PROJECT TITLE

StackSense: Making Any Gym Stack Weight Machine Smarter

STUDENT/TEAM INFORMATION

Team Name if any: **Team # on Canvas you have self-signed-up for:**	*Team 9*
Team member 1 (Team Lead) (Last Name, name: student ID: UCI email, picture):	*Cheng, Leyang* Removed in public copy
Team member 2 (Last Name, name: student ID: UCI email, picture):	N/A

ABSTRACT (15 points)

(Summarize your project (motivation, goals, system design and results). Max 300 words).

A smart scale is usually the first fitness device most beginners use, but it doesn't record the quality of each workout. A gym coach would provide feedback, but it's too expensive for beginners. Thus, a sensor-based AI assistant device would be a better choice.

This project implements StackSense, a low-cost Internet of Things device attached to the weight stacks of fitness machines to measure range of motion. A VL53L1X Time-of-Flight distance sensor (laser-based) module is aligned upwards above the weight stacks to measure the lift displacement, while an LSM6DSO Accelerometer records the acceleration along the same axis. Both sensors connect to an ESP32 on a LILYGO T-Display through the I2C bus. The ESP32 processes the sensor data and segments individual movements, providing immediate feedback of counts through the display and buzzer. After a set, the cached data are uploaded in packed binary form via Wi-Fi to a Javalin server deployed on an AWS cloud server. The server later unpacks and reconstructs the data, calculates the range of motion, the duration of concentric and eccentric phases, and the peak speed for each movement, and stores it in MySQL. A web dashboard then visualizes the data and provides a prepared AI prompt for analysis.

The System was tested in a public gym. In a representative set of ten movements, 2519 samples were collected within 46.5 seconds at an effective sampling rate of 54 Hz, and the data was transmitted in 12595 bytes within two seconds; the server-segmented results matched real-time counts on the device. The analysis quantified the control quality that the introduction argues is invisible.

INTRODUCTION (15 pts)

Motivation/Background (3 pts)

(Describe the problem you want to solve and why it is important. Max 300 words).

For fitness beginners, a gym coach is expensive, whereas a sensor-based AI assistant with workout motion tracking is not.

Hiring a coach is one solution, but a fitness coach is usually expensive, especially for fitness beginners. The people who need the correction most are the least likely to pay a lot. Even then, the feedback is verbal and not quantitative.

A good repetition (rep) is not simply push and release. How you control the movement is a large part of the training effect—a rep pushed steadily and returned slowly keeps the muscle under tension far longer than the same rep thrown up and dropped back onto the stack. In other words, to get a good effort, you need to control your muscles: push slowly and return smoothly.

The problem is that lifting quality is invisible. A fitness beginner cannot tell whether the last five reps were as controlled as the first five, or whether the range of motion quietly shortened as fatigue set in. Feelings are hard to record, and effort is hard to measure. To solve this problem, we need something that can monitor and record the workout.

A stack machine is usually the first machine fitness beginners try. Thus, this project focuses on motion tracking for stack machines.

Project Goals (6 pts)

(Describe the project general goals. Max 200 words).

The goal of this project is to build a low-cost sensor node that records the movement of a pin-loaded stack fitness machine and quantifies the quality of every repetition.

The node sits above the top of the weight stacks and moves along with the stacks. A VL53L1X Time-of-Flight distance sensor (laser-based) module measures how far the stack travels, directly capturing the distance of lift motion. An LSM6DSO accelerometer would also capture acceleration during lifting. An ESP32 on a LilyGo T-Display would save this data and separate the set into reps (repetitions). The display on the LilyGo would show the count on screen and provide audio feedback through a passive buzzer.

The device could work on its own or with Wi-Fi. The user could use two buttons on the device to start, end, or cancel a set. After a set is completed and the device connects to Wi-Fi, it uploads the recorded data to the AWS cloud server. The server stores each set in a MySQL database and computes the per-rep analytics: range of motion, concentric duration, peak velocity, etc. A web dashboard would present the visualization and analysis of the user's data.

Assumptions (3 pts)

(Describe the assumptions (if any) you are making to solve the problem. Max 180 words).

I assume that the weight stack the device sits on is constrained by fixed guide rods and lifts vertically.
I assume the surface the sensor points at is diffuse enough to return the laser. (The surface should not be metal; otherwise the laser might be reflected somewhere else).
I assume that the user would cooperate with the device, using buttons to confirm a set has finished. Continuous non-stop cycling is out of scope.
The system serves one user on one machine at a time; high concurrency is not currently being considered.
I assume the gym network is safe for this prototype to transmit through HTTP without authentication or encryption.

SYSTEM ARCHITECTURE (20 pts)

(Describe the final architecture you have implemented listing sensors, communication protocols (Wi-Fi, BLE, ...), cloud services and user interfaces. Include a block diagram of the system. Max 300 words).

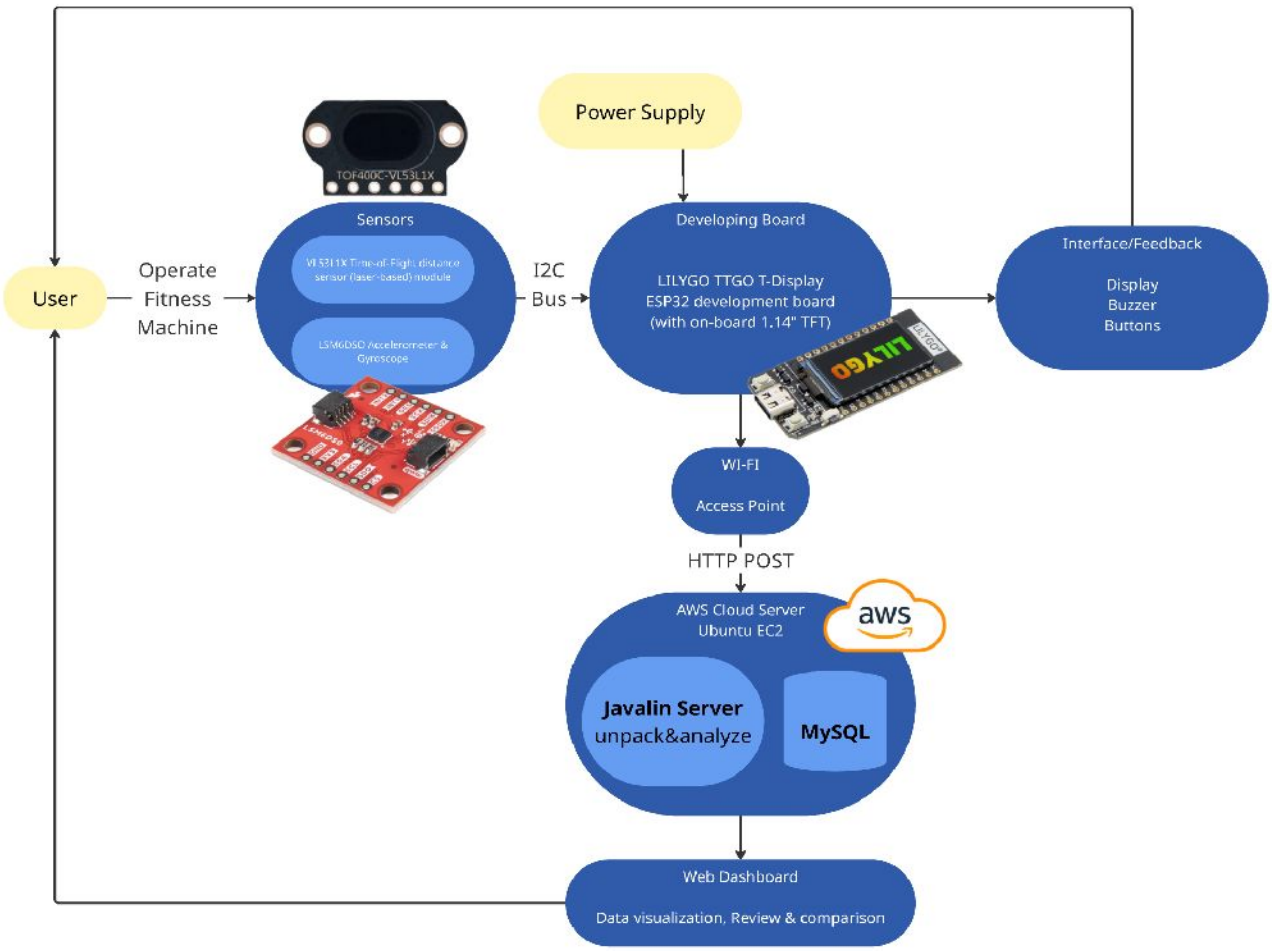
This project implements Architecture 2 (Sensor - WIFI - Cloud). The block diagram shown below.

While the user operates the fitness machine, pushing and releasing cause the stacks the device sits on to lift up and down. Two sensors, a VL53L1X time-of-flight module (laser-based) and an LSM6DSO accelerometer, track the distance of motion: The ToF (time-of-flight) module faces upward and measures the distance from the moving weight stack to the fixed panel above it. LSM6DSO records acceleration along the same axis. Two sensors share a single I2C bus, connected to the ESP32's SCL (GPIO 22) and SDA (GPIO 21)

I2C bus carries sensor data to the ESP32 and prepares it for sending. For the network path, the ESP32 joins the access point in station mode. When a set is complete, the ESP32 uploads the data as an HTTP/1.1 POST request over TCP/IP, carrying raw binary data. Each upload is keyed by device ID and set number.

An AWS EC2 instance hosts a Javalin server that receives the request and unpacks the binary to reconstruct the data. The server then computes the data: range of motion, concentric and eccentric duration, peak velocity, etc. The data are then stored in a MySQL database on the same AWS instance.

For user interfaces: the machine has an on-board TFT display and a buzzer to provide feedback while counting repetitions. Two push buttons mark the start/end and cancel of a set. After a set is completed, the display shows the messages "Sending…" and "Completed." Afterward, a web dashboard visualizes the stored sets. The web dashboard also provides a copy&paste prepared AI prompt for the user to discuss their workout effort with their own AI chatbot.



FINAL LIST OF HARDWARE COMPONENTS (5 pts)

(Write the final list and quantity of the components you have included in your system)

Component/part	Quantity
LILYGO TTGO T-Display ESP32 development board (with on-board 1.14" TFT)	1
VL53L1X Time-of-Flight distance sensor (laser-based) module	1
LSM6DSO Accelerometer & Gyroscope	1
Push buttons	2
buzzer	1
Power supply	1

PROJECT IMPLEMENTATION (30 PTS)

Tasks/Milestones Completed (15 pts)

(Describe the main tasks that you have completed in this project. Max 250 words).

Task Completed	Team Member
Integrated the hardware: wired the I2C bus for the ToF and IMU accelerometer, and added two interrupt-driven buttons.	Leyang Cheng
Tested how noisy the sensors actually are. The ToF reads at about 53 Hz with roughly 2 mm of noise. The IMU reads 1.03 g at rest, not 1.0, so I tare it before every set. I also found that just bumping the desk creates 2.5 g spikes.	Leyang Cheng
Built the signal processing chain: per-set tare baseline, tare the baseline at the start of each set, smooth the readings with a moving average, and estimate velocity from a 6-point window	Leyang Cheng
Implemented rep detection through a state machine, switching between Idle, Up, and Down, using peak/valley detection with hysteresis and	Leyang Cheng

minimum-ROM rejection; each rep completion would have feedback from the buzzer and display. The IMU would also record acceleration for further data analysis on the cloud server.	
Set up the sample buffer so a whole set can be stored and uploaded later, about 150 seconds of raw data in a fixed 40 KB array	Leyang Cheng
Build cloud-side functions to decode binary uploads and store them in MySQL. Check the full data workflow from HTTP POST to decode, validate, and store.	Leyang Cheng
Build a visualization & analysis dashboard on the cloud server. Deploy under my own domain xxxmnalx.com: https://stacksense.xxxmnalx.com/	Leyang Cheng
Collect real data at the gym	Leyang Cheng
Provide copy&paste prepared AI prompts for users to analyze their training in their own AI chatbot.	Leyang Cheng

Challenges/Roadblocks (5 pts)

(Describe the challenges that you have faced and how you solved them if that is the case. Max 300 words).

The biggest challenge I have faced is BLE. I first built Architecture 1: ESP32 -> BLE phone gateway (app) -> Cloud. The device worked, but the phone gateway was the problem. I learned to use BLE with the nRF Connect app, which can read and write characteristics but cannot forward anything to a server, which left me with Web Bluetooth or a native iOS application. Web Bluetooth isn't supported in Safari or iOS Chrome because of an iOS feature. Building an iOS app is beyond my current ability, so my solution is to switch to Architecture 2, where the ESP32 connects directly to the access point. The trade-off is that it's harder for the user to change the Wi-Fi name and password.

Another challenge I have is getting the set data to the server. My first plan was to upload each set as JSON, the format I was most familiar with. That plan didn't work because of the ESP32's limited RAM. Using a smaller data structure would let me collect and send more data to the server. My solution is to switch to packed binary data to minimize payload size. I then tried passing the samples in a GET query string as in Lab 4, which failed for a different reason: Jetty rejects a request line longer than its header size limit, so it refused larger sets before my handler ever ran. My solution was to use a POST request carrying the samples as packed binary (five bytes per sample), declared as application/octet-stream.

Tasks Not Completed (5 pts)

(Describe the tasks that you originally planned to complete but were not completed. If all tasks were completed, state so. Max 250 words).

Task	Reason
Utilize AWS Bedrock to build a server-side natural-language coaching feedback	Not implemented. Bedrock is billed per token, and the server would call it on every set. Given the development fee and potential deployment risk, an alternative is a formatted analysis prompt the user can copy into their own AI chatbot to deliver the same interpretation without the cost.

WEAK POINTS / FUTURE WORK (15 pts)

(Mention at least two points of your project that have room for improvement. These points can be additions to the existing project setup (components) or improvement of the current implementation. Max 200 words).

The first weak points would be identity, isolation, and security. The system currently assumes one user, one machine, and one server, so it has no account system. Every set is stored under a device ID, and every visitor to the dashboard could see all of them. Upload also travels as plain HTTP, so anyone on the same gym network can read a set or gorgo one. For future work, I would want to improve the cloud server by implementing registration with per-user data isolation, an authenticated delete path, and TLS with a signed device token on upload.

Another limitation of the current time-of-flight sensor is that it depends on the surface above it returning the laser. A polished or angled panel is specular, and the returned signal collapses when the node is tilted even slightly, which is why the current design assumes a prepared target surface. A possible future improvement would be to improve the outer case so the ToF sensor points to the gym floor, using a magnetic base to attach to the side of the stack.

FINAL DEMO VIDEO (60 pts)

Please include a link to a demo video (5 min max) showing the setup, the components and functionality of your IoT system. A Google drive link is preferred.

https://youtu.be/7v7RHENa6LA

SOURCE CODE (25 pts)

Please include a link to the source code of your project. A link to a repository (like GitHub) is preferred.

https://github.com/xxxmnalx/StackSense