# MARVIN: Adaptive AI Companion for Personalized Human-Robot Interaction

Md. Rakibul Hasan[1†], Sudipa Saha[1†], Md. Mamun Hossain[1], A B M Shawkat Ali[1], Md. Mahmudul Hasan Maruf[1]

[1]Department of Computer Science and Engineering, Bangladesh University of Business and Technology (BUBT), Dhaka, 1216, Bangladesh.

Contributing authors: rakibhasan@bubt.edu.bd; sahadipa1999@gmail.com; mamunhasan.cs@gmail.com; mamunhasan.cs@gmail.com; maruf77663@gmail.com;
[†]These authors contributed equally to this work.

**Abstract**

This paper introduces MARVIN (Multi-faceted Adaptive Robotic Virtual Intelligence Nucleus), an innovative robotic pet designed to redefine human-robot interaction. Inspired by the companionship of traditional pets, MARVIN integrates advanced sensors (visual, auditory, tactile) for responsive environmental perception, health monitoring, and energy efficiency. With reinforcement learning, it continuously adapts its behavior to align with user preferences, expressing emotions and engaging in meaningful conversations via natural language processing.It can also control smart home devices through natural commands. This paper addresses challenges in creating a "living" pet robot and discusses ethical and societal implications. Future work aims to advance autonomous navigation, emotion recognition, and environmental awareness.

**Keywords:** Intelligent Companion, User Recognition, Behavior Adaptation, Machine Learning, Natural Language Processing, Human-Robot Interaction, Personalization, Adaptive Systems, Autonomous Systems, Voice Recognition, Smart Home Automation, Social Robotics, Multi-Modal Interaction, Deep Learning, Human Activity Recognition.

# 1 Introduction

The field of robotics has made impressive strides in recent years, producing robots capable of performing complex tasks with increasing precision and efficiency. However, despite advancements in artificial intelligence (AI) and machine learning, current robotic systems often fall short in providing emotionally intelligent interactions that foster genuine companionship. Existing research in robotic companionship, such as the work of Breazeal et al. (2021) on sociable robots, and Dautenhahn (2018) on human-robot interaction, primarily focuses on task-oriented performance rather than the development of deep emotional connectivity. These systems tend to rely on pre-defined scripts and responses, limiting their ability to adapt dynamically to individual user emotions, preferences, and environmental cues, which results in interactions that often feel predictable and lacking in authenticity.

**MARVIN** (**M**ulti-faceted **A**daptive **R**obotic **V**irtual **I**ntelligence **N**ucleus) is designed to address these gaps by incorporating a unique fusion of adaptive learning, sensory integration, and emotional intelligence, making it a pioneering example in emotionally responsive robotics. Powered by advanced deep and reinforcement learning techniques, MARVIN's system can express a spectrum of emotions—ranging from joy and curiosity to nuanced states like frustration or confusion. Unlike current robotic companions, MARVIN's emotionally rich state machine is built to respond dynamically to user interactions and environmental stimuli through voice, touch, and visual cues, creating a contextually aware, lifelike engagement. The research presented in this paper explores the architecture, emotional model, and adaptive behavior that underpin MARVIN, highlighting innovations that distinguish it from existing robotic companionship models.

The current market for robotic companions, often targeted at alleviating loneliness, especially in aging populations and urban settings, still lacks affordability, emotional adaptability, and ethical integration. Studies, such as Kuo et al. (2022) on assistive social robots and Bartneck et al. (2020) on human perceptions of robotic emotion, underline these limitations, pointing out that most robotic pets today remain task-focused with minimal emotional responsiveness. MARVIN addresses this gap by delivering a highly interactive, adaptive robotic companion capable of meaningful emotional exchange. Through its personalized learning system, MARVIN seeks to elevate the standard of human-robot interaction,

providing users with a new level of companionship that responds to emotional and contextual nuances.

This research advances the field of pet robotics by addressing key limitations in adaptability, emotional intelligence, real-time responsiveness, and personalization. MARVIN incorporates parallel task processing capabilities to enhance operational speed and responsiveness, and introduces the following contributions:

- **Enhanced Emotional Intelligence:** MARVIN's emotion engine is designed to express a wide range of emotions, allowing for authentic and personalized interactions. This feature addresses the limited emotional expressiveness commonly seen in existing pet robots, fostering a more engaging user experience.
- **Adaptive Learning through Reinforcement:** By utilizing reinforcement learning, MARVIN adapts to individual user preferences and behavior patterns over time. This adaptive capability overcomes the static interaction models found in many contemporary robotic companions, resulting in progressively more personalized and intuitive interactions.
- **Advanced Natural Language Processing (NLP):** With sophisticated NLP capabilities, MARVIN can engage in meaningful, context-aware conversations. This feature surpasses the command-based communication systems in most existing pet robots, providing users with a richer and more conversational experience.
- **Comprehensive Sensor Integration:** MARVIN is equipped with a wide array of sensors—including visual, auditory, and tactile—which enable it to respond dynamically to environmental changes. This sensory integration, coupled with parallel task processing using FreeRTOS and C++ for real-time efficiency, significantly enhances situational awareness and enables faster, adaptive responses.
- **Parallel Task Processing for Optimization:** By leveraging FreeRTOS to manage tasks across multiple cores, MARVIN can execute tasks in parallel, optimizing processing speed. This parallelism addresses a critical limitation in traditional pet robots that process tasks sequentially, leading to faster response times and more fluid interactions.
- **User Experience Evaluation:** This research includes extensive user testing to evaluate MARVIN's effectiveness in companionship, entertainment, and assistance. The findings will provide insights into the benefits and limitations of an emotionally intelligent and adaptive robotic companion, contributing to an improved understanding of practical applications in this field.
- **Ethical Design Considerations:** The development of MARVIN includes a focus on ethical considerations, such as privacy, consent, and psychological impact. Ensuring respect for user autonomy and well-being, MARVIN's design addresses crucial ethical aspects often overlooked in robotic companion development.

Through these contributions, MARVIN represents a major advancement in pet robotics by combining parallel task processing, FreeRTOS, C++,

and reinforcement learning to deliver a responsive, adaptive, and ethically designed robotic companion.

In summary, MARVIN contributes to the evolving field of robotics by enriching the emotional and interactive dimensions of robotic companionship, moving beyond traditional task-oriented frameworks. Developed in C++ for efficiency and performance, MARVIN's architecture includes an emotion engine and behavioral framework that encapsulate a wide range of human emotions, such as playfulness and empathy. This research strives to create a more holistic, engaging, and ethically responsible model for pet robotics, integrating real-time processing with emotionally attuned behavior.
This work ultimately aims to bridge the gap between utilitarian robotic design and emotionally intelligent, context-aware human-robot interaction. By doing so, it sets a new standard in the field of pet robotics, promoting a future where robotic companions can genuinely enrich the lives of their users through nuanced and adaptive interactions.

## 2 Background Study

Recent advancements in robotics research have focused on enhancing autonomy, adaptability, and emotional intelligence in robotic systems across various domains such as social interaction, smart home integration, agriculture, and human-robot interaction. While motivation-driven decision-making, natural language processing, and IoT integration have improved robotic capabilities, limitations remain. These include challenges in achieving dynamic adaptation, comprehensive contextual understanding, and realistic emotional expression.
Existing systems often face restrictions like fixed parameters, limited sensor integration, and difficulties in real-world adaptability. Addressing these gaps, our research on MARVIN introduces a multi-model robotic companion.

### 2.1 Literature Review

In the course of our research, we systematically reviewed 25 papers, synthesizing a comprehensive literature review based on the insights garnered from this corpus of scholarly works.

**Table 1**: Previous research works in terms of objectives, used methods, and possible research gaps

| **Title** | **Research Purpose** | **Used Methods & Tools** | **Contribution** | **Research Gaps** |
|---|---|---|---|---|
| Selection of Actions for an Autonomous Social Robot [1] | Develop an external USB Braille keyboard for visually impaired users. This keyboard is designed for computers, focusing on accessibility and usability for Braille input. | Motivation-driven decision-making, Utilizes Lorentz's motivation model, Computes internal drives and external states, Selects actions based on dominant motivation, Potential for reinforcement learning. | Contributes by proposing a motivation-based decision system, integrating internal needs and external stimuli, emphasizing autonomy and survival in robot behavior, applying Lorentz's motivation model for decision-making, and demonstrating improved autonomy in a social robot. | Limitations include a simplified model, fixed parameters, limited consideration of emotional factors, dependency on predefined drives, and challenges in predicting real-world interactions. |
| Integration of service robots in the smart home by means of UPnP: A surveillance robot case study [2] | Integration home appliances through a single robot, Integration of service robots in the smart home | Deep Learning , Speech Recognition Systems,Natural Language Processing (NLP) | Focused to Implementing Speech Recognition System,Integration of Facial Expression and Natural Interaction,Improvement in Accuracy and Previous Research | Language Limitation,Limited Knowledge Sources,Less Accuracy and Reliability,Limited Emotional Expression,Limited Contextual Understanding of complex questions. |
| Development of an integrated IoT-based greenhouse control three-device robotic system [3] | Developing a solar-powered cablebot equipped with a spectroscopic camera and an array of other sensors,Designing and developing a four-legged ground robot (agbot) | IoT,cable-driven parallel robot,cablebot,sensor box,interface app | Focused on Intregating Three-Device System (3DS),Development of Autonomous Devices,IoT Integration,Real-World Implementation and Data Collection | On-site testing with real measurements is not possible,The system is not fully functional in open area environments for low temperatures,The degrees of movement of the robot are not perimeter movement in space. |
| Deep Learning and Sentiment Analysis for Human-Robot Interaction [4] | Building an autonomous humanoid robot,Apply deep learning and nlp within the field of Human-Robot Interaction, Train the robot to reply to a natural language question | Deep learning, Sentiment analysis, Word embeddings,Natural language processing,Generative Conversational Agent (GCA),QiChat | Focused on Implementing feature extraction and classification,Implementation of sentiment analysis system | Limited Scope of Sentiment Analysis, Real-world Environment Challenges,Limited Dialogue Complexity. |

### 2.2 Hardware Requirements

MARVIN's design incorporates a diverse set of hardware components carefully selected to meet the demands of its advanced functionalities. The central processing unit is the ESP32-S3, a versatile microcontroller known for its dual-core processing capabilities, built-in Wi-Fi and Bluetooth support, and secure, low-power design. This microcontroller is well-suited for IoT applications and provides ample processing power for multitasking while ensuring efficient energy use. For visual data processing, the ESP32S3-CAM, which features a 2-megapixel camera capable of capturing images for various computer vision tasks, alongside a micro-SD slot for local storage. This module enables engage in real-time image processing, an essential component of its face recognition and detection capabilities. In addition to visual processing, MARVIN includes a 1.8-inch TFT display with a resolution of 128x160 pixels, providing clear visual feedback to users. This display is integral to MARVIN's ability to convey emotions and statuses, offering an interactive and expressive interface. For physical expression and movement, MARVIN integrates a servo motor, which allows for precise angular positioning through pulse-width modulation. This component enables MARVIN to execute subtle gestures, enhancing the robot's ability to interact naturally with its environment.

Audio input and output are managed by an I2S microphone and an 8-ohm, 3-watt speaker, respectively. The I2S microphone is designed to capture high-quality digital audio signals, which are processed for voice recognition and interaction tasks. This microphone, coupled with the speaker, allows MARVIN to engage in verbal communication, responding to commands and delivering audio feedback, creating a more immersive user experience. Additionally, a touch sensor is included for direct user interaction, translating tactile inputs into electrical signals that trigger corresponding actions within MARVIN's operating system.

To support movement analysis and environmental awareness, MARVIN is equipped with a gyroscopic sensor with nine degrees of freedom. This sensor enhances MARVIN's ability to understand its spatial orientation and movements, which is critical for responsive interaction and stability in dynamic settings. Together, these hardware components form a cohesive and responsive system that enables MARVIN to perform a wide range of tasks, from emotional expression to environmental interaction, laying the groundwork for a versatile and adaptive robotic companion.

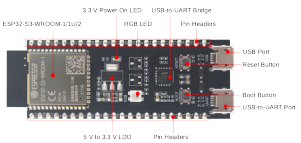
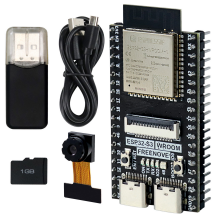
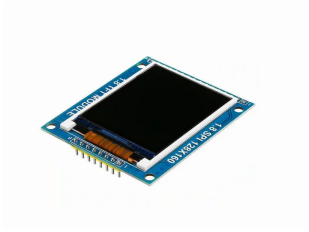
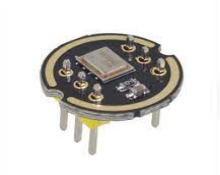
| Component | Description | Specifications |
|---|---|---|
| ESP32-S3[1]  | Main processing unit with powerful CPU, built-in security, Wi-Fi/Bluetooth, and low-power design, ideal for IoT applications. | CPU: Xtensa LX7, RAM: 128-512 KB, Flash: 4-16 MB |
| ESP32-CAM  | Visual input with a 2-megapixel camera, dual-core processor, Wi-Fi/Bluetooth, and micro-SD slot for storage. | Camera: 2 MP, Storage: Micro-SD supported |
| TFT_Display  | 1.8-inch, 128x160 resolution IPS display, ideal for showing emotions with full viewing angles. | Display: 128x160, Driver IC: ST7735S |
| Servo_Motor  | Rotary actuator with precise angular positioning using PWM, operates in 4.8-6V range, ideal for robotic movement. | Voltage: 4.8-6V, Torque: 60-300 RPM |
| I2S_Microphone  | Voice input for sound capture using I2S interface, providing high-fidelity digital audio signals. | Interface: I2S, Application: Voice recognition |
| Speaker  | Speaker for audio output, converts electrical signals to sound; | Speaker:8ohm 3W |
| Touch_Sensor   | Touch_Sensor for sensitive input, converts human touch signals to electrical signals; | Touch:Resistive |
| Gyroscopic   | Gyro for 3 dimensional input, used for movement analysis; | Gyro:9 DOF |



**Table 2**: Hardware Requirements for MARVIN

## 3 Methodology

The fig-1 architecture integrates multiple components to create an adaptive and responsive robotic system capable of interacting intelligently with users. The methodology involves two primary blocks: Robotics and Internet of things.

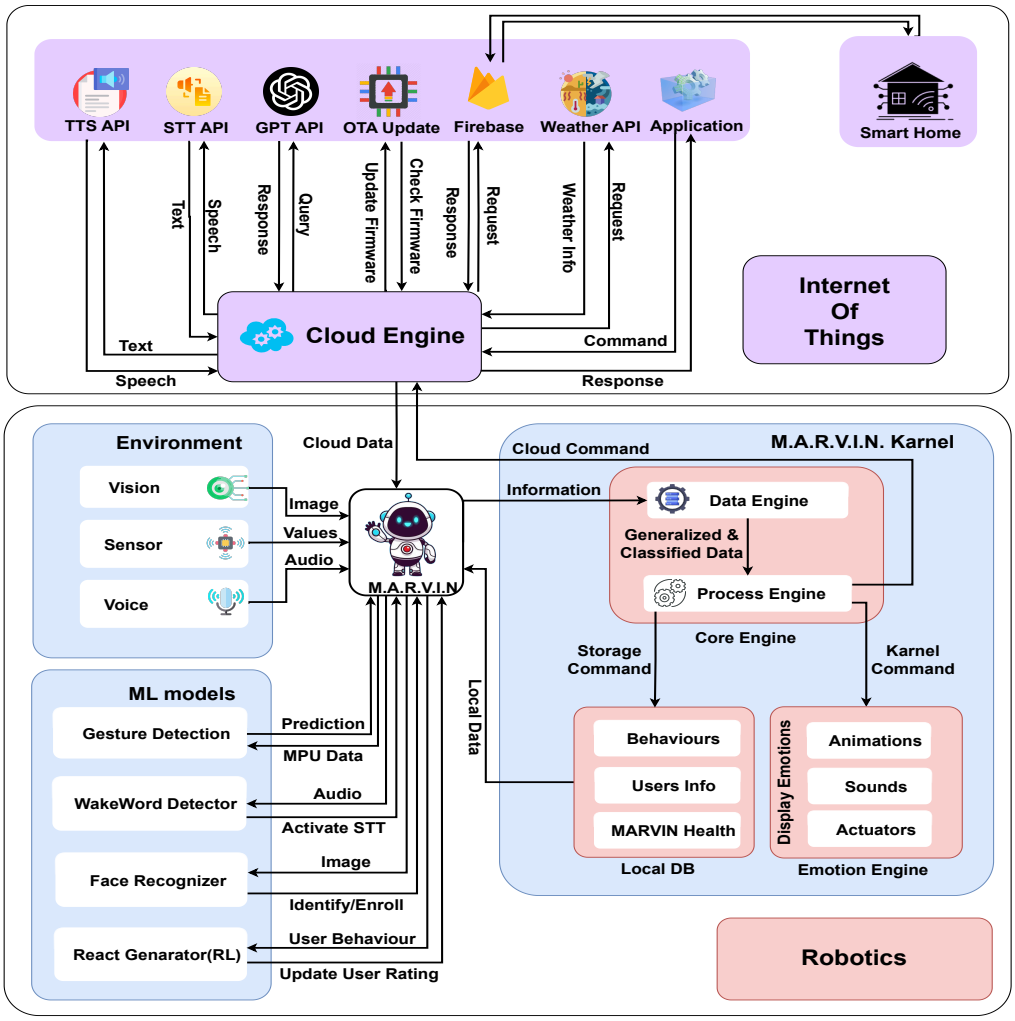



**Fig. 1**: Methodological Diagram

## 3.1 Robotics Block

This block is composed of three distinct modules. The Environment Module gathers Vision, auditory, and sensory data. The ML Module processes movement analysis, wake word detection, face recognition, and reaction generation. Finally, the Kernel Module generalizes and processes all incoming information, ensuring seamless integration and decision-making within the system.

### 3.1.1 Environment Module

The Environment Module collects three types of sensory data: vision, auditory, and physical interactions. Vision input from the camera is used for facial recognition, auditory input from the microphone enables wake word detection and conversation, and sensory data from touch and IMU sensors facilitate gesture and interaction detection. These inputs are processed using specialized machine learning models, including a Wake Word Detector for voice-activated system control, a Face Recognizer for user identification and personalized interactions, and a React Generator to adapt responses based on user feedback and interaction history.

### 3.1.2 MARVIN Kernel Module

It is the core of the system, which includes the Data Engine, Core Engine, Local Database, and Emotion Engine. The Data Engine classifies and organizes incoming information, making it readily accessible for processing by the Core Engine. This core engine executes commands, communicates with the local database, which contains user data, behavioral definitions, and system health metrics, and orchestrates tasks across the subsystems. The Emotion Engine further enhances the user experience by generating emotional responses through animations, sounds, and actuator-driven movements, making interactions more engaging and lifelike.

### 3.1.3 ML Module

The multimodal ML Module integrates vision, auditory, and motion-based inputs to enhance system intelligence and adaptability. Gesture Detection 3.6 leverages MPU sensor data and touch sensor data to predict user movements, and action enabling intuitive interaction. The Wake Word Detector 3.3 processes audio signals to identify activation commands, triggering speech-to-text functionality. Face recognition 3.5

utilizes image data for user authentication and enrollment, facilitating personalized experiences. Additionally, the React Generator, driven by an advanced emotion algorithm, dynamically adapts system responses based on user behavior and interaction history, continuously refining personalization through an adaptive user rating mechanism. All the models are discussed below.

### 3.2 IoT Block

This plays a crucial role in enabling seamless communication between the Cloud Engine and connected devices, such as smart home systems and robotics. It processes and executes commands received from the Cloud Engine, facilitating real-time automation and control. Through various cloud-based APIs, including speech-to-text, text-to-speech, artificial intelligence-based decision-making, over-the-air firmware updates, and weather data retrieval, the IoT module ensures context-aware and intelligent responses. This integration enhances user experience by allowing connected devices to operate efficiently, adapt to environmental conditions, and provide personalized interactions based on processed data. The IoT module ensures a cohesive connection between cloud intelligence and physical devices, making the system more responsive and adaptive.

Overall, MARVIN employs a combination of real-time local data processing and cloud-based resources to create an intelligent, context-aware companion that adapts to user behavior. This hybrid approach ensures high responsiveness, continuous learning, and personalization, allowing it to evolve dynamically based on interaction patterns and feedback, thus delivering an emotionally resonant and functional robotic experience.

### 3.3 Wake Word Detection Model

The wake word detection system[5] was designed to recognize the keyword "MARVIN" from real-time audio input using a combination of digital signal processing and deep learning models. The methodology consists of three main components: Data Acquisition Block, Training Block, and Deployment Block.

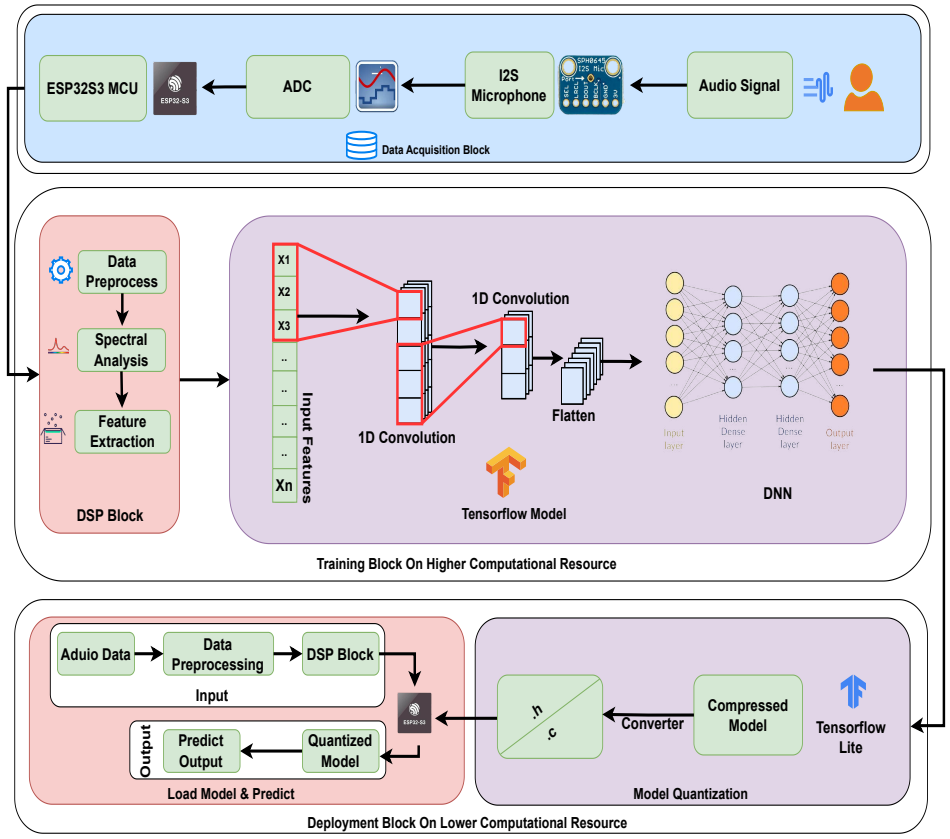



**Fig. 2**: Wakeword Model

### 3.3.1 Data Acquisition Block

The system utilizes an ESP32-S3 microcontroller unit as the core processing unit. The wake word detection pipeline begins with capturing raw audio input through an I2S microphone. The microphone's analog signal is converted to a digital signal using an Analog-to-Digital Converter (ADC) before being processed by the MCU. This raw audio data is then stored and sent for further processing.

### 3.3.2 Training Block on High Computational Resources

Before training the model, the raw audio data undergoes preprocessing to extract relevant features. The DSP module consists of two key operations:

**i)Spectral Analysis:** Converts the time-domain audio signal into a frequency-domain representation, helping to analyze speech components.

**ii)Feature Extraction:** Extracts Mel-Frequency Cepstral Coefficients, which capture essential speech characteristics. These extracted features serve as input for training the wake word detection model.

The wake word detection model was trained using Edge Impulse, leveraging convolutional neural networks (CNNs). The model training follows these steps:

- The extracted MFCC features are fed into a 1D Convolutional Neural Network (CNN). The first convolutional layer detects spatial patterns in the input feature maps.
- The processed output is passed through additional 1D convolutional layers to refine feature representation, capturing deeper speech characteristics.
- The feature maps are flattened before being processed by a Deep Neural Network (DNN) with multiple hidden layers.
- The final output layer performs binary classification, indicating whether the wake word is detected or not.

To improve model robustness, data augmentation techniques such as pitch shifting and background noise addition were applied during training. The model was then fine-tuned for accuracy, ensuring reliable wake word detection in real-world scenarios.

### 3.3.3 Deployment Block on Lower Computational Resources

Once trained, the quantized model is optimized for deployment on resource-constrained hardware. The process includes:

**i)Model Quantization:** The trained TensorFlow model is compressed using TensorFlow Lite, reducing computational overhead while maintaining detection accuracy.

**ii)Deployment on ESP32-S3:** The quantized model is loaded onto the ESP32-S3, where it continuously listens for audio input.

**iii)Wake Word Prediction:** The ESP32-S3 runs real-time inference, where incoming audio data is preprocessed and passed through the quantized model. If the model's confidence score exceeds a predefined threshold, the system triggers an activation response, allowing further voice-based commands.

This structured pipeline ensures an efficient, low-latency, and reliable wake word detection system for MARVIN, enabling voice-based interactions while maintaining power efficiency.

### 3.4 Speech To Text with Voice Activity Detection

Voice Activity Detection (VAD) is a technique used to determine whether a segment of audio contains speech. As shown in Figure 3, the methodology implemented in the VADCoreESP32 library[6], which uses an ESP32 microcontroller to process audio data from an I2S microphone, apply Fast Fourier Transform (FFT), and detect speech based on energy thresholds within a specific frequency range (300 Hz – 3400 Hz). The system initializes by configuring the I2S interface to receive audio data with specified parameters, including sample rate, bit depth, and buffer settings. To enhance low-amplitude speech signals, a gain factor is applied while ensuring that values remain within the 16-bit integer range to prevent distortion. The FFT transformation converts the time-domain audio signal into the frequency domain, where energy levels within the speech frequency range are analyzed. Speech is detected when the computed energy surpasses a predefined threshold. The VAD process runs continuously in a task loop, monitoring elapsed time and stopping if no speech is detected within a specified period. It utilizes FreeRTOS for efficient task scheduling, allowing dynamic adjustments based on speech activity. This implementation provides a robust and efficient solution for speech-to-text applications, smart assistants, and voice-controlled systems by integrating FFT-based analysis, gain adjustment, and dynamic time management.

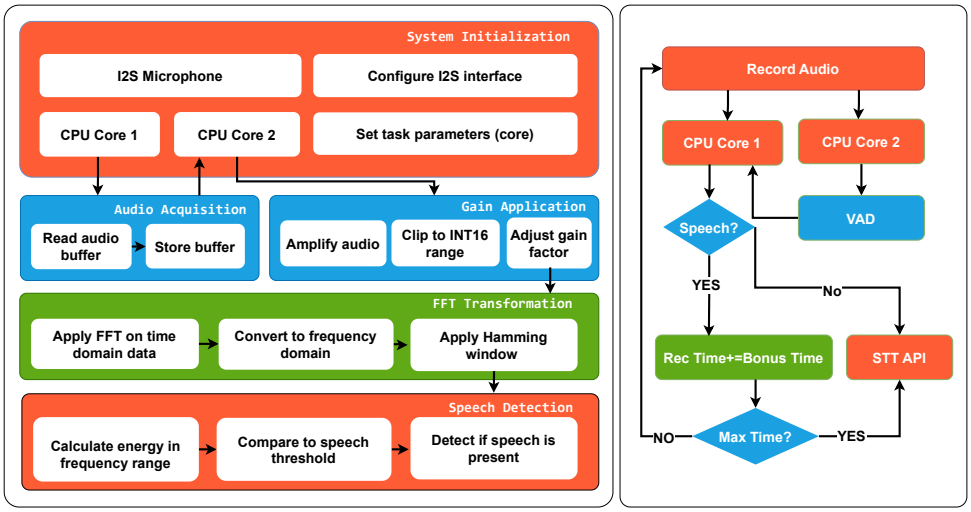



**Fig. 3**: Voice Activity Detection

**Algorithm 1** I2S Initialization

**Require:** $i2sPort$, $i2sBckPin$, $i2sWsPin$, $i2sDataPin$

**Define I2S Configuration:**
  Set mode ← `MASTER | RX`
  Set sample rate ← `VAD_SAMPLE_RATE`
  Set bits per sample ← `16-bit`
  Set channel format ← `ONLY_LEFT`
  Set communication format ← `I2S | I2S_MSB`
  Set interrupt allocation flags ← `LEVEL1`
  Set DMA buffer count ← `8`
  Set DMA buffer length ← `512`
  Disable APLL usage
  Enable TX descriptor auto-clear
  Set fixed master clock ← 0
**Install I2S Driver:**
  Call `i2s_driver_install(i2sPort, i2s_config, 0, NULL)`
**End Algorithm**

## 3.5 Face Recognition Model

The ESP-WHO framework is designed to process(240×240) images, employing a combination of sophisticated algorithms for face detection and recognition. For face detection, the framework utilizes the MTCNN

**Algorithm 2** Applying Gain to Audio Data

**Require:** *data* (array of 16-bit integers), *length* (size of the array)
**Ensure:** Modified *data* with applied gain

**for** $i = 0$ to $length - 1$ **do**
    $data[i] \leftarrow data[i] \times GAIN\_FACTOR$
    **if** $data[i] > INT16\_MAX$ **then**
        $data[i] \leftarrow INT16\_MAX$
    **else if** $data[i] < INT16\_MIN$ **then**
        $data[i] \leftarrow INT16\_MIN$
    **end if**
**end for**
**End Algorithm**

**Algorithm 3** Speech Detection

**Require:** *vReal* (array of FFT magnitude values)
**Ensure:** Returns **true** if speech is detected, otherwise **false**

Initialize $energy \leftarrow 0$
Compute $startIndex \leftarrow \frac{\text{SPEECH\_FREQ\_MIN} \times \text{FFT\_SIZE}}{\text{VAD\_SAMPLE\_RATE}}$
Compute $endIndex \leftarrow \frac{\text{SPEECH\_FREQ\_MAX} \times \text{FFT\_SIZE}}{\text{VAD\_SAMPLE\_RATE}}$
**for** $i = startIndex$ to $endIndex - 1$ **do**
    $energy \leftarrow energy + vReal[i] \times vReal[i]$
**end for**
Compute $energy\_avg \leftarrow \sqrt{\frac{energy}{(endIndex - startIndex)}}$
**if** $energy\_avg >$ SPEECH_THRESHOLD **then**
    **return true** ▷ Speech detected
**else**
    **return false** ▷ No speech detected
**end if**
**End Algorithm**

(Multi-task Cascaded Convolutional Networks) aka MTMN[7] algorithm in conjunction with MobileNet.

**Algorithm 4** VAD Task Management

**Require:** $listening = \textbf{true}$, $maxTime$, $bonusTime$
**Ensure:** Handles voice activity detection (VAD) and stops after conditions are met

Initialize $startTime \leftarrow$ current time in milliseconds
Initialize $lastDetectionTime \leftarrow startTime$
Initialize $elapsedTime \leftarrow 0$
**while** $listening$ **do**
    $currentTime \leftarrow$ current time in milliseconds
    **if** vadDetect() **then**
        $lastDetectionTime \leftarrow currentTime$
        Print "Speech Detected"
    **end if**
    $elapsedTime \leftarrow currentTime - startTime$
    **if** $elapsedTime \geq maxTime$ **or** $(currentTime - lastDetectionTime \geq bonusTime)$ **then**
        $listening \leftarrow$ **false**
        $recording \leftarrow$ **false**
        **break**
    **end if**
    Wait for 10 milliseconds
**end while**
**End Algorithm**

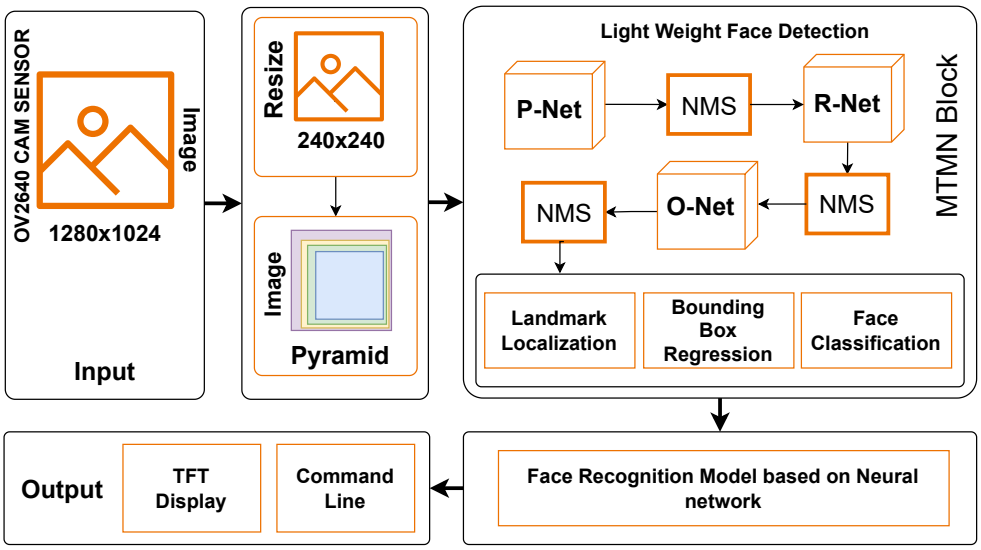



**Fig. 4**: Face Recognition Model

The system starts with an OV2640 camera sensor capturing images at 1280x1024 resolution. The captured image is resized to 240x240 for efficient processing. A pyramid representation of the image is created, likely for multi-scale face detection.

### 3.5.1 Pyramid

An image pyramid is created by resizing the original image to multiple scales (both downscaled and sometimes upscaled versions). This produces several copies of the same image at different sizes. Each scaled image is fed into the network, and candidate face regions detected at these scales are later mapped back to the original image coordinates.

- Scaling: The original image is resized to different scales, forming the pyramid.
- Detection: The P-Net processes each scaled image to propose potential face regions.
- Aggregation: These candidate regions are then adjusted back to the original scale and refined through subsequent stages (R-Net and O-Net) using techniques like non-maximum suppression to merge overlapping detections.

### 3.5.2 MTMN Block

MTMN[7] is a lightweight Human Face Detection Model, which is built around a new mobile architecture called MobileNetV2 and Multi-task Cascaded Convolutional Networks, and is specially designed for embedded devices. The MTMN pipeline consists of three cascaded neural networks, each progressively refining the detection results:

- Proposal Network (P-Net): A small fully convolutional network that scans the image at multiple scales to propose candidate face regions.
  Outputs: Face classification scores (is a region likely to contain a face?), bounding box regression offsets (to refine the coordinates), facial landmark location predictions (coarse).
- Refine Network (R-Net): Takes the candidate face regions (bounding boxes) from P-Net, crops them to a fixed size, and refines the detection.
  Outputs: Higher-accuracy face classification scores, more precise bounding box regression and landmark predictions (more refined) using Non-Maximum Suppression (NMS). The method applies NMS to eliminate overlapping or redundant bounding boxes. This ensures that only the best bounding boxes are forwarded to the next stage.
- Output Network (O-Net): Operates on the remaining candidate regions after R-Net filtering.
  Outputs: Final face classification, final bounding box regression and more accurate landmark localization.

### 3.5.3 Multitask Loss Function

Each of the three networks is trained with a multi-task objective, which includes:

- Classification Loss: Distinguishes faces vs. non-faces.
- Bounding Box Regression Loss: Refines the coordinates of the face bounding boxes.
- Landmark Localization Loss: Predicts the coordinates of key facial landmarks (e.g., left eye, right eye, nose, left mouth corner, right mouth corner).

### 3.5.4 Face Recognition

After detecting and processing the face, it is passed through a Face Recognition Model based on a neural network for identity verification or classification.Then results are displayed on a TFT(Thin Film Transistor) display or returned via a command-line interface.

## 3.6 Gesture Model

The gesture model[8] depicted involves multiple stages, beginning with raw accelerometer data (accX, accY, accZ) that captures different gestures like "shake." This raw data is processed through spectral analysis, extracting key features based on frequency and amplitude, which are then fed into a neural network.

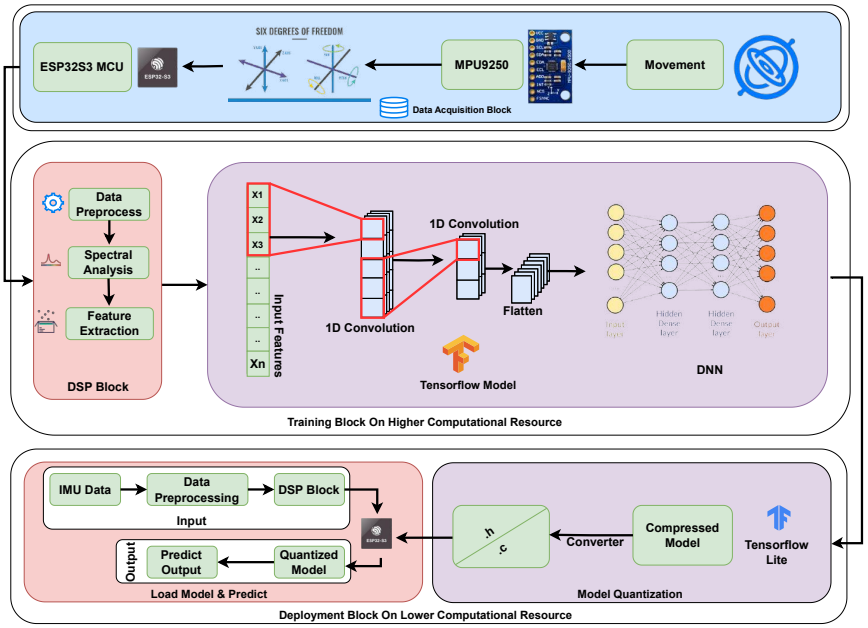



**Fig. 5**: Gesture Model

The network architecture includes an input layer with 192 features, followed by several dense layers with neuron dropouts to prevent overfitting. Finally, the model outputs classifications into four gesture classes, indicating its role in gesture recognition tasks based on motion data.

### 3.7 Home Automation

Home automation segment seamlessly take natural english command from user. Key functions include check roles, analyze, manage device and policy, control mode and state, make decision, and handle request, each performing specific tasks to automate the home environment.

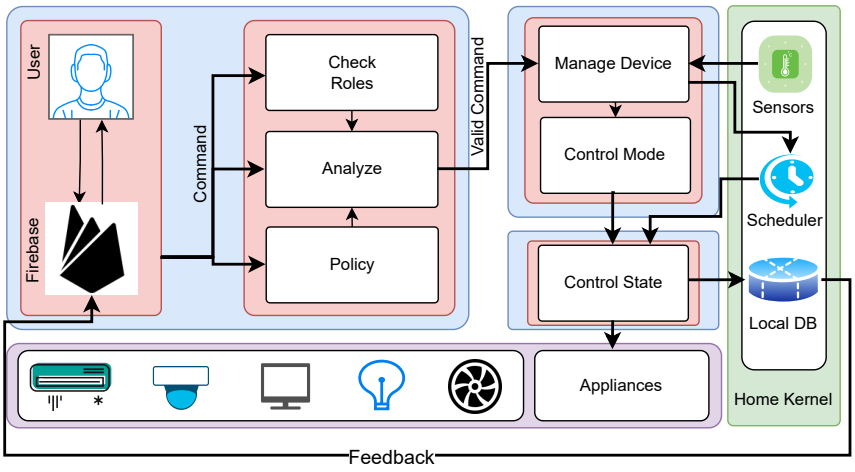



**Fig. 6**: Home Automation Section

**User & Firebase :** User represents the end-user who interacts with the home automation system. Firebase serves as the communication back-end or cloud service[9]. The user's commands are sent here first (or stored/retrieved from here) before being forwarded to the internal modules of the home automation system.

**Command-Processing Modules :**

- Check roles: Verifies the user's permissions or roles (e.g., admin vs. guest). If the user is not authorized to perform a certain action, the command is rejected.
- Analyze: Interprets the user's command (e.g., determining which device to control, what state to set). It may also consider context or additional data. Making sub command from the raw command, extracting command name and state and cloud command(Eg. turn off room fan, open window etc.), if there exist a device name included into the home environment. Cloud commands are made of command name, state and device kernel name and finally pushed into the firebase data bucket.
- Policy: Applies security or usage policies. For instance, certain devices may only be controlled during specific hours, or only certain users may have access.

**Device Management & Control :** Once the command is validated, it goes through these modules to effect changes on actual devices:

- Manage Device: Acts as a manager or orchestrator that routes the valid command to the correct appliance or subsystem.
- Control Mode: Determines how the command should be applied (e.g., manual mode vs. scheduled/automatic mode[10]). This module might also handle toggling between different operating modes (e.g., energy-saving mode, auto mode, smart mode).

- Control State: Sets or updates the current operational state of the device (e.g., turning the light "ON," setting temperature to 24°C). Essentially, this is where the final state instructions are sent to the appliances.
- Appliances: The physical devices (e.g., fan, lights) that receive the final command and carry out the action.

**Home Kernel & Supporting Modules :** Sensors collect real-time data (e.g., temperature, humidity, motion) which can inform the control logic or user policies. Scheduler automates tasks based on time or events (e.g., turning off lights at 10 PM, adjusting AC temperature in the morning). Local DB stores system settings, schedules, sensor readings, or user/device profiles locally for quick access and offline operation. Home Kernel is the underlying core or operating environment that coordinates all the system processes. It provides the runtime platform for scheduling, device management, and sensor integration.

## 3.8 Task Handler

Marvin is designed to efficiently distinguish between tasks that are processed locally and those requiring cloud access. For instance, commands such as "come here," "turn left," "dance," or "turn right" are treated as local tasks [**?** ] and are executed immediately without needing cloud connectivity.

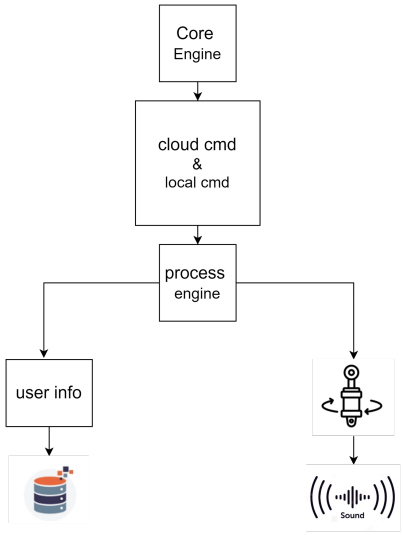



**Fig. 7**: Task Handler

Figure 7 illustrates these straightforward actions, as they involve direct control over Marvin's movements and devices, allowing for quick, responsive performance. However, more complex commands, which depend on external data or advanced processing, are classified as cloud tasks. These are managed by connecting to the necessary online services to retrieve information or complete operations (IoT).

## 4 Experimental Result Analysis

The performance of MARVIN[ 8 ] was assessed by evaluating a number of important parameters. We averaged all of the findings after benchmarking them more than 100 times both noisy and quiet environment. They are described in further detail below:

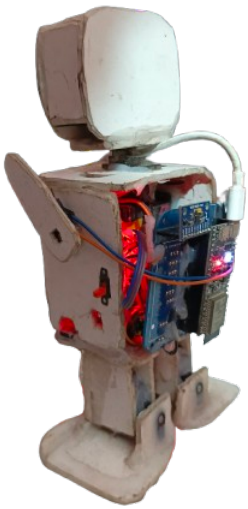


**Fig. 8**: MARVIN hardware architecture (left back view)

### 4.1 Wakeword Detection Results

Wakeword performance in various situations and thresholds is shown in the table **??**. Confusion matrix, accuracy, false & missed wakeword rate

**Table 3**:
Caption

are also given. Avarage timing for wakeword detection is 316 ms and memory consumed 13 kb. A threshold of 0.95 yields the best accuracy (96.67%) in a quiet setting, but it also has a 1% greater rate of missed wakewords than 0.9. Noise impacts detection performance because it causes a minor decrease in accuracy (up to 94.17% with 0.95 threshold) and an increase in the rates of false wake words and missed wake words in noisy situations. Rate of false positives:

$$\text{False Wake Word Rate} = \frac{FP}{FP + TN}$$

Rate of false negetives:

$$\text{Missed Wake Word Rate} = \frac{FN}{FN + TP}$$

Overall effectiveness :

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

### 4.2 Voice Activity Detection Results

The Fig. 9 shows a VAD overview, where energy levels (in blue) are plotted along with smoothed energy (in red) and detected speech segments (in green). The graph highlights periods of speech activity, with peaks in energy corresponding to detected speech intervals.

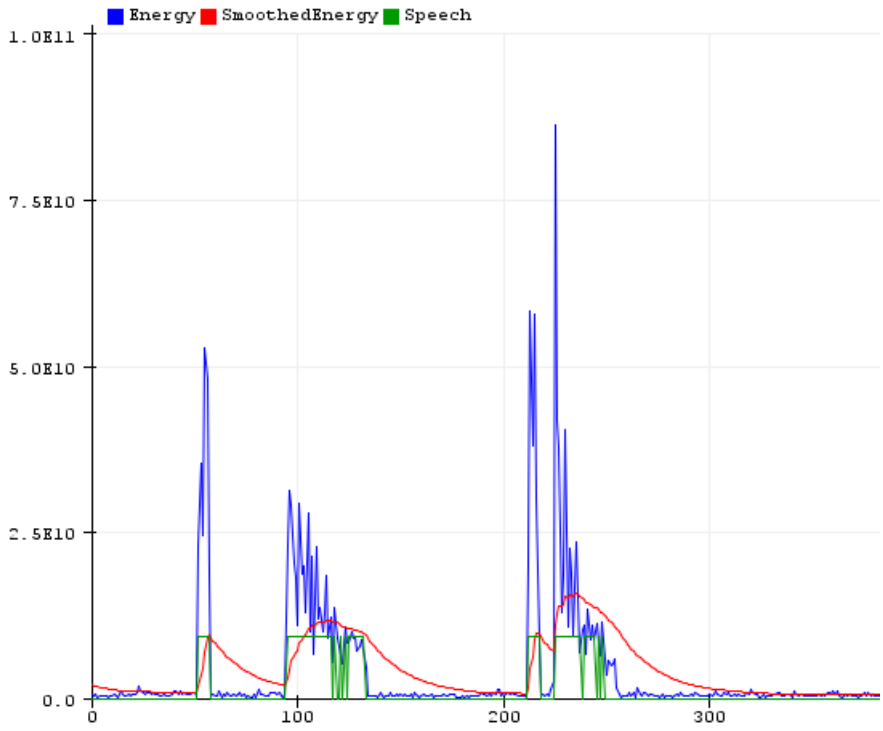



**Fig. 9**: VAD Overview

### 4.3 Face Recognition & Detection Results

Multi resolution search from a displaced position using a face model is a great approach. We use face recognition model by ESP32 and collect result in table **??**. Avarage timing for detection and recognition 554 ms and memory 80 kb. Average accuracy is 79.60%.


**Fig. 10**: Face Recognition Overview

The image 10 shows a facial recognition system identifying some known faces with specific IDs. The system successfully differentiates between recognized individuals and effectively flagging unknown faces.

## 5 Conclusion and future work

This study conducts a comparative analysis between graph-based clustering and traditional clustering methods to model armed conflicts in Bangladesh and compares the performance of graph neural networks (GNNs) against conventional machine learning (ML) algorithms, particularly artificial neural networks (ANNs), to predict fatalities. This study aims to understand armed conflict dynamics and forecast mortality rates, which are the key components of conflict research. Graph-based clustering algorithms capture the intrinsic relational structure of contradictory data, whereas classic clustering methods focus on feature engineering and distance measurement. The findings highlight the usefulness of graph-based clustering techniques in providing insights into the dynamics and structure of violent conflicts in Bangladesh. Additionally, the comparison demonstrates that GNNs outperform some conventional ML techniques in predicting lethality, underscoring the importance of understanding the relational relationships within the data. Overall, the study shows the utility of graph-based methodologies and GNNs in comprehending and forecasting patterns of armed conflict.

Graph-based clustering approaches offer valuable insights into complex data structures, yet they possess several limitations. The datasets with few occurrences that could affect the prediction were filtered out. These biases within datasets can distort graph representations, impacting clustering outcomes. Skewed or incomplete data, along with outliers, can

compromise the accuracy of the clustering algorithm. We also worked on a selected feature set. High-dimensional or non-linear data can pose challenges for graph construction and parameter specification, hindering the effectiveness of graph-based clustering. Furthermore, we made an effort to reduce some computational complexity by utilizing KNN to compute the adjacency matrix while only taking 20 nodes into account. Computational complexity presents scalability issues for large datasets, while traditional graph construction methods may struggle with identifying clusters of varying densities or irregular shapes.

In the future directions, some additional data sources, such as social media and satellite imagery can be integrated to enhance the accuracy and granularity of conflict analysis. Also, investigating alternative graph-based modeling techniques, such as community detection algorithms like Louvain algorithm, Infomap algorithm, Label propagation algorithm, etc. could offer new insights and potentially improve clustering accuracy. Lastly, applying the methodology to other contexts of conflict, such as social unrest or political instability, could broaden the scope of the research and uncover common patterns or underlying structures across different conflict scenarios. The knowledge collected from this research can help in designing more reliable conflict analysis models and tactics for conflict prevention and resolution in Bangladesh.

## Declarations

**Funding:** None.
**Conflict of Interest:** The authors declare that they have no conflict of interest.
**Availability of data and materials:** Not applicable.
**Code availability:** Not applicable.
**Authors' contributions:** D.N.S, S.P.S, M.M.H, and M.A.R wrote the main manuscript text. S.P.S, M.M.H, and M.A.R did the experimentation and data analysis part. D.N.S. coordinated the whole study and reviewed the manuscript. All authors analyzed the data, discussed the results, and approved the final manuscript.
**Ethical Approval:** There are no studies by any of the authors in this article that used humans or animals as subjects.
**Consent for publication:** all authors agreed to publish.
**Consent to participate:** Consent to publish.

**Competing interests:** The authors have no relevant financial or non-financial interests to disclose.

## References


[1] Castro-González, Á., Malfaz, M., Salichs, M.A.: Selection of actions for an autonomous social robot. In: Social Robotics: Second International Conference on Social Robotics, ICSR 2010, Singapore, November 23-24, 2010. Proceedings 2, pp. 110–119 (2010). Springer

[2] Borja, R., De La Pinta, J.R., Álvarez, A., Maestre, J.M.: Integration of service robots in the smart home by means of upnp: A surveillance robot case study. Robotics and Autonomous Systems **61**(2), 153–160 (2013)

[3] Thomopoulos, V., Bitas, D., Papastavros, K.-N., Tsipianitis, D., Kavga, A.: Development of an integrated iot-based greenhouse control three-device robotic system. Agronomy **11**(2), 405 (2021)

[4] Atzeni, M., Reforgiato Recupero, D.: Deep learning and sentiment analysis for human-robot interaction. In: The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers 15, pp. 14–18 (2018). Springer

[5] TheZeroHz: GitHub - TheZeroHz/ESpeech: Free speech to text module for ESP32 with WakeWord & NLP. No API Key or any account is needed. [Online; accessed 2025-02-25]. https://github.com/TheZeroHz/ESpeech/tree/main

[6] TheZeroHz: GitHub - TheZeroHz/VADCoreESP32: ESP32 Voice Activity Detection Library. [Online; accessed 2025-02-25]. https://github.com/TheZeroHz/VADCoreESP32/tree/main

[7] Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. IEEE signal processing letters **23**(10), 1499–1503 (2016)

[8] TheZeroHz: GitHub - TheZeroHz/MotionSense-ESP: A real-time gesture detection system using an IMU (Inertial Measurement Unit) and an ESP-based microcontroller with TensorFlow Lite (TFLite) for lightweight machine learning inference. This project enables motion-based control by recognizing predefined gestures with IMU sensor data and processing them on an ESP32. [Online; accessed 2025-02-25]. https://github.com/TheZeroHz/MotionSense-ESP

[9] Sarkar, S., Gayen, S., Bilgaiyan, S.: Android based home security systems using internet of things (iot) and firebase. In: 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), pp. 102–105 (2018). IEEE

[10] Pujari, U., Patil, D.P., Bahadure, D.N., Asnodkar, M.: Internet of things based integrated smart home automation system. In: 2nd International Conference on Communication & Information Processing (ICCIP) (2020)