City University of Hong Kong

2021 – 2022 Sem B

EE3220 System-on-Chip Design

Lab Report 1

Lab 1: Embedded System Design with STM32F401RE ARM Board

Lab 2: Interrupt Programming

Objective of the Experiment

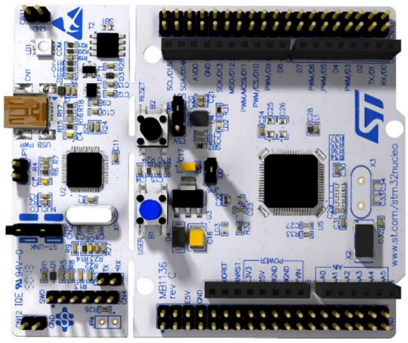
Lab1:

- Explore the Nucleo-F401RE board and STM32 microcontroller (STM32-Nucleo-F401RE)
- Create a new Mbed OS projects using Keli Studio Cloud
- Communicate between the board and the computer via USART serial communication with PuTTY
- Write C/C++ programs on LED control with user interrupt.
- Program and debug the board from the Keil Studio Cloud

Lab2:

- Learn the serial communication and GPIOs with the Mbed platform
- Familiarize the interrupt programming on Mbed
- Configure the program time and serial interrupts for the ARM Cortex M4 Microcontroller on the board using the Mbed library
- Handling Multiple Interrupts
- Program and debug the board from the Keil Studio Cloud

Apparatus Used in The Experiment



Hardware:

STM32-Nucleo-F401RE (F401RE) (Image 1) (STMicroelectronics, 2021), is a Microcontroller board featuring the STM32F401RE Microcontroller that provides a flexible method for users to try out build prototypes by choosing from the several combinations of performance and power consumption features. This development board provides Mbed and Arduino Uno V3 connectivity support, and the ST morpho headers.

Image 1: STM32-Nucleo-F401RE

- **Software**:

  Keil Studio Cloud, is a free Cloud-based IoT development tool and browser-based IDE for Arm-based microcontrollers developed by ARM. It provides a cloud-hosted workspace (arm Developer, 2022), a modern C/C++ editor with IntelliSense and git integration (Keil Studio Cloud, 2022). Users can edit their projects from any computer. In addition, Keil Studio Cloud provides mighty web debugging features.

PuTTY, is an open-source SSH and a Telnet client for the Windows platform developed by Simon Tatham (PuTTY, 2022). The serial connection of PuTTY is the main function of lab sections.

Procedure of The Experiment

Lab1:

There are a total of 5 parts with 4 Checkpoints of Lab1:

1. Introduce an overview of the STM32-Nucleo-F401RE and various development tools.
2. Create the new Mbed project in Keil Studio Cloud, write a C/C++ code about blinking an LED on the F401RE. Build the project and save the generated binary. Copy the binary to the board, then observe the result. (Checkpoint 1)
3. Create another Mbed project, write a C/C++ code about controlling LED with press button. Build the project, save the generated binary, and load it to the board. Open PuTTY, set up the serial connection between the board and the host. Observe the output and the difference in both the PuTTY serial connection and the board when a button is pressed. (Checkpoint 2)
4. Modify the code about controlling LED with serial commutation. Repeat the steps above, then type the number 0 or 1 in the serial connection windows and observe the variation of the development board. (Checkpoint 3)
5. Introduce the debug function in Keil Studio Cloud. Setting up the debug then debugging the modified program in part 4 with setting up breakpoints at line 21. Then repeat the process to toggle the LED, observe the changing of the board and registers r0 and r11, as well as variable “command” when the debugging procedure is running. (Checkpoint 4)

Lab2:

There are a total of 3 parts with 2 Checkpoints and 1 Bonus of Lab2:

1. Create the new Mbed project in Keil Studio Cloud, write a C/C++ code about blinking an LED with interrupting. Build the project and save the generated binary and load it to the board. Open PuTTY sets the serial connection between the board and computer and observes the result. Then, change the “Ticker” keyword to “Timeout”. Repeat the previous step to observe the difference. (Checkpoint 1)
2. Create the new Mbed project in Keil Studio Cloud, write a C/C++ code about multiple interrupts handling. Repeat the build and save and PuTTY steps in part 2. Then, type any number into serial connection windows and observe the variation of the development board. (Checkpoint 2)
3. Set up breakpoint debug and observe the project in Part 3. (Bonus)

Results

Lab1:

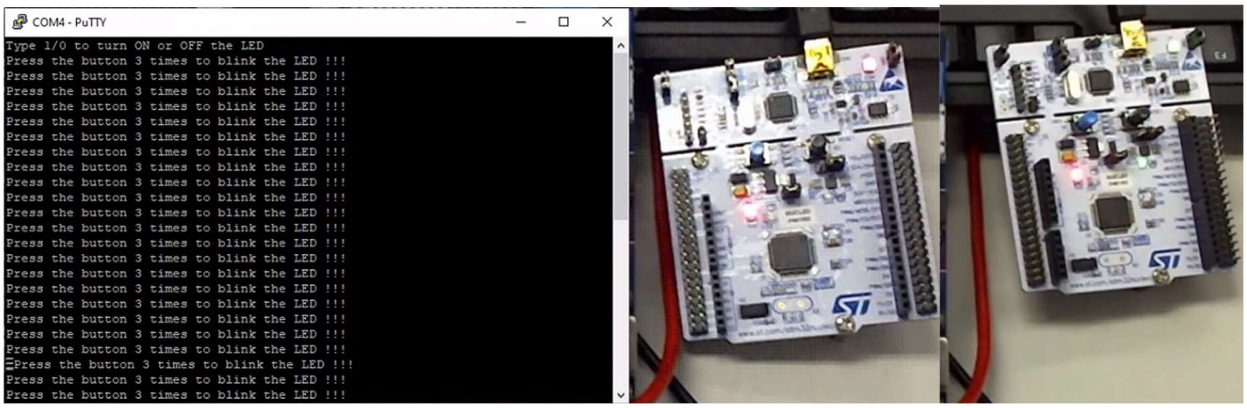


Image 2: Observation of lab 1 check point 1

1. **Check point 1:**

   LED blinking for every second.

2. **Check point 2:**

   The number of times the LED blinking depends on the times of pressing button.

   PuTTY screens show the output “Press the button # times to blink the LED” which # is the number of times.

3. **Check point 3:**

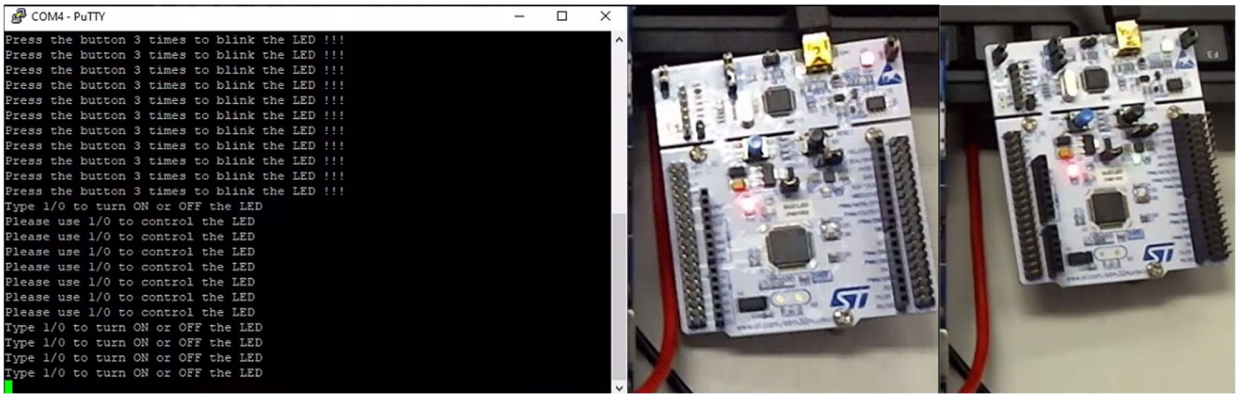


Image 3: Observation of lab 1 check point 3

PuTTY screens show the output “Type 1/0 to turn ON or OFF the LED”.

1) **Input 0 at PuTTY screens:** LED turn off.
2) **Input 1 at PuTTY screens:** LED turn on.
3) **Other input:** PuTTY screens show the output “Please use 1/0 to control the LED”.

4. **Check point 4:**

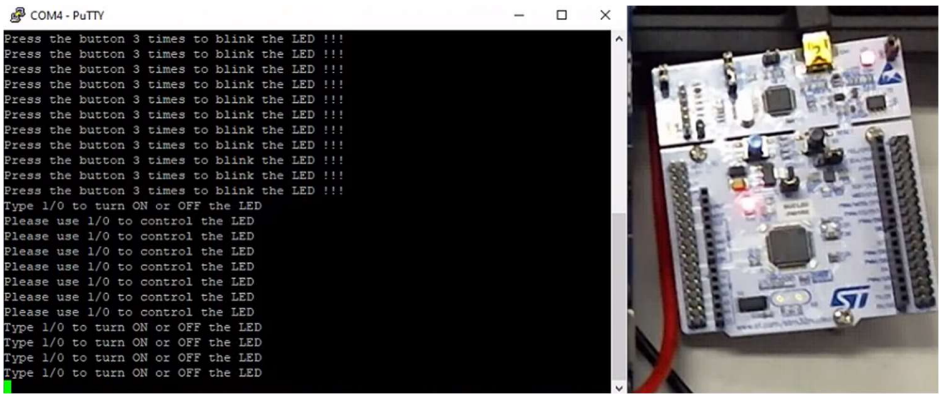


Image 4: Observation of lab 1 check point 4

Press the Continue button, the debugger continues to run the code until reach the breakpoint (line 21).

The LED turns off at this point. The registers r0 and r1 value changed.

Press the Step Over button twice, the debugger continues to run the code.

The LED turns on.

- **Lab2:**

1. **Check point 1:**

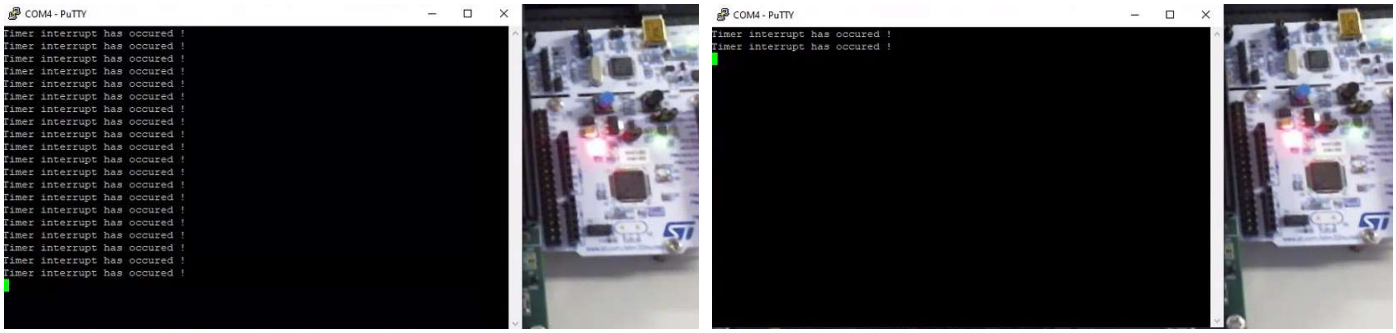


Image 5: Observation of lab 2 check point 4

Original Code (Ticker class in line 5):

The LED blink every second, time interruption signal received and print message" Timer interrupt has occurred!" then sleep for 8 seconds, and repeat.

Modified Code (Timeout class in line 5):

The LED blink every second, time interruption signal received and print message" Timer interrupt has occurred!" then stop after the time interruption and repeat.

2. Check point 2:

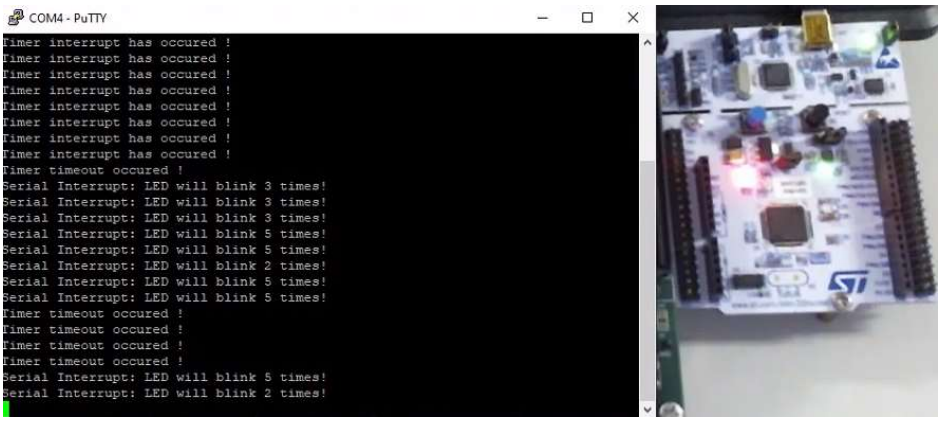


Image 6: Observation of lab 2 check point 2 after input the number 5 and 2 at PuTTY screens

1) If no number input at the beginning, LED blinks 4 times once with time interrupt and print the message “Timer timeout occurred!” after the time interruption.
2) Then, the LED blink for number of times depending on the input number with serial interrupt and prints the message “Serial Interrupt: LED will blink # times!” during the serial interrupt, which # is the input number.
3) If there is no input afterwards, the LED will not turn off.

3. Bonus Part:

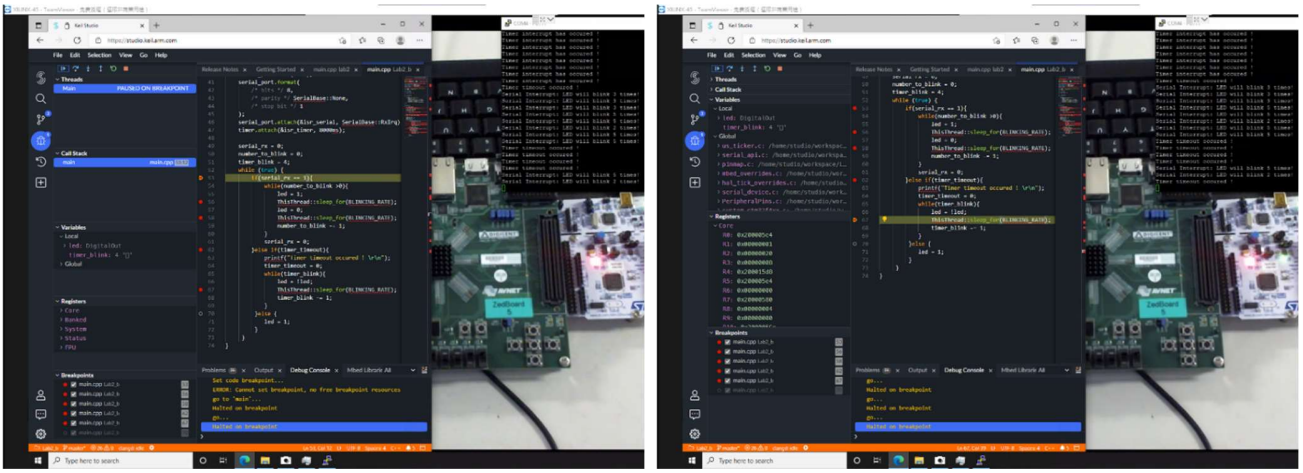


Image 7: Observation of lab 2 bonus part

We set several breakpoints to observe the variation, after the program starts () and after the timeout occurred ().

Analyzed Results

The observed result of the lab is matched with the expected result from the lab instructions to a large extent, in which the behavior of the board and LED are matched with what we had written in the code.

In lab1, as shown in the code, the duration and blinking frequent of the LED are controlled with the function sleep_for() within class ThisThread which provided by the Mbed library. The output and input of the board such as the LED and button can be set by DigitialIn and DigitialOut. The serial port can be set by the BufferSerial, for input and output to the serial which can be interacted with the serial port with PuTTY.

The value of registers r0 and r1 observed in lab1 within the debug mode is updated when the program is not pause as these two registers are the callee-saved register for local variables.

In lab2, as shown in the code, the Ticker class can define a time object which ticks for time for a duration, the Timeout class can define a time object which will timeout for a duration set by the code. The timer interrupt occurred if the timer.attach function is called, same as for the serial interrupt if the serial.attach function is called.

Discussion

Latency

Since online mode is adopted for lab sessions, the equipment for the lab is connected to the lab computer then to the internet, and the only way for students to control that equipment is by TeamViewer which causes latency on controlling the remote computer, such as editing the C/C++ code on the Keli Studio Cloud on the remote computer or inputting for the serial communication. Also, the video quality is variable due to the consistence of the connection between the student and the lab is constant.

Inconvenience

As mentioned above, online mode is adopted for lab sessions, other than the technical difficulties, there are other inconveniences caused by the limitations of adopting online lab sessions. Such as there are only a limited number of TAs to serve the students which the number is much higher than the number of TA staff. Thus, some operations require interaction with the hardware, which was inconvenient as the only way to do so is through the TA. Other than that, the camera angle and environmental brightness will affect the video quality, as the students have no way to adjust them.

Conclusion and Summary

Overall, in Lab 1, we learned the overview and several features of the Nucleo-F401RE board, STM32F401RE microcontroller, and Keil Studio Cloud. We created a few Mbed OS projects then wrote a C program inside the project. The program includes blinking the LED and controlling the blinking with the development board's button or serial communication. We recognized the serial connection between a computer and the development board using PuTTY. Lastly, we practiced debugging the project with Keil Studio Cloud.

For Lab 2, we understood how to use interrupt on the Keil Studio Cloud using the Mbed OS 6 and the development board. Furthermore, we observed and enrolled the difference between Timeout and Ticker class in Mbed OS 6. There are multiple interruptions that we recognized and handled as well.

Reference

arm Developer. (2022). *Arm Keil Studio Cloud User Guide*. Retrieved from arm Developer: https://developer.arm.com/documentation/102497/1-5/Arm-Keil-Studio

Keil Studio Cloud. (2022). Retrieved from arm KEIL: https://www.keil.arm.com/

PuTTY. (2022). *PuTTY*. Retrieved from PuTTY: https://www.putty.org/

STMicroelectronics. (2021, 12). *STM32 Nucleo-64 boards Data brief*. Retrieved from STMicroelectronics : https://www.st.com/resource/en/data_brief/nucleo-f401re.pdf

EE3220: System-On-Chip Design Lab 1: Embedded System Design with STM32F401RE ARM Board.

EE3220: System-On-Chip Design Lab 2: Interrupt Programming.