LAB 2 Report

Task 1: GPIO Interrupt

- *(a) Introduction and High-Level Description*
    - The purpose of task 1 is to let STM32 respond to two external interrupts, and then indicate when the interrupts occurred. The solution to this requirement is using EXTI0 and EXTI8 to detect the external interrupts and print out whether EXTI0 or EXTI8 has been triggered when the corresponding interrupts happened.
    - After an interrupt occurred, the program will run the code in the interrupt function and return to the main function by clear an interrupt flag.
    - Wire connection:

Table 1: Task 1 schematic

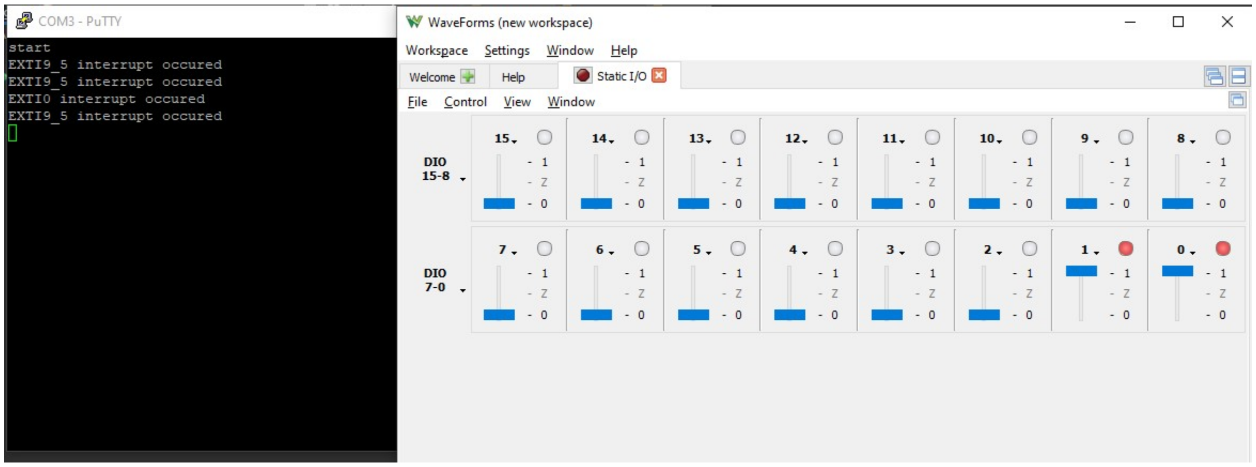
	Connected to	Connected to
Interrupt EXTI0	Pin D4	Analog Discovery
Interrupt EXTI8	Pin D5	Analog Discovery

    - Flow chart is in the appendix
- *(b) Low Level Description*
    - Interrupt is special. It can be triggered when the program is running. By initialization, the corresponding pin can be used to trigger interrupt. Here the task

required students to use EXTI0 and EXTI8. The available pin on the board is restricted. According to Arduino Uno V3 connector table[1, ch. 7.2, pp. 25], students choose Pin D4(GPIO J0) for EXTI 0 and Pin D5(GPIO C8) for EXTI 8.

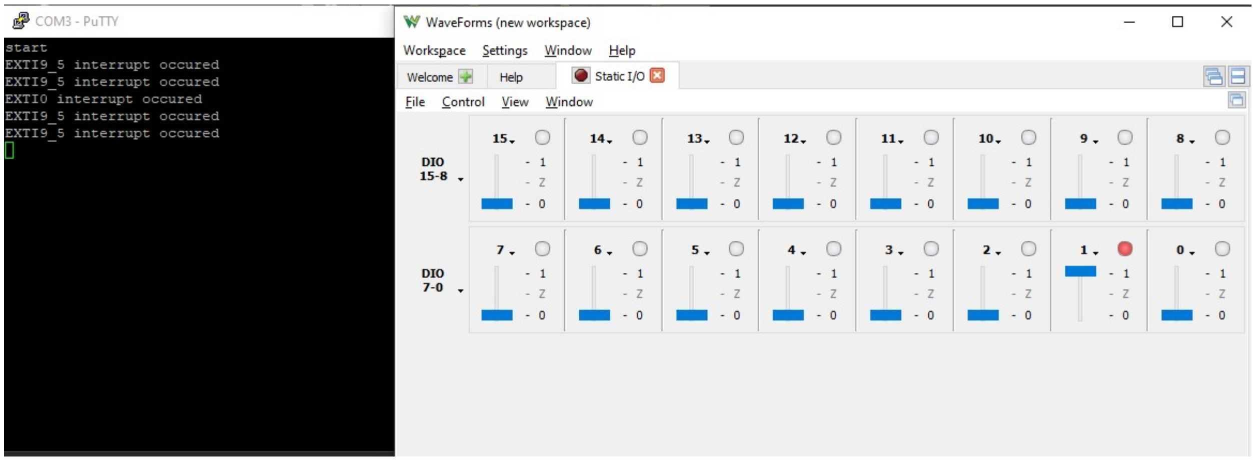
  - For the register part, students first enable the GPIO setting as the part in LAB1. To enable the interrupt, students need to open `SYSCFG->EXTICR[0]` to enable GPIOJ. Then Enable NVIC for interrupt 0, the priority position need to be processed, `NVIC->ISER[0] = (uint32_t) 1 << (6);` [2, ch. 4.2.1, pp. 185] .For EXTI0, priority 6. Then open the interrupt character such as bitmask, falling edge and rising edge.
  - For the HAL part, students use the same way as LAB1 to initialize the GPIO port. Then enable the NVIC interrupt setting and the corresponding IRQ by `HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);`. [3, ch. 10.2.4, pp. 151]
  - `HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin);` is the call back function for the HAL part. And the call back function is integrated into the `EXTI0_IRQHandler()` function. It needs a few loop in the call back function to make sure the interrupt flag is clear. Otherwise, incorrect interrupt will be triggered.

- *(c) Results and Analysis:*
  - Program works as expected.
  - It is hard to figure out what is missing when the interrupt is not working
  - Results are as follow, Figure 8.



```
start
EXTI9_5 interrupt occured
EXTI9_5 interrupt occured
EXTI0 interrupt occured
EXTI9_5 interrupt occured
```

Figure 1: Task 1 result 1



```
start
EXTI9_5 interrupt occured
EXTI9_5 interrupt occured
EXTI0 interrupt occured
EXTI9_5 interrupt occured
EXTI9_5 interrupt occured
```

Figure 2: Task 1 result 2

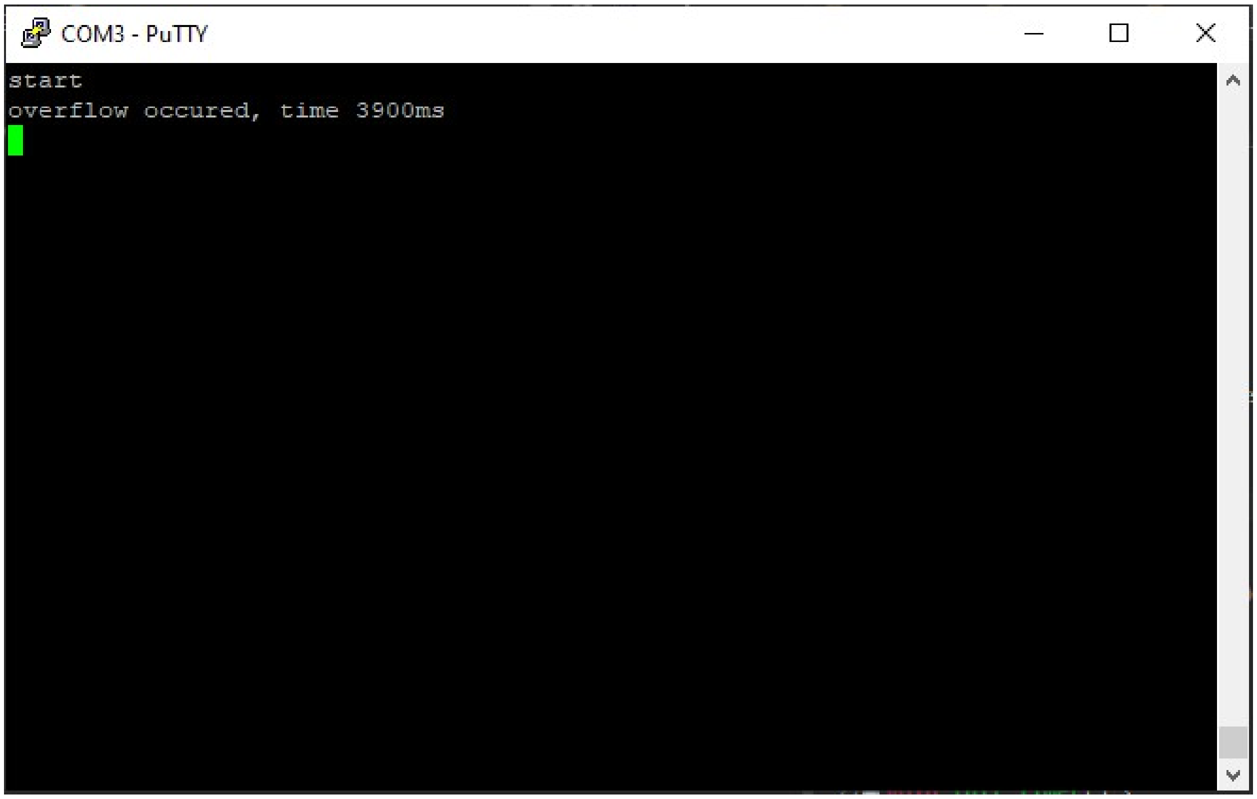
Task 2: Simple Timer: Register Implementation

- *(a) Introduction and High-Level Description*
  - In this task, one of the basic timers in the microcontroller is used. The target is to display elapsed time in tenths of seconds. Timer should be properly configured

and should be as accurate as possible. Pre-scaler and auto-reload value should be determined.

- No schematic for this task.
- Flow chart is in the appendix

- *(b) Low Level Description*
    - Students will use TIM6, a simple timer, and the Interrupt priority position is 54.
    - Students need to determine the prescaling register and auto-reload register here to make the timer overflow once a tenth of a second. The calculation are as follow:
        - $Overflow\ Time = \left(\frac{PSC+1}{CLK}\right)(ARR+1)$
    - To set the timer to 10Hz. Pre-scaler will be set to 10799 by code `TIM6->PSC = 0x2A2F;` And Auto-reload value will be set to 999 by code `TIM6->ARR = 0x03E9;`.
    - After that, use `TIM6->EGR |= 0x0001;`, `TIM6->DIER |= 0x0001;`, and `TIM6->CR1 |= 0x0001;` to generate update events to auto reload, enable update interrupts, and start the timer properly. [4, ch. 28.4, pp. 1090-1091]
    - The callback function is the same as task 1.
- *(c) Results and Analysis:*
    - Program works as expected.
    - It is hard to figure out what is missing when the interrupt is not working
    - PSC and ARR are both for students to determine, which is a bit confusing.
    - Results are as follows, Figure 9.



```
start
overflow occured, time 3900ms
```

Figure 3: Task 2 Result

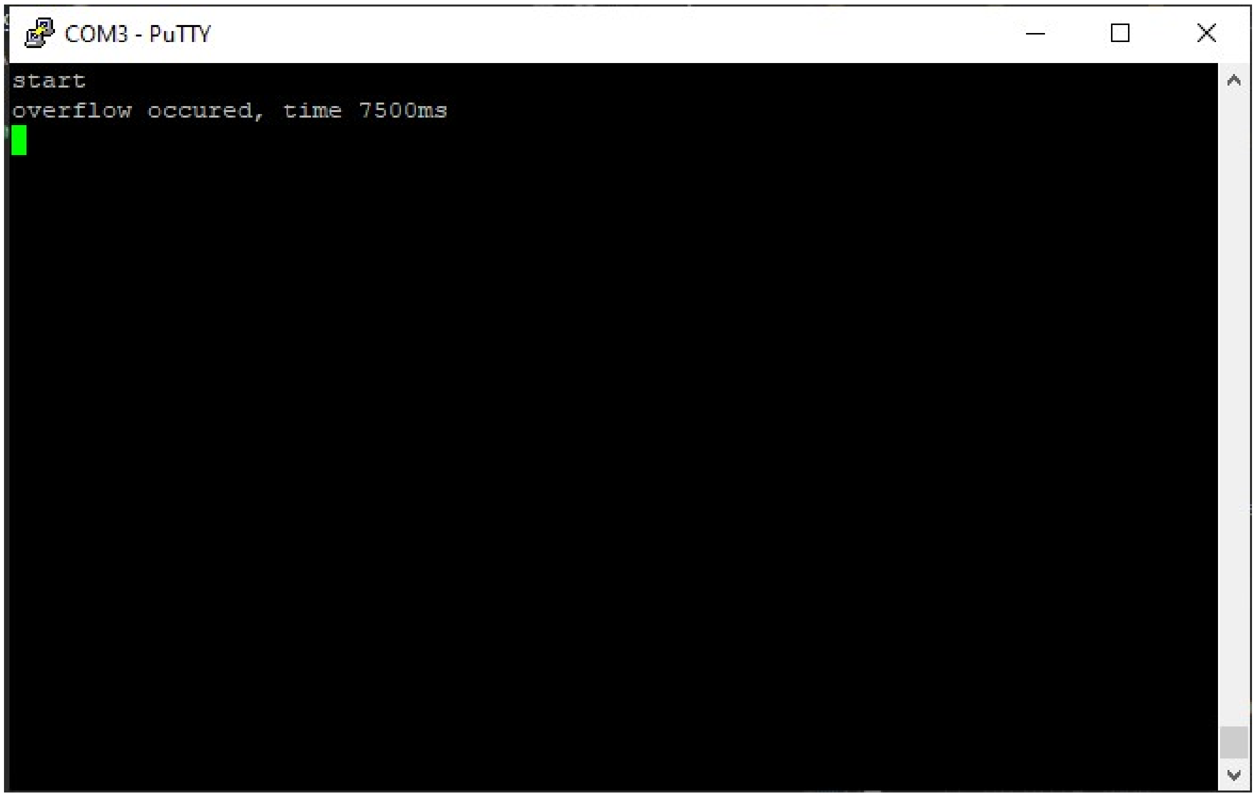
Task 3: Introduction to User Interface

- *(a) Introduction and High-Level Description*
    - The target for this task is the same as task 2. But students need to use HAL function to fulfill the implementation.
    - No schematic in this task.
    - Flow chart is in the appendix
- *(b) Low Level Description*
    - The initialization of this task is in HAL function. Open clock, NVIC are the same as task1. To open Timer, `HAL_TIM_Base_Init(&htim6)`, and `HAL_TIM_Base_Start_IT(&htim6)` to configure and open the timer. Like the

GPIO initialization in LAB1, a type of pointer, `TIM_HandleTypeDef` is needed to configure the timer. [3, ch. 90.2, pp.1704]

- The calculation of the prescaler and period is different to the register part. The calculation are as follow [5, ch. 11.2.1, pp. 315]
  - $Overflow\ Time = \frac{Clock}{(Prescaler+\ )(Period+1)}$
- `htim6.Init.Prescaler = 5399;` and `htim6.Init.Period = 1999;` are used to set the Prescaler and the Auto-reload Value to get 10Hz from the timer.
- The call back function works the same as task 1.

- *(c) Results and Analysis:*
  - Program works as expected.
  - Results are as follows, Figure 9.



```
start
overflow occured, time 7500ms
```

Figure 4: Task 3 Result

Task 4: [Depth] Reaction Time Game

- *(a) Introduction and High-Level Description*
  - In this task, students need to combine the interrupt functions in Task 1, 2, and 3 to create a reaction time game. The Game flashes the screen after a random delay and measures the time it takes for the player to react and press the push button.
  - The interrupt settings are the same as the previous part. The implementation concentrates on how to get the program work.
  - Flow chart is in the appendix, Figure 10.
  - Wire connection:

Table 2: Task 4 schematic

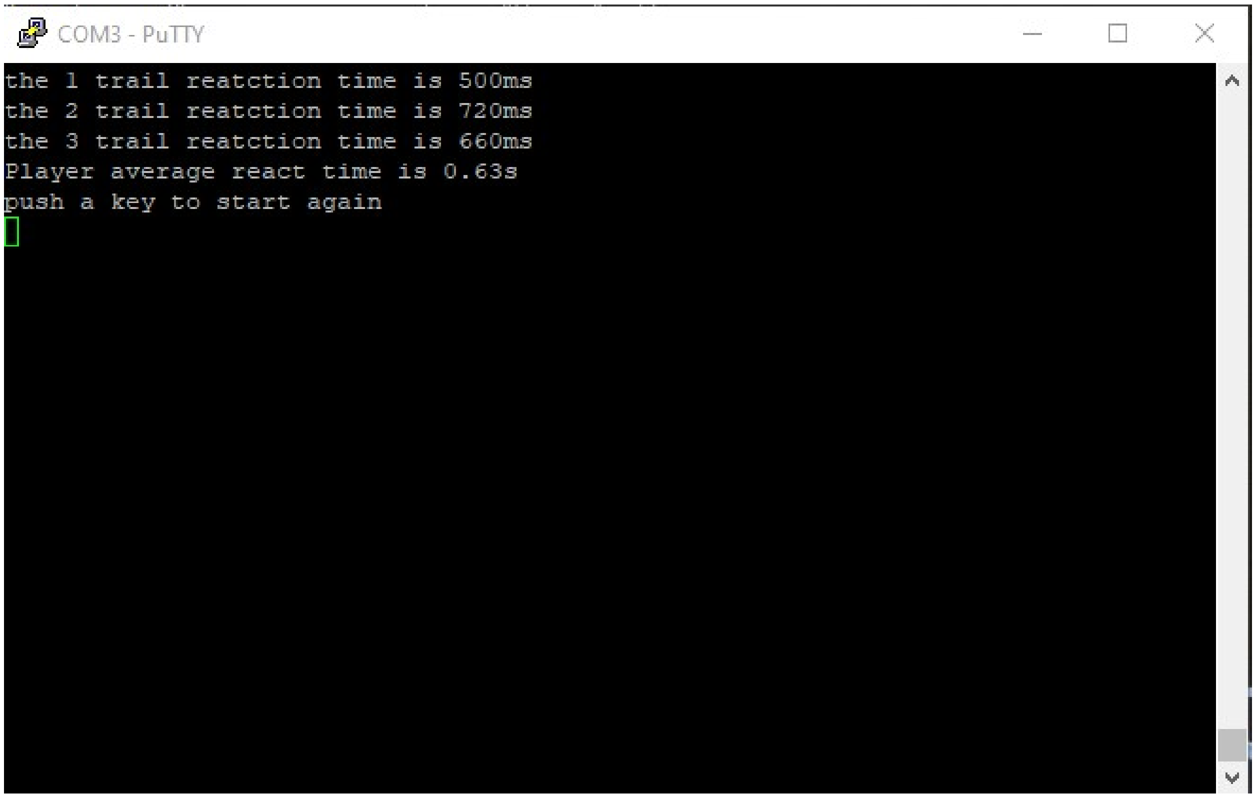
	Connected to	Connected to
User Pushbutton on board	USER_B	(No connection, on board pushbutton)
Restart pushbutton	Pin D4	Analog discovery

- *(b) Low Level Description*
  - The random delay time is generated by the random function rand() and srand(). Here srand() gets a random seed. Students use time(NULL) function to get current time as a random seed and them use rand() to generate a random number. The

delay time here is between 0.5s to 4.5s. Students use `(50 + 10 * (rand()%40))*10` to get 500 to 4500(ms) delay time.

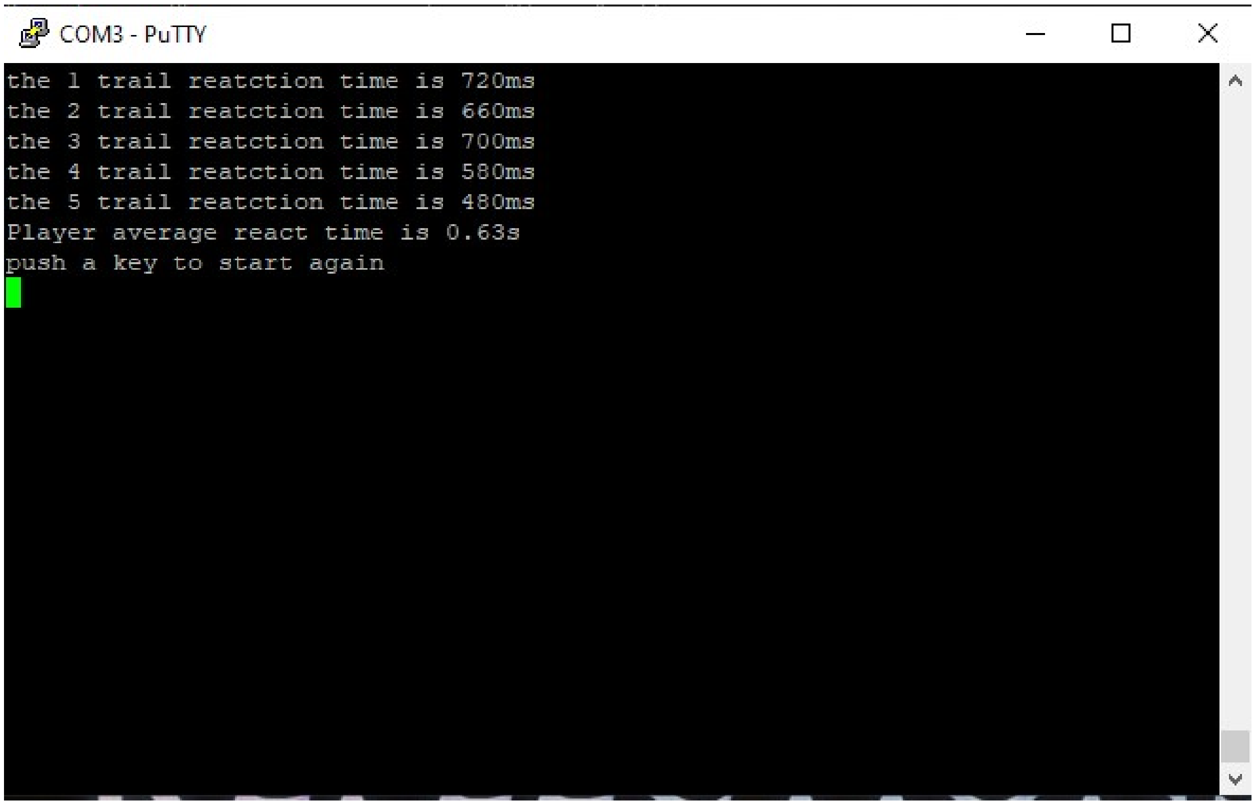
  - The screen is cleared before each trail. Then the overflow counter in the timer interrupt is also cleared after the delay. The program will record the counter number when GPIO interrupt is triggered. Then the result is record in an array to calculate the average react time.
  - Students use the User pushbutton, which is GPIO A0 for react time, and Pin D4 connected to the Analog Discovery for restart the game. Here `NVIC_SystemReset();` is used to restart the program[citation].
  - The overflow time is 10ms, so the accuracy of react time is 10ms.
  - When the number of trail is larger than 5, the program will clear the first data in the array and move the remaining one to previous array space, the newest result is always stored in the last space of the array.

- *(c) Results and Analysis:*
  - The program worked as expected
  - The react time is little longer than expected, students are unsure if the function HAL_Delay() used in blinking function will increase the react time.
  - The result are as follow:



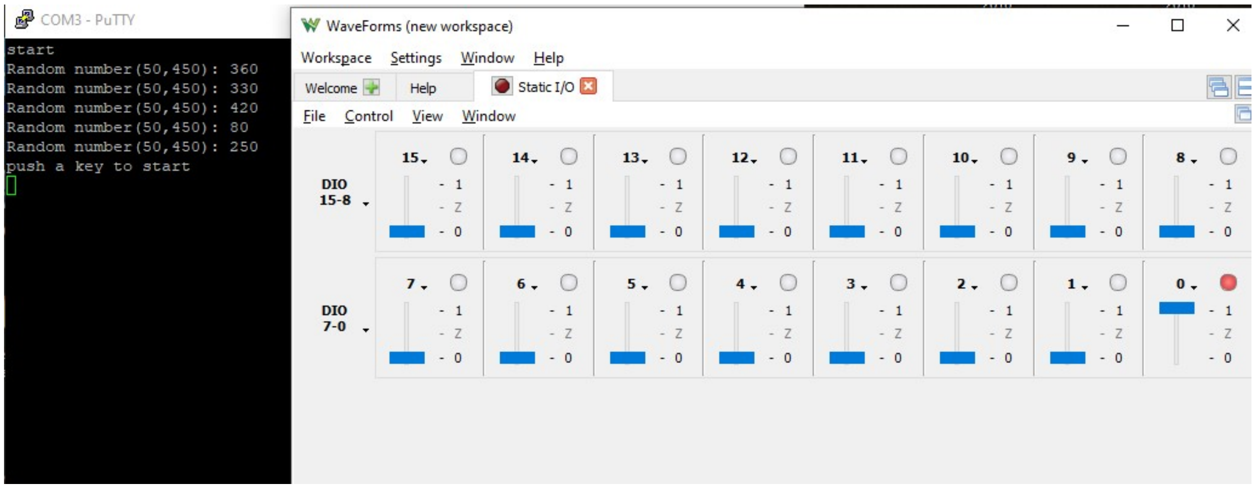
```
the 1 trail reatction time is 500ms
the 2 trail reatction time is 720ms
the 3 trail reatction time is 660ms
Player average react time is 0.63s
push a key to start again
```

Figure 5: Task 4 Result 1 Average score



```
the 1 trail reatction time is 720ms
the 2 trail reatction time is 660ms
the 3 trail reatction time is 700ms
the 4 trail reatction time is 580ms
the 5 trail reatction time is 480ms
Player average react time is 0.63s
push a key to start again
```

Figure 6: Task 4 Result 2 Average score



```
start
Random number(50,450): 360
Random number(50,450): 330
Random number(50,450): 420
Random number(50,450): 80
Random number(50,450): 250
push a key to start
```

Figure 7: Task 4 Result 3 GPIO Interrupt Restart

Conclusion

Through Task1, we got an idea about how to let the STM32 be triggered by an external interrupt. In this case, either the Analog Discovery board or a physical push pull button. We learned how to link EXTIs to desired ports and how to set the ISR codes correctly.

For Task 2, we used a built-in timer to trigger the interrupt. We defined the clock speed, period, and the starting point for the clock through registers. With that, we know the Hz of the timer and we can then calculate how long it will take for each period. In Task 3, we achieved the same target as Task 2 but with a different solution. We used another basic timer with a different system clock speed. So, we must recalculate the period and starting point for the timer. And this time, we modified all the settings with HAL.

Finally, we combined all the stuff we learned from previous tasks to write a game which can test reaction time. Users should click the push button as soon as they see the scream blink. The program will record the time interval between the blink and the click. After 5 trials, the program will print the average reaction time. Users can reset the program at any time by pressing the other button on the board.

The material we learned from Lab 2 can be used to create a clock. Since it supports both count down and count up functions. The clock will not only track the real-world time but can also tell you how many seconds or minutes to a certain point. It can also be used as part of the program which will automatically turn off TVs or monitors after not being interacted with for a specific time period. And if users make any input, it will trigger the interrupt and shut down the turn off program.

Citation

[1] *RM2033 User manual,* STMicroelectronics, NV, USA, 2018, pp. 25.

[2] *RM2053 Programming manual,* STMicroelectronics, NV, USA, 2019, pp. 185.

[3] *RM1905 User manual,* STMicroelectronics, NV, USA, 2017, pp. 150-153.

[4] *RM0410 Reference manual,* STMicroelectronics, NV, USA, 2018, pp. 1080-1091.

[5] *Mastering STM32,* Carmine Noviello, USA, 2018, pp. 315.

Appendix

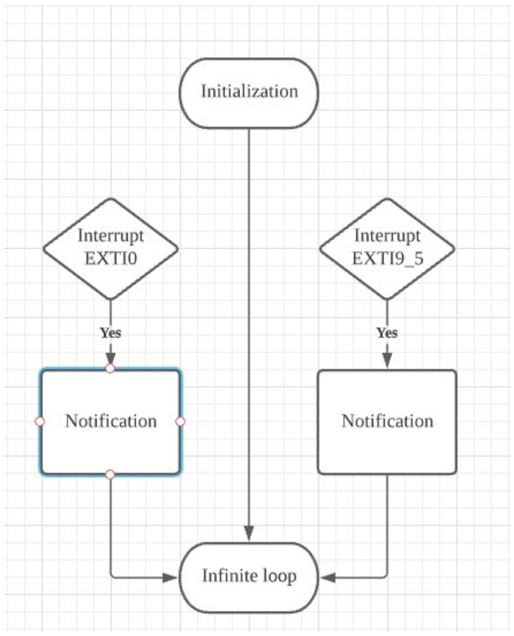


Figure 8: Task 1 flowchart

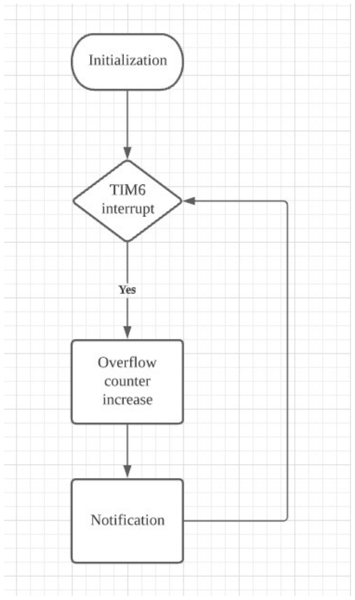


Figure 9: Task 2 flowchart

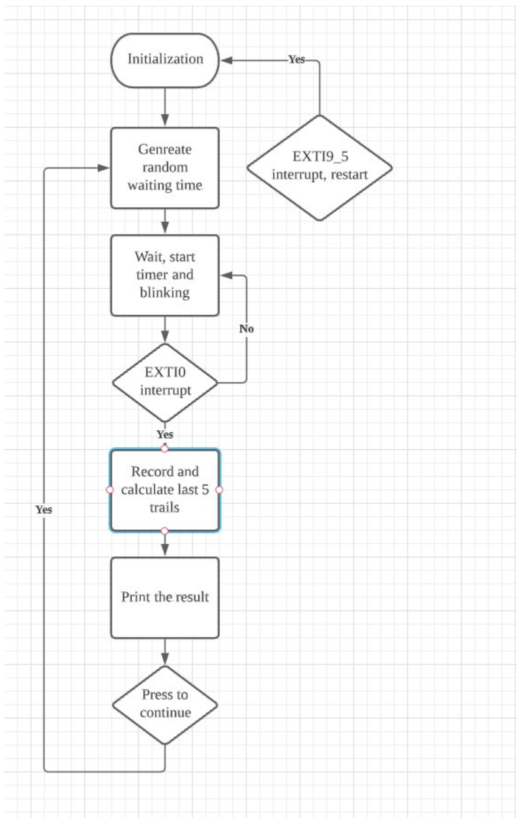

Initialization
Yes
Genreate random waiting time
EXTI9_5 interrupt, restart
Wait, start timer and blinking
No
EXTI0 interrupt
Yes
Record and calculate last 5 trails
Yes
Print the result
Press to continue

Figure 10: Task 3 flowchart