A11 Assignment – ESE516-SPRING 2021

DUE DATE: Saturday April 27 2022 before 11:59pm EST (By almost midnight). To be submitted on Google Drive of Team Folder on a folder called A11. GROUP PROJECT

Remember: Please submit your complete Atmel Project on Google Drive. The project must be complete (must have everything so we can compile it!).

1. Drivers for a sensor

Continuing with developing our system, for this assignment, please develop a driver for one of the sensors of your system. We recommend taking the time to tackle the sensor that you believe is the most critical for your particular project to work. If your sensors are simple enough, please consider tackling at least two – that will reduce the work you will do later on!

Some weeks ago we ordered the external sensors of your system. You can develop the driver for one of these sensors if you don't have the board to work with.

Please make the driver with a *C and *H file as seen in class. You can make it yourself, or port a driver from the internet (if porting an existing driver, please make sure to copy any license comments it needs for its use. Please ensure the license allows for your use of the driver).

You can use the project from A11 to test your driver. This is also the base project for the rest of the class! The Annex will explain the steps to use this firmware on your project.

Please provide a video of you explaining the driver and showcasing that it works. You may use CLI commands to show the sensor functioning.

A11 RUBRIC

Points	Rubric
-200 points	If the project is incomplete or has missing files, it will incur this penalty.
200 points	A video is submitted that explains how your driver works, and SHOWS the sensor working with your driver.

Annex: A11 project

The A11 project can be found on the Google Drive on the A11 assignment folder. This project will be a good step to work on your final project!

Steps:

Note: For ALL your MQTT Topics, please add the string "_Tx" to your topics, where "x" is your team number. The idea is to have unique topics so we don't have crosstalk amongst teams!

You can find your team number here: https://docs.google.com/spreadsheets/d/1zJV80lDCvYRMDdVxJOanEcSNMR1fx5_j3uOuxio_GzY/edit?usp=sharing

- Change the bootloader project on the A11 code for your bootloader code of A9.
- Add your code to make OTAUs work (CLI code "fw") from A10.
- The code of A11 already has the I2C driver, so no need to add it.
- If you are using the accelerometer:
  - Add the accelerometer code from A8
    - `platform_write`
    - `platform_read`
- If you are using the Neotrellis:
  - Add the Neotrellis code from A8
    - `CLI_NeotrellisSetLed`
    -
- If you are using the Sparkfun Quick OLED:
  - Uncomment lines 74-77 from UiHandlerThread.c. This will start the LCD screen. If correctly wired, you should see a circle, a line, and the text ESE516.
- If you are using a distance sensor (US-100)
  - DistanceSensor.c is added for you as a starter, you can check it out

The starter project does the following:

- Topics are defined on "WifiHandler.c"
- When the button SW0 is pressed, it sends a message to the "`TEMPERATURE_TOPIC`"
  - See function "`extint_detection_callback`" on `WifiHandler.c`
- It will send data of the IMU when the CLI command "imu" is typed
  - See `CLI_GetImuData` on "CliThread.c"
- It subscribes to a Game, IMU and LED topic
  - See `mqtt_callback()` on "WifiHandler.c", case `MQTT_CALLBACK_CONNECTED`
    - `mqtt_subscribe` subscribes to a topic

        - The functions on the third parameter will be called when the MCU gets a message on that topic – check those functions to see how an MQTT message can be parsed!
            - `SubscribeHandlerGameTopic`
            - `SubscribeHandlerLedTopic`
            - `SubscribeHandlerImuTopic`

- The device can send data on the topics it subscribed to
    - The system defines RTOS queues on WifiHandler.c
    - It also provides APIs for different parts of the system to add data to a queue
    - Check functions "`WifiAddGameDataToQueue`", "`WifiAddImuDataToQueue`" for example. Other parts of the program can add data to the queue, and then the WifiHandler thread will read from the queues and process them (send the data) if there is any data available.
        - See function `MQTT_HandleTransactions.` This function will call a function for each topic that we are sending data to, checking if anything has been added to the queue and if so, it sends data.
        - This function call `MQTT_HandleImuMessages` for example. This will get the data from the queue and send the message.

- The program has a good example on how to use queues to transmit information. The following is the program flow on how a "game" data packet is transmitted from the MQTT broker to the device.
    - MQTT broker send a "game" packet. A game packet is defined as a JSON formatted string, with only one parameter (game), and an array of integers (0 to 15), one detailing a button press for each current move. The array can be up to 20 moves in length
        - You can simulate this pressing the "Initialize Game" button, which sends a game packet.
    - MCU is subscribed to the `GAME_TOPIC_IN` topic
    - The topic is received and processed in the function `SubscribeHandlerGameTopic.` If the data packet is correct, it adds the data to the `xQueueGameBufferIn` queue using the API function `ControlAddGameData`
    - On "ControlThread.c". the control thread is in the `CONTROL_WAIT_FOR_GAME` state. It is waiting for a game packet in the queue
    - Once it receives the move, it orders the UI thread to show the move, and then it will wait for a signal from the UI thread that it has finished showing the moves and capturing the new move

        - Note: Currently, the UI thread (on UiThread.c) will just show button 0 and 15. It is up to the groups doing the Simon says game to change this to show the moves in the game packet received. The Control thread waits for this in the `CONTROL_PLAYING_MOVE` state

        - After showing the move, the UI thread waits for the user to input one button press. If you are doing the Simon says game, you have to modify this to input the number of moves + 1 to continue the game!

    - The control thread receives a new move from the UI thread and adds the new game packet to the `xQueueGameBuffer` queue using the `WifiAddGameDataToQueue API`
    - The WiFi thread will eventually check if there is anything to be sent on the `GAME_TOPIC_OUT topic.` It will see there is a new data packet and will send it to the cloud (function `MQTT_HandleGameMessages).`