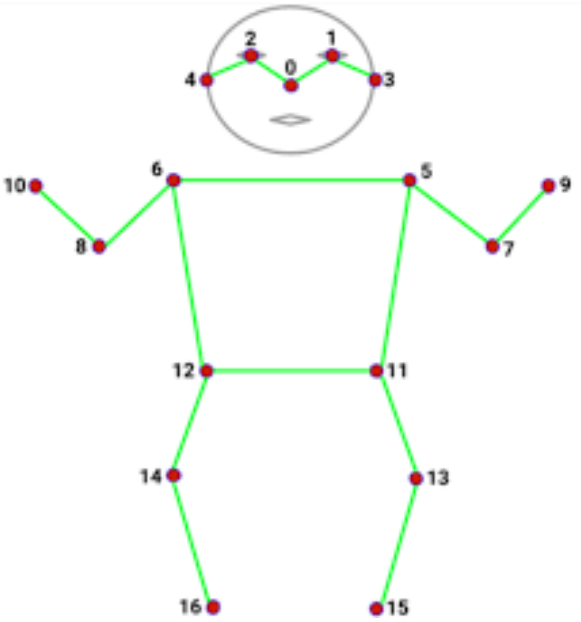

Fig. 3. The 17 key-points mapped by Movenet Pose Estimation Model

squat model analyzes every three frames instead of every single frame. Feedback will only be changed if the repetition depth is hit for this metric.

Push-Up Form Analysis: Push-up form was evaluated on four metrics: shoulder level, elbow extension, body alignment, and elbow flare. Shoulder level was measured by the angle between the left and right shoulder keypoints relative to the horizontal axis; shoulders should not tilt past 25°—if they do in any frame, feedback prompts "Keep your shoulders steady during push-ups!" otherwise "Good control over your shoulders!". Elbow extension is measured as the angle at the elbow joint (shoulder→elbow→wrist). At the top of the rep this angle must reach at least 80° (to approximate a full 90° extension)—if not, feedback says "Extend your arms fully at the top!" otherwise "Good full extension!". Body alignment was computed via the angle between shoulder and hip keypoints relative to the horizontal. It must remain under 10° of failure unless you will see, "Keep your core engaged and maintain a straight body!" as opposed to, "Great body alignment!". Elbow flare was quantified by the normalized horizontal offset of the elbow from the body's midline (elbow offset ratio). This ratio should stay below 0.3—if it exceeds this threshold, feedback reads "Tuck your elbows in closer to your body!" otherwise "Good elbow position!". To reduce frame-to-frame noise, we analyze every three frames and use each rep's extreme (max or min) angle to decide feedback.

3.4 GUI

The web-application was built in Python using the Tkinter GUI toolkit to create windows, buttons, and canvas elements, and OpenCV (cv2) together with PIL's Image and ImageTk modules to capture, process, and display live video frames from the PicamZero. Threading is used to run the continuous video-capture loop without freezing the interface, while the `after()` method schedules regular screen updates. User interactions—starting/stopping recording, saving workouts, naming files—are handled by Tkinter callbacks that accumulate frames into an

in-memory list and then write out an MP4 via OpenCV's video-writer API when the session ends. Analysis of the video is called after the session ends, and the corresponding analyzed workout video is placed in the same folder as all other workout data. File-I/O for workout metadata (timestamps, durations) is managed alongside video export so that each recording is paired with its corresponding data file. All layout, resizing, and event-binding logic lives in a single `GUI.py` module that glues together the camera, the recorder, and the Tkinter front-end. This GUI will be run on booting The Hub.

4 SOURCE CODE

Our source code can be accessed via Github through the following link:
https://github.com/abigailmurphy/SAFE-Personal-Trainer-Kit

5 CHALLENGES

In this section, we cover the key technical challenges we encountered at each phase of development, their impact on progress, and the solutions or workarounds we devised.

5.1 Migration to Raspberry Pi Environment

Challenge

Our pose-estimation and workout-analysis codebase relied on desktop-only Python packages (e.g. specific TensorFlow wheels, OpenCV builds, and custom C-extensions). On the RPi 5, many of these binaries weren't available or had mismatched architecture requirements.

Impact Initial on-device runs of our Python script aborted immediately with a confusing C-extension error:

```
ValueError: numpy.dtype size changed, may indicate binary incompatibility.
Expected 96 from C header, got 88 from PyObject
```

This wasn't just a missing import—it stemmed from a binary mismatch between NumPy 2.1.3 and the `simplejpeg` extension used by `picamera2`. Tracking it down required rebuilding key packages from source (using `pip –no-binary :all:`), repinning NumPy, and re-validating our virtual environment—costing nearly two weeks of debugging.

Resolution

- Created a minimal, reproducible `venv` with only the exact subset of packages that MoveNet and our post-processing scripts needed.
- Switched to TensorFlow Lite for Python on the Pi and rebuilt OpenCV from source with only core modules.
- Documented an RPi-specific install guide in our repo's README to streamline future setup.

5.2 Real-Time Inference Performance on Raspberry Pi 5

Challenge

Running MoveNet lightning at 480×360 resolution resulted in ~1 frame every 2 seconds, causing delayed skeleton overlays and innacurate feedback. Hardware constraints of the RPi 5 (quad-core CPU, no GPU) meant our multi-threading optimizations had diminishing returns.

Impact User experience degraded—feedback lag led to misaligned rep counts and form alerts.

Mitigation
- **Preview → analysis:** Added a preview mode allowing users to ensure they're in frame before recording workout. Recorded workouts are then passed to our analysis script for pose estimation and genarating feedback on form.
- **Pipeline optimizations**: Moved video capture and display into separate threads to avoid blocking.

Result Achieved a more stable ∼ 2 FPS end-to-end, acceptable for well informed feedback.

5.3 Memory Footprint during Post-Workout Analysis

Challenge

When analyzing longer recorded sessions (30+ seconds), our in-memory buffer of frames and keypoints caused the Pi to exhaust its 8 GB RAM and swap heavily.

Impact Crashes during end-of-session summaries; analysis jobs sometimes never completed.

Solution:
- **Writing analyzed frames directly to disk**: Switched from accumulating arrays in RAM to writing analyzed workouts to disk frame-by-frame.

Outcome Post-workout analysis now consistenly completes in under 30 seconds without swapping or OOM failures.

5.4 GUI Integration

Challenge

The GUI was originally developed and tested on laptops using built in webcams and local processing. This created issues when migrating to the Raspberry Pi. Camera control methods were tightly coupled with platform specific implementations, and several interface elements did not render properly when the window was resized on the Pi.

Impact Camera functionality simply did not work and we were unable to show any video or record anything, since the existing code was written for laptops which had built in webcam support. Additionally, because interface elements did not render properly, some buttons were not visible on our 5 inch screen, and certain components like the camera feed could take up too much space, making the GUI difficult to navigate during use and sometimes impossible.

Solution We refactored the GUI's camera logic into a standalone module with general purpose functions for capturing and saving video. This allowed the GUI to support both laptop and Raspberry Pi environments through dependency injection. For the layout issues, we modified widget placements and anchor settings to make the interface fully responsive, ensuring all interactive elements remained accessible regardless of screen size.

Outcome The final GUI runs reliably on the Raspberry Pi with a responsive layout and working camera integration. These changes allowed the system to support full workouts from start to finish, including video capture, analysis, and playback through the same unified interface.

5.5 Form Analysis Method

Challenge

After completing the pose estimation for a frame or repetition, the next step was to determine if the form of that repetition was correct. Our initial and ideal solution was to find a dataset and train

classification models based on "correct" or "incorrect" labels. However, there seemed to be no existing datasets that contained the type of data we needed. We were able to find datasets with many videos of certain exercises, but none that were labeled correctly. We were left with needing to either collect or label our own dataset, or find another method.

Impact Not using a classifier model to determine form meant we had to think of other ways to use the given pose estimations. We were looking for a method that could be flexible for different people and settings but still precise.

Solution Our solution to this was to use the angles of edges created by the pose estimation key points to determine factors about form. For example, depth is determined based on the measured angle between edge connecting the hip and knee points, and the edge connecting the ankle and knee points.

Outcome While this servers well for the scope of this project, it lacks some flexibility and nuance. People with different body compositions may require different metrics for different exercises. A person with longer legs generally appears to lean forward more than somebody with shorter legs. A classification model trained on a large dataset, and made to prevent overfitting, could help diversify our model. If we were to persue this further, using a classification model would be better to explore.

5.6 Establishing Bluetooth Communication Between Pico and RPi5

Challenge

One of the most significant technical hurdles we faced was establishing a stable Bluetooth Low Energy (BLE) connection between the Raspberry Pi Pico (Armband) and the Raspberry Pi 5 (The Hub). The SAFE Armband collects essential workout data—heart rate, repetition counts, and user button signals—which needed to be transmitted wirelessly to the Hub for display and analysis. Wi-Fi was not a viable option (due to constraints like eduroam's network limitations), making BLE the only practical solution. However, existing documentation for BLE communication between a Pico and a Pi 5 was scarce and often incomplete. We encountered numerous compatibility issues and cryptic errors when experimenting with libraries like `bleak`, `pybluez`, and the native `bluetooth` module on the Pi 5.

Impact This lack of documentation and the unstable communication attempts significantly stalled development. Without a functioning BLE link, we couldn't validate sensor data or proceed with user interface integration. Since the BLE connection formed the backbone of our system, progress in almost every other area was blocked.

Solution After extensive trial and error, we developed a custom BLE communication strategy. The Pico was programmed to advertise itself using a unique device name ("Pico") and a custom service UUID. On the Raspberry Pi 5, we used `bleak` for asynchronous scanning and connecting, specifically targeting devices broadcasting the chosen name and UUID. By writing custom BLE advertising code for the Pico and implementing robust scanning and connection logic on the Pi 5, we finally achieved reliable communication. We further debugged the connection process using raw BLE packet logging and tools like `bluetoothctl` and `nRF Connect`.

Outcome Our solution, while somewhat unconventional, resulted in a stable, repeatable BLE communication framework between the Pico and Pi 5. This enabled seamless data transfer and allowed us to build

higher-level SAFE system functionality on top of this solid foundation. With the BLE link established, users could now rely on the SAFE system for safety, efficiency, and progressive workout improvement.

5.7 Sensor Calibration and Accuracy

Challenge

When developing the SAFE Armband, we faced immediate challenges with sensor calibration. The accelerometer, critical for detecting repetition-based movement, was initially too noisy to provide reliable rep counts. Small movements were often misread as full repetitions, while legitimate reps were sometimes ignored. Simultaneously, the heart rate sensor struggled to provide consistent and accurate readings due to signal interference and improper contact with the skin. These calibration issues jeopardized the integrity of our workout data.

Impact Without reliable sensor data, our entire system risked being ineffective. Inaccurate rep counts would erode user trust, and unreliable heart rate readings could lead to unsafe workouts. Ensuring precision in both sensors was crucial to building a product that users could depend on for accurate feedback.

Solution

We approached the problem systematically: for the accelerometer, we collected test data across various movement profiles and implemented a dynamic thresholding algorithm to distinguish true repetitions from noise. For the heart rate sensor, we refined the placement mechanism on the armband, adjusted analog sampling intervals, and filtered out irregular spikes using a moving average algorithm. Calibration was continuously tuned using live test sessions and comparative validation against commercial devices.

Outcome After several iterations, both sensors reached a level of accuracy suitable for our project. The accelerometer now counts reps with over 95% accuracy in most common exercises, and the heart rate sensor consistently reports readings within a 5 BPM range of Fitbit and Apple Watch devices.

6 MEMBER CONTRIBUTIONS

6.1 Samuel

Created the initial version of the GUI and worked on integrating it with the Raspberry Pi, Pico, and camera system. Focused on setting up and troubleshooting the heart rate sensor on the Pico and helped establish the Bluetooth connection between the Pico and Raspberry Pi to enable reliable data transmission. Also designed and 3D-printed the enclosures for both the SAFE Hub and the Armband, assembling the hardware into a functional and user-ready form.

6.2 Abbie

Created the squat evaluation part of the model, and then integrated the squat and push-up models together into one code that could run. When the decision was made to do post-processing of the video rather than live feedback, she formatted the form-evaluation code accordingly. She was also a part of integrating the GUI into the RPi using the camera rather than computer input, adding form evaluation to the flow of the GUI, and formatting the output videos and playbacks to integrate seamlessly.

6.3 Facundo

Integrated Gyroscope, Button, and Heart Rate Monitor in the Pico to gather the Armband's data. Additionally, configured the Bluetooth connection between the RPi5 and Pico in order to consolidated the gathered information and provide optimal user experience.

6.4 Ethan

Assisted with research and integration of the MoveNet pose estimation model into the SAFE workout app in addition to developing the push-up evaluation logic and code. After dependency issues were resolved and the GUI was in a runnable state, he worked alongside Abbie to resolve issues with performance of the model during inference. He helped overcome this challenge by moving the video capture component of the pipeline to its own separate thread to avoid blocking (thus increasing CPU utilization and increasing throughput). In the last week leading into the presentation, he helped to resolve issues with the GUI that caused video playback of analyzed workouts to be drastically sped up, resulting in a more polished display of results.

REFERENCES

[1] FSA-2104-CAPSTONE-11. 2021. FormFit: AI-Powered Fitness Form Evaluation. https://github.com/FSA-2104-CAPSTONE-11/FormFit Accessed: 2024-03-22.
[2] Teresa Loboa. 2024. AI-Trainer: AI-Based Personal Trainer. https://github.com/LoboaTeresa/AI-Trainer Accessed: 2024-03-22.
[3] Luka Sajina. 2021. *Machine Learning-Based Pose Estimation for Fitness Applications.* Technical Report. University of Rijeka, Department of Informatics. https://www.inf.uniri.hr/images/studiji/poslijediplomski/kvalifikacijski/Sajina_kvalifikacijski.pdf Accessed: 2024-03-22.
[4] TensorFlow. 2024. MoveNet: Ultra Fast and Accurate Pose Detection Model. https://www.tensorflow.org/hub/tutorials/movenet Accessed: 2024-03-22.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009