Assignment 1 Thread Programming and Device driver in Zephyr RTOS (100 points)

In this assignment, you are required to develop a HC-SR04 sensor driver in Zephyr RTOS 1.14.0 running on Galileo Gen 2. The driver module should be placed in /Zephyr/drivers/sensor/hc-sr04 directory. When the driver is initialized, it should create two instances of HC-SR04 sensors. Here are few requirements:

1. Configuration options and CMake file should be added for kernel build and should be properly defined for the Zephyr kernel on Galileo board. For instance, you can add configuration parameters in Kconfig file for Galileo board (/Zephyr/boards/X86/galileo/Kconfig). Example configuration options include the names of the sensors (HCSR0 and HCSR1), the gpio pins to be used as echo and trigger pins.
2. The device works only in one-shot mode. The sensor driver api should consists of the following functions:
   - *sensor_sample_fetch:* the sample fetch call triggers a new measurement if there is no on-going measurement operation. The measured distance should be saved in a per-device buffer for the following read. The buffer of 32-bit integer can accommodate one distance measure and should be cleared upon a *sample fetch* call.
   - *sensor_channel_get*: a distance measure (an integer in centimeter) saved in the per-device buffer is returned. If the buffer is empty, the channel get call is blocked until a new measure arrives. Note that, if there is no on-going sample fetch operation, the device should be triggered to collect a new measure. After each channel get operation, the buffer is cleared.
   - *sensor_attr_set*: A timeout duration parameter (in microsecond) should be provided in each HCSR sensor. A blocked channel get call should return -1 when the timeout duration is expired.

Once you have the driver for HC-SR04, you will develop an application that takes the distance measure and records the measure plus timestamp in an internal buffer. Your implementation should be added to /Zephyr/samples/HCSR_app. The application should use a shell module to accept commands from console. The commands include:

1. Select_HCSR *n* (n=0, 1, or 2, to enable none, HCSR0, or HCSR1)
2. Start_HCSR *p* (to start collecting *p* distance measures from the selected HCSR sensor, where $p \le 256$. The interval between consecutive measures is set to 0.5 seconds.)
3. Dump_HCSR *p1 p2* ($p1 \le p2$, to dump (print out) the distance measures in centimeter and timestamps recorded in previous measurement on console)

Your application and driver should avoid any busy waiting. Hence, you should use interrupt-driven approach for the echo signal of HCSR devices, and thread sleeping function. Also, the timestamp for each distance measure represents the elapse time since the beginning of the measurement. You can use x86 TSC to compute the elapse time in microsecond.

In the assignment, you have a chance to work on Zephyr gpio device driver to develop the driver for HC-SR04. The gpio driver is based on several Zephyr components, such as gpio_dw, gpio_pcal9535a, i2c_dw, pci, and interrupt. It will be a good exercise to examine and study how these components are designed and implemented. For the assignment, you are asked to compile a report to detail what you learned from studying these components and your implementations. The report should be formatted to single-space, 11-point font, and be limited to 8 pages. Please note that you need to do this part of the assignment individually. No team work will be accepted.

Due Date

TBD

What to Turn in for Grading

- Create a working directory, named “cse522-assgn01-LastName_FirstInitial”, for the assignment to include
  - A pdf report to include a description of your implementation. Don’t forget to add your name and ASU id in the report.
  - A patch to include all your changes in the Zephyr 1.14.0 commit for the implementation of HCSR-04 sensor driver and a sample application.
  - A readme text file with all the commands you use. Alternatively, you could have a script file with comments inside on how to use it.
- Compress the directory into a zip archive file named cse522-assgn01-LastName_FirstInitial.zip. 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.
- 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 and TA to drop a submission.
- The assignment must be done individually. No collaboration is allowed, except the open discussion in the forum on Canvas.
- Here are few general rule for deductions:
  - Compilation error -- 0 point for the part of 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.