REALTIME ECG MONITORING SYSTEM

AADITYA PRAKAASH J -22L201

KRISHNAKANTH A –22L230

MITHUN HARI K -22L232

MUGUNDAN K –22L236

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: ELECTRONICS AND COMMUNICATION ENGINEERING

of Anna University

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

FACLUTY GUIDES:

Dr. K.Vasanthamani

Dr. S. Mohandass

Abstract—**The primary objectives of this project include acquiring clean ECG signals using the AD8232 module and processing SpO_2 and pulse rate data via the MAX30102 sensor. The ESP32 is used to interface with the sensors and perform digital signal sampling. Captured data is transmitted to Firebase for temporary cloud storage. An encoder- based Transformer model is developed and trained using Jupyter Lab to classify arrhythmias. The pretrained model is used for real-time ECG signal classification, with results visualized on a Streamlit-based web interface that also plots live ECG waveforms.**

I. CHAPTER 1: INTRODUCTION

1.1BACKGROUND

CVDS ARE A MAJOR GLOBAL HEALTH THREAT [1]. WEARABLE ECG SYSTEMS USING IOT AND AI NOW ALLOW REAL-TIME, NON-CLINICAL HEART MONITORING.

1.2PROBLEM_STATEMENT

CURRENT ECG DEVICES ARE EXPENSIVE AND LACK REAL-TIME ARRHYTHMIA DETECTION [2]. THERE'S A NEED FOR A SMART, PORTABLE, LOW-COST SYSTEM.

1.3OBJECTIVE

TO BUILD A COMPACT ECG SYSTEM USING ESP32, SENSORS, AND A PRETRAINED DEEP MODEL FOR REAL-TIME ARRHYTHMIA CLASSIFICATION ON A WEB DASHBOARD.

1.4SCOPE

SUPPORTS REAL-TIME ECG, SPO_2, AND PULSE MONITORING WITH FIREBASE SYNC, CLASSIFYING FIVE ARRHYTHMIA TYPES USING DEEP LEARNING.

II. LITERATURE SURVEY

2.1 Overview

Explores ECG classification methods using CNNs, Transformers, and hybrid models. Reviews key studies shaping our system design.

2.1.1 Dual Model: XGBoost + Attention-based DL

Hybrid model merges XGBoost with attention-enhanced DL for better feature learning. Real-time deployment is not explored.

2.1.2 CNN-LSTM with FFT Features

Combines CNN and LSTM for spatial-temporal features with frequency-domain input. Ignores deployment on embedded systems.

2.1.3 Deep Learning on MIT-BIH Dataset

Trains deep neural networks for arrhythmia detection; matches expert accuracy. Model complexity limits edge deployment.

2.1.4 Transfer Learning with GANs

Uses distant transfer learning and GANs for dataset generalization. High compute cost makes it unsuitable for real-time use.

2.1.5 CNN Features + XGBoost

Pretrained CNN features classified by XGBoost show strong accuracy. Generalizability and validation still lacking.

2.1.6 XGBoost with Robust Features

Uses handcrafted features with XGBoost for arrhythmia classification. Ignores model interpretability and real-time use.

2.1.7 Hybrid CNN-Transformer

Combines CNN and Transformer models without R-peak detection; highly accurate. Doesn't consider hardware constraints.

2.1.8 Lightweight DCETEN Model

1D-CNN + Transformer + pruning for IoT-friendly ECG classification. Great accuracy but lacks real-world dataset tests.

2.1.9 Constrained Transformer Network

Adds loss constraints to balance performance across imbalanced classes. Still underperforms on rare arrhythmias.

2.1.10 Unsupervised Transformer for Anomaly Detection
Detects ECG anomalies using only normal training data. Strong results, but edge deployment remains unexplored.

III. System Design

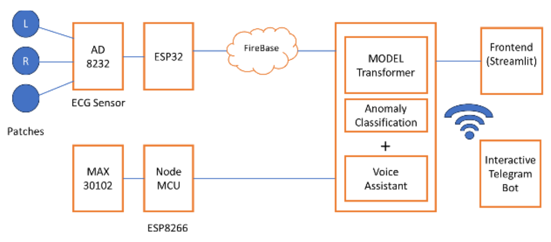


Fig 3.1 Block diagram of the proposed system architecture

3.1 OVERALL SYSTEM ARCHITECTURE

The system comprises three key layers: data acquisition, processing, and visualization. In the acquisition layer, ECG signals are obtained using AD8232 and heart rate/oxygen saturation are measured using the MAX30102 sensor. The ESP32 and NodeMCU ESP8266 handle signal digitization and communication. The processed data is uploaded to Firebase Realtime Database for temporary storage. A web application developed using Streamlit fetches this data, applies the pre-trained Transformer model for classification, and plots real-time ECG signals for end users. This architecture ensures low latency and high availability, making the system suitable for continuous health monitoring and early detection of cardiac anomalies.

3.2 HARDWARE COMPONENTS

3.2.1 ECG Patch and AD8232

The AD8232 analog front-end module is used to acquire ECG signals from electrode patches placed on the subject's chest. It is designed specifically for biopotential measurements and provides built-in signal conditioning including amplification and filtering. It delivers a clean analog ECG waveform, which is fed into the ADC pin of the ESP32. The electrodes used are reusable ECG pads connected to the AD8232 through 3-pin wires (RA, LA, and RL).

3.2.2 ESP32 Microcontroller

The ESP32 microcontroller is utilized for digitizing ECG data from the AD8232 module. With its 12-bit ADC, it samples the analog ECG waveform and temporarily buffers the signal. It ensures accurate real-time conversion and communication with Firebase using Wi-Fi. The ESP32 is also responsible for sampling control, handling lead-off detection, and ensuring signal quality before uploading.

3.2.3 NodeMCU and MAX30102

The MAX30102 is an integrated optical sensor used for measuring SpO_2 (oxygen saturation) and pulse rate via photoplethysmography (PPG). It works in conjunction with the NodeMCU ESP8266, which reads raw IR and red-light data from the sensor and processes it using the open-source SpO_2 algorithm. The NodeMCU then transmits the BPM and SpO_2 readings via serial to the Streamlit dashboard for real-time display

3.3 SOFTWARE STACK

3.3.1 Jupyter Lab (Model Training)

The model training and evaluation were carried out in Jupyter Lab. ECG segments were loaded, preprocessed, and normalized. Custom scripts were used to extract features from the P, Q, R, S, and T waves. The labeled dataset was split into training and testing sets before being passed to a Transformer-based model built with TensorFlow.

3.3.2 TensorFlow (CNN Transformer)

The core of the classification model is based on a CNN-Transformer encoder. The Transformer layers capture long-term dependencies in ECG signals while CNN layers extract local features like wave morphology. TensorFlow was used to build, train, and export the model. After achieving satisfactory accuracy and loss metrics, the trained model was saved in ".keras" format for use in the deployment stage.

3.3.3 Streamlit (Web Server)

Streamlit was used to develop a lightweight, interactive web dashboard to monitor patient vitals. The interface connects to Firebase, fetches the latest ECG samples, applies the trained model, and displays the predicted arrhythmia class. It also shows BPM, SpO_2, and calculated BP (SBP/DBP) values

3.4 SIGNAL EXTARCTION AND SIGNAL PROCESSING

3.4.1 ECG SIGNAL CHARACTERISTICS

An ECG (Electrocardiogram) signal reflects the electrical activity of the heart and consists of repetitive cycles of five major waveform components: P, Q, R, S, and T. These components represent different phases of cardiac depolarization and repolarization. The R-wave is typically the most prominent and is used as a reference for beat segmentation. Key parameters such as amplitude, interval, and duration of these waves help identify arrhythmic patterns. ECG signals are typically non-stationary and sensitive to noise, requiring careful processing before classification.

3.4.2 NOISE REMOVAL USING ANALOG FILTERS

To ensure high signal fidelity before digitization, analog filters were employed at the hardware level. A combination of high-pass and low-pass filtering was implemented using the AD8232 module's built-in filtering circuitry. These filters suppress baseline drift, muscle noise, and high-frequency interference from power lines. Additionally, the system uses lead-off detection pins to identify when electrodes are improperly connected, preventing false readings. This preconditioning stage ensures that only physiologically relevant signal content is passed to the ADC.

3.4.3 PQRST FEATURE EXTRACTION

Once the ECG signal is digitized, each beat is isolated using R-peak detection algorithms. From each segmented beat, temporal and amplitude-based features are extracted from the P, Q, R, S, and T points.

These include:
- P-wave amplitude and duration
- QRS complex width and R-peak height
- ST segment slope
- T-wave polarity and duration
- RR interval

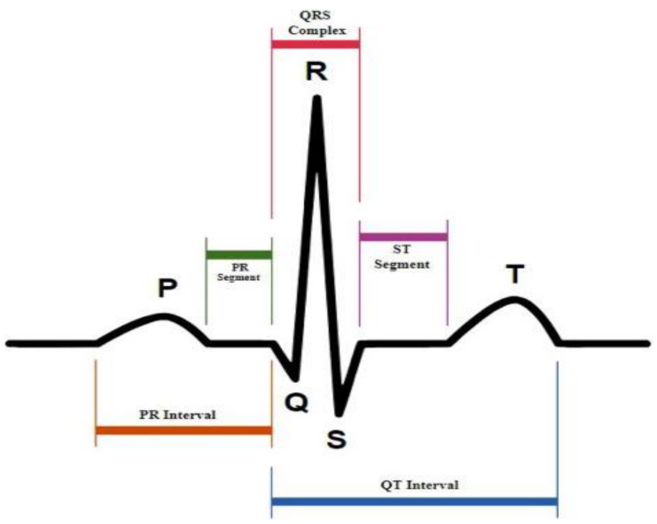


Fig 3.2 ECG signal

These features (as shown in Fig 3.2) are medically relevant and play a crucial role in identifying arrhythmias such as LBBB, RBBB, PVC, and APC. The extracted values are structured as feature vectors, serving as input to the model.

3.4.4 DATASET PREPROCESSING AND NORMALIZATION

Before training, all feature vectors and raw ECG signals are normalized to ensure consistency across samples. Z-score and Min-Max normalization techniques are applied to scale signals to a standard range (typically [-1, 1]). This prevents the model from being biased towards larger amplitude components. Additionally, ECG sequences are zero-padded or truncated to a fixed length of 100-time steps to maintain input dimensionality. This preprocessing improves convergence during model training and reduces variance in prediction.

3.4.5 INTERFACING MAX30102 USING NODEMCU

The MAX30102 sensor is used for measuring heart rate and blood oxygen saturation (SpO_2) through photoplethysmography (PPG). It operates using infrared (IR) and red LEDs, which emit light through the skin and detect changes in blood volume. The output obtained is shown in Fig 3.3. These changes correlate with the heartbeat and oxygen concentration. For this project, the MAX30102 module was connected to a NodeMCU ESP8266 board via the I^2C interface. The SDA and SCL lines of the sensor were wired to GPIO4 (D2) and GPIO5 (D1), respectively. The sensor was initialized and configured using the official MAX30105 Arduino library.
A fixed-size buffer was used to store PPG readings, which were processed using the open-source SpO_2 and BPM extraction algorithm provided by Maxim Integrated. The results were printed in serial format (BPM and SpO_2) and read by the Streamlit web interface running on the laptop. This serial link enabled real-time visualization of pulse and oxygen saturation alongside ECG data.
This modular approach ensured that both electrical and optical signals were processed in parallel, providing a more complete picture of the user’s cardiovascular health.

3.5 MACHINE LEARNING MODEL

This chapter describes the design and implementation of the deep learning model used for ECG signal classification. The approach leverages a Transformer encoder architecture to effectively learn both local and long-range dependencies in ECG waveforms. The model is trained and validated using a real-world dataset from the WFDB (Waveform Database), which contains annotated ECG records. The chapter also presents evaluation metrics such as accuracy and loss, which were used to assess model performance.

3.5.1 ENCODER TRANSFORMER MODEL ARCHITECTURE

The Transformer encoder architecture is adopted for its capability to model temporal relationships without relying on sequential recurrence. The model accepts ECG signals of fixed length and processes them through an embedding layer followed by stacked multi-head self-attention blocks. These blocks allow the model to learn how different parts of the waveform relate to each other — even across distant time steps. Each attention block is followed by layer normalization and residual connections to maintain gradient stability. After attention, the output passes through feed-forward dense layers with dropout for regularization. The final classification layer uses a SoftMax activation to output the predicted arrhythmia class from five predefined categories: Normal, LBBB, RBBB, PVC, and APC.

The model architecture is designed to be lightweight enough for deployment while preserving sufficient complexity to capture the subtle variations in ECG morphology.

3.5.2 TRAINING AND VALIDATION IN JUPYTER LAB

Model development and experimentation were performed in the Jupyter Lab environment. ECG signals were loaded from the WFDB waveform database using the wfdb Python library. R-peak detection and beat segmentation were applied directly to the waveform, followed by length standardization to ensure uniform input. During training, the dataset was split into training and validation sets using an 80:20 ratio. The model was trained using categorical cross-entropy as the loss function and Adam optimizer with a learning rate of 0.001. Data augmentation techniques such as noise injection and random shifting were applied to increase variability and prevent overfitting. Model checkpoints and early stopping were implemented to optimize training time and preserve the best-performing version of the model.

3.5.3 ACCURACY AND LOSS METRICS

Model performance was evaluated using two key metrics: accuracy and categorical cross-entropy loss. Accuracy measures the percentage of correctly classified heartbeats, while the loss quantifies how well the predicted class probabilities match the true labels. Over the training epochs, a consistent decrease in loss and an increase in validation accuracy were observed, indicating stable learning. The final model achieved high classification accuracy on unseen validation data, with strong generalization across arrhythmia classes. Confusion matrix analysis further revealed low misclassification rates between structurally similar beats like LBBB and RBBB.

These results validate the effectiveness of the encoder-based Transformer architecture for ECG signal classification in real-time healthcare applications.

4. IMPLEMENTATION AND RESULTS

4.1 Realtime ECG Acquisition using ESP32

The ESP32 was used to collect ECG signals from the AD8232 module. It reads analog values using its 12-bit ADC at a sampling rate of approximately 360 Hz. The digitized ECG samples are temporarily stored in a buffer, and once the buffer is filled (e.g., 360 samples per batch), the data is uploaded to Firebase Realtime Database under a user-specific path. Lead-off detection logic is included to ensure reliable signal acquisition and to prevent data corruption due to sensor disconnection. The connections given are shown in Fig 4.1

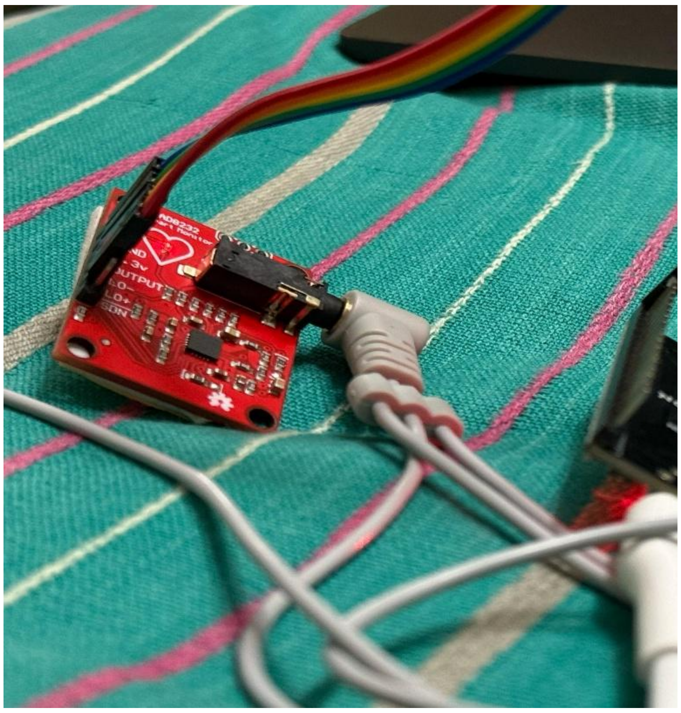


Fig 4.1 Interfacing diagram of ESP32 with AD8232

4.2 Firebase Communication Protocol

The ESP32 communicates with Firebase via Wi-Fi. Firebase credentials and tokens are configured during device initialization. Once authenticated, the ESP32 pushes ECG batches into the "/UsersData/<UID>/ecgReadings" path as JSON arrays. Each batch is timestamped using millis() to maintain order. Additionally, the system periodically checks and deletes older data if the sample count exceeds a threshold, thereby preventing database overflow and improving real-time access performance.

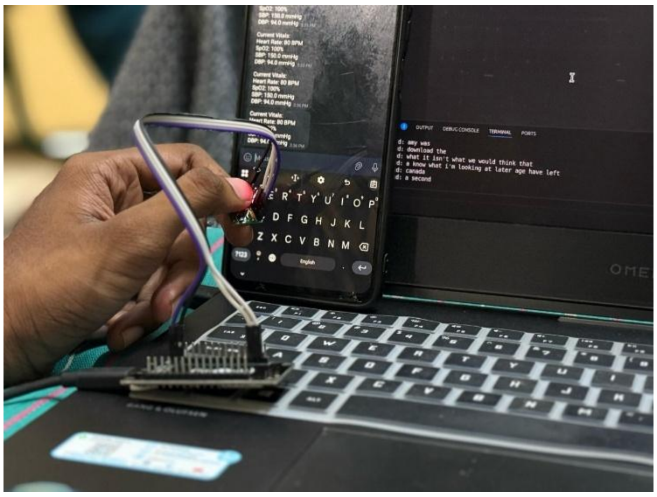


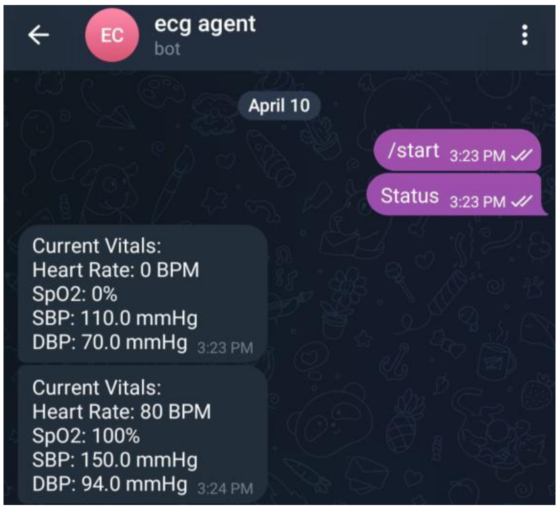
Fig 4.2 Interfacing diagram of MAX30102 with NodeMCU

4.3 Laptop-Based Model Interface

In this architecture, the actual inference process is performed on a laptop using Python and Streamlit. The pre-trained Transformer model, saved in ".keras" format, is loaded at runtime. The Streamlit app continuously polls Firebase for the latest ECG batch, preprocesses the signal (normalization and padding), and feeds it into the model for classification. This approach allows the use of a complex deep learning model without exceeding the resource limits of the ESP32.

4.4 Integration of Telegram Bot

Telegram bot sends alerts for hardware errors and responds to "status" requests with real-time vitals from Firebase.



ecg agent
bot

April 10

/start 3:23 PM

Status 3:23 PM

Current Vitals:
Heart Rate: 0 BPM
SpO2: 0%
SBP: 110.0 mmHg
DBP: 70.0 mmHg 3:23 PM

Current Vitals:
Heart Rate: 80 BPM
SpO2: 100%
SBP: 150.0 mmHg
DBP: 94.0 mmHg 3:24 PM

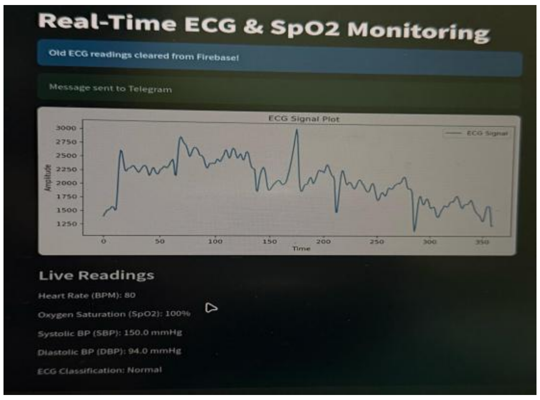
Fig 4.3 Output of Heart rate, SpO2, SBP, DBP as telegram message

4.5 Voice Assistant for Emergency Alerts & Reminders

Offline voice assistant detects SOS keywords for emergency alerts and gives scheduled reminders using gTTS.

4.6 Result and Analysis

The system performed stable real-time ECG monitoring, accurate arrhythmia classification, and multi-vital tracking with cloud and bot integration.



Real-Time ECG & SpO2 Monitoring

Old ECG readings cleared from Firebase!

Message sent to Telegram

Live Readings

Heart Rate (BPM): 80

Oxygen Saturation (SpO2): 100%

Systolic BP (SBP): 150.0 mmHg

Diastolic BP (DBP): 94.0 mmHg

ECG Classification: Normal

Fig 4.4 Final output

ACKNOWLEDGMENT

WE WOULD LIKE TO SINCERELY EXPRESS OUR HEARTFELT GRATITUDE TO OUR PRINCIPAL, **DR. K. PRAKASAN**, FOR PROVIDING US WITH THE OPPORTUNITY AND NECESSARY FACILITIES TO SUCCESSFULLY CARRY OUT THIS PROJECT.WE ALSO WISH TO EXPRESS OUR SINCERE THANKS TO **DR. V. KRISHNAVENI**, PROFESSOR AND HEAD, DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING, FOR THE ENCOURAGEMENT AND SUPPORT THAT SHE EXTENDED TOWARDS THIS PROJECT WORK.

WE ARE GRATEFUL TO OUR PROGRAMME COORDINATOR, **DR. M. SANTHANALAKSHMI**, ASSOCIATE PROFESSOR, DEPARTMENT OF ECE, TUTOR **DR. K. THIYAGARAJAN**, ASSOCIATE PROFESSOR, DEPARTMENT OF ECE AND COMMITTEE MEMBERS. **DR. K. VASANTHAMANI**,

ASSISTANT PROFESSOR (SR. GR) AND **DR. S. MOHANDASS**, ASSISTANT PROFESSOR FOR ALWAYS BEING APPROACHABLE AND FOR THE ADVICE AND CONSTRUCTIVE FEEDBACK REGARDING VARIOUS ASPECTS OF THE PROJECT. FINALLY, WE WOULD LIKE TO THANK ALL MY DEPARTMENT SENIORS AND STUDENT COLLEAGUES WHO HAVE HELPED US IN COMPLETING THIS PROJECT WITHOUT ANY HASSLES.

REFERENCES

1. G. B. Moody and R. G. Mark, “The impact of the MIT-BIH Arrhythmia Database,” *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.
2. T. Ince, S. Kiranyaz, and M. Gabbouj, “A generic and robust system for automated patient-specific classification of ECG signals,” *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 5, pp. 1415–1426, 2009.
3. O. Yildirim, U. Baloglu, R. Tan, and A. Acharya, “A deep learning model for automated ECG classification,” *Expert Systems with Applications*, vol. 120, pp. 103–111, 2019.
4. H. Rajpurkar et al., “Cardiologist-level arrhythmia detection with convolutional neural networks,” *arXiv preprint arXiv:1707.01836*, 2017.
5. M. Schirrmeister et al., “Deep learning with convolutional neural networks for EEG decoding and visualization,” *Human Brain Mapping, vol. 39, no. 2, pp. 539–554, 2018.*
6. A. Daoud and M. Bayoumi, "Efficient ECG classification using deep learning," *in 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), 2019, pp. 731-735.*
7. P. de Chazal et al., "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," *IEEE Transactions on Biomedical Engineering, vol. 51, no. 7, pp. 1196-1206, 2004.*
8. S. Hong et al., "ECG arrhythmia classification using a 2-D convolutional neural network," *IEEE Access, vol. 7, pp. 92871-92880, 2019.*
9. J. Yao and R. W. Smith, "ECG signal classification using deep learning and a stacked LSTM model," *in 2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS), 2021, pp. 568-573.*
10. S. M. A. Basha, K. J. Kiran, and B. Rajasekaran, "Deep learning-based approach for ECG classification," *in 2020 IEEE International Conference on Advances in Computing, Communication, and Control (ICAC3), 2020.*