EC544 Networking the Physical World
2022 Spring

Project Proposal —
A Data Encrypted Neck Healthcare System Based on FRDM Sensor

Team05
Xiaopeng Huang (xphuang@bu.edu)
Tianyi Xu (tyx@bu.edu)

Introduction:

Nowadays people are facing all kinds of circumstances where they need to be sitting for a long time, either for studying or working in the office. As time passes, this would easily lead to some physical illness unconsciously. Therefore, in this class, we would like to develop a healthcare embedded system aiming to solve this problem based on some hardwares we have. FRDM development board becomes a good choice for its small size and powerful characteristics.
The FRDM-K64F-AGM01 is a 9-axis sensor toolbox Demonstration Kit including the FRDM-K64F MCU board and the FRDM-STBC-AGM01 Sensor Shield. “AGM” stands for Accelerometer, Magnetometer and Gyroscope, which serves as a stabilized compass that can sense the movements (rotation, acceleration, etc.) of the board, and generate an accurate orientation estimate.Inspired by this powerful sensor system, we want to develop the FRDM board into a wearable healthcare device to track people’s body position, for example, checks how long one has been sitting or not moving his neck. In the end, information will be sent to another terminal device – Raspberry Pi 3/4, representing the user, for visualization.
We regard these output data as sensitive personal information. Meanwhile, gyroscope and accelerometer information is streaming in realtime at high frequency. Therefore,

before the data is being sent to the user, it needs to be encrypted and safely stored inside the Raspberry Pi. Finally, Raspberry Pi 3/4 should be able to retrieve the stored ciphertexts and decode them into a readable result.
As thus we can divide the whole project into several main tasks:

a. Plan the movements. Use FRDM-K64F-AGM01 sensors to detect movements and generate data with Freedom Sensor Fusion Toolbox.
b. Encrypt the data and store the ciphertexts.
c. Fetch the ciphertexts from the distributed storage and decrypt them to readable messages to the user.

Here we can foresee some potential problems that might exist in all the tasks. Certain algorithms need to be implemented to convert gyroscope data to useful information that reflects people's position. We also need to utilize an efficient and safe encryption and decryption method. In this process, a feasible distributed storage is also required. See sections **Risk Management** and **Technical Approach** for details.

Resources:

Hardware:

- FRDM-K64F board
- FRDM-K64F-AGM01 Sensor Shield (Including Gyroscope / Accelerometer / Magnetometer)
- Raspberry PI 3 / 4
- Cat-6 Ethernet Patch Internet Cable
- USB cables
- Routers / Switch
- Micro SD cards

Software:

- Freedom Sensor Fusion Toolbox GUI
- Gyro Stabilized Compass Algorithm
- Intelligent Sensing Framework
- Sensor Fusion Library
- Arm Mbed OS
- Raspberry Pi OS
- (AWS S3 / EC2 hosts, as a backup plan for shortage of Raspberry Pi supply)

Risk Management:

- As the sensors we selected are recording and transmitting data in a streaming manner, encrypting it on the air would be a computational challenge. We should evaluate the bottleneck of this streaming encryption and find the balance between accuracy (the amount of data we would execute in a unit time) and the computational power available on the board.
- As the FRDM-K64F board is responsible for both collecting data from sensors and encrypting data on-board, short-term on-board memory space is another foreseeable limitation. A well-balanced pipe-line should be considered in the design.
- Moreover, since these data would be transmitted to several different storage tanks for long-term storage, one strategy should be considered to safely slicing or splitting data into batches.
- Packages may be lost in the transmission phase. Tolerance of these lost packages should be accounted for in the design of the distribution storage system, for both data integrity and decryptibility.
- Since we are facing a supply-chain shortage, it's risky to rely on the existence of Raspberry Pi in our project, though using Raspberry Pi means we would use local network only, have more control over the switch and router, and no need to config complicated dynamic routing and tunneling. For the worst case scenario, we would transmit to and retrieve from multiple AWS storage hosts, instead of using local storage buckets on the local network.

Technical Approach:

This project is divided into the following three modules for development.

1. Sensor and Positioning Algorithm
2. Sensor Data Encryption and Decryption Algorithm
3. Distributed Storage System

Sensor and Positioning Algorithm:

This module can be further divided into two submodules: Sensing data collection and positioning algorithm. We need to utilize the sensor to detect movements and generate data. Then the generated data should be performed as the input to our algorithm and information could be output to reflect the position of the human body.

For sensing and data collection, FRDM-K64F-AGM01 provides a powerful GUI called Freedom Sensor Fusion Toolbox. Basically we need to set up the sensing environment inside the development board and select a sensor algorithm. To set up, we need to use the USB cable to connect the development board to the PC. Then the FRDM-STBC-AGM01 sensor shield needs to be plugged onto the development board and device driver needs to be installed. There are several default algorithms aiming at different applications, e.g. 2D Automotive Compass, Rotation, Gaming Handset and so on. Here we want to implement Gyro Stabilized Compass, which makes use of Accelerometer, Magnetometer and Gyroscope to provide an accurate orientation estimate even in the presence of high linear acceleration and the presence of an external magnetic disturbance. After the environment is set up, we want to fix the development board on the user's neck and start collecting data. During the sensing, data will be collected and displayed from the three gyroscope axes measurements in units of deg/s.

Now we have a real time streaming data shown on the GUI, we need to further convert it to intuitional information that reflect the people's positioning, and hopefully, gives a feedback when people are keeping still with an unhealthy posture, e.g. a very small angle between the neck and horizontal axis, for a long time. The positioning algorithm should take the real time data as input and manage the data allocation inside the development board. The development board may have a limited memory for this real time, so a quick and valid data allocation like stack needs to be implemented. It will also have a power loop which can be halted as needed to check the angle all the time. In the positioning algorithm, we need to have an adjustable threshold of angle. When the angle between the user's neck and the horizontal axis is less than the threshold, the beeper on the development will be turned on as a warning. Only if the user changes its position and recovers to his normal sitting position, the beeper can be turned off. To realize this warning function, we need to check the development board manual to control the peripheral GPIOs.

Sensor Data Encryption and Decryption Algorithm:

As we stated in the introduction, body positioning data should be regarded as sensitive data, therefore an encryption algorithm is required. The design matrix of this encryption algorithm should at least contain the following aspects,computational complexity, security, and consistency.

For computational complexity, it contains two main aspects. One is the computational complexity of encrypting the raw data into multiple pieces of ciphertext, and the other would be retrieving these ciphertext from a well-defined distributed storage system and decode the raw data out from them .We would compare the properties of different

approaches for storage encoding and decoding methods, including Reed-Solomon code and Fountain Codes. These are well-known algorithms for the distributed storage systems, which corresponds to the module we would discuss in detail in the following section.
As a pipelined design, the sensing data would be captured and temporarily stored in the memory and feed both the body position analysis module and this real-time encryption module. To obtain a practical performance for the encryption module, on-board acceleration hardware components should be considered as the key part of encryption code design. As stated in the datasheet, hardware encryption supports DES, 3DES, AES, MD5, SHA-1 and SHA-256 algorithms. A deep investigation over these accelerations modules should be done, in order to expand the full potential of this development board and therefore have the best output rate of the encryption process. Resource restrictions limit the size of the batch we could choose for one round of encryption. As stated in the data sheet, on-board memory space is 256 KB. Encryption module would consume data batches stored by the sensor and generate new encrypted dataframes. Moreover, the sensor data is in a streaming manner while the encryption module is in a batched manner, meaning that the sensor would continuously send data while the encryption module is processing the previously stored data. Two round-robin memory structures should be designed for sensor data storage in the memory, and the time of encryption should match the time one of these round-robin memory structures have been fully consumed by the incoming sensor data.
On the other hand, the storage mechanism of the encrypted data should also be designed. As an I/O process, we should not assume that the transmission of one encrypted ciphertext would consume a relatively stable amount of time. Hence encrypted ciphertext should not be stored in the memory once the encryption process is complete. From another perspective, if the transmission task follows the encryption task in a procedural fashion, the uncompleted transmission would block the encryption task, which leads to an in-consistence of the encryption of raw data.

Distributed Storage System

One distributed storage system should be defined as a storage system which contains multiple storage nodes or storage tanks for data storage. We introduce this concept to the project to illustrate the ability of one distributed storage algorithm to partially store data and the property of partial-inconsistency, in another word, only the whole collection of encrypted data could be used to decrypt the raw data. We are planning to design a linux-based file encoding and decoding system and the distributing system. Files in this system would contain the encrypted data from the encryption module running on the Fremdom board. File names should be defined in an encrypted way to increase the security level by reducing the capability of gathering different batches together. Decryption process would find files from different nodes of this distributed system and gather files belonging to the same batch to run the decryption algorithm, which returns the raw data.

Responsibilities:

Xiaopeng Huang will mainly focus on the Sensor and Positioning Algorithm.
Tianyi Xu will mainly focus on the Sensor Data Encryption and Decryption Algorithm.
The last part, Distributed Storage System would be a joined module.

Milestones:

Week 1: FRDM sensing data collection

The FRDM-K64F-AGM01 sensor environment is expected to be set up and successfully collect the data shown on the GUI. Implement the sensor with several user tests in physical tests and examine the results.

Week 2: User position algorithm based on sensing data

Develop the algorithm to convert the real time data to physical information that reflects the user's sitting position. Study the FRDM peripherals and be able to control some of them, e.g. beeper, to realize a feedback function.

Week 3: Encrypt and Decrypt Algorithm

The encrypt process would consume one small batch of data (e.g. 32KB) as input and generate multiple (e.g. 3) encrypted data segments as output. These outputs would be stored as a binary file or pass to the transmission module for data distribution and be sent to different data nodes.

Week 4: File storage strategy and transmission

This contains three parts, the encrypted binary file storage mechanism, the transmission between Freedom board and storage nodes, and the file storage mechanism on the storage nodes. Tests could be done on a linux based system or AWS.

Week 5: On-board data fetch and encryption

Here we should finish both sensor algorithms and encryption algorithms respectively, it’s time to combine these two modules together and optimize the on-board performance.

Week 6: Raspberry Pi / AWS storage and connection setup

If we have Raspberry Pi in hands, configure the local network and set up tunneling for communication between Freedom board and Raspberry Pi.

If we don’t have Raspberry Pi by then, setup AWS EC2 or S3 as the linux environment to store the data. Set up the secure connection between Freedom board and AWS hosts. Then run scripts to retrieve data from these hosts to prove the consistency of the packages.

Week 7: Optimize and integrate.

We left one buffer week to better integrate different modules, and system-level debugging, since multiple modules are designed and implemented in this project.