Assignment 3 Coap Server and Distance Sensing (100 points)

IoT devices are often small, remotely deployed and infrequently serviced. These devices are also network-constrained and limited in both computing resources and power consumption. As a consequence, to establish communication with IoT devices, we should only consider the protocols that address IoT data connectivity as well as the constrained environment.

In this assignment, you are requested to implement several components/applications to demonstrate the operations and communication of an IoT device in Zephyr RTOS (version 2.6.0) running on MIMXRT1050-EVKB board. Your task includes:

1. Port an existing HC-SR04 driver to Zephyr version 2.6 for MIMXRT1050_EVK. The driver can be found in https://github.com/inductivekickback/hc-sr04. Modifications should be done such that it can report the distance measure in inches (with 2 decimal digits) and consist of a Kalman filter to reduce any measurement noise of HC-SR04 sensors.
2. Develop a Zephyr application that serves as a COAP server using Zephyr's network stack and COAP library. The server should enable the following resources:
    a. /sensor/hcsr_i where i=0 and 1 for two HC-SR04 sensors. A get method request retrieves the current distance measure. The sensors are COAP observing resources and a notification should be sent to all observers whenever there is a change of distance measurement larger than 0.5 inches.
    b. /sensor/period. The interface is used to define the sampling period of HC-SR04 sensor measurement which is set by a put request method.
    c. /led/led_n where n= r, g, and b for a RGD led. A put method request sets the led to 0 (OFF) or 1 (ON) and a get method request retrieves the current led status.
    d. /.well-known/core interface for resource discovery.

Your MIMXRT1050_EVK board can be connected to an Ethernet-based LAN or directly connected to a computer with an Ethernet cable. If there is a DHCP server on the LAN, you can enable dhcp service to acquire a dynamic ipv4 address, as shown in the Zephyr's sample application /samples/net/dhcpv4_client. Otherwise, a static address can be set for the board. For peripheral connections, please use the following Arduino pins:

(1) Trigger and echo pins of hcsr_0: D3 and D6.
(2) Trigger and echo pins of hcsr_1: D8 and D9.
(3) Led RGB pins: D2, D5, and D12.

In the current Zepyr source tree, there is a COAP server sample application /samples/net/sockets/coap_server/src/coap-server.c. You may want to study its implementation carefully or re-use the code partially in your development. To test your COAP server, we will use a Chrome extension, Copper for Chrome (Cu4Cr) CoAP user-agent, to browse and interact with Internet of Things devices via COAP protocol. The user-agent and the installation procedure can be found in https://github.com/mkovatsc/Copper4Cr.

Note that the Zephyr COAP library is based on several IETF RFCs. However, not all parts of these RFCs are supported. For this assignment, simple method invocations and resource discovery can be done using the library. Also, during my testing of dhcp on MIMXRT1050_EVK, the following configuration is needed:

CONFIG_NET_UDP_MISSING_CHECKSUM=y

Due Date

The due date is 11:59pm, April 6.

What to Turn in for Grading

- Your submission is a zip archive, named RTES-LastName-FirstInitial_03.zip, that includes
  - An application folder, named *project_3*, to include all your implementation of the assignment. The folder should contain *CMakeLists.txt, prj.conf, readme.txt*, and a source file directory *src,* a board device tree overlay directory "*boards*".
  - A patch file to contain all changes to Zephyr v2.6 for the HC-SR04 driver. Please use the following diff command to generate the patch (assuming the only change in the Zephyr source is the HC-SR04 driver except any on-tree sample applications):

    ```
    diff –rauN --exclude=samples /(original zephyr tree) /(modified zephyr tree) >
            patch_project3
    ```

  - The readme text file describes all the commands you use to build and run your program.
- Note that any object code or temporary files should not be included in the submission. Submit the zip archive to the course Canvas by the due date and time.
- Please make sure that you comment the source files properly and the readme file includes a description about how to make and use your software. Don't forget to add your name and ASU id in the readme file.
- There will be 20 points penalty per day if the submission is late. Note that submissions are time stamped by Canvas. If you have multiple submissions, only the newest one will be graded. If needed, you can send an email to the instructor to drop a submission.
- The assignment must be done individually. No collaboration is allowed, except the open discussion in the forum on Canvas. The instructor reserves the right to ask any student to explain the work and adjust the grade accordingly.
- Failure to follow these instructions may cause deduction of points.
- Here are few general rule for deductions:
  - Cannot compile or compilation error -- 0 point for the assignment.
  - Must have "–Wall" flag for compilation -- 5-point deduction for each warning.
  - 10-point deduction if no compilation or execution instruction in README file.
  - Source programs are not commented properly -- 10-20-point deduction.
- ASU Academic Integrity Policy (http://provost.asu.edu/academicintegrity), and FSE Honor Code (http://engineering.asu.edu/integrity) are strictly enforced and followed