StrikeSense: Wearable IMU-Based Strike Classification System

Landon Wright, Timothy Azinord, Benjamin Bolton, Shem Fils-Aime

Department of Electrical Engineering and Computer Science
University of Central Florida, Orlando, Florida, 32816

Abstract* — *This paper presents the design and implementation of StrikeSense, a wearable sensor system for real-time classification of athletic strikes. The system integrates IMU sensors, wireless communication via BLE, and a machine learning pipeline to analyze motion data. Experimental results demonstrate reliable performance with low latency and high classification accuracy.

Index Terms — Bluetooth Low Energy, inertial sensors machine learning, wearable devices, embedded systems, computer vision

I. INTRODUCTION

Sports serve as a platform to display human performance with technology often supplementing athletes' growth. Approaches have grown from strictly statistical to using IoT approaches. Within the realm of martial arts, there is a lack of innovative and robust approaches compared to other sports.

The biggest obstacle that technology can assist with is injury avoidance. Injuries are natural due the nature of the sport however they also come from ill advised technique. The entry fee for solving this problem is identifying strikes. Prior approaches have attempted to do this through the use of sensors [# for citation] or computer vision [8]. The approach that has found the most success is one which combines the strengths of both [# for citation]. Using an IMU strictly for capture forces users to wear multiple IMU's for systems to get a full understanding of strikes. This downside is not favorable due to its negative effect on athlete performance. Using a strict computer vision approach runs the risk of losing value with occlusion. A combined approach uses the strengths of each to cover the weakness of the other. We continue this approach with the novel addition of feedback being added in order to stimulate athlete development.

Strike Sense captures kinematic data through IMU's mounted on athletes' wrist as well as camera feeds fed through computer vision models. This data is then passed through models trained on similar data to identify strikes being thrown. The output of these pipelines are then consolidated by one more model which identifies the strikes. Capture and classification is done in real time with feedback being provided on a per session basis. Feedback and raw data is accessible through a web based dashboard. Users have their own account associated with data to prevent overlapping feedback. Feedback is defined as raw strike counts per session, raw strike counts historically., heartrate, and recommended combos.

This approach builds upon proven prior approaches and provides novel features which foster growth on an individual level. Its room to grow is immense as it is currently being designed with users who are new to the sport in mind.. With sufficient data and adoption, it can be used to assist high level athletes which will push the edge in competition, raising overall peak human performance in one of the most difficult sports to do today.

II. SYSTEM ARCHITECTURE

A. System Overview

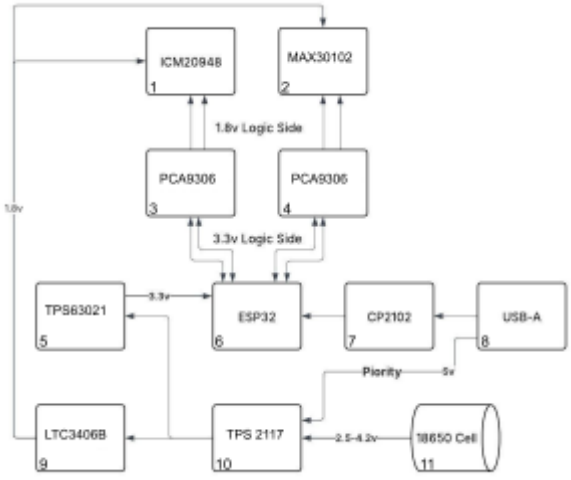

Figure 1: Hardware Block Diagram

ICM20948 – 9-axis IMU for motion and orientation tracking (accelerometer, gyroscope, magnetometer). **MAX30102 –** Optical sensor for heart rate and SpO_2 measurement using photoplethysmography. **PCA9306 –** I^2C level shifters interfacing 1.8 V sensors with the 3.3

V ESP32. **TPS63021 –** Buck-boost converter providing a stable 3.3 V supply. **ESP32 –** Main microcontroller for data processing, control, and wireless communication. **CP2102 –** USB-to-UART bridge for programming and serial communication. **USB-A (1046) –** Provides 5 V input power and host interface. **LTC3406B –** Buck converter generating a 1.8 V rail for sensors. **TPS2117 –** Power multiplexer switching between USB and battery input. **18650 Li-ion Cell –** Primary battery source (2.5 - 4.2 V).

B. Data Flow

The data flow of the StrikeSense system is organized into multiple parallel pathways serving distinct roles, either real-time inference or user-feedback. The primary data path starts at the sensor layer, where the ICM-20948 captures 3-axis motion data including acceleration, angular velocity, and magnetic field measurements. Our current implications samples at 60Hz to align with a 60fps camera feed simplifying synchronization with the computer vision system and improving the downstream classification algorithm. This sampled data is transmitted via an I2C bus to the central processing unit.

The processing unit, the ESP32-S3, complies and timestamps incoming sensor data. Optional preprocessing such as sensor fusion, may be applied generating orientation estimates. The data is then formatted into packets and transmitted over Bluetooth Low Energy(BLE) to the host system.

At the host level, the received IMU data forms the primary input to the machine learning pipeline. In addition to the sensor data, the host-system incorporates a computer vision stream that is processed and synchronized with the IMU data to be used as an auxiliary input to the classification algorithms. This gives complementary spatial context improving classification.

In parallel, a secondary data pathway is dedicated to physiological monitoring. A wrist-mounted heart rate sensor captures biometric data (bpm) and is transmitted wirelessly to the host device. Unlike the IMU and computer vision data streams,the heart rate data is not used in the inference process but instead is provided to the user for performance monitoring.

Finally all processed data is displayed on a simple and intuitive front-end application.This multi-path data flow architecture allows the efficient capture, processing, and transmission of the multiple data streams while maintaining low-latency, synchronization and reliability.

III. HARDWARE DESIGN

A. Processing Unit

The processing unit for the strike sense is the ESP32-S3 which serves as the primary controller for data acquisition, preprocessing, and wireless communication. This microcontroller was selected do to its combination of providing high-processing capabilities, integrated wireless support, and low-power operation, making it well suited for a wearable embedded application

The ESP32-S3 features a dual core processor operating up to 240MHz, this provides sufficient computational resources to handle real-time sensor data acquisition, preprocessing and wireless transmission. Memory resources on the ESP32-S3, include on-chip SRAM and external PSRAM which are sufficient and enable efficient buffering of sensor data prior to transmission

The ESP32-S3 also includes an internal I2C bus which will be utilized with the onboard sensors to sample at a fixed rate of 60 Hz and an integrated analog-to-digital converter (ADC) which will be utilized to monitor the battery voltage, providing battery percentage estimations

In addition to data acquisition the ESP-32 is responsible for some light-weight preprossing prior to wireless transmission including, time stamping data then organizing into structured packets for transmission, and optionally applying sensor fusion algorithms.

B. Sensor Subsystem

The sensor subsystem of the Strike Sense unit consists of an inertial measurement unit (IMU) and an optical heart rate and pulse oximetry sensor. These components enable the system to capture both motion dynamics and physiological data. The combined use of these sensors allows for synchronized acquisition of kinematic and physiological data, enabling comprehensive analysis of user activity.

1) Inertial Measurement Unit Overview

An inertial measurement unit (IMU) enables real-time motion and orientation tracking by combining an accelerometer, gyroscope, and magnetometer into a single device. The accelerometer measures linear acceleration, the gyroscope captures angular velocity, and the magnetometer provides absolute orientation using the Earth's magnetic field, enabling full 3D motion characterization. In wearable systems, IMU selection requires balancing accuracy, power consumption, size, and cost, as higher degrees of

freedom improve tracking performance but increase system complexity and energy demand [1]–[3].

Sensor	DoF	Size (mm)	Pwr (mA)	$
MPU6050	6	4x4	~3.9 mA	$
BNO055	9	3.8x5.2	~12 mA	$$$
ICM20948	**9**	**3x3**	**~3.0 mA**	**$$**

Table 1: IMU Comparison

The **ICM20948** was selected due to its integrated 9-axis capability, compact size, and medium cost. Additionally, it is widely supported with existing libraries and documentation, simplifying development and integration into the system.

2) Heart Rate and Pulse Oximeter Overview

A pulse oximeter measures physiological signals such as heart rate and blood oxygen saturation (SpO_2) using photoplethysmography (PPG). In the Strike Sense system, this sensor provides real-time insight into user exertion by detecting changes in blood volume through light absorption. Red and infrared LEDs are used to measure variations in blood flow, enabling extraction of heart rate and estimation of oxygen saturation. Key design tradeoffs include size, power consumption, accuracy, and level of integration, all of which must be balanced for a compact wearable system.

Sensor	Sp02	Size (mm)	Pwr (mA)	$
AD8232	No	5.6x2.8	~0.17 mA	$$
MAX30100	Yes	4x4	~1.5 mA	$
MAX30102	**Yes**	**5.6x3.3**	**~1.0 mA**	**$$**

Table 2: HR Comparison

The **MAX30102** was selected due to its high level of integration, combining LEDs, photodetectors, and analog front-end circuitry into a single compact package, which simplifies PCB design and minimizes external components. Compared to earlier devices such as the MAX30100, it provides improved signal quality and measurement reliability. While the AD8232 was considered for its very low power consumption, it is an ECG-based solution that requires additional analog circuitry and processing, making it less suitable for this application. Its power savings were ultimately unnecessary given the system constraints, and the added design complexity outweighed its benefits.

C. Power System

The major source of energy for the system is made up of a collection of 18650 protected lithium-ion cells. This is because the cell is energy-dense and has a long lifespan. Additionally, the voltage of each cell is approximately 3.7 V, and the cell is fully charged when the voltage is approximately 4.2 V. However, the voltage is set to approximately 3.0 V in order to prevent cell degradation. Additionally, the protected cell has in-built safety features such as overcharge protection, over-discharge protection, and overcurrent protection. Moreover, in order to monitor the voltage in real-time, there is a voltage sensing circuit made up of a voltage divider and an ADC in the ESP32. This ensures that the voltage is in the correct range for the micro-controller so that there is no risk of damage. Additionally, the voltage divider is made up of high-value resistors in order to prevent quiescent current being drawn from the battery.

Power Source management is achieved through the TPS2117 power multiplexer. This is the component that selects between both the USB and the battery as inputs. It is configured so that it prioritizes USB input when present. And the multiplexer seamlessly transfers the system to load from the battery to the external supply. This addition to the circuit is essential to prevent voltage drops or interruptions that can lead to system resets or brown outs. Additionally it has controlled inrush current that minimizes stress on the power supply during startup. Together these features enable efficient and reliable power management.

To accommodate the varying input voltage of the battery the primary logic rail is generated using the TPS63021 boost buck converter which produces 3.3 V output. . Unlike traditional linear or single mode switching regulators the boost buck converter transitions between step up and step down modes and has a constant output voltage across the entire input range. Choosing this approach served us multiple benefits: it first allowed for high efficiency across the battery discharge and reduced the thermal dissipation

compared to linear regulators. Careful selections of critical passive components like inductors and capacitors optimized the efficiency and output ripple.

The device incorporates a secondary output rail of 1.8V using the LTC3406B buck switching regulator. This power rail serves as a source of voltage components such as sensors and other digital circuitry. Since the input of this regulator is a much higher regulated voltage a buck topology provides efficiency with minimal power loss. And it is recommended because it is well suited for compact designs because of its low external component count.

Connected Sources	**Battery Terminal Voltage**	**USB Vbus voltage**	**1.8v regulator input voltage**	**3.3v regulator input voltage**
USB(5v)	**3.3mV**	**5.22V**	**5.22V**	**5.22V**
Battery(3.7v	**3.69V**	**0.008mV**	**3.69V**	**3.69V**
USB(5v) + Battery(3.7v)	**3.69V**	**5.22V**	**5.22V**	**5.22V**

Table:3 In its current configuration the power mux should give priority to VBUS when both sources are connected or allow voltage from either source through when a single source is connected

IV. SOFTWARE DESIGN

A. Embedded Firmware

The embedded firmware running on the ESP32-S3 is responsible for sensor acquisition, data preprocessing, and wireless data transmission.

Upon power-up or reset, the system initializes all required hardware peripherals, including the I2C interfaces and Bluetooth Low Energy stack. If initialization fails, the node enters a safe termination state. Following successful initialization the inertial measurement unit (IMU) is calibrated, and the device begins BLE advertising.

The system will remain idle until a synchronized start signal is received from the master device. This start command ensures the synchronized acquisition of multiple data streams.

Once the start signal is received, the node transitions into an active sampling state. The IMU is sampled continuously at 60Hz. Configurable within the software the sampled data will either be transmitted raw, in the form of 3-axis accelerometer, gyroscope, and magnetometer data or passed through an altitude heading reference system AHRS, computing quaternion based orientation estimated and then transmitted.

Sampled data is then time stamped and formatted into structured packets and then transmitted to the host system via BLE. The node will continue streaming until a stop signal is received, at which point the node safely terminates or remains in an idle state.

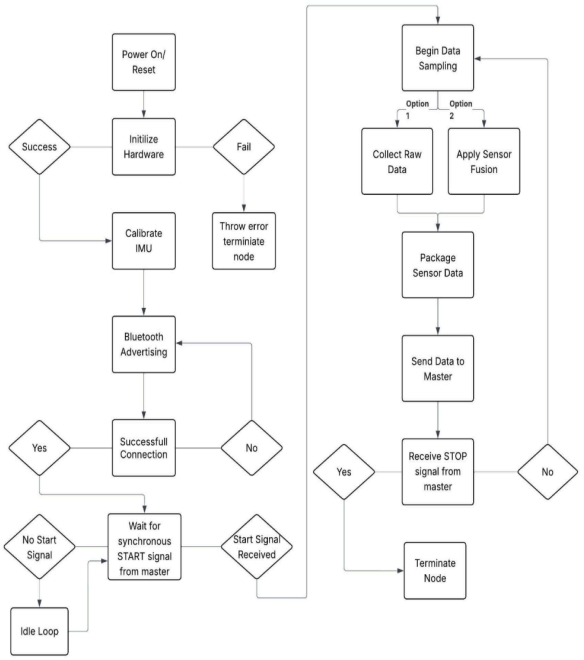

Figure. 3 Embedded State Diagram

In addition to motion sensing, the system supports heart rate monitoring using an onboard optical sensor. Due to the sensitivity of photoplethysmography(PPG) measurements to motion, heat rate acquisition is not performed concurrently with action motion sampling, heart rate measurements will be initiated on-demand by the user, typically before and/or after training when the device and user are stationary. This design choice avoids interference from motion induced noise in biometric sensing, while still providing useful feedback to the user.

The firmware is implemented using a software stack composed of widely supported libraries. The Arduino core provides the runtime environment and hardware abstraction, while the Wire library handles I²C communication with the IMU. Sensor configuration and data acquisition are managed using the Adafruit ICM-20948 driver. Quaternion-based orientation estimation is performed using a sensor fusion library, and wireless communication is implemented using the NimBLE stack, which provides a lightweight BLE GATT server optimized for low-power data streaming.

The current firmware utilizes an approximate 9.3% of RAM and 16.3% of flash memory. This leaves plenty of headroom for any future enhancements.

B. Machine Learning Pipeline

The machine learning pipeline is responsible for taking raw sensor data and classifying it into one of three strike types: hook, jab, or uppercut. The IMU captures six channels of motion data, consisting of three axes of acceleration and three axes of angular velocity, sampled at 50 Hz. This raw stream is segmented into fixed-length sliding windows, where each window is sized to roughly contain a single punch based on the sampling rate and the average number of strikes thrown per recording. Overlapping windows with a reduced stride are used during training to increase the number of available samples, while non-overlapping windows are used during testing to avoid inflating evaluation metrics.

The classification model uses a multi-stage architecture built in PyTorch. The first stage is a convolutional neural network (CNN) that extracts local spatial and temporal features from the windowed input. Convolutional filters scan across the time axis of each window, identifying short-duration patterns such as the initial acceleration spike of a punch or the rotational signature that distinguishes a hook from a jab. The output of the CNN is a compressed feature map that preserves the most informative characteristics of the raw signal.

This feature map is then passed into a bidirectional Long Short-Term Memory (LSTM) network. While the CNN captures localized patterns, the LSTM is designed to model longer temporal dependencies across the full duration of a strike. Running bidirectionally allows the network to read the sequence in both the forward and reverse directions, giving it context from both the wind-up and the follow-through of a punch. An attention mechanism sits on top of the LSTM output and learns to assign higher importance to the most discriminating timesteps within each window rather than relying solely on the final hidden state. The attended output is then passed through a fully connected layer with softmax activation to produce a probability distribution over the three strike classes.

Training the model required addressing several challenges tied to the relatively small dataset. The training set consisted of roughly 723 samples, and the class distribution was imbalanced, with hooks being the most represented strike type. To handle this, weighted cross-entropy loss was applied to penalize misclassifications on underrepresented classes more heavily, and a WeightedRandomSampler was used in the data loader to oversample minority classes during each epoch. Regularization techniques including dropout at a rate of 0.4, L2 weight decay in the Adam optimizer, and batch normalization were applied throughout the network to combat overfitting. Early stopping with patience was used to halt training when validation accuracy plateaued, and the best checkpoint was saved for final evaluation.

Model validation was performed using 5-fold cross validation, which rotates the training and validation splits so that every sample is validated exactly once. The fold whose performance was closest to the mean across all folds was selected for final testing, avoiding the bias of simply picking the best-performing fold. Balanced accuracy was used as the primary evaluation metric so that each class contributed equally to the score regardless of sample count. The learning rate was reduced from an initial value of 5e-3 to 1e-3 to accommodate the sensitivity of LSTM gate transitions, and a ReduceLROnPlateau scheduler was used to automatically halve the rate when progress stalled. During inference, a confidence threshold of 0.70 is applied so that only predictions where the model is at least 70% certain are counted as valid classifications.

C. Front-End Application

Users are able to interact with the system through a simple and easy to use interface built upon React. This dashboard has 3 main functions:

- Provide users a way to view all raw data
- Provide users a way to view feedback
- Store historic data on a per user basis which is enforced through individual accounts

When first opened, Users are welcomed with an option to sign in or to create an account

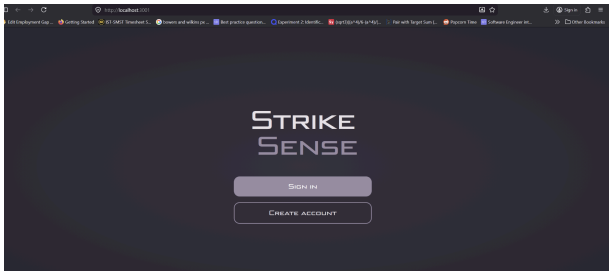

Figure 3:Landing page

A new user would create an account whose information is then stored in a PostgreSQL database. Once account creation is confirmed, users are then routed to sign in with credentials. Upon signing in, they are welcomed by the dashboard which serves as an all in one view.

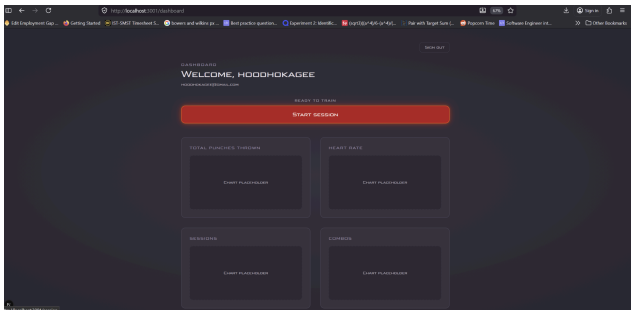

Figure 4: Dashboard view

From this Dashboard, users are able to start their sessions, view historical data, as well as sign out. Once a session is started, all information necessary for accurate data capture and processing is asked of the user on a separate page seen in the next figure.

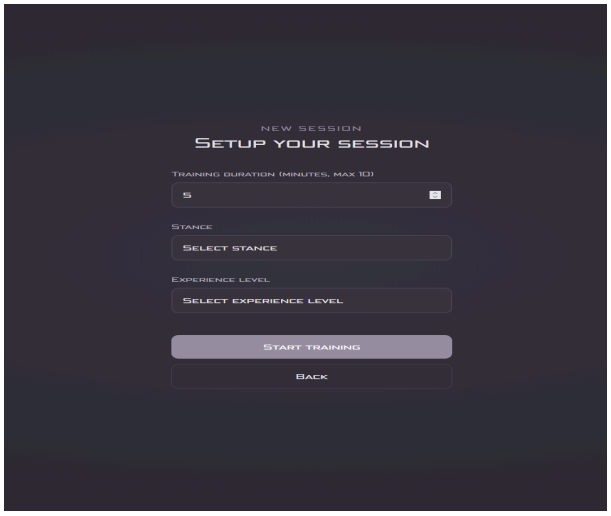

Figure 5: Session Setup

With this submission complete, the session of data capturing begins with only a timer being visible indicating how much time is left. After one session, Users are able to either begin another with the same setup, change the setup and start anew, or continue back to session feedback through the dashboard.

The backend of the system is composed mainly of python and postgresql. Python provides the code needed to connect the webcam and IMU through bluetooth and Postgresql serves as a way to store users.

The backend of the system is built on Node.js with Express and PostgreSQL, while Python handles the data collection and machine learning layers. PostgreSQL was chosen as the database because of its reliability, support for structured relational data, and its ability to handle the kind of linked records that a multi-user session tracking system requires.

The database schema is organized around three core tables: users, sessions, and strike_results. The users table stores each account's credentials and profile information. When a new user signs up through the React interface, the Express backend hashes their password using bcrypt before writing it to the database. This ensures that raw passwords are never stored, protecting user data even in the event of a database breach. Each user is assigned a universally unique identifier (UUID) generated natively by PostgreSQL, which serves as their primary key across the entire system.

The sessions table is linked to the users table through a foreign key relationship. Every time a user starts a new training session from the dashboard, a new row is created in the sessions table that references their user ID. This connection is what allows the system to associate all captured data with the correct individual. The sessions table also records metadata such as timestamps, session duration, and the setup parameters the user selected before beginning data capture.

The strike_results table stores the classified output from the machine learning pipeline on a per-session basis. Each row references a session ID, creating a chain of ownership from strike result to session to user. This relational structure means the dashboard can query a single user's entire history of sessions and strike data without any risk of pulling in another user's records. It also makes it straightforward to aggregate data over time, which is what powers the historical feedback view on the dashboard.

Authentication between the frontend and backend is handled through JSON Web Tokens (JWT). When a user logs in with valid credentials, the Express server issues a signed token that the React frontend stores and includes with each subsequent request. This token-based

approach keeps the system stateless on the server side, meaning the backend does not need to maintain active login sessions in memory. Instead, each request is independently verified by checking the token's signature, which simplifies scaling and reduces overhead. Python handles the data collection side, managing the Bluetooth connection to the IMU sensor and the webcam feed for the computer vision pipeline. These scripts run as child processes spawned by the Express server, and device connection status is held in memory on the server rather than written to the database, keeping the schema focused on persistent user and session data.

V. IMPLEMENTATION AND INTEGRATION

The hardware was assembled on 4 individual printed circuit boards that connect using pin headers. The reason for modularizing our version 1 design was that it allowed for sectionized debugging..

1) Main Board

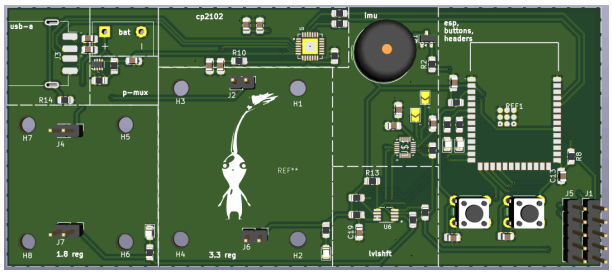

Figure 6: Main Board Version 1 (size: 121.5mm x 53mm)

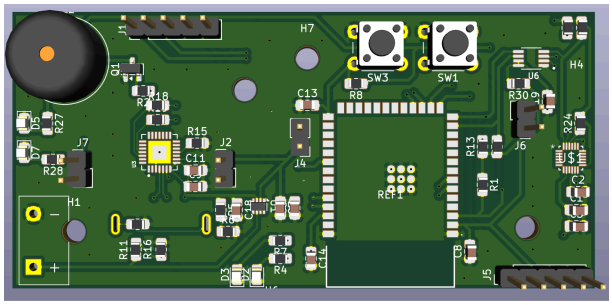

Figure 7: Main Board Version 2 (size: 82 mm x 39 mm)

2) 3.3v Regulator Board

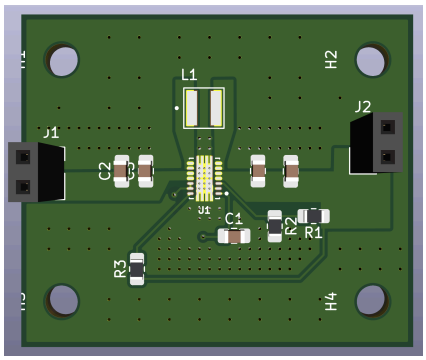

Figure 8: 3.3v Regulator Board Version 1 (size: 39 mm x 43 mm)

3) 1.8v Regulator Board

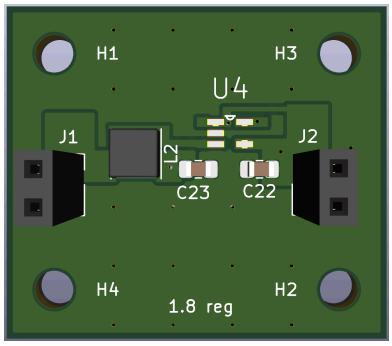

Figure 9: 1.8v Regulator Board Version 1 (size: 32mm x 28mm)

4) Heart Rate Monitoring Board

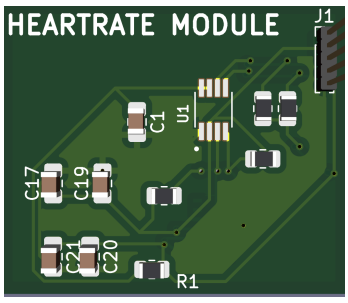

Figure 10: Heart Rate Sensor Board Version 1 (size: 28.4mm x 23.4mm)

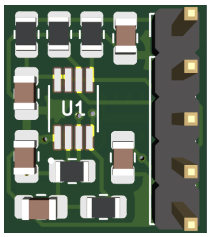

Figure 11: Heart Rate Sensor Version 2 (size: 12.45mm x 14.1mm)

VI. RESULTS

The voltage regulators demonstrated high accuracy, with the 3.3 V rail achieving ~0.076% error (≈99.924% efficiency) and the 1.8 V rail achieving ~0.861% error (≈99.139% efficiency).

Upgrading from the MAX30100 to the MAX30102 reduced heart rate error from ±30 BPM to ±2 BPM, corresponding to a ~93.3% improvement. Additionally, the 1.8 V regulator board area was reduced by ~73.6%, significantly improving hardware compactness.

The 9-axis IMU provides a steady stream of raw motion data for real-time monitoring. While initial larger hardware introduced instability, the reduced footprint is expected to improve measurement consistency and overall system reliability.

The IMU machine learning pipeline currently is able to identify with 80% accuracy. The computer vision pipeline is being trained with a hand collected dataset. The current model being used to capture movement keypoints for training is YOLOv11.

VII. CONCLUSION

This work presents a wearable system for real-time motion and physiological monitoring, integrating IMU and pulse oximetry with embedded processing. Results demonstrate strong improvements in sensing accuracy and significant reductions in hardware size.

Future work focuses on further miniaturization for consumer usability and improving machine learning performance through expanded datasets and model refinement.

References

[1] Titterton, D. H., and Weston, J. L., *Strapdown Inertial Navigation Technology*, 2nd ed., IET, 2004.

[2] Sabatini, A. M., "Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 7, pp. 1346–1356, 2006.

[3] Madgwick, S. O. H., "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," Univ. of Bristol, 2010.

[4] Allen, J., "Photoplethysmography and its application in clinical physiological measurement," *Physiological Measurement*, vol. 28, no. 3, pp. R1–R39, 2007.

[5] Kamal, A. A., Harness, J. B., Irving, G., and Mearns, A. J., "Skin photoplethysmography—A review," *Computer Methods and Programs in Biomedicine*, vol. 28, no. 4, pp. 257–269, 1989.

[6] Verkruysse, W., Svaasand, L. O., and Nelson, J. S., "Remote plethysmographic imaging using ambient light," *Optics Express*, vol. 16, no. 26, pp. 21434–21445, 2008.

[7] Maxim Integrated (now Analog Devices), *MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health*, Datasheet, 2018.

[8] Maxim Integrated, *MAX30100 Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health*, Datasheet, 2014.

[9] Jabbr Technologies ApS, "Jabbr.ai | AI for Combat Sports," Accessed: Mar. 27, 2026. [Online]. Available: https://jabbr.ai/