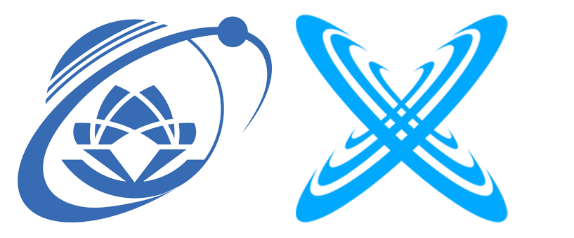
**UNIVERSITY OF INFORMATION TECHNOLOGY, VNU-HCM**

**FACULTY OF COMPUTER NETWORKS AND COMMUNICATION**



**FINAL PROJECT REPORT**

WIRELESS EMBEDDED NETWORK SYSTEMS COURSE

**GENERAL INFORMATION**

**Project title**: Wireless embedded system for human activities recognition using ESP32 and Jetson Nano

**Group: 3**

**Members of the project:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Name** | **Student ID** | **Class** |
| 1 | Tống Võ Anh Thuận | 21522652 | MMCL2021 |
| 2 | Trần Công Hải | 21520811 | MMCL2021 |

**Instructor:** Assoc Prof. Lê Trung Quân

**Report:** 5/8/2023 – 20/12/2023

**Table of Contents**

[I. Introduction - Overview 3](#_Toc154003519)

[II. System architecture 3](#_Toc154003520)

[A. System components and related definitions 3](#_Toc154003521)

[B. System diagram 4](#_Toc154003522)

[III. System realization 5](#_Toc154003523)

[A. ESP32 Configuration and CSI Data Collection 5](#_Toc154003524)

[B. CNN Model Development and Training 6](#_Toc154003525)

[C. MQTT Broker Deployment and Android App Development 7](#_Toc154003526)

[D. Code and Dataset Availability 7](#_Toc154003527)

[IV. Conclusions - Key achievements 7](#_Toc154003528)

[V. References 8](#_Toc154003529)

1. Introduction - Overview

Human activity recognition (HAR) has witnessed a surge in recent years due to its vast potential in healthcare, fitness monitoring, smart homes, and various other applications. Traditional HAR methods typically rely on cameras or wearable sensors, raising concerns about privacy and inconvenience. This project aims to address these limitations by developing a wireless embedded system that leverages the combined power of the ESP32 microcontroller and the Jetson Nano edge AI platform for accurate and non-intrusive HAR by utilizing Channel State Information – CSI signal data obtained via Wi-Fi communication.

1. System architecture
2. System components and related definitions
3. ESP32

The ESP32 is a versatile and popular microcontroller and system-on-chip (SoC) designed for embedded systems and IoT (Internet of Things) applications. Its advantages are powerful features, low cost, and extensive community support. ESP32 is commonly used in a wide range of applications, including home automation, industrial automation, smart devices, robotics, and more. Its affordability, combined with its powerful capabilities, has contributed to its widespread adoption in the maker and IoT communities.

1. **Jetson Nano**

NVIDIA Jetson Nano is a small, low-cost, and power-efficient computer module designed for embedded artificial intelligence (AI) and machine learning (ML) applications, thanks for its affordability and capability for AI experimentation and development. It is targeted at developers or professionals, who are working on projects that involve AI and computer vision. Jetson Nano is commonly used in projects involving image and video processing, object detection, data prediction, and other AI-related tasks.

1. **HiveMQ**

HiveMQ is a MQTT (Message Queuing Telemetry Transport) broker that is designed for handling large-scale, real-time IoT (Internet of Things) applications. MQTT is a lightweight and efficient messaging protocol specifically designed for low-bandwidth, high-latency, or unreliable networks. Therefore making it well-suited for IoT and M2M (Machine to Machine) communication. HiveMQ is utilized in various industries and applications, including smart homes, industrial automation, healthcare, and more, where efficient and reliable communication between devices is essential. It plays a crucial role in the MQTT ecosystem, providing a robust infrastructure for building scalable and secure IoT solutions.

1. **Android Studio**

Android Studio is the official integrated development environment (IDE) for Android app development. It is provided by Google and is designed to simplify the process of building, testing, and deploying Android applications. Android Studio offers a comprehensive set of tools and features for developers to create high-quality Android apps for smartphones, tablets, smart TVs, wearables, and other Android-powered devices. In this system, we use Android Studio to develop an application for users to login and observe human activities.

1. **System diagram**

A diagram of a computer chip

Description automatically generated

1. **CSI Data Acquisition via ESP32 Nodes:**

* Two ESP32s as CSI Collectors: Two ESP32 microcontrollers act as sensor nodes, each equipped with Wi-Fi transceivers to capture CSI data.
* Wi-Fi Communication for Data Relay: The ESP32 nodes communicate wirelessly via Wi-Fi to efficiently transfer CSI data between them.

1. **Data Transmission to Jetson Nano:**

* Serial Port for Reliable Delivery: The receiver ESP32 node sends the collected CSI data to the Jetson Nano via a serial port connection, ensuring a robust and streamlined data transfer.

1. **Activity Recognition on Jetson Nano:**

* Segmentation and Preprocessing: The Jetson Nano segments the CSI data into meaningful activity segments and performs necessary preprocessing to prepare it for inference.
* AI-Powered Inference with TensorRT: A pre-trained CNN model optimized for CSI-based HAR is deployed and executed on the Jetson Nano via the TensorRT engine. This significantly accelerates inference speed and optimizes resource utilization.
* Real-Time Activity Classification: The CNN model analyzes the processed data and classifies human activities in real-time.

1. **MQTT-Based Communication for Real-Time Updates:**

* HiveMQ as the MQTT Broker: The Jetson Nano publishes the activity recognition results to a HiveMQ MQTT broker, enabling seamless data exchange with other connected devices.

1. **Android App for Visualization and Interaction:**

* Subscribing to MQTT Topics: An Android application subscribes to the relevant MQTT topics to receive real-time activity predictions from the Jetson Nano.
* User-Friendly Display: The Android app presents the activity information in an intuitive and user-friendly manner, providing real-time insights into human behaviors.

1. System realization
2. **ESP32 Configuration and CSI Data Collection**
   1. ESP32 CSI Toolkit Deployment: The two ESP32 nodes were configured to run applications built using the ESP32 CSI Toolkit, a specialized framework designed for CSI data collection within the ESP-IDF environment.
   2. CSI Data Acquisition: The Toolkit enabled efficient capture of CSI data from Wi-Fi signals, reflecting subtle changes in the wireless environment caused by human movements.
   3. Dataset Creation: A comprehensive dataset was generated, comprising 525 segments of 200 packets each, totaling 105,000 packets. These segments were carefully labeled across three distinct activities: **Raising Right Arm, Raising Left Leg, and Stretching Out**.
3. **CNN Model Development and Training**
   1. Model Architecture: A Convolutional Neural Network (CNN) model was designed specifically for CSI-based human activity recognition, featuring the following architecture:
      1. Input Layer: Shape (200, 55, 1), accommodating the dimensions of CSI data segments.
      2. Convolutional Layer: Extracting spatial features from the input CSI data.
      3. Pooling Layer: Downsampling the feature maps for dimensionality reduction.
      4. Flatten Layer: Converting the pooled feature maps into a one-dimensional vector.
      5. Dense Layers (2): Fully connected layers for high-level feature learning.
      6. Dropout Layers (2): Preventing overfitting during training.
      7. Output Layer: Three neurons, corresponding to the three activity classes.
   2. Model Training: The CNN model was trained on the collected CSI dataset, learning to accurately classify the three target activities. Google Colab was used entirely for the training phase.
   3. Training result: The model has a validation loss of 0.2469 and validation accuracy of 0.9324. ([Link to Jupyter notebook file](https://github.com/thu4n/ESP32-WiFi-Sensing/tree/master/notebooks/3_act_CSI_CNN_train.ipynb))

A screenshot of a computer screen

Description automatically generated

1. **MQTT Broker Deployment and Android App Development**

* HiveMQ Cluster Establishment: A cluster was created within the HiveMQ Cloud platform's free plan, providing a reliable MQTT broker to facilitate real-time communication between the Jetson Nano and the Android app.
* Android App Development: Using Android Studio, a basic mobile application was developed in Java to visualize activity recognition results in real-time. The app's core functionalities include:
  + MQTT Client Integration: The app establishes a secure connection to the HiveMQ broker using appropriate MQTT client libraries.
  + Topic Subscription: The app subscribes to the specific MQTT topic “predict” where the Jetson Nano publishes activity predictions.
  + Data Handling: Upon receiving activity predictions, the app parses and extracts relevant information for user-friendly display.

1. **Code and Dataset Availability**

* **Dataset:** The CSI dataset that we collected and used for this project is publicly available at: <https://drive.google.com/drive/folders/1rnbM4t8aVR6NuMp-QE3FwWFJ-vZKpR8Y?usp=sharing>
* **Code:**

**+ ESP32 and Jetson Nano:**

<https://github.com/thu4n/ESP32-WiFi-Sensing>

**+ MQTT Client Android Application:**

<https://github.com/thu4n/NT131-MQTT-Client>

1. Conclusions - Key achievements

* In conclusion, our project has achieved the main contents that are specified from the beginning. Those are:
* Understanding the functionalities of ESP32 and Jetson Nano.
* Using machine learning techniques to recognize human activities based on CSI data extracted from Wi-Fi signals.
* Developing a wireless embedded network system between ESP32 and Jetson Nano based on the understanding from the two previous points.
* However, since the system is in the experimental process and the project’s time constraint, there are some limitations that right now we can’t surmount. Those disadvantages can be known as:
* The prediction can be incorrect if we deploy the system in different places or the activities aren’t from the same person when we train the model. Because the CSI signals are not the same if they are collected from different people and in different environments.
* We use only 2 threads when running the model in Jetson Nano. To be more specific, one thread is used for collecting the CSI signals and analyzing those, the other one is used for publishing the messages to the MQTT broker. As a result, some CSI signals are lost during the model’s process, making the accuracy of the prediction is lower than expected.
* In the future, if we have sufficient time, beside of fixing those problems, we will add more predicted activities such as: walking, jogging, jumping,… and develop more service for users, for example: improving the application performance and features, developing a web application for easier access, etc...

1. References
2. [ESP32 CSI Toolkit (stevenmhernandez.github.io)](https://stevenmhernandez.github.io/ESP32-CSI-Tool/)
3. [Get Started - ESP32 - — ESP-IDF Programming Guide release-v4.3 documentation (espressif.com)](https://docs.espressif.com/projects/esp-idf/en/release-v4.3/esp32/get-started/index.html)
4. S. M. Hernandez\*, M. Touhiduzzaman\*, P. E. Pidcoe and E. Bulut, “**Wi-PT: Wireless Sensing based Low-cost Physical Rehabilitation Tracking**,” in Proc. of 2022 IEEE International Conference on E-health Networking, Application & Services (IEEE Healthcom 22), Genoa, Italy, October 2022.
5. S. M. Hernandez and E. Bulut, "**WiFi Sensing on the Edge: Signal Processing Techniques and Challenges for Real-World Systems**," in IEEE Communications Surveys & Tutorials, vol. 25, no. 1, pp. 46-76, Firstquarter 2023, doi: 10.1109/COMST.2022.3209144.
6. H. Choi, M. Fujimoto, T. Matsui, S. Misaki and K. Yasumoto, "**Wi-CaL: WiFi Sensing and Machine Learning Based Device-Free Crowd Counting and Localization**," in IEEE Access, vol. 10, pp. 24395-24410, 2022, doi: 10.1109/ACCESS.2022.3155812.
7. [Quick Start Guide :: NVIDIA Deep Learning TensorRT Documentation](https://docs.nvidia.com/deeplearning/tensorrt/quick-start-guide/index.html)
8. [Leveraging TensorFlow-TensorRT integration for Low latency Inference — The TensorFlow Blog](https://blog.tensorflow.org/2021/01/leveraging-tensorflow-tensorrt-integration.html)
9. [tensorflow/tensorrt: TensorFlow/TensorRT integration (github.com)](https://github.com/tensorflow/tensorrt)
10. [HiveMQ Introduction :: HiveMQ Documentation](https://docs.hivemq.com/hivemq/latest/user-guide/index.html)
11. [sachintha00/Android-Login-UI: Mobile Login UI Design](https://github.com/sachintha00/Android-Login-UI)