The American University in Cairo
School of Sciences and Engineering

Embedded Systems
CSCE 4301- 02
Project 1 Report: Cooperative Scheduler

Submitted by:
Hadeel Mabrouk, 900163213
Eman Darwish, 900172070

Submitted to:
Dr. Mohamed Shalaan
Spring 2021

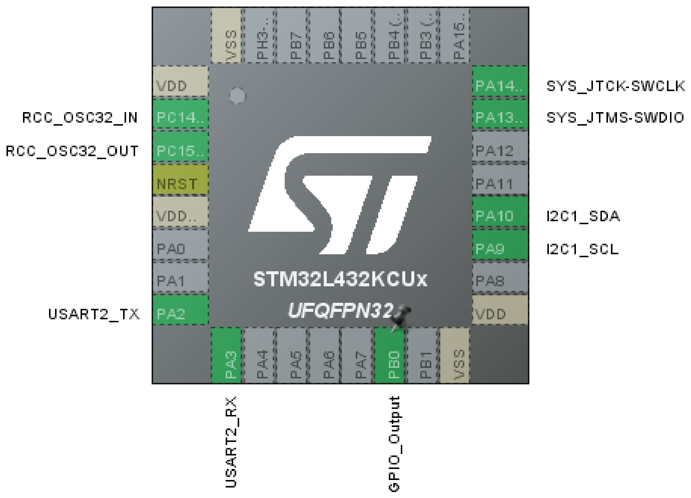
The purpose of this project is to develop a cooperative scheduler for embedded systems supporting 8 priority levels (lower values indicate higher priorities). The scheduler is implemented using linked lists with a complexity of O(N) for enqueuing and O(1) for dequeuing, where N is the number of scheduled tasks.

- **Cooperative Scheduler Implementation:**
  - **Ready Queue:** It is mainly a linked list of nodes, where each node is composed of a pointer of the ready task, and its priority. Enqueuing and Dequeuing to and from the ready queue can be done using QueTask() and Dispatch() respectively.
  - **Delayed Queue:** It is a linked list of nodes, where each node is composed of a pointer of the ready task, its original priority, and its delay. Enqueuing a task in this queue can be done using ReRunMe() passing the desired delay. The tasks are sorted in a descending order based on the delay value, which is decremented every 50 ms using SysTick Timer using DecreasePriorities() routine. Whenever the delay of the task at the head of the delayed queue is 0, it gets popped from the delayed queue and pushed to the ready queue again. Note that DecreasePriorities() routine gets executed inside the ISR of the SysTick_Handler.
  - Note: All the following applications and unit tests are implemented using STM32L432KCUx Nucleo board and are developed using Keil MDK-ARM V5 IDE and tested with the help of TeraTerm.
- **Unit Tests:**
  - **ReadyQueue Test:** A simple C program that tests the functionality of QueTask() and Dispatch() functions scheduling simple C functions.
  - **ReRunMe Test:** A simple application composed of one task that toggles User LED (LD3), and uses ReRunMe(10) to toggle every 500 ms.
    - How to Build and Run:
      - CubeMX Configurations:
        - From the available boards, select STM32L432KCUx.
        - In Pinout & Configuration tab:
          - System Core > RCC:
            - LSE: Crystal/Ceramic Resonator
          - System Core > SYS:
            - Debug: Serial Wire
            - Timebase Source: SysTick
          - Enable PB3 as GPIO_Output
      - After Code Generation:

            - In Core > Src:
                - Add the files in Unit Tests > ReRunMe Test.
                - Add the .c files in the Scheduler Directory.
            - In Core > Inc:
                - Add the .h files in the Scheduler Directory.
        - Build the project and load it on the Nucleo-32 Board.
        - Notice the flashing of the User LED (LD3) every 500 ms.
    - Hardware Components:
        - Nucleo-32 Board
        - USB-to-Micro USB Cable

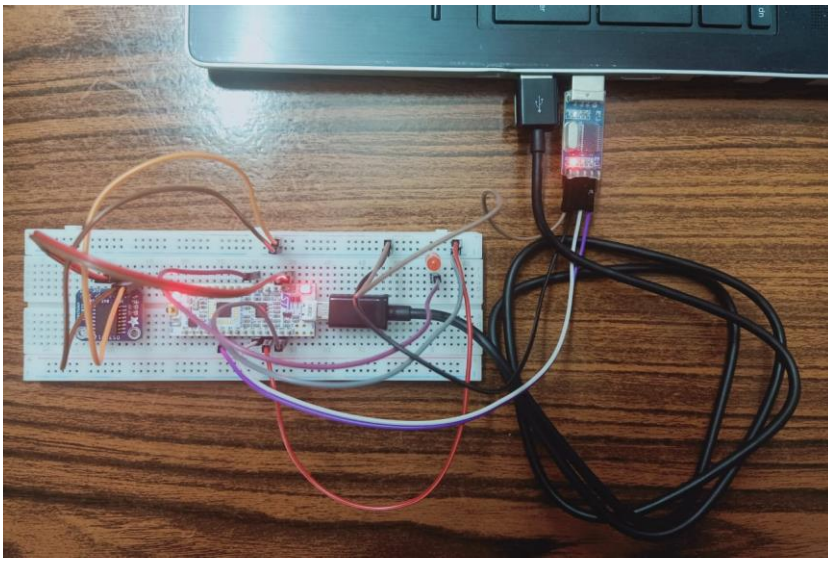
- **Demo Applications:**
    - **Ambient Temperature Monitor Application:** The purpose of this application is to Read the ambient temperature using DS3231 RTC (over I2C bus) sensor every 30 sec, and then produce an alarm through an external LED flashing when the temperature exceeds a threshold that is given by the user through TeraTerm terminal emulator using an asynchronous serial link (UART2) connected via a USB-to-TTL module.
        - Tasks:
            - Sensor Setting: to set the temperature integer and fractional portion register addresses, as well as the control register.
            - Read Threshold: to read the desired temperature threshold value from the user from using UART2.
            - Check Temperature: to periodically check the ambient temperature every 30 seconds and set the alarm flag if it exceeds the threshold value.
            - Toggle LED: to flash the external LED in case of an alarm.
        - Logic:
            - Sensor Setting, Read Threshold, and ToggleLED tasks are enqueued at the beginning of the program, with a priority of 1, 2, and 4 respectively.
            - Then, at the end of the Read Threshold task, Check Temperature gets enqueued with a priority of 3.
            - After that, the Check Temperature task reruns itself every 30 seconds, and the Toggle LED task reruns itself every 250 ms.
            - To generate the tick interrupts every 50 ms, we used the following command in the SystemClock_Config() function in main.c: HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/20);

- How to Build and Run:
  - CubeMX Configurations:
    - From the available boards, select STM32L432KCUx.
    - In Pinout & Configuration tab:
      - System Core > RCC:
        - LSE: Crystal/Ceramic Resonator
      - System Core > SYS:
        - Debug: Serial Wire
        - Timebase Source: SysTick
      - Connectivity > USART2
        - Mode: Asynchronous
        - Baud Rate: 115200 (8N1)
      - Connectivity > I2C1
        - I2C: I2C
      - Enable PB0 as GPIO_Output



  - After Code Generation:
    - In Core > Src:
      - Add the files in Demo Applications > Ambient Temperature Monitor
      - Add the .c files in the Scheduler Directory.
    - In Core > Inc:
      - Add the .h files in the Scheduler Directory.
  - TeraTerm:
    - Open TeraTerm application, and establish a new serial connection to USB-to-Serial Comm Port.

        - From Setup > Serial port.. > adjust the speed to be equal to the Baud Rate (115200).
    - Build the project and load it on the Nucleo-32 Board.
    - Enter the threshold value (the user has a buffer of 5 characters, they can use 2 digits for the integer and fractional partions each, separated by a dot).
    - Notice the output ambient temperature, and its effect on the LED flashing.
- Hardware Components:
    - Nucleo-32 Board
    - DS3231 RTC Sensor
    - External LED
    - USB-to-Micro USB Cable
    - USB-to-TTL module
- Below, is an image of how the connections should look like.

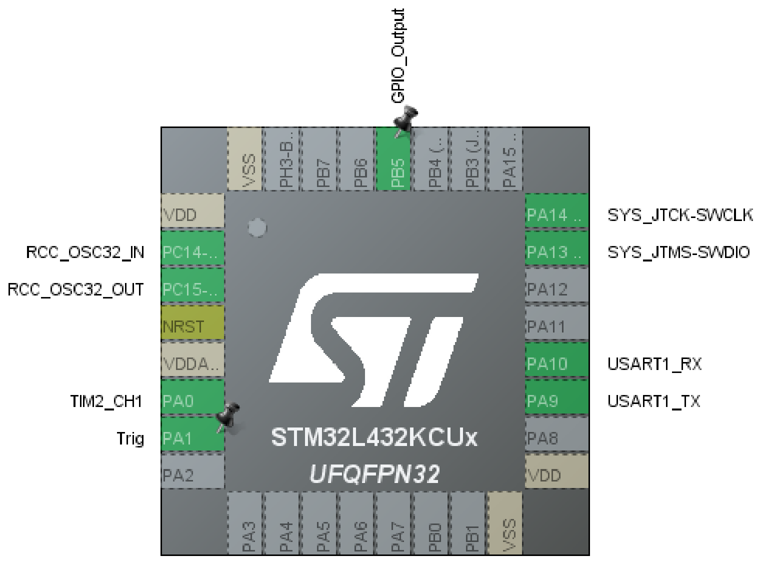


- A demo video can be found on this link.

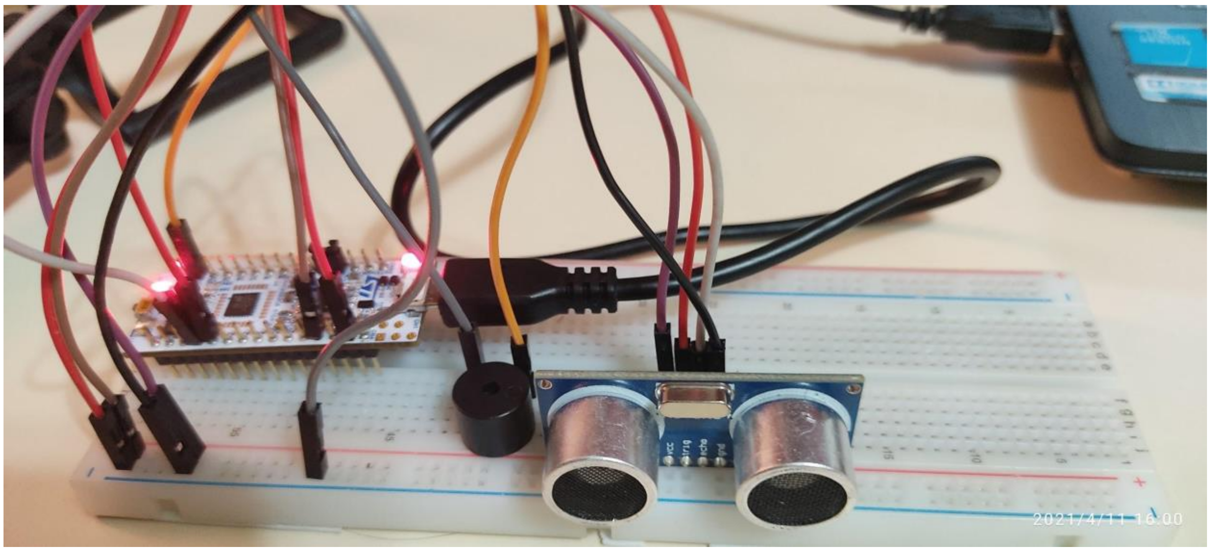
- **Parking Sensor Application:** The purpose of this application is to read the distance between the ultrasonic sensor HC-SR04 and accordingly the alarm buzzer will produce a sound reflecting the distance between the object and the ultrasonic sensor
  - Design:
    - Tasks:
      - ReadDistance: to send 10uS Trig and receive Echo from the difference between the rising edge and the falling edge of TIM2 and calculate the distance between the body and sensor using the equation:

        Distance = echo time * speed of sound /2
      - ToggleBuzzer: toggling the buzzer reflecting the distance between the object and the sensor.
    - Logic:
      - ReadDistance and ToggleBuzzer tasks are enqueued at the beginning of the program, with a priority of 1 and 2 respectively.
      - Then, the ReadDistance task reruns itself every 2 seconds, and the Toggle LED task reruns itself according to the new value of the distance.
      -
  - How to Build and Run:
    - CubeMX Configurations:
      - From the available boards, select STM32L432KCUx.
      - In Pinout & Configuration tab:
        - System Core > RCC:
          - LSE: Crystal/Ceramic Resonator
        - System Core > SYS:
          - Debug: Serial Wire
          - Timebase Source: SysTick
        - Connectivity > USART1
          - Mode: Asynchronous
          - Baud Rate: 9600
        - TIM2 (As a General Purpose Timer):
          - Clock Source: Internal Clock
          - Channel1: Input Capture Direct Mode
          - Polarity Selection: Both Edges

        - Enable PB5 as GPIO_Output
        - Enable PA1 as GPIO_Output and label it Trig
- Hardware Components:
    - Nucleo-32 Board
    - Ultrasonic sensor HC-SR04
    - Buzzer
    - USB-to-Micro USB Cable
    - USB-to-TTL module



    - After Code Generation:
        - In Core > Src:
            - Add the files in Demo Applications > Parking Sensor
            - Add the .c files in Scheduler Directory
        - In Core > Inc:
            - Add the .h files in Scheduler Directory
    - TeraTerm (Optional):
        - uncomment the Tera Term code
        - Open TeraTerm application, and establish a new serial connection to USB-to-Serial Comm Port.
    - Build the project and load it on the Nucleo-32 Board.
    - Place an object in front of the ultrasonic sensor and vary the distance.
    - Notice the change in duration of the beeps reflecting the distance
- Below, is an image of how the connections should look like.



- A couple of demo videos can be found on those links:
  - General Demo
  - Displaying Distance Demo

- **Acknowledgement:**

We would like to express our appreciation to The American University in Cairo (AUC) for its continuous support, and the Embedded Systems Course, under the supervision of Professor Mohamed Shalan.