CSSE4011 Prac 2
Seismic Activity Monitoring Network

1 Assessment

- Git due 8am Friday in week 6 .
- Due in your lab session in week 6 .
- Course Marks: 15%
- Electronic Course Profile Pass Hurdle: Optional submission.

2 Resources

- NRF52840 USB Dongle
- ESP32-C3
- B-L475e-IOT01A Platform
- Ultrasonic Ranger

3 Introduction

This practical will involve expand your previous practical by creating a Seismic Activity Monitoring Network (SAMN) that is used to monitor seismic activity with motion sensors and an ultrasonic sensor, over a Bluetooth connection. This practical introduces you to Bluetooth wireless communications, sensor interfacing and sensor data display. This practical also introduces you to the Zephyr RTOS Bluetooth Low Energy Stack Communications library. The goal is to understand the basics of Bluetooth and to gain familiarity with using advanced Zephyr RTOS libraries. Figure 1 shows the setup.

3.1 Using Zephyr RTOS with BLE

To use Zephyr RTOS 's BLE libraries, you must revert to the Zephyr RTOS git version 3.1. Follow the tutorial link. If you do not revert to git version 3.1, then BLE will not work on the platforms.

BLE Examples from Zephyr RTOS : `samples/bluetooth/central`

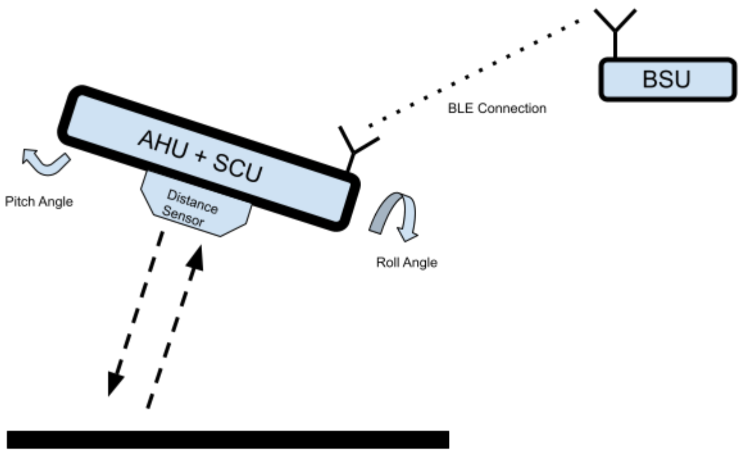

Figure 1: Overview of Base Station Unit (BSU, Application Host Unit (AHU) and Sensor Control Unit (SCU))

4 PART 1 Tasks - Base Station (BSU) BLE Wireless Communications

The Zephyr RTOS provides libraries for BLE Wireless data connections. The BSU and AHU are capable of Bluetooth Low Energy (BLE) communications and is supported by Zephyr RTOS . In this practical, you will look at developing libraries for BLE scanning and Data transfer.

4.0.1 Repository Setup - BSU

```
csse4011_repo/
├── p2/
│   ├── bsu/
│   │   ├── prj.conf
│   │   ├── CMakeLists.txt
│   │   └── src/
│   │       └── main.c
│   └── dv/
└── oslib/
    └── bsu_drivers/
        └── bsu_hci/
            ├── bsu_hci.c
            └── bsu_hci.h
```

You repo structure should be setup as shown in the directory tree above. Where inside the repo there will be a *p2 /* directory that contains p2 specific application code for the bsu, and a *oslib/* directory that contains any library code. Any code in *oslib/* should be treated as a module that you can easily add to another application and not tightly coupled to p2 bsu code. That is, they should compile without depending on other oslib modules.

Note: the file/directory names used above are for demonstration purposes, you may name them appropriately as required. You may also add additional files/folders where required. For the README.md: you must have a HCI section to document the hardware signals and firmware protocol used.

4.1 Design Tasks

Design tasks must be submitted via git in week 6 . You MUST have a `README.md` file that shows the following

- Title
- First Name, Second Name and Student ID
- Brief Description of functionality achieved for each design task.
- Folder structure
- List any references used
- Any instructions regarding your source
- Any user instructions

The `README.md` file must be written in markdown format.

4.1.1 Design Task 1A: Base Station Unit (BSU)

Create a Base Station Unit (BSU) using the NRF52840 USB Dongle . The BSU should be able to send/receive BLE Advertising packets. The NRF52840 USB Dongle should be able to filter and only receive from the AHU+SCU. The output should be viewed in a serial terminal and should have a JSON format.

Use the Zephyr RTOS BLE library. You should include a Shell command to request a single sensor measurement from an arbitrary sensor (specified by the Device ID - DID from the previous prac HCI specification), on the SCU. You must be able to return any of the DID values from the previous prac.

Table 1: Extra Device IDs (See previous prac HCI specification

Device ID (DID)	Device
12	Ultrasonic (raw value)
13	Orientation (Pitch, Roll)
14	Altitude (m)

```
//Example Commands

ble g <DID ID>
// 2 example commands
// Get the latest accelerometer X reading
// Get the latest ultrasonic ranging reading
ble g 3
ble g 12
```

```
//Example Received Output
{<system time (in seconds) received by base>, <DID>: [VALUE 1, VALUE 2, VALUE3]}
```

4.1.2 Design Task 1B: BSU BLE Scanning

Use the Zephyr RTOS BLE library, to scan for the AHU+SCU platform. You should include the provided Zephyr RTOS BLE Library shell commands to start/stop BLE scanning.

4.1.3 Design Task 1C - Continuous Sampling and Data Viewer (DV)

Add a shell command to start/stop continuous sampling of orientation and altitude data from the AHU+SCU. Table 2 show the sampling parameters. The received output must follow the previous specified output JSON format.

Table 2: Sampling Parameters

	Period (s)
Minimum Sample Period	2
Maximum Sample Period	30

```
// Continuous sample start and stop commands of orientation and altitude
// Start continuous sampling (every 5s)
ble c s <sample time>
ble c s 5
// Stop sampling
ble c p
```

Create a data viewer using a suitable framwork - e.g. python with the matplotlib and the pyserial library. Your script should accept the JSON output from the BSU. The BSU should receive orientation and altitude data from the AHU+SCU. You must display the orientation and altitude (fine and course), in the correct units and as a time series graph. All upload data values must be timestamped and this should be viewed on a time series graph.

Git Submission

You are required to submit all code created (e.g. python scripts). Create a `dv` folder in your `p2` folder, for any python code files. Any references should be listed in the README.md.

4.1.4 Flow Charts and Message Passing Diagram

Draw flow charts of the BSU's BLE connection handling and BLE scanning. Also provide a messgae passing diagram of the BSU's BLE request/receive interaction with the AHU.

5 PART 2 Tasks - Measuring Altitude and Orientation

Use the sensors on the SCU (via the HCI) to measure the altitude and orientation prose of the SCU/AHU, as you hold it above the ground.

5.1 Repository Setup

You should also maintain your `p2` folder for AHU+SCU, in your git repo.

```
csse4011_repo/
 └── p2/
 │   ├── ahu/
 │   │   ├── prj.conf
 │   │   ├── CMakeLists.txt
 │   │   └── src/
 │   │       └── main.c
 │   └── scu/
 │       ├── prj.conf
 │       ├── CMakeLists.txt
 │       └── src/
 │           └── main.c
 └── oslib/
     ├── ahu_drivers/
     │   └── ahu_hci/
     │       ├── ahu_hci.c
     │       └── ahu_hci.h
     └── scu_drivers/
         └── scu_hci/
             ├── scu_hci.c
             └── scu_hci.h
```

You repo structure should be setup as shown in the directory tree above. Where inside the repo there will be a *p2 /* directory that contains p2 specific application code for the ahu, and a *oslib/* directory that contains any library code. Any code in *oslib/* should be treated as a module that you can easily add to another application and not tightly coupled to p2 ahu/scu code. That is, they should compile without depending on other oslib modules.

Note: the file/directory names used above are for demonstration purposes, you may name them appropriately as required. You may also add additional files/folders where required. For the README.md: you must have a HCI section to document the hardware signals and firmware protocol used.

5.2 Design Tasks

Design tasks must be submitted via git in week 6 . You MUST have a `README.md` file that shows the following

- Title
- First Name, Second Name and Student ID
- Brief Description of functionality achieved for each design task.
- Folder structure
- List any references used
- Any instructions regarding your source
- Any user instructions

The `README.md` file must be written in markdown format.

5.2.1 Part 2A AHU BLE Connection

The AHU must be able to receive requests from the BSU and then transmit the required DID value. Use the Zephyr RTOS BLE Library.

5.2.2 Part 2B AHU Orientation Calculation

You must process sensor readings on the AHU, to the orientation prose angles. You must use the LSM6DSL to compute the orientation prose of the SCU+AHU. This should consist of the pitch and roll angles. You do not need to calculate the yaw angle but you may do so with the LIS3MDL heading value. You must use the correct Device ID (DID) values.

5.2.3 Part 2C AHU Altitude Calculation

You must process sensor readings on the AHU, to calculate the altitude height (in m) of the AHU+SCU. You must use the following sensors to measure altitude:

- LPS22HB Barometer
- HCSR04 Ultrasonic Ranger (connected to the SCU)

The ultrasonic ranger must be fixed and connected to the SCU and must move with the same orientation of the SCU. The ultrasonic ranger must face downwards to a surface. You must be able to compensate the altitude for changes in orientation (e.g. pitching up/down or rolling clockwise/anticlockwise). You must use the correct DID ID values.

The altitude should be measured up to a minimum distance of 0.1m to a max distance of 0.8m. You must show two measurements:

- Fine: 0.1m to 0.8m in 1mm resolution. Note accuracy can be upto 25mm.
- Course: Elevation (in meters) above sea level.

5.2.4 Part 2D AHU Orientation Level Indicator with RGB

Use the AHU's onboard RGB LED (SK68) to indicate the level of the AHU+SCU (See table 3). This indications should be continuous and should update every 2s.

Table 3: Level Indication

Angle	Colour
$0\circ$ (level)	Green
$> 45\circ$ (up)	Blue
$< -45\circ$ (down)	Red

Refer to SK688 Bit manipulation example.

5.2.5 AHU+SCU Flow Charts and Hardware Schematic

Draw flow charts of the AHU's BLE connection handling and the SCU's altitude and orientation calculation. Also provide a hardware schematic of the AHU+SCU configuration.

6 Academic Integrity

While the assessment is done in a group, there are specified individual assessment items. You should feel free to discuss aspects of C programming and assessment specifications with fellow students, and discuss the related APIs in general terms. You should not actively help (or seek help from) other students with the actual coding of your assessment, other than normal code sharing with your group partner It is cheating to look at another group student's code and it is cheating to allow your code to be seen or shared in printed or electronic form, with other groups. You should note that all submitted code will be subject to automated checks for plagiarism and collusion. If we detect plagiarism. or collusion (outside of the base code given to everyone), formal misconduct proceedings will be initiated against you. If you're having trouble, seek help from a member of the teaching staff. Don't be tempted to copy another student's code. You should read and understand the statements on student misconduct in the course profile and on the school web-site: https://www.itee.uq.edu.au/itee-student-misconduct-including-plagiarism

7 Criterion

The prac demonstrations are marked according to the criterion outlined in the table below. You must pass the pre-demo checks before your submission is marked. **All code assessed for the prac must be your own work.**

7.0.1 Pre Marking Checks

The following criteria **must** be met **before** you are allowed to demo.

Check	**P/F**
Your latest prac code **must** be in git.	
Your git repository must be up to date.	
Your prac code must build without errors.	
Your README.md has been included.	

Failure to meet pre-demo checks will mean that your prac will not be marked.

7.0.2 Prac Criteria

Design Task 1A: BSU	
0	BLE connectivity is not correctly implemented with Zephyr RTOS and does not function.
1	BLE connectivity is not implemented correctly with Zephyr RTOS and only partially functions.
2	BLE connectivity is correctly implemented with Zephyr RTOS and only partially functions with Shell commands.
3	BLE connectivity is correctly implemented with Zephyr RTOS and fully functions with Shell commands.
Design Task 1B: BLE Scanning	
0	BLE Scanning is not correctly implemented with Zephyr RTOS and does not function.
1	BLE Scanning is not implemented correctly with Zephyr RTOS but the shell command only partially functions.
2	BLE Scanning is correctly implemented with Zephyr RTOS but only partially functions with the shell command.
3	BLE Scanning is correctly implemented with Zephyr RTOS fully functions with the Shell command.
Design Task 1C: Continuous Sampling and Data Viewer	
0	No data displayed or viewed on an online data viewer.
1	Some data or output can be viewed on the online data viewer dashboard but it is not realtime or shows erroneous values.
2	Data viewer fully shows the data received from the BSU.
BSU Flowcharts and Message Passing Diagram	
0	No flowcharts or message passing diagram is provided.
1	flowcharts and message passing diagram are provided but there are some errors in the either .
2	flowcharts and message passing diagram are correct.

Design Task 2A: Orientation Calculation	
0	No orientation calculation is done.
1	Orientation calculation is implemented in myoslib correctly and but errors occurs.
2	Orientation calculation is correctly implemented and transfers the data to the PC but errors may occur in angle resolution $> 10\circ$.
3	Orientation calculation is correctly implemented and transfers the data to the PC, with acceptable angle resolution $<= 5\circ$.
Design Task 2B: Altitude Calculation	
0	No altitude calculation is done.
1	Altitude is implemented but large errors occurs when calculating the measurements.
2	Altitude calculation is correctly implemented and transfers the data to the PC but errors may occur in distance resolution for either fine ($> 50mm$ or course measurements.
3	Altitude calculation is correctly implemented and transfers the data to the PC with acceptable distance resolution for either fine ($< 50mm$ or course measurements
Design Task 2C: Orientation Level Indicator	
0	Not implemented or does not work.
1	Has a working implementation but colours indicators are incorrect for the angle orientation of the AHU+SCU.
2	Fully implemented and works without errors.
AHU+SCU Flowcharts and Hardware Schematic	
0	No flowcharts or Hardware Schematic is provided.
1	Flowcharts and Hardware Schematic are provided but there are some errors in the either.
2	Flowcharts and Hardware Schematic are correct.

Code Style	
0	One or more style errors found in one tutor selected file.
1	No style errors found in **one** tutor selected file.
Code Organisation	
0	Required repo organisation is not followed.
1	Required repo organisation is followed..
Late Penalty	
	-10% per day up to 7 calendar days.