Information Technology Course
Module: Individual Project
Supervised by Prof. Dr. Andreas Pech

Smart lighting system for electricity saving with Red pitaya and Ultrasonic sensor

Submitted by

Md Zahid Hasan
Matriculation:1396470
Email: md.hasan5@stud.fra-uas.de

***Abstract*— To optimize energy efficiency in modern office environments, this paper proposes a machine learning algorithmic approach comprising a two-stage hybrid human detection system utilizing a Red Pitaya board and an SRF02 ultrasonic sensor. Initially, the signal is processed to distinguish between human and non-human presence in real time. In the first stage, the model detects activity based on changes in distance threshold, while the second stage uses a Convolutional Neural Network (CNN) to confirm whether the presence is human or non-human. Despite the challenges of ultrasonic signal interference, the CNN model demonstrates high effectiveness, achieving an accuracy of 99.53%. Featuring a comprehensive GUI for real-time monitoring, this study validates that affordable ultrasonic hardware combined with deep learning offers a robust solution for intelligent lighting automation.**

Keywords*— Red Pitaya, Signal Processing, Spectrogram, Short Time Fourier Transform, Convolutional Neural Network, Graphical User Interface.

I. INTRODUCTION

In the era of smart cities and modern infrastructure, managing energy efficiency in buildings has become a critical challenge. While saving energy is essential, it should not come at the cost of user comfort. One of the most effective ways to reduce energy waste is to detect whether a room is occupied, potentially cutting energy consumption by up to 30%. However, the technologies currently used to detect human presence often fall short [1]. Traditional motion sensors, such as Passive Infrared (PIR) sensors, are cheap and common, but they have a major flaw: they frequently fail to detect people who are sitting still. This leads to "false negatives," where lights turn off on an occupant simply because they aren't moving. On the other hand, advanced solutions like cameras or thermal imaging are accurate but consume too much power and raise serious privacy concerns regarding surveillance. To overcome these limitations, this project proposes a novel human detection system that is non-contact, privacy-preserving, and highly accurate. By integrating ultrasonic echolocation with a state-aware Deep Learning pipeline, this system aims to solve the problem of detecting static occupants.

The proposed system utilizes a client-server architecture to handle data acquisition and analysis efficiently. The hardware setup consists of an SRF02 ultrasonic rangefinder, a cost-effective sensor capable of measuring distances from 16 cm to 6 meters, connected via an I2C bus to a Red Pitaya STEMlab 125-14. The Red Pitaya acts as the server; it is a versatile System-on-Chip featuring a dual-core ARM Cortex-A9 processor and FPGA technology. Running a custom C program, the Red Pitaya captures 25,000 Analog-to-Digital Converter (ADC) data samples per frame. This raw acoustic data is then transmitted via UDP (User Datagram Protocol) over Ethernet to a laptop acting as the client, where a Python application processes the signals. Fig 1 shows an Ultrasonic Sensor connected with Red pitaya board.

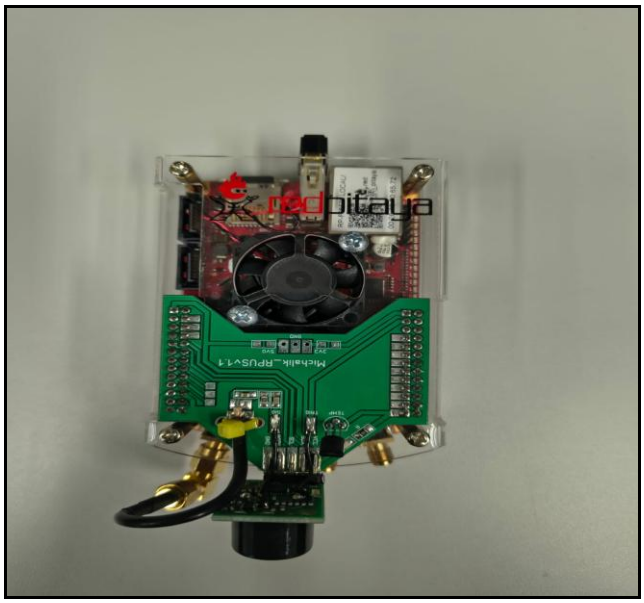

Fig. 1. Red Pitaya Board with Ultrasonic Sensor.

To promote energy efficiency and computational economy, the system operates on a specialized "two-stage" detection logic controlled via a Graphical User Interface (GUI). In the initial phase, known as Activity Detection, the application employs a robust "Distance Variation Threshold" algorithm. This process continuously compares the calculated distance of the current frame against the previous frame. If the difference exceeds a user-defined threshold of 10 cm, the system flags an "Activity" event, indicating sudden movement or environmental changes; this immediately activates the Red Pitaya's onboard LED7 indicator and initializes a countdown timer. Once triggered, the system transitions to the second stage: CNN

Classification. Here, a custom 2D Convolutional Neural Network (CNN) is employed. The raw 1D ultrasonic signals are converted into 2D spectrograms to capture frequency intensity over time, which are then analyzed by a PyTorch-based model to classify the source as "Human" or "Non-Human." By combining these methods, the system ensures intelligent responsiveness. If the CNN continuously confirms a "Human" presence, the internal timer resets, keeping the LED7 illuminated to simulate active lighting. Conversely, if the CNN detects a "Non-Human" object or the timer expires without confirmation, the system automatically extinguishes the light and reverts to the idle stage, effectively preventing energy waste on false positives while ensuring lights remain on for static occupants.

II. Methodology

A. Data Collection

To develop a system for detecting human versus non-human subjects, valid data for both categories is essential, as high-quality data is part and parcel of any successful project; without authentic data, the system cannot be developed effectively. To build this project, ADC data was collected using a C-based Red Pitaya library (prebuilt as a UDP client and installed in the laboratory). The experimental setup utilized the sensor's I2C interface to read from the ultrasonic proximity sensor. For data collection, the sensor was mounted on a metal stand at three different heights: 210 cm, 230 cm, and 250 cm, measured from the sensor head to the ground. The collected data was then separated into two distinct categories human and non-human to ensure the dataset was robust and representative. Fig 2 shows the distance measurement from sensor head to floor.

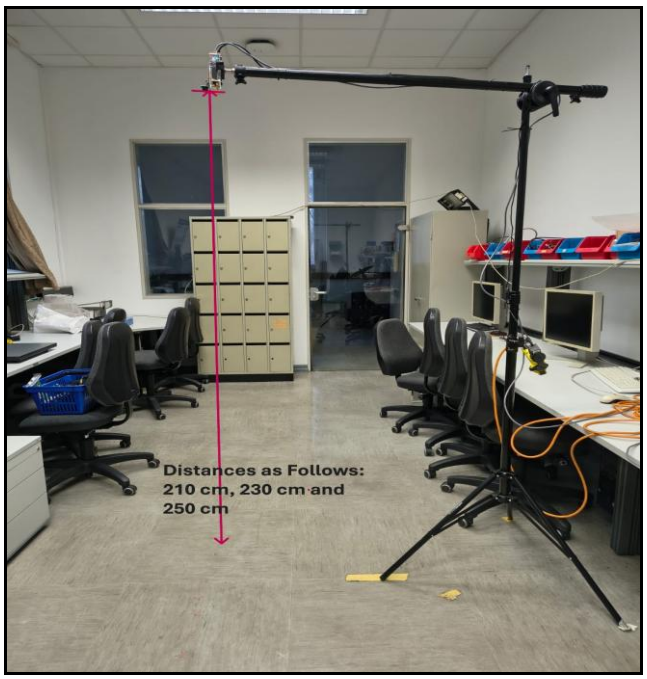

Fig. 2. Shows distance reading from sensor head to floor.

- Human data was collected from three people in a room under various conditions. These included sitting on a chair, standing, and wearing either a jacket or regular clothing (like a T-shirt). Data for the jacket and regular clothing sessions was recorded separately because different materials reflect sound differently, which changes the signal patterns. In total, 99,500 samples were collected in this category, making up 53.35% of the entire dataset. Fig 3 shows an example of a person sitting on the chair.

Fig. 3. Human data collected where human seated on a chair.

- Non-human data was gathered under varied conditions, including scenarios like a chair with a basket or jacket, and an office table with a book or laptop. In this category, 87,000 signal samples were collected, accounting for 46.65% of the total data. Although the sample sizes for the human and non-human classes were not exactly equal, the difference was insignificant. However, a data balancing technique was used during the training of the CNN model to ensure it did not become biased toward the majority class. Fig 4 shows an example of non-human data collection.

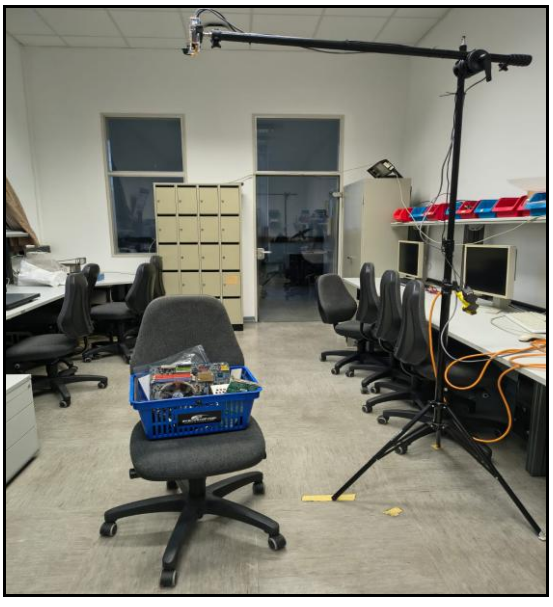

Fig. 4. Non-Human data collected with a basket on the top of a Chair.

B. *Data preprocessing*

Data preprocessing involves transforming and cleaning collected data to address issues such as inconsistencies, noise, and missing values before feeding it into a machine learning model. Clean and well-structured data is essential for achieving high model accuracy. Furthermore, removing irrelevant features helps prevent overfitting, which improves prediction accuracy and ensures the model can generalize well to unseen real-world data [2].

In this project, the pre-processing pipeline transforms raw ultrasonic time-series signals into consistent time–frequency representations for the deep learning model. All measurements were recorded using a Red Pitaya system and stored as CSV files, where each row represents a single waveform. Each ADC signal contains 25017 packets out of which the first 17 serves as header. For every signal, the first 5,517 samples are removed because they contain the initial transmit pulse. The remaining columns are interpreted as the actual signal. To ensure meaningful analysis, any signal shorter than a single STFT window (2,048 samples) is discarded. Fig 5 shows raw ultrasonic time-domain signals for Human and Non-Human classes.

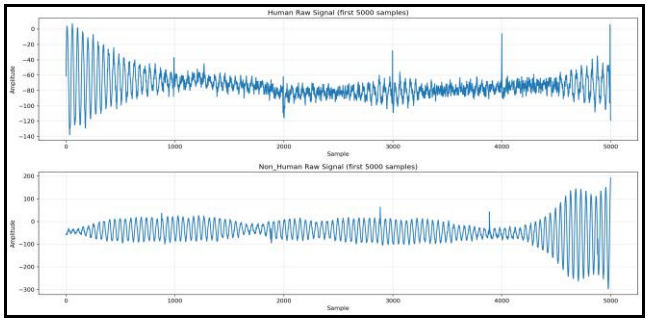

Fig. 5. Raw ultrasonic time-domain singnals.

Each valid 1-D signal is then converted into a two-dimensional spectrogram using the scipy.signal.spectrogram function, which internally performs a Short-Time Fourier Transform (STFT). The sampling frequency is fixed at 1953125 MHz. A Hamming window of 2,048 samples is applied with a 50% overlap (1,024-sample hop size), requesting the magnitude of the complex STFT coefficients. With these settings, every spectrogram is made up of 1,025 frequency bins (because for a window length NPERSEG, the unique frequencies are NPERSEG/2 + 1). These frequency points start at 0 Hz (no oscillation) and go up to the Nyquist frequency, which is the highest frequency that can be correctly represented for the given sampling rate. The number of time steps (columns of the spectrogram) is not fixed; it depends on how long the original signal is, because longer signals produce more time windows. All spectrogram values are stored as 32-bit floating-point numbers (float32), which is a common numerical format that provides a good balance between precision and memory usage.

To handle the full dataset efficiently, the CSV files are processed class by class in chunks. For every row that passes the length check, its spectrogram is appended to an in-memory list, and the corresponding label is encoded as 1 for Human and 0 for Non-Human. Once all files are processed, the list of spectrograms is converted into a NumPy array. If all spectrograms share the same dimensions, they are stacked into a single dense 3-D tensor with the shape (186,500, 1025, 18); otherwise, an object array is used to accommodate minor length variations. The labels are stored in a separate NumPy vector of type int64. Finally, spectrograms are saved in NumPy format, forming the pre-processed dataset for model training. Example spectrograms for both classes can then be visualized as time–frequency heatmaps, where color intensity reflects the magnitude.

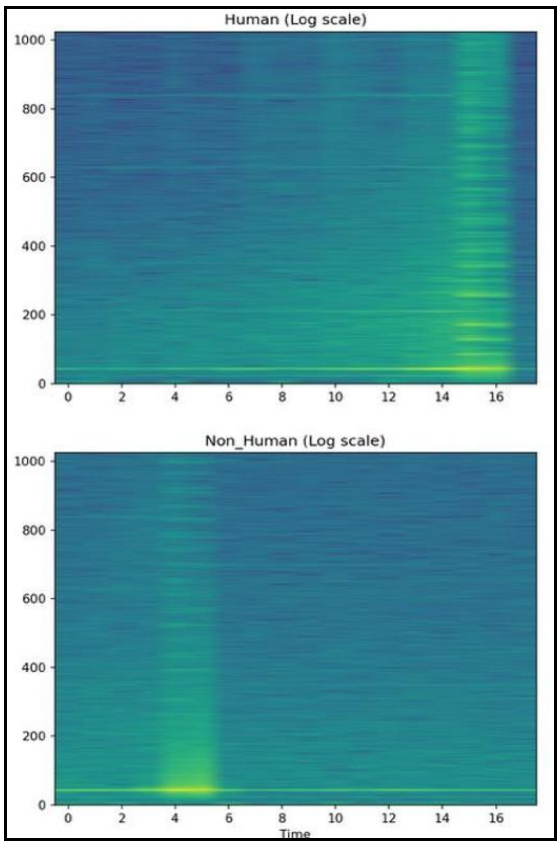

Fig. 6. Spectrogram images of both human and non-human sample.

C. *Machine Learning*

The preprocessed spectrogram images were fed into a 2D Convolutional Neural Network (CNN) SpectrogramCNN model to classify human and non-human subjects. This model was implemented using PyTorch, chosen for its flexibility and efficiency. The model architecture is presented below.

The input layer accepts a single-channel spectrogram with dimensions of 1025 × 18. This is followed by four convolutional blocks designed to extract features from the data.

Convolutional Blocks:

- Block 1: Consists of a 2D convolutional layer with 1 input channel and 32 output channels. This is followed by Batch Normalization, a ReLU activation function, a Max Pooling layer (2×2), and a Dropout layer (0.25).
- Block 2: Increases the depth to 64 output channels, followed by the same sequence of Batch Normalization, ReLU, Max Pooling, and Dropout.
- Block 3: Further increases the depth to 128 output channels with an identical layer structure.

- Block 4: Expands to 256 output channels. Unlike the previous blocks, this ends with an Adaptive Average Pooling layer (outputting a fixed 4×1 size) followed by Dropout.

Flatten Transition:

Converts the 3D blocks of features (Channels × Height × Width) into a single long vector (1D array) so the linear layers can process it.

Fully Connected Layers:

The output from the convolutional blocks is flattened and passed to a series of fully connected (Linear) layers:

- First Linear Layer: Maps the flattened features (1,024 units) to 512 units, followed by ReLU activation and a Dropout layer (0.5).
- Second Linear Layer: Maps 512 units to 128 units, followed by ReLU activation and Dropout (0.5).
- Output Layer: Finally, the last linear layer maps the 128 units to 2 outputs, corresponding to the two classes: Human and Non-human.

The figures below illustrate the visual representation and detailed architecture of the CNN model.

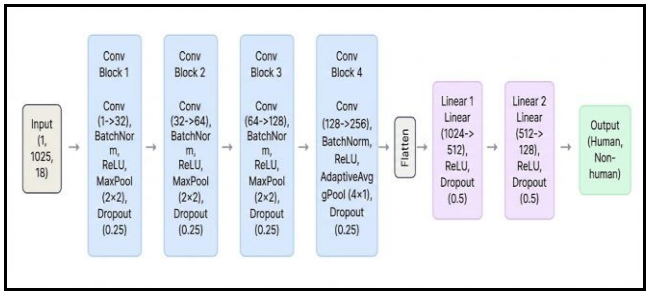

Fig. 7. Visual Representation of SpectrogramCNN model.

SpectrogramCNN Model Architecture

Layer Group	Layer (Type)	Details	Output Shape (Batch Size, C, H, W)	Parameters
Input	-	1-channel Spectrogram	(N, 1, 1025, 18)	0
conv_block1	Conv2d	32 filters, kernel 3x3, stride 1, padding 1	(N, 32, 1025, 18)	320
	BatchNorm2d	Epsilon: 1e-05, Momentum: 0.1	(N, 32, 1025, 18)	64
	ReLU	-	(N, 32, 1025, 18)	0
	MaxPool2d	Kernel 2x2, stride 2	(N, 32, 512, 9)	0
conv_block2	Conv2d	64 filters, kernel 3x3, stride 1, padding 1	(N, 64, 512, 9)	18,496
	BatchNorm2d	Epsilon: 1e-05, Momentum: 0.1	(N, 64, 512, 9)	128
	ReLU	-	(N, 64, 512, 9)	0
	MaxPool2d	Kernel 2x2, stride 2	(N, 64, 256, 4)	0
conv_block3	Conv2d	128 filters, kernel 3x3, stride 1, padding 1	(N, 128, 256, 4)	73,856
	BatchNorm2d	Epsilon: 1e-05, Momentum: 0.1	(N, 128, 256, 4)	256
	ReLU	-	(N, 128, 256, 4)	0
	MaxPool2d	Kernel 2x2, stride 2	(N, 128, 128, 2)	0
conv_block4	Conv2d	256 filters, kernel 3x3, stride 1, padding 1	(N, 256, 128, 2)	295,168
	BatchNorm2d	Epsilon: 1e-05, Momentum: 0.1	(N, 256, 128, 2)	512
	ReLU	-	(N, 256, 128, 2)	0
	AdaptiveAvgPool2d	Output size (4, 1)	(N, 256, 4, 1)	0
-	Flatten	-	(N, 1024)	0
fc_block1	Linear	In: 1024, Out: 512	(N, 512)	524,800
	ReLU	-	(N, 512)	0
	Dropout	p = 0.5	(N, 512)	0
fc_block2	Linear	In: 512, Out: 128	(N, 128)	65,664
	ReLU	-	(N, 128)	0
	Dropout	p = 0.5	(N, 128)	0
Output	Linear	In: 128, Out: 2 (human/non-human)	(N, 2)	258
			Total Trainable Parameters:	979,522

Fig. 8. Detailed Architecture of SpectrogramCNN model.

The model design prioritizes resource efficiency. It is a lightweight architecture optimized for fast training and inference, containing a total of 979,522 trainable parameters.

D. Project Architecture

When the sensor system is initialized via the human detection Python application, an SSH connection is established to the Red Pitaya OS, and the data acquisition program (dma_with_udp_faster) is executed. After a brief initialization pause, the application performs a UDP handshake with the server using the -i 1 command. Once the handshake is successful, the system enters a continuous data acquisition and processing loop. A UDP message is sent to the Red Pitaya to request the most recently acquired sensor data. In response, the system returns a packet containing a header (with metadata for distance calculation) and 25,000 raw ADC samples. This data is received by the Python application, plotted on the GUI, and prepared for processing. This loop strictly controls the throughput, ensuring exactly two valid signals are processed per second to optimize performance. This given figure delineates the workflow of the whole system.

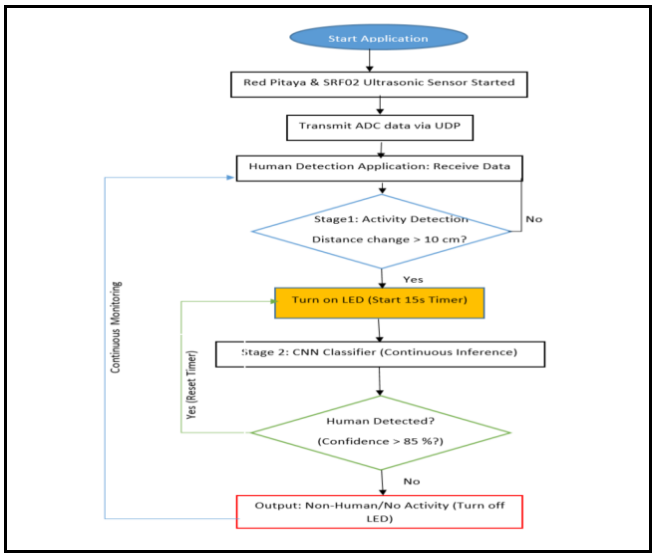

Fig. 9. Workflow of the whole system.

The application employs a two-stage detection cycle: first, activity is detected based on distance changes, and second, the system is sustained using deep learning. A worker thread continuously processes the incoming raw signals.

In the "Activity Detection" stage, the physical distance of the object is calculated by extracting the dmax value from the packet header and converting it to centimeters. The current distance is then compared to the previous reading. If the distance changes by more than a configurable threshold (default: 10 cm), the event is classified as "Activity," and an LED is activated immediately.

The LED control is executed within the Red Pitaya OS by writing the hexadecimal value 0x80 directly to the hardware memory register 0x40000030. This activates the GPIO pin connected to LED7. Simultaneously, a 15-second internal timer is initiated.

Once the system is active, a continuous "CNN Classification" mode is engaged to verify human presence. Valid signals are converted into spectrograms using the Short-Time Fourier Transform (STFT) with a Hamming window and processed by the PyTorch 2D CNN model. This

model analyzes spectral features to output a probability score.

When the initial 15-second timer expires, the current CNN prediction is evaluated. If "HUMAN" presence is confirmed with sufficient confidence (default: >85%), the timer is reset to maintain illumination. Conversely, if "NON-HUMAN" is predicted, a command is sent to deactivate the LED. This logic ensures the light remains active only while human presence is verified by the deep learning model.

E. Graphical User Interface (GUI)

This section describes the Graphical User Interface (GUI) of the Python application, which is built using the PyQt6 framework to provide a simple dashboard for monitoring the system. The top part of the window features a large graph with a black background that plots the live raw signal received from the sensor, allowing the user to visually inspect the data waveform. Fig 10 shows the GUI data waveform graph.

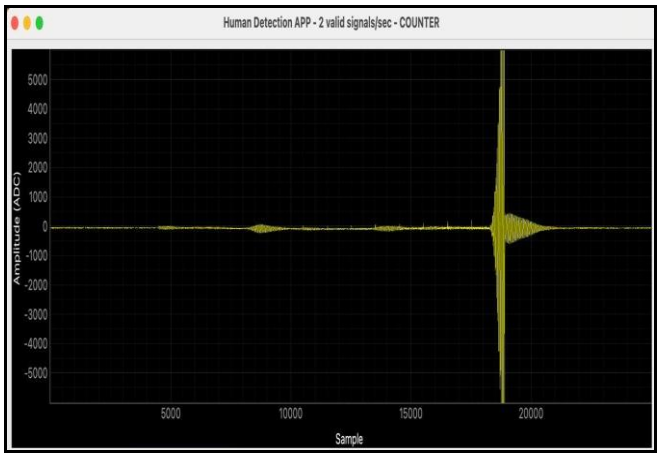

Fig. 10. Data waveform graph of the GUI application.

In Fig 11, below the data waveform graph, the "Primary Status" area displays the most important information using large, clear indicators. There are two big buttons labeled "HUMAN" and "NON-HUMAN" that act as visual alerts which is initially shows Grey color; the "HUMAN" button blinks green when a person is detected, and the "NON-HUMAN" button blinks red when the signal is classified as non-human. Next to these buttons, the interface shows the current status of the hardware LED (on or off) and LED stays on timer count, the specific distance of the object in centimeters, and the confidence percentage (probability) of the signals that is human or non-human. There is also an "Activity" label that changes from "IDLE" (grey) to "ACTIVE!" (orange) whenever the system detects significant movement based on distance threshold.

The "Detection Settings" section allows the user to adjust the system's sensitivity by changing the "Distance Threshold." Change in the distance gap between the sensor and the object is considered for activity detection. The user can enter a value in centimeters (default is 10 cm), and the system will use this to decide how much change in distance gap is considered as an activity. After that, the user can adjust the LED timer in seconds (initially 15s) to control how long the LED stays on. This section also includes a rate monitor that displays the actual processing speed to ensure the system is handling exactly two valid signals per second. Distance is taken from the packet header.

At the bottom of the window, It provides the information about the application is connected with red pitaya UDP server or not and the loaded model with accuracy and also detailed statistics and system messages. It displays running counters for the total number of "Human," "Non-Human," and "Uncertain" detections since the session started. It also shows the time it took for the model to make the last prediction (inference time) and tracks the number of "Broken" signals to warn the user about data loss. The entire system is controlled by two main buttons at the very bottom: "Start Sensor Detection" to begin the connection and analysis, and "Stop Sensor Detection" to end the process.

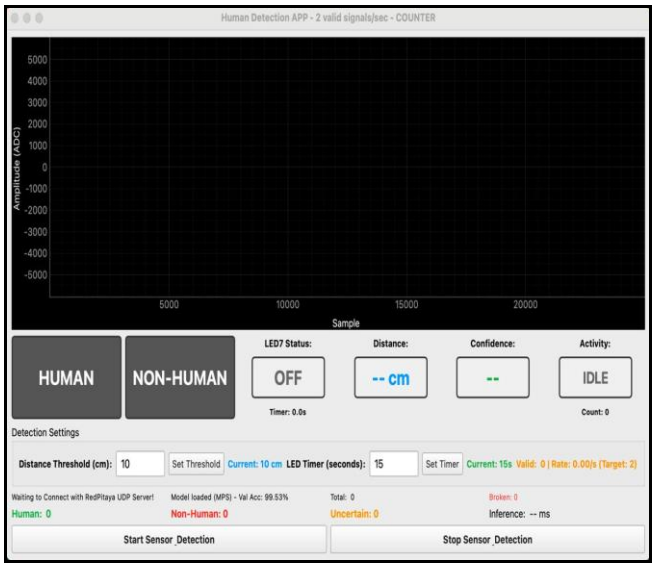

Fig. 11. Detection monitoring and settings of GUI.

The User Interface (UI) in fig 12 confirms a successful connection between the client application and the Red Pitaya UDP server. The graph visualizes the real-time plot of incoming ultrasonic signals, highlighting that valid non-human signal samples are successfully received and processed.

Fig. 12. Non-human (Floor static environment) detected where led off.

Fig 13 shows our UI when a non-human object (a chair) has been pushed under the sensor, and the LED turns on due to the detective activity.

Fig. 13. Detection of a Non-Human object (a chair) with the LED ON due to significant distance change.

Fig 14 clarifies the sequence of events: the system initially tracked a static environment (non-human data). When a person suddenly entered the sensor's range, the significant distance change triggered the "Activity" state. Simultaneously, the CNN accurately classified the signal as "Human," which caused the LED7 to remain continuously ON.

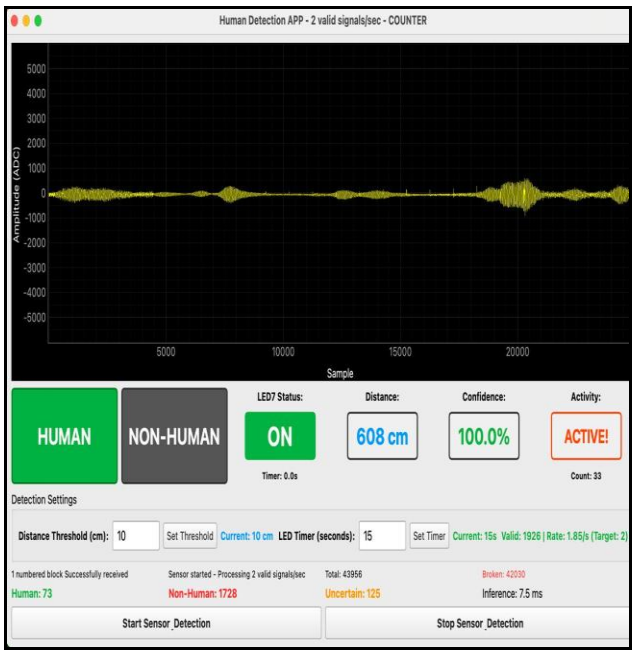

Fig. 14. Human detected GUI with continueous LED ON.

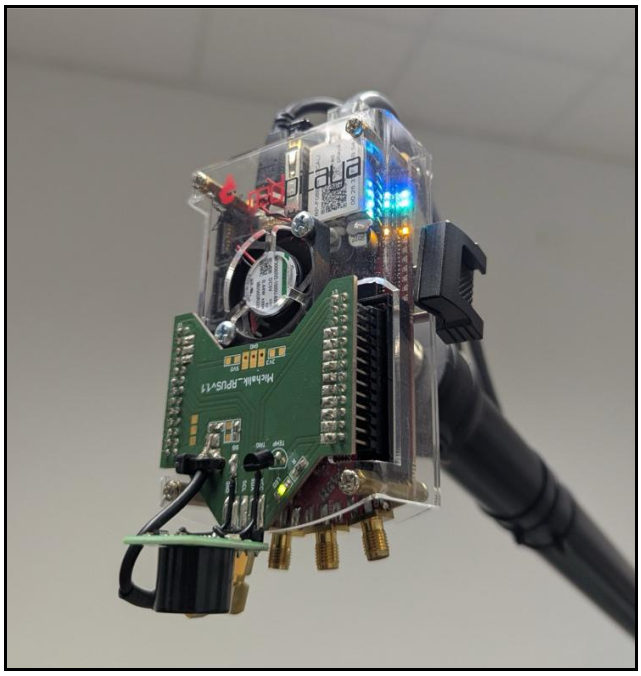

Fig. 15. LED (yellow light) turns on when HUMAN is detected.

III. RESULTS

The SpectrogramCNN model was evaluated using a randomized 80/20 split on the spectrogram dataset, designating 149,200 signals for training and 37,300 signals for validation. Training was conducted over 15 epochs, requiring approximately 76 minutes. The training dynamics indicated a robust learning process; the model achieved a validation accuracy of 99.53% with a minimal validation loss of 0.0104 at epoch 14, which was selected as the optimal checkpoint. The consistent decrease in loss paired with stable validation accuracy suggests that the weighted loss function effectively mitigated class imbalance, preventing model overfitting.

The classification Accuracy performance on the validation set of 37,300 signals, confirmed the system's high precision. As detailed in the confusion matrix (see Figure 16), the model demonstrated exceptional discriminative capability:

- True Positives: 19,647 Human signals were correctly identified.
- False Positives: Only 41 Non-Human signals were incorrectly classified as Human.
- True Negatives: 17,476 Non-Human signals were accurately rejected.
- False Negatives: 136 Human signals were missed.

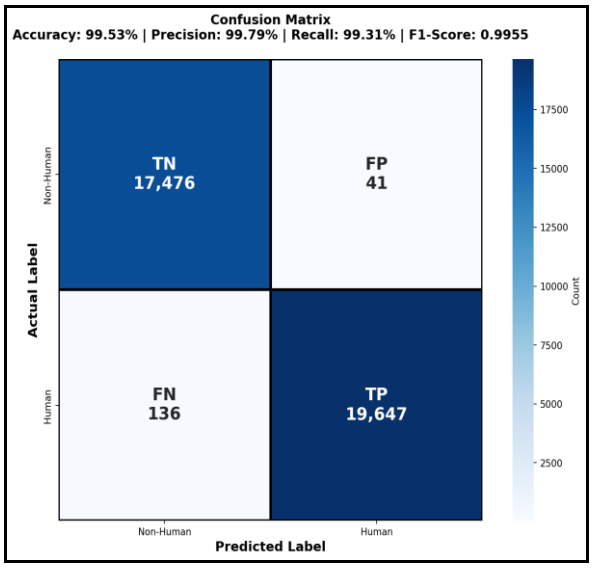

Fig. 16. Detailed confusion matrix.

Quantitatively, the model achieved a precision of 99.79% and a recall of 99.31% for the Human class. The low False Positive rate (41 out of 17,517 non-human signals) indicates the system's high reliability for positive identification. To assess deployment robustness, we analyzed the model's prediction confidence. Results indicate that 98.47% of the validation signals were predicted with a confidence score exceeding a 95% threshold. Within this high-confidence subset, the model achieved an accuracy of 99.96%. This suggests that the model possesses strong generalization capabilities and is well-suited for real-time deployment where high certainty is required.

The system's performance was validated through a series of trials. It successfully detected human presence in every instance, activating the LED on the Red Pitaya board. The system also proved effective at distinguishing targets; while moving inanimate objects (like a chair) into the zone triggered an initial reaction, the system correctly classified them as non-human and turned off the LED.

The table below illustrates the accuracy metrics for the real-time detection of humans and non-humans. The system demonstrated consistent classification reliability. In the real-time trials, six individuals were detected, consisting of four unknown and two known subjects.

TABLE I. REALTIME TESTING RESULT.

NR.	Data type Actual	Classified as Human	Classified as Non-Human
1	Floor	No	Yes
2	Human 1	Yes	No
3	Chair	No	Yes
4	Human 2	Yes	No
5	Human 3	Yes	No
6	Human 4	Yes	No
7	Table	No	Yes
8	Human 5	Yes	No
9	Human 6	Yes	No

IV. DISSCUSSION

The two-step system performs well in real time testing, striking a balance between energy saving and accurate detection. By using the simple distance check first, the system only turns on the powerful CNN model analysis when it sees movement, which saves power. However, there are still some issues. The model sometimes gets confused by some unseen objects that are soft or shaped like humans, such as backpack or chairs with jacket on them. It also struggles with partial views of chairs entering the sensor area and inference from the floor, which causes the LED to stay on unintentionally. Additionally, the system was not tested in completely new environments, so its performance in different room layouts is unknown. Future updates aim to teach the model to recognize a wider variety of objects and use simpler methods to boost accuracy while maintaining system speed and simplicity.

REFERENCES

[1] P. Chaudhari, Y. Xiao, M. M.-C. Cheng, and T. Li, "Fundamentals, Algorithms, and Technologies of Occupancy Detection for Smart Buildings Using IoT Sensors," Sensors, vol. 24, no. 7, pp. 2123–2123, Mar. 2024, doi: https://doi.org/10.3390/s24072123

[2] P. Peace, J. Chris, and L. victor, "Data Preprocessing for AI Models," Nov. 11, 2024. https://www.researchgate.net/publication/385707249.