Grant Agreement Number:
619193-EPP-1-2020-1-BE-EPPKA2-CBHE-JP

Co-funded by the Erasmus+ Programme of the European Union

DigiHealth-Asia

Deliverable 2.2 Development of communication systems and protocols

Work package	WP2
Task	Task 2.2
Due date	15th May 2023
Submission date	5th June 2023
Deliverable lead	UGent
Version	0.1
Authors	Adnan Shahid (UGent), Ingrid Moerman (UGent), Rafia Mumtaz (NUST), Amir Qayyum (CUST), Mammona Qudsia (CUST), Pradorn Sureephong (CMU), Surapong Uttama (MFU), Ariuntuul Garidkhuu (MNUMS), Mend-Amar Majig (NUM)
Reviewers	Daud Abdullah Asif (NUST), Suwit Wongsila (CMU)

© DigiHealth-Asia Consortium 2021-2024

Table of Contents

© DigiHealth-Asia Consortium 2021-2024

Executive Summary

Deliverable 2.2 is associated with Task 2.2 of Work Package 2 Development. This deliverable focuses on defining the communication architecture and protocol required for the reliable transfer of sensor data in three pilot cases: Pilot Case 1 – cardiovascular monitoring (Pakistan), Pilot Case 2 – mobility disorder monitoring (Thailand), and Pilot Case 3 – remote patient consultation (Mongolia). In each pilot case, there are three communication entities: sensor, edge, and cloud.

The deliverable begins by presenting an overall system architecture for each pilot case, providing a clear description of the components used in each communication entity. For each pilot case, there are two levels of communication: a) between the sensor and edge, and b) between the edge and cloud. The deliverable describes the communication technologies and protocols employed at each level. Due to the varying availability of communication interfaces and diverse application requirements for each pilot case, different communication technologies are utilized at each level, and these are described in detail.

1. Introduction

This deliverable describes communication architecture and protocols for the three pilot cases. For each pilot, we recap the system architecture that we reported in Deliverable 2.1 [1]. The system architecture comprises of three communication entities sensor, edge and cloud. The sensor is used for acquiring monitoring data (e.g., ECG, temperature, activity, PPG, etc.) from people. The edge is utilized for short-term machine learning inference, filtering, and raising alarms. On the other hand, the cloud is employed for machine learning model training and long-term analysis. Once the model training is completed, it is transferred from the cloud to the edge for inference.

This deliverable describes the communication architecture and protocols between a) sensor and edge, and b) edge and cloud. The communication technologies differ among the pilot cases due to variations in the availability of communication technology interfaces and the application requirements, such as latency, packet error, throughput, etc.

2. Pilot case 1 – Cardiovascular monitoring (NUST)

2.1. Overall architecture

The overall system architecture of cardiovascular monitoring is shown in Figure *2-1*. It consists of four main parts i.e., sensors, edge device, cloud and web-based GUI. For the cardiovascular pilot case, the following parameters will be measured:

- Body temperature
- Blood Oxygen Saturation (SpO2)
- Heart Rate
- Blood Pressure
- Electrocardiogram (ECG)
- Photoplethysmography (PPG)
- Respiration/Breathing Rate

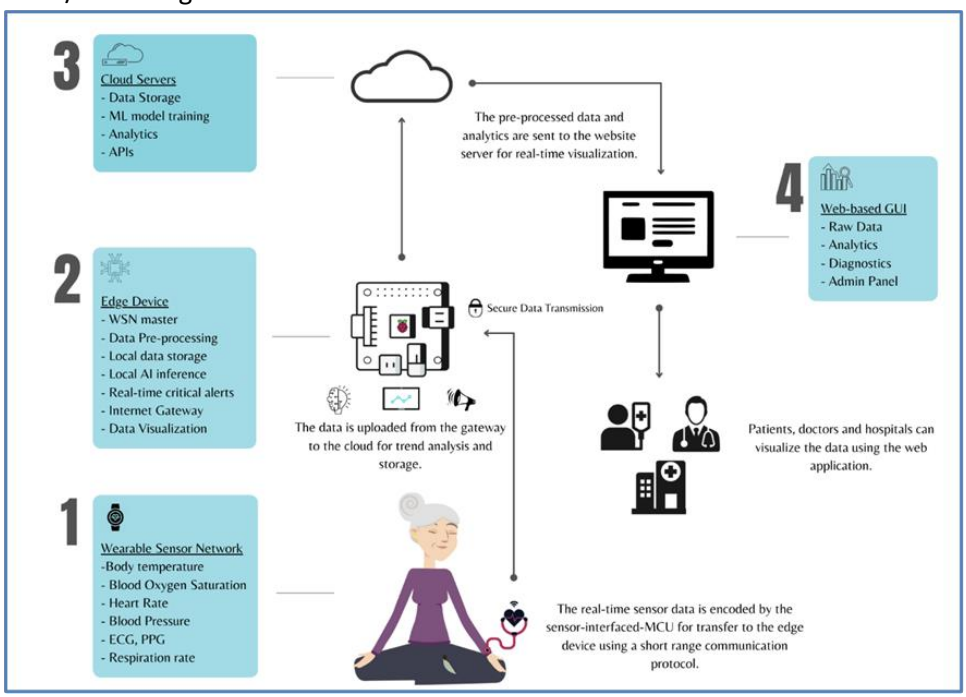

Figure 2-1:System Architecture

Edge Device:

The edge device consists of two parts: i. sensor with microcontroller (Arduino UNO), ii. Microcontroller sending data to the RaspberryPi as shown in Figure *2-2*. The edge device is monitoring the following health vitals currently which are explained in detail below. The other sensors will be integrated into the system subsequently.

1. ECG
2. SPO_2

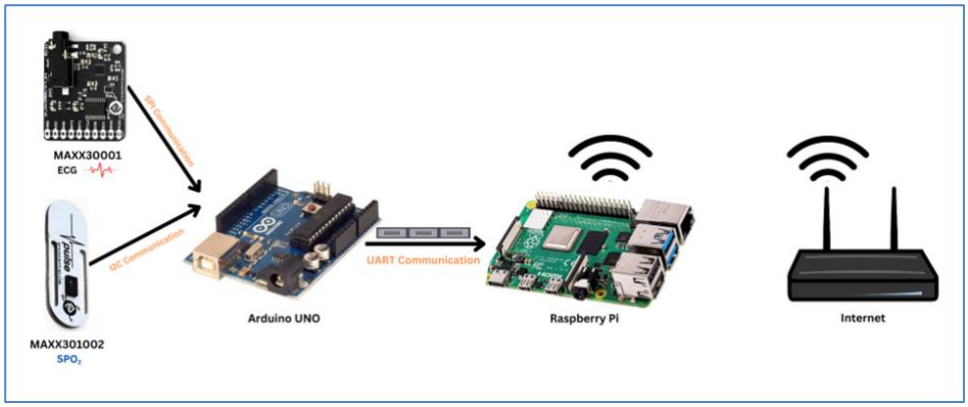

Figure 2-2: Edge Device Architecture

2.2. Communication between sensor and edge:

The sensors communicate with the Arduino using their respective communication protocols: SPI for the ECG sensor and I2C for the SPO2 sensor. Both of these communication protocols are reliable and support high transmission rates with minimal or no data loss. The microcontroller then reads the data from the sensors in the form of their respective data packets, processes it, and writes the processed data via serial communication to the Raspberry Pi. This data is transmitted through serial communication using UART in the form of data packets to the Raspberry Pi, which must be read separately in order to store each packet individually.

ECG Sensor:

The communication protocol utilized between the ECG sensor MAXX30001 and the Arduino microcontroller is Serial Peripheral Interface (SPI), which is a synchronous, full-duplex communication protocol. SPI is widely employed in scenarios where high-speed data transmission with minimal errors or packet loss is required. Given that the ECG data in our case is transmitted at a frequency of 125 Hz and a baud rate of 57600, relying on any wireless protocol for this transmission between the sensor and the microcontroller (in the edge device) could introduce risks of data loss.

The SPI connection uses four pins:

MOSI (Master Out Slave In): This pin is used to send data from the Arduino to the MAX30001.

MISO (Master In Slave Out): This pin is used to receive data from the MAX30001 to the Arduino.

SCLK (Serial Clock): This pin provides the clock signal for data synchronization.

CS (Chip Select): This pin is used to activate the MAX30001 for communication.

Data Packets:

The data packets in the given code are formatted according to the ProtoCentral OpenView format, which consists of the following parts:

Packet header, data payload, and packet footer. The packet header contains the start bytes, packet length, and packet type, while the packet footer contains the stop byte.

In the sendData() function, the ECG and BioZ samples are placed in the data payload. The data payload is an array of 12 bytes, where the first 4 bytes contain the ECG sample, the next 4 bytes contain the BioZ sample, and the remaining bytes are used for additional information (such as heart rate, which is set to zero in the given code).

Baud Rate and Data Transmission Rate:

The baud rate for serial communication between the Arduino board and RaspberryPi is set to 57600 bps using the Serial. Begin (57600) function in the setup() function. This baud rate ensures fast data transmission, enabling the Arduino to keep up with the data acquisition rate.

The data acquisition rate is determined by the delay between consecutive samples. In the given code, there is a delay of 8 milliseconds between samples, which corresponds to a sampling frequency of 125 Hz (1 / 0.008 s).

The data is transmitted between sensor and the edge device is first transmitted by the SPI or I2C protocol between the sensor and the microcontroller, followed by transmission between the microcontroller and the edge device using wired UART communication protocol.

This complete transmission process can be summarized as:

Arduino Setup: The Arduino is configured to communicate with the MAX30001 sensor using the SPI (Serial Peripheral Interface) protocol. The sensor is initialized, and the Arduino starts reading ECG and BioZ data.

Serial Communication: Arduino's Serial object is initialized with a baud rate of 57600 bits per second. The baud rate determines how fast the data is transmitted between the Arduino and the Raspberry Pi. A higher baud rate would allow for faster data transmission but could also increase the risk of data corruption or loss if the receiving device (Raspberry Pi) cannot keep up.

Data Packet Formatting: ECG and BioZ data is formatted into a custom data packet. The data packet consists of a header, payload, and footer. The header contains synchronization bytes (CES_CMDIF_PKT_START_1 and CES_CMDIF_PKT_START_2), the length of the data packet (DATA_LEN), and the type of data (CES_CMDIF_TYPE_DATA). The payload contains the actual ECG and BioZ data, and the footer contains a zero byte and a packet stop byte (CES_CMDIF_PKT_STOP).

Data Transmission: The formatted data packet is transmitted to the Raspberry Pi using Arduino's Serial. Write() function. This function sends the data bytes one by one over the UART connection to the Raspberry Pi.

Raspberry Pi Reception: The Raspberry Pi listens to the UART connection and receives the transmitted data bytes. It is essential that the Raspberry Pi's serial connection is also configured with the same baud rate as the Arduino (57600) to ensure correct data reception. The Raspberry Pi can then process the received data and perform further analysis or visualization.

SPO_2 Sensor:

The sensor relates to Arduino microcontroller that reads the values using I2C communication, then this data is transmitted to RaspberryPi using UART via serial communication. Arduino reads the sensor data and sends it to the Raspberry Pi. The communication setup can be explained further as:

Arduino setup: In the provided code, the Arduino sets up the MAX30102 sensor and initializes the Serial communication with a baud rate of 57600 (Serial.begin(57600)). The baud rate defines the speed at which data is transferred, and both the devices (Arduino and Raspberry Pi) must use the same baud rate for successful communication.

Reading sensor data: The Arduino reads the sensor data by calling the 'sensor.readSensor()' function, which retrieves the red and infrared LED intensity values (REDD and IRR).

Creating data packets: The Arduino then creates data packets containing the sensor data and some header and footer information. Each data packet is 15 bytes long and includes start indicators, data length, data type, REDD, IRR, and end indicators.

Sending data packets: The Arduino sends the data packets to the Raspberry Pi through the serial. Write() function. The data packets are transmitted one byte at a time at the specified baud rate of 57600.

Raspberry Pi setup: On the Raspberry Pi side, a UART communication library (e.g., PySerial) needs to be set up to read the data packets from the Arduino. The Raspberry Pi's UART library should be configured with the same baud rate as the Arduino (57600).

Reading data packets: The Raspberry Pi listens for incoming data packets through the UART interface. When a data packet is received, it processes the packet by reading the header information (start indicators, data length, and data type) and extracting the sensor data (REDD and IRR).

Data processing: Once the Raspberry Pi has the sensor data, it can further process the information for various applications, such as calculating SpO2 levels, heart rate, or other health metrics.

Data Packet:

The data packet for the MAX30102 sensor in the provided code is structured as follows:

Header (5 bytes):

Byte 0: 0x0A (Start of packet byte 1)

Byte 1: 0xFA (Start of packet byte 2)

Byte 2: Data length (lower 8 bits)

Byte 3: Data length (upper 8 bits)

Byte 4: 0x02 (Data packet type)

Data payload (8 bytes):

Bytes 5-8: RED LED data (4 bytes, stored as a 32-bit integer)

Bytes 9-12: IR LED data (4 bytes, stored as a 32-bit integer)

Footer (2 bytes):

Byte 13: 0x00 (Unused byte)

Byte 14: 0x0B (End of packet)

The data packet has a total size of 15 bytes. It starts with a 5-byte header that includes start bytes, data length, and packet type information. The 8-byte payload contains the RED and IR LED data from the MAX30102 sensor. Finally, the packet ends with a 2-byte footer to mark the end of the packet. This data packet structure allows efficient and organized transmission of sensor data from the Arduino to the Raspberry Pi via UART communication.

Baud Rate and Data Transmission Rate:

In this sensor setup, the baud rate is set to 57600 bits per second. The baud rate determines the speed at which data is transmitted between the Arduino and Raspberry Pi through the UART communication protocol. Both devices must be configured to use the same baud rate to ensure successful and reliable data transfer.

The data transmission rate, or data frequency, depends on both the baud rate and the time delay between sending data packets. In the provided code, a 10ms delay (delay(10)) is introduced between sending each data packet. This means the system sends 100 data packets per second (1 second / 0.01 second). With a 15-byte data packet, the data transmission rate becomes 15 bytes * 8 bits/byte * 100 packets/second = **12,000 bits per second**.

It's important to note that the data transmission rate is significantly lower than the baud rate (12,000 bps compared to 57,600 bps). This difference provides a safety margin to ensure reliable data transmission without overwhelming the receiving device or causing data loss.

2.3. Communication between edge and cloud:

The Raspberry Pi acts as an edge device, receiving sensor data from the Arduino via UART serial communication and then uploading it to the cloud. To achieve this, the Raspberry Pi needs to communicate with a cloud service over the internet. The communication between the Raspberry Pi and the cloud involves an internet connection, a cloud service, a communication protocol (such as MQTT or HTTP), authentication and security mechanisms, and various tools for processing, storing, accessing, and visualizing the data.

The integral parts communication process between the edge device (Raspberry Pi) and the cloud is as follows:

Internet connection: The Raspberry Pi needs an active internet connection to communicate with the cloud. This connection can be established through Wi-Fi, Ethernet, or even cellular networks using a USB dongle or a compatible HAT (Hardware Attached on Top).

Cloud service: You need to choose a cloud service to store and manage the data sent by the Raspberry Pi. Popular cloud service providers for IoT applications include Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and IBM Cloud.

Communication protocol: To transfer data between the Raspberry Pi and the cloud, a communication protocol is required. The most common protocols for IoT applications are MQTT (Message Queuing Telemetry Transport) and HTTP (Hypertext Transfer Protocol).

MQTT: This is a lightweight, publish-subscribe protocol designed for constrained devices and low-bandwidth, high-latency, or unreliable networks. MQTT is advantageous for IoT applications due to its low power consumption, small data packets, and real-time communication capabilities.

HTTP: This is the protocol used to transfer data over the World Wide Web. It is a request-response protocol, which makes it suitable for transmitting data between the Raspberry Pi and the cloud. However, HTTP may be less efficient than MQTT for real-time IoT applications due to its higher overhead and latency.

Authentication and security: When transmitting sensitive data like ECG and SpO2 signals, it's crucial to ensure that the data is secure. To protect the data, authentication, and access control mechanisms provided by the chosen cloud service can be used.

Data processing and storage: Once the data reaches the cloud, it can be stored, processed, and analyzed based on your requirements. Cloud platforms offer various tools and services for data storage (e.g., databases), data processing (e.g., stream processing), and data analytics (e.g., machine learning).

3. Pilot case 1 – Cardiovascular monitoring (CUST)

3.1. Overall architecture

The architecture for remote monitoring of cardiovascular patients is shown in Figure 3-1.

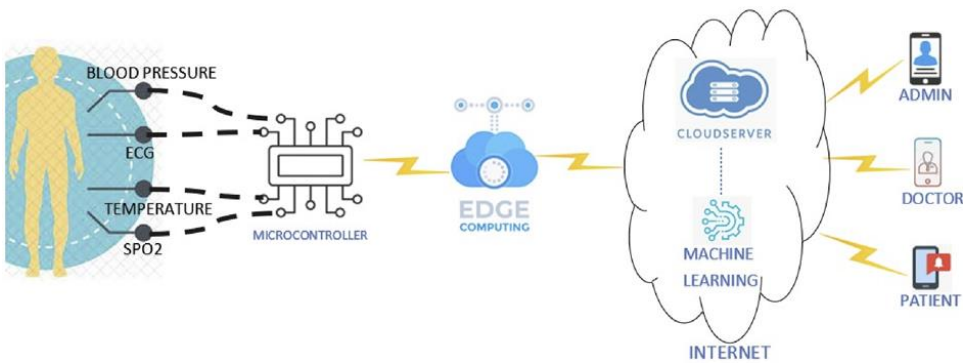

Figure 3-1: Remote Monitoring System for Cardiovascular Patients by CUST

A network of sensors is used to acquire the medical vital data such as ECG, Heart rate, SPO2 and body temperature of cardiovascular patients. These sensors are interfaced to an ESP32 microcontroller. Acquired data is transmitted wirelessly to the edge device for further processing and storage at regular intervals. Edge device receives the data from the sensor network and performs following tasks:

- Local data storage
- Machine Learning based inference
- Generation of alert messages
- Transmission of preprocessed data to the cloud

After processing, the data from the edge device is transmitted to the Cloud server. Main purpose of the cloud server is the storage of patient data, training of Machine Learning models and long-term Machine Learning based inference. Data in the cloud will be used for long term inference based upon machine learning. The outcome of this inference will be available to the patient and the caregivers, which is the final layer of the system. At the user end, a mobile application will be used to view patient records and history, visualize data analytics and receive warning alerts. Using the mobile application, doctors can view the patient record and history. Mobile applications will also be used to share the alerts and critical decisions from the edge device or from the cloud based upon long-term machine learning predictions. These alerts and warning messages are conveyed to different stakeholders. These stakeholders include the patient or the care givers that include doctors, nurses and paramedics.

3.2. Communication between sensor and edge

The MLX90614 and MAX30102 sensors that are used to monitor Temperature, Heart rate and SPO2, are interfaced with an ESP32 microcontroller using an I2C communication interface. This involves connecting the data pins and voltage/ground pins of the sensors to the corresponding pins on the ESP32 microcontroller. The ESP32 microcontroller is configured to communicate with the sensors using I2C. This involves initializing the I2C peripheral on the ESP32 microcontroller, setting the appropriate I2C communication speed, and configuring the ESP32 microcontroller's I2C pins.

The ADS8232 sensor (ECG sensor) is interfaced with an ESP32 microcontroller using an analog communication interface. The output pin of the ADS8232 sensor is connected to an analog input pin on the ESP32 microcontroller. ESP32 microcontroller is configured to read the analog signal from the ADS8232 sensor.

The ESP32 microcontroller and Raspberry Pi (Edge) communicate with each other using Wi-Fi as the primary wireless connection method. Both the ESP32 and Raspberry Pi have built-in Wi-Fi capabilities, allowing for straightforward wireless communication between the two devices. Data can be transferred over the established Wi-Fi connection. Additionally, Bluetooth is also available as an alternative option for wireless communication. Both the ESP32 and Raspberry Pi support Bluetooth, providing an alternative wireless communication method. Bluetooth can serve as a backup option if a Wi-Fi connection is not available between the microcontroller and the edge device.

If the ESP32 and Raspberry Pi are communicating using Wi-Fi, they can use several communication protocols over the wireless network. The preferred choices for this pilot case will be:

1. MQTT: Message Queuing Telemetry Transport (MQTT) is a lightweight, publish-subscribe protocol that is commonly used for IoT applications. The ESP32 and Raspberry Pi can communicate using MQTT over Wi-Fi.

2. If the ESP32 and Raspberry Pi are communicating using HTTP as a protocol over Wi-Fi, then TCP/IP protocols will be used for communication. TCP/IP provides reliable, ordered, and error-checked delivery of the data.

3.3. Communication between edge and cloud

The communication between the edge device (Raspberry Pi) and the cloud server is established through an internet connection. The edge device receives data from the sensor network and performs pre-processing on the received data. Once the data is pre-processed, it is transmitted to the cloud server for storage and training of Machine Learning Models.

The communication between the Edge device (Raspberry Pi) and the cloud server can be established using a wired internet connection.

The Edge device (Raspberry Pi) and cloud server are communicating using Wi-Fi over the internet. This can be accomplished using several communication protocols, however the preferred choices for this pilot case will be.

1. Raspberry Pi can be configured to use HTTP requests to send data to a server or retrieve data from a server using REST APIs.
2. Raspberry Pi can be configured to use MQTT. Message Queuing Telemetry Transport (MQTT) is a lightweight, publish-subscribe messaging protocol that is commonly used for IoT applications. MQTT allows edge devices to send data to a cloud server. Also, cloud servers can send data to the edge device using MQTT.

4. Pilot case 2 – Mobility disorder monitoring (CMU and MFU)

4.1. Overall architecture

Pilot case II involves the development of a mobility disorder monitoring system which involves two partners working together. MFU and CMU collaborate to develop a sustainable system to prevent mobility disorder in the elderly due to fall. The entire system architecture of Thai partners is illustrated in Figure *4-1*. MFU focuses on fall-risk assessment at healthcare centers while CMU intends to monitor the elderly's activities and exercises at home.

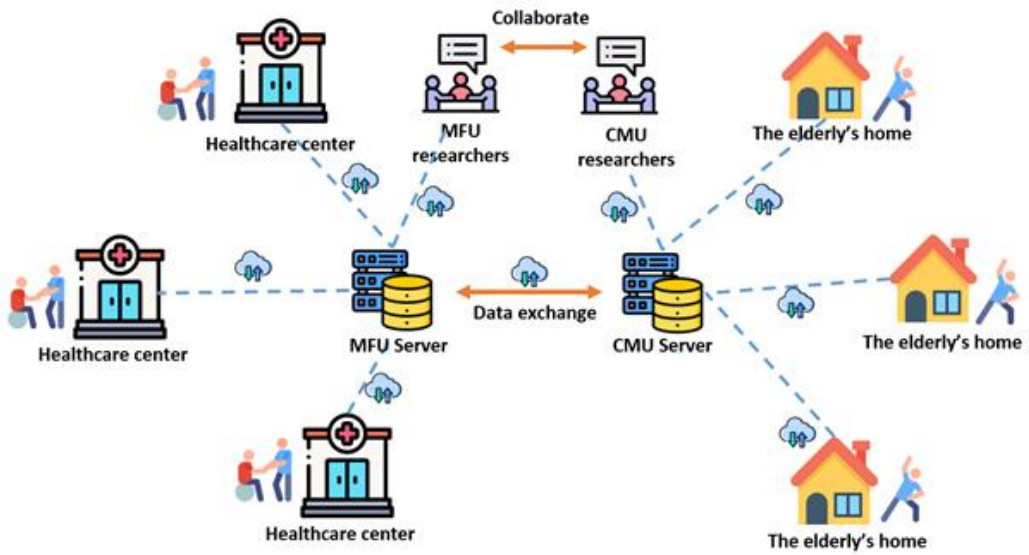

Figure 4-1: Mobility disorder monitoring system architecture in Pilot Case 2 by MFU and CMU.

4.2. CMU use case

For the CMU use case, extensive work has been undertaken to monitor home exercise routines for the elderly to prevent mobility disorders. The architecture of the CMU pilot case, as depicted in Figure below, features a user interface tailored specifically for elderly users. This interface is connected to an edge computing system that utilizes a depth camera sensor to capture the user's exercise posture. The output from the edge system is displayed on a TV, allowing elderly users to interact with it. Serving as the primary component for assisting the elderly in their exercise routines, the edge computing system collects and analyzes the exercise posture data captured by the depth camera sensor. Subsequently, this data is transmitted to the cloud for storage and further processing.

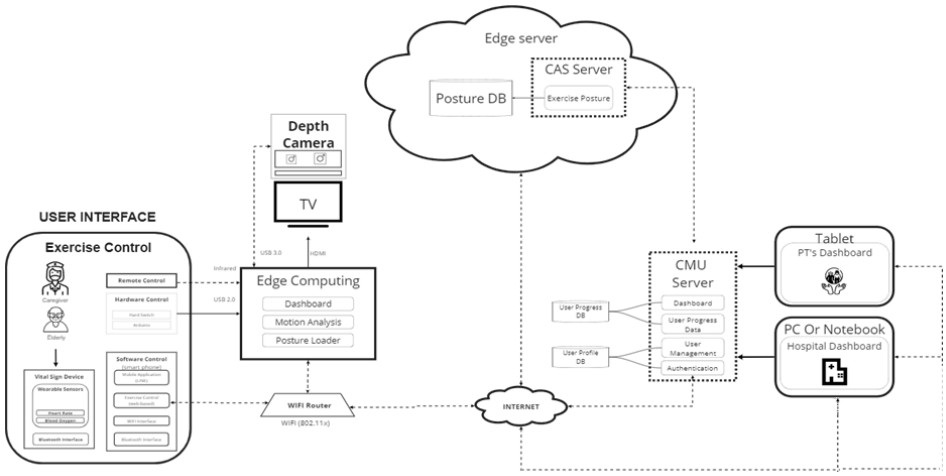

Figure 4-2: Overall Architecture for CMU pilot case

The Cloud storage serves dual purposes. Firstly, it stores the exercise posture data, which can be retrieved by the edge system to provide guidance to the elderly during their exercises. Secondly, the cloud server receives progress updates from elderly users, facilitating the generation of a dashboard that allows physical therapists to monitor the exercise behavior of the elderly. The communication between the edge system and the cloud is established through a dedicated protocol designed for secure and efficient data transfer of the exercise posture data. The architecture aims to offer a user-friendly interface for elderly users to engage in exercises while effectively collecting and processing data in the edge system and cloud for further analysis and feedback.

4.2.1. Communication between sensor and edge

The CMU pilot communication system comprises two parts: motion detection and heart rate detection. The motion sensor communication between the depth camera sensor and the edge system is established through a reliable and efficient USB 3.0 connection. With its high bandwidth, the USB 3.0 connection ensures accurate capture and analysis of the elderly exercise body data. Additionally, this connection guarantees secure and low-latency transmission of the sensor data to the edge system, which is vital for real-time processing of the body data. This enables the edge system to quickly and accurately detect exercise posture and provide feedback to elderly users, ensuring that they perform the exercise correctly.

On the other hand, the heart rate sensor, which utilizes the MAX30102 heart rate oximeter chip, communicates with a wearable device, such as a smartwatch, via I2C communication. The communication schematic is depicted in Figure *4-3*. The I2C is a reliable and widely used communication protocol for short-range communication between integrated circuits. One of the key benefits of I2C communication is its ability to facilitate communication between multiple devices with

only two wires, making it a cost-effective solution for connecting multiple sensors and devices to a single MCU. Furthermore, I2C communication supports bidirectional data transfer, allowing the wearable device to receive heart rate data from the sensor and send control signals back to the sensor. The edge computing device for the heart rate sensor is based on the ESP32 Dual Core MCU (Lily Go®).

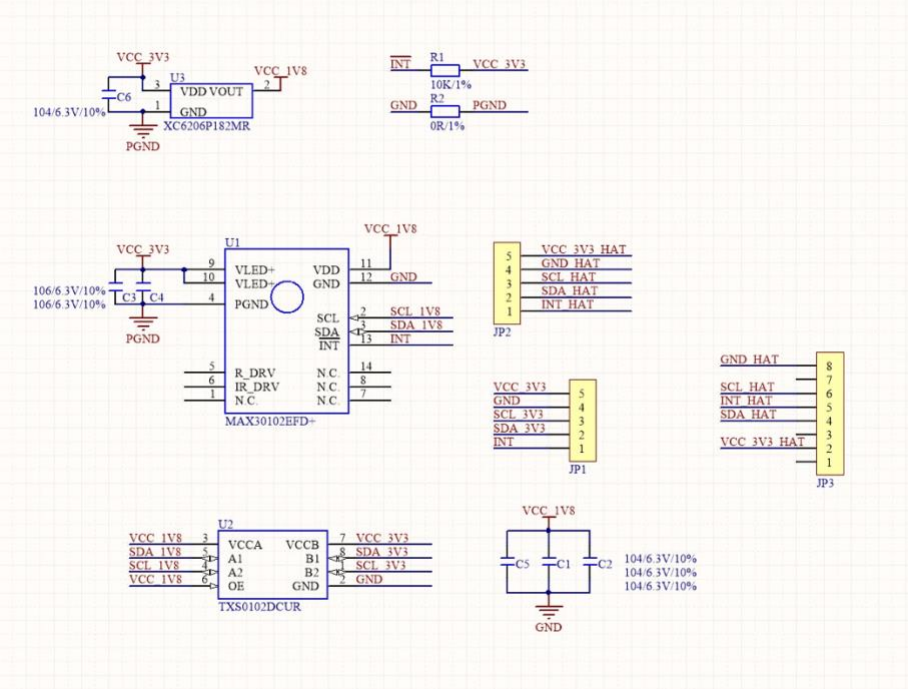

Figure 4-3: Communication schematic for heart rate monitoring sensor

The CMU pilot utilized two key wireless technologies: Wi-Fi and Bluetooth. Wi-Fi (802.11x) communication was employed for motion detection edge computing to establish a connection with the server through the internet. This allowed for seamless data transfer and communication between the edge computing system and the server, enabling real-time analysis and processing of motion data.

In addition, Bluetooth communication was utilized for the smartwatch to establish connections with mobile devices or the motion detection edge. Bluetooth technology provides a short-range wireless connection, making it suitable for connecting devices in close proximity. By utilizing Bluetooth, the smartwatch could efficiently exchange data with mobile devices or the motion detection edge, facilitating convenient monitoring and control of the CMU pilot system. Both Wi-Fi and Bluetooth technologies play essential roles in the CMU pilot, enabling wireless data communication and connectivity between different components of the system. Wi-Fi ensures robust and high-speed communication over longer distances, while Bluetooth enables seamless connectivity between devices in close proximity. Together, these wireless technologies enhance the functionality and usability of the CMU pilot system.

In the CMU pilot for mobility disorder prevention, the communication protocol utilized within the architecture is TCP/IP (Transmission Control Protocol/Internet Protocol). TCP/IP is a widely adopted protocol suite for network communication, specifically designed for reliable and secure data transmission over the internet. By implementing TCP/IP, the CMU pilot ensures robust connectivity between the various components of the system, including the edge computing devices, servers, and user interfaces. This protocol enables seamless transmission of exercise posture data and progress updates from the edge system to the cloud storage, allowing for storage, analysis, and further

processing. Moreover, TCP/IP facilitates the generation of a user-friendly dashboard for physical therapists, enabling them to monitor and follow up on the exercise behavior of elderly users.

4.2.2. Communication between edge and cloud

For CMU, the communication between the edge system and the cloud is established through a wireless communication system, specifically utilizing Wi-Fi. This decision was driven by the need to securely collect and transmit exercise data of elderly users to the cloud for storage and further processing. Wi-Fi technology offers a convenient and reliable wireless communication method, well-suited for transmitting large amounts of data over long distances with robust security measures. Moreover, Wi-Fi is user-friendly, making it an ideal choice for the CMU system.

In the case of CMU, the communication between the edge system and the cloud is established using WiFi, which provides a convenient and efficient means of data transfer. WiFi boasts several advantages over other communication methods, including its widespread availability, affordability, and ease of use. Additionally, WiFi supports high-speed data transfer, a critical factor in transmitting the substantial exercise data collected by the system.

For CMU, the commonly employed protocol for wireless communication between the edge system and the cloud is Dual Channel 802.11 b/g/n. This protocol was selected for its reliability and ability to facilitate high-speed data transfer. Operating in the 2.4GHz and 5GHz frequency bands, it minimizes interference and enhances overall wireless communication performance. Moreover, this widely compatible protocol offers practicality for the research project, ensuring seamless connectivity across various devices. With built-in security features like WPA2 encryption, the integrity and confidentiality of the data transferred between the edge system and the cloud are upheld, protecting it from unauthorized access.

4.3. MFU use case

4.3.1. Overall architecture

MFU focuses mainly on fall risk assessment by developing IoT devices to evaluate the elderly's performance according to SPPB tests including Five-time sit-to-stand test, Time up and Go test, Standing Balance test and Gait speed test. The collected data from the sensors will be transmitted to edge layer relied on a device such as Raspberry Pi for storage, preprocessing, and inferencing later. The use case can be illustrated in Figure *4-4*.

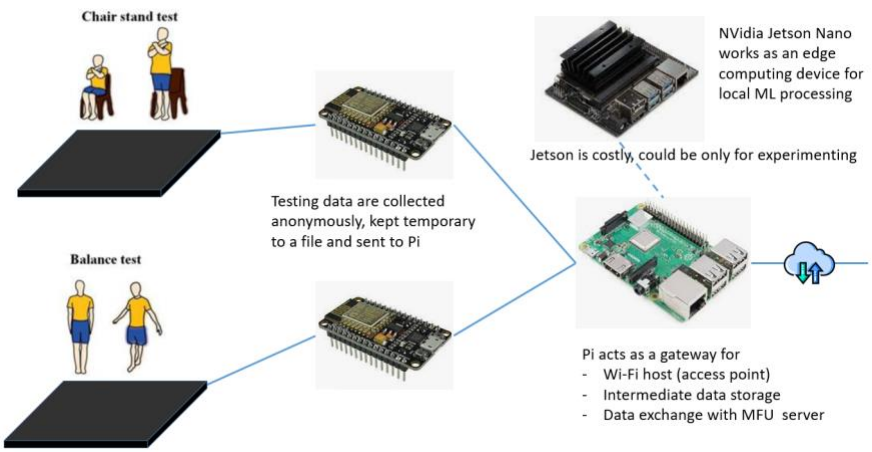

Figure 4-4: Example of data collection, processing and communication on MFU's use case in pilot case 2

Subsequently the processed data from the edge layer will be transferred to the cloud which is our own server for training machine learning model and the server then returns the model to the edge layer for further inferencing. The system architecture for MFU is shown in figure below.

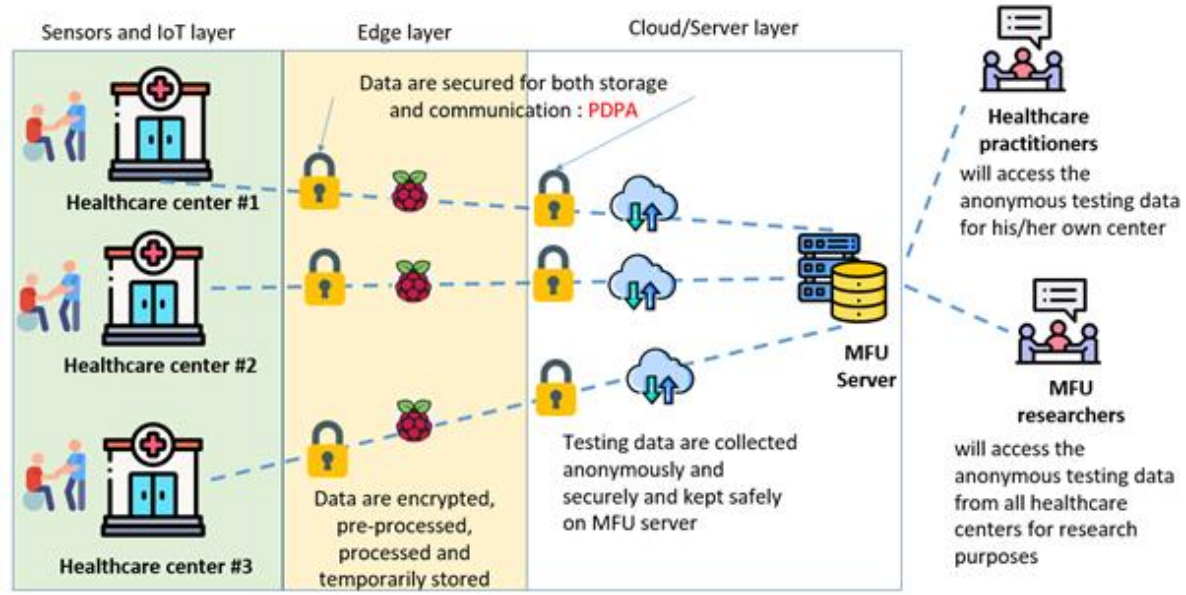

Figure 4-5:The system architecture implemented by MFU in pilot case 2

4.3.2. Communication between sensor and edge

Our developed IoT devices are designed to gather data from sensors and store it locally on an SD card, as well as transmit it remotely to the edge device. At the sensor layer, we connect the sensors and microcontroller unit (MCU) using wires and then wirelessly submit the sensor data to the edge device using Wi-Fi. Although the NodeMCU ESP32 MCU supports both Wi-Fi and Bluetooth, we prefer Wi-Fi due to its simpler setup on the actual site, longer range, and ability to connect to multiple devices simultaneously. Wi-Fi also allows us to configure the edge device as a local access point, ensuring continuous local communication between the IoT devices and the edge device even when the internet is unavailable.

Given our preference for Wi-Fi, we have various available protocols to choose from, including TCP/UDP, MQTT, and HTTP/HTTPS. However, our preferred protocol is HTTP/HTTPS due to its compatibility with our plan to set the edge device as a local access point/server. Additionally, HTTP/HTTPS enables us to implement RESTful services using software tools like Flask or Django, installed on the Raspberry Pi. This environment is based on Python, which facilitates subsequent inference of machine learning models.

By utilizing Wi-Fi and HTTP/HTTPS, we create a seamless communication pathway between the edge layer and the cloud, enabling efficient data transmission and implementation of RESTful services. This integrated approach empowers our IoT system with the ability to store and process data locally while also leveraging cloud resources for advanced analysis and inferencing.

4.3.3. Communication between edge and cloud

The communication between edge and cloud will be through the Internet via broadband service of ISP. From our survey, the target healthcare centers are provided with broadband Internet with is more stable than cellular connection. The edge device or Raspberry Pi could be connected to an Internet access point with or without wire.

Similar to the previous layer, the communication protocol between edge and cloud will be HTTP/HTTPS in a pattern of python-based RESTFUL services. This will be compatible with model training on the server/cloud and supports the web-based interfaces to the healthcare practitioners and researchers. In addition, HTTP/HTTPS RESTFUL services will be the main communication protocol for exchanging data between the two partners: MFU and CMU.

5. Pilot case 3 – Remote patient consultation (NUM and MNUMS)

5.1. Overall architecture

The pilot use case involves the development of a diagnostic system API for dentistry, based on digital intraoral images of patients taken by several dentists. The data collection and model creation stages involve the use of an intraoral camera to take digital images of patients, which are then uploaded to a cloud server using a mobile or web application. In Figure NUM_MNUMS, the general architecture of the pilot case 3 is shown.

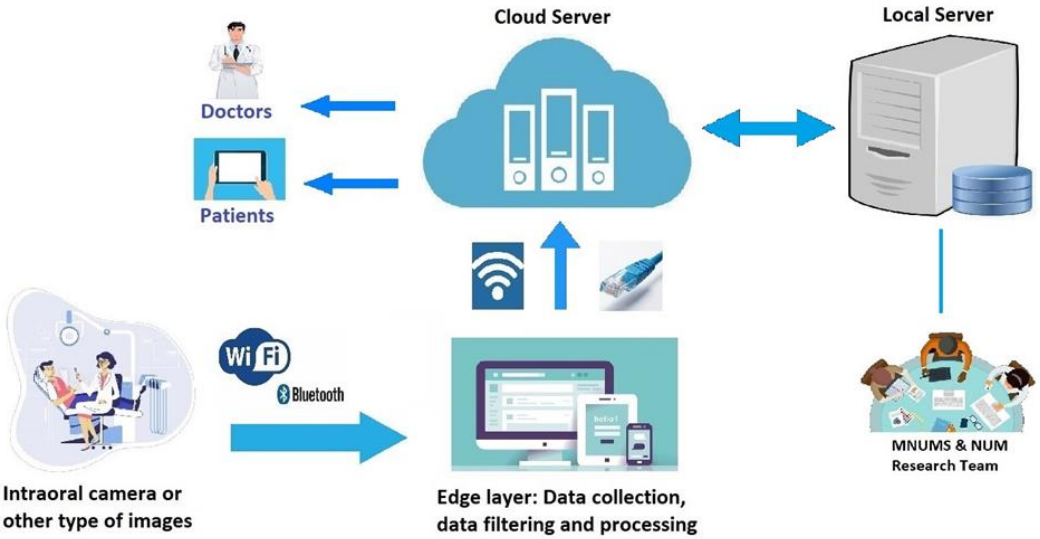

Figure 5-1:System diagram for the pilot use case 3

The data is pre-processed and filtered on the edge side to ensure its quality, and a machine learning algorithm is developed based on the uploaded data. Once the model is trained, it is deployed to the cloud server. Patient-doctor consultations can then be conducted either through a dedicated application or in person. To safeguard patient privacy, the system only collects age and gender information, ensuring that sensitive data is protected. Additionally, the mobile application incorporates a data filtering function to prevent the upload of unsuitable or inappropriate data. Overall, the system architecture involves a cyclic flow of data from intraoral camera images to machine learning model to patient-doctor consultation terminal.

5.2. Communication between sensor and edge

The intraoral cameras used in the pilot case study 3 will transfer the digital images using a USB cable. This means that the communication between the sensor and the edge will be via a wired USB connection, which provides a reliable and fast data transfer between the camera and the mobile phone or desktop computer. Upon the conclusion of this pilot case study, there are plans to deploy the system onto the internet to enable individuals located in remote areas to access it. Furthermore, a variety of intraoral cameras equipped with Bluetooth, Zigbee, and Wi-Fi capabilities may be employed.

The data collection plan for Pilot Case 3 involves the use of intraoral cameras to transfer digital images using a USB cable to a mobile phone or desktop computer. We do not utilize any wireless technologies such as Bluetooth or Wi-Fi in this process. The intraoral cameras we use are equipped with a 5-pinhole connection port and draw power from a DC 5V+/-0.5V (1.5A) source via USB. Therefore, the communication between the sensor and the edge device is conducted through a wired USB connection.

The communication protocol used by the intraoral cameras when transferring digital images through a USB cable is the USB protocol. This protocol specifies the communication between devices and their host controllers, including data transfer rates, power management, and device identification. The USB protocol is a widely adopted standard for connecting and communicating between devices, making it a reliable choice for our data transfer needs.

5.3. Communication between edge and cloud

Communication between the edge device and the cloud server is established through an internet connection. Once the data is filtered and pre-processed on the edge device, it is transferred to the cloud server for storage and further data processing. This data transfer is typically carried out using network protocols such as HTTP or MQTT, depending on the specific application requirements. The cloud server can be hosted on platforms such as Amazon Web Services or MATLAB Production Server. It is responsible for storing the pre-processed and cleaned data, as well as training the AI model based on this data. The local server is also used for long-term development and maintenance purposes.

The communication between the edge device and the cloud server can be achieved through various means, including wired connections using USB cables, wireless connections using Wi-Fi, or cellular network connections such as 4G internet. The choice of communication method depends on factors such as network availability and reliability, the amount of data being transmitted, and the system's security requirements.

For the development and pilot study of the remote consultation system for dental patients, the communication protocol between the edge device and the cloud server will primarily rely on cable or Wi-Fi connections. However, for the general deployment of the system, other communication protocols such as Bluetooth or 4G may also be utilized. The choice of communication protocol depends on several factors, including the availability of network infrastructure, the distance between devices, the required data transmission speed, and the power consumption limitations of the devices. Therefore, the system may support multiple communication protocols to provide flexibility and ensure efficient data transmission.

6. Conclusion

The deliverable presents the communication architecture and protocols for the three pilot cases. The overall system architecture for all the pilot cases is divided into three communication entities: sensor, edge, and cloud. The deliverable specifies the communication protocols and technologies used between a) sensor and edge, and b) edge and cloud. The information provided in the deliverable will be valuable for prototyping the three pilot cases and will be included in the final deliverables D2.4, D2.5, and D2.6.

References

[1] Deliverable 2.1. [Online]. Available: https://digihealth-asia.eu/wp-content/uploads/2022/07/Deliverable-D2.1.pdf.