

Micro-Controller Experiment

Week7

Teacher: 廖裕評 Yu-Ping Liao

TA: 陳大荃 Da-chuan Chen, 陳恩妮 En-ni Chen

Class Rules

1. No drink besides water.
2. Bring a laptop and breadboard if needed.
3. Ask us TAs to sign and borrow development boards. Do not sign or ask others to sign for you without TAs' permission.
4. Arriving 10 minutes after the bell rings will be regarded as absent.
5. If you damage any borrowed equipment, you have to pay for it.

Homework Rules

1. Includes: A. Class content, B. Class exercise, C. Homework (screenshot or video)
2. Editing software: MS PowerPoint
3. File format: PDF
4. Filename: "date_group_studentID_name.pdf", like "0916_第1組_11028XXX_陳OO.pdf"
5. The homework deadline is 23:59 of the day before the next class. If you are late, then your grade will be deducted.

Contact

If you encounter any problems with this class, please get in touch with us with the following E-mails:

1. Teacher, Prof. Yu-Ping Liao 廖裕評 : lyp@cycu.org.tw
2. TA, Da-chuan Chen 陳大荃 : dachuan516@gmail.com
3. TA, En-ni Chen 陳恩妮 : anna7125867@gmail.com

Or visit 篤信 Lab353 for further questions.

Outline of the Week

1. PWM introduction
2. PWM Project.
3. Homework 7-1.
4. Homework 7-2.
5. Homework 7-3.
6. Homework 7-4 Bonus.

The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large, solid red speech bubble is centered on the page, pointing downwards.

PWM Introduction

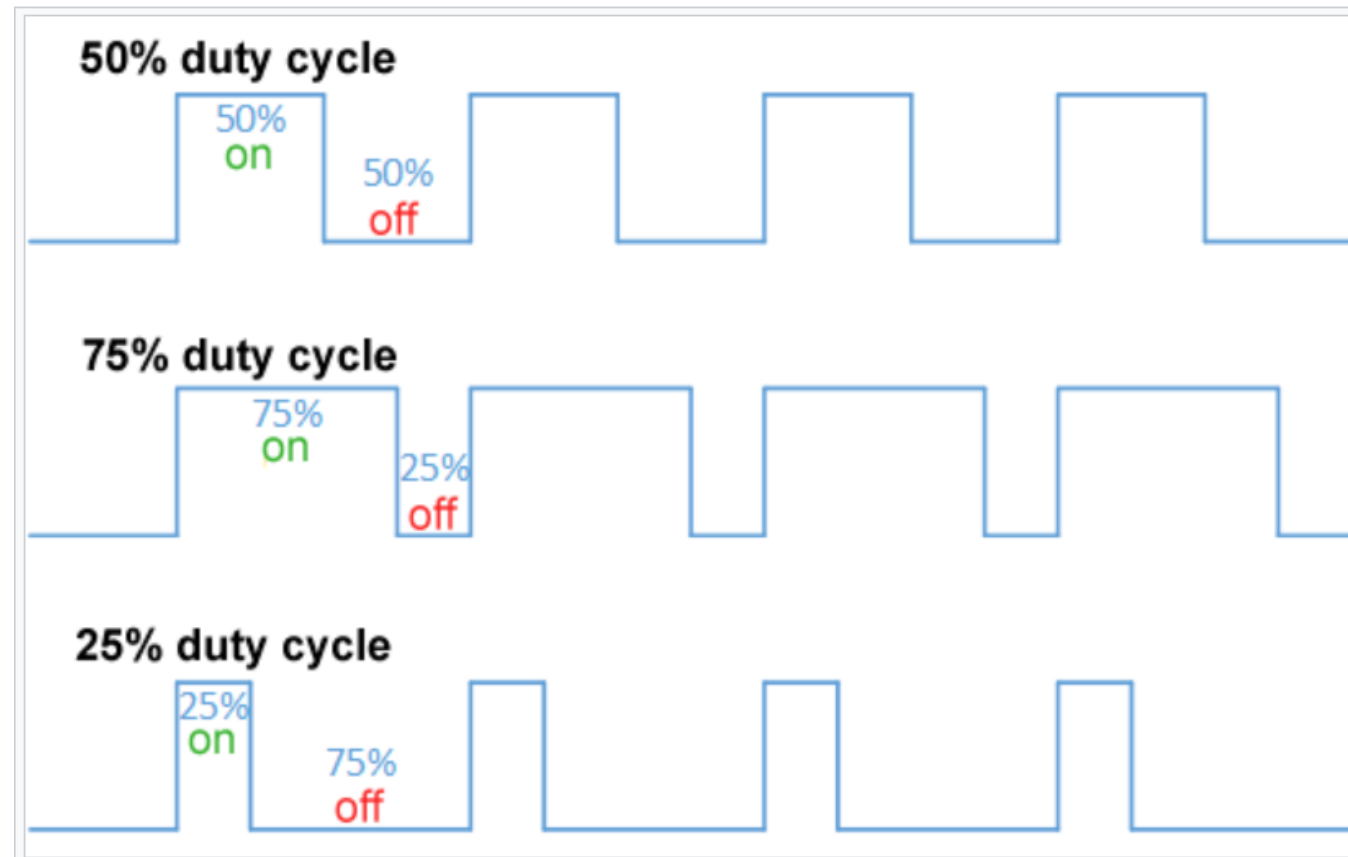
PWM

Pulse-width modulation (PWM) is a technique for generating analog signals using pulse waves. Typically, the converted pulse wave has a fixed period, but the duty cycle of the pulse wave varies depending on the magnitude of the analog signal.

The **duty cycle** and **frequency** of a PWM signal determine its behavior.

Duty Cycle

The term duty cycle describes the proportion of on time to the regular interval or period of time.



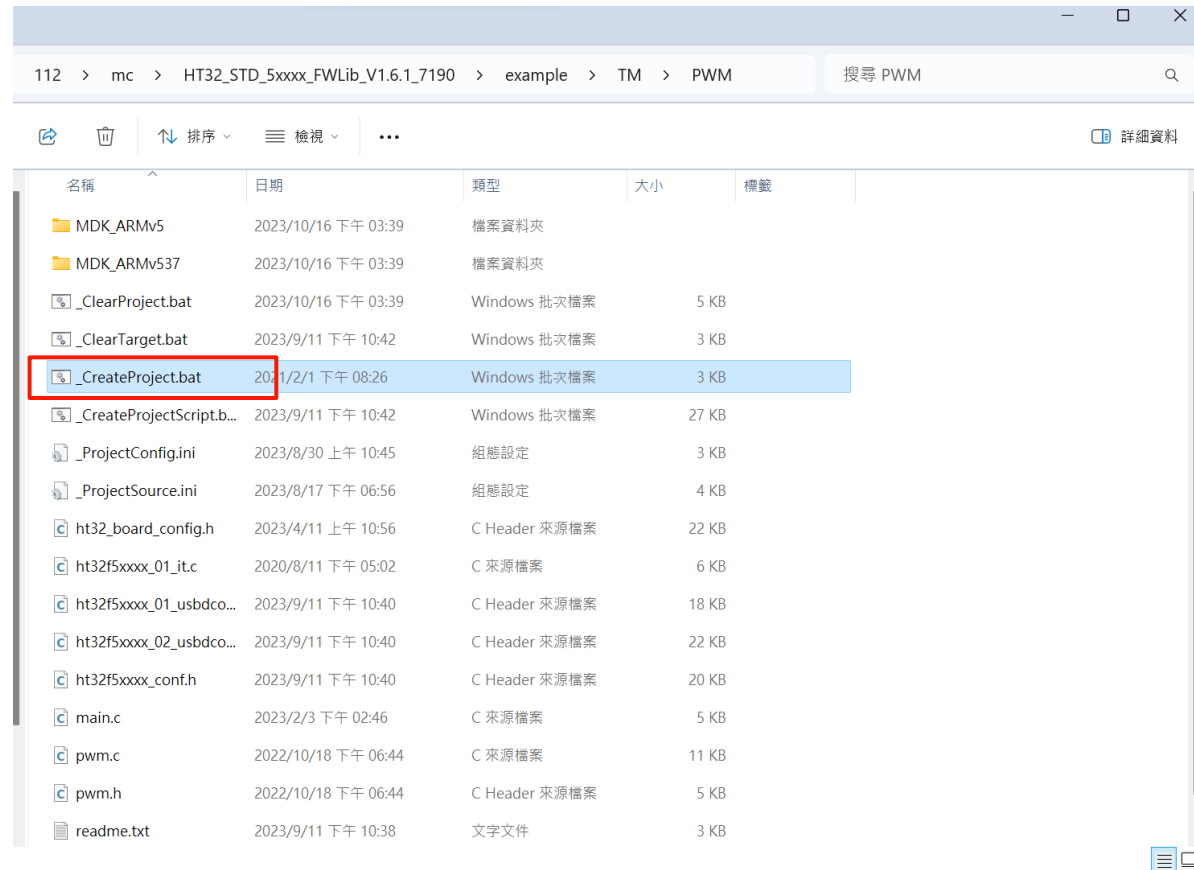
The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large red speech bubble is centered on the page, pointing downwards.

PWM Project

1. Execute “_CreatProject”

1. Go to “~/HT32_STD_5xxxx_FWLib_V1.5.1_7084/example/TM/PWM”.

2. Double click “_CreateProject.bat”.



2. Launch project

The image shows two windows from a development environment. The left window is a file explorer showing the directory structure of a project. The right window is the uVision IDE showing the project configuration and the main.c file.

File Explorer (Left):

- Path: 12 > mc > HT32_STD_5xxxx_FWLib_V1.6.1_7190 > example
- Files and folders listed:
 - MDK_ARMv5 (2023/9/26 下午)
 - MDK_ARMv537 (2023/9/26 下午)
 - _ClearProject.bat (2023/9/26 下午)
 - _ClearTarget.bat (2023/9/11 下午)
 - _CreateProject.bat (2021/2/1 下午)
 - _CreateProjectScript.b... (2023/9/11 下午)
 - _ProjectConfig.bat (2022/5/26 上午)
 - _ProjectConfig.ini (2022/5/26 上午)
 - ht32_board_config.h (2022/4/12 下午)
 - ht32f5xxxx_01_it.c (2018/8/3 下午)
 - ht32f5xxxx_01_usbdco... (2023/9/11 下午)
 - ht32f5xxxx_02_usbdco... (2023/9/11 下午)
 - ht32f5xxxx_conf.h (2023/9/11 下午 10:40)
 - i2cm.h (2016/5/25 下午 02:39)
 - main.c (2020/8/26 下午 07:06)

uVision IDE (Right):

- Project: Project_52352
- Files in Project:
 - User
 - main.c
 - ht32f5xxxx
 - pwm.c
 - Config
 - CMSIS
 - MDK-ARM
 - USBD_Library
 - Library
 - Retarget
 - Utilities
 - Doc
- main.c content:

```
1  /*  
2  * @file    TM/PWM/main.c  
3  * @version $Rev:: 6712 $  
4  * @date    $Date:: 2023-02-03 #$  
5  * @brief    Main program.  
6  *  
7  * @attention  
8  *  
9  * Firmware Disclaimer Information  
10 *  
11 * 1. The customer hereby acknowledges and agrees that the program technical documentation, including the  
12 * code, which is supplied by Holtek Semiconductor Inc., (hereinafter referred to as "HOLTEK") is the  
13 * proprietary and confidential intellectual property of HOLTEK, and is protected by copyright law and  
14 * other intellectual property laws.  
15 *  
16 * 2. The customer hereby acknowledges and agrees that the program technical documentation, including the  
17 * code, is confidential information belonging to HOLTEK, and must not be disclosed to any third parties  
18 * other than HOLTEK and the customer.  
19 *  
20 * 3. The program technical documentation, including the code, is provided "as is" and for customer reference  
21 * only. After delivery by HOLTEK, the customer shall use the program technical documentation, including  
22 * the code, at their own risk. HOLTEK disclaims any expressed, implied or statutory warranties, including  
23 * the warranties of merchantability, satisfactory quality and fitness for a particular purpose.  
24 *  
25 * <h2><center>Copyright (C) Holtek Semiconductor Inc. All rights reserved</center></h2>  
26 *  
27 *  
28 */ Includes -----*/
```

- Build Output:

```
After Build - User command #1: fromelf --bin -o ".\HT32\52352\Obj\HT32.axf.bin" ".\HT32\52352\Obj\HT32.axf"  
".\HT32\52352\Obj\HT32.axf" - 0 Error(s), 0 Warning(s).  
Build Time Elapsed: 00:00:11
```

main

```
54 int main(void)
55 {
56     PWM_Init();
57
58     PWM_UpdateDuty(PWM_CH0, PWM_DUTY_50);
59     PWM_UpdateDuty(PWM_CH1, PWM_DUTY_25);
60     PWM_Cmd(ENABLE);
61
62     Delay(5000);
63
64     PWM_UpdateDuty(PWM_CH0, PWM_DUTY_0);
65     PWM_UpdateDuty(PWM_CH1, PWM_DUTY_75);
66
67     Delay(5000);
68
69     PWM_Cmd(DISABLE);
70
71     Delay(5000);
72
73     PWM_SetFreq(PWM_FREQ_12K);
74     PWM_UpdateDuty(PWM_CH0, PWM_FREQ_12K * 0.25);
75     PWM_UpdateDuty(PWM_CH1, PWM_FREQ_12K * 0.75);
76     PWM_Cmd(ENABLE);
77
78     while (1);
79 }
```

PWM Initialization Setting

PWM Enable

Set Duty Cycle as X%

PWM Disable

Set CRR value

F12"PWM_FREQ_12K"->PWM.h

```

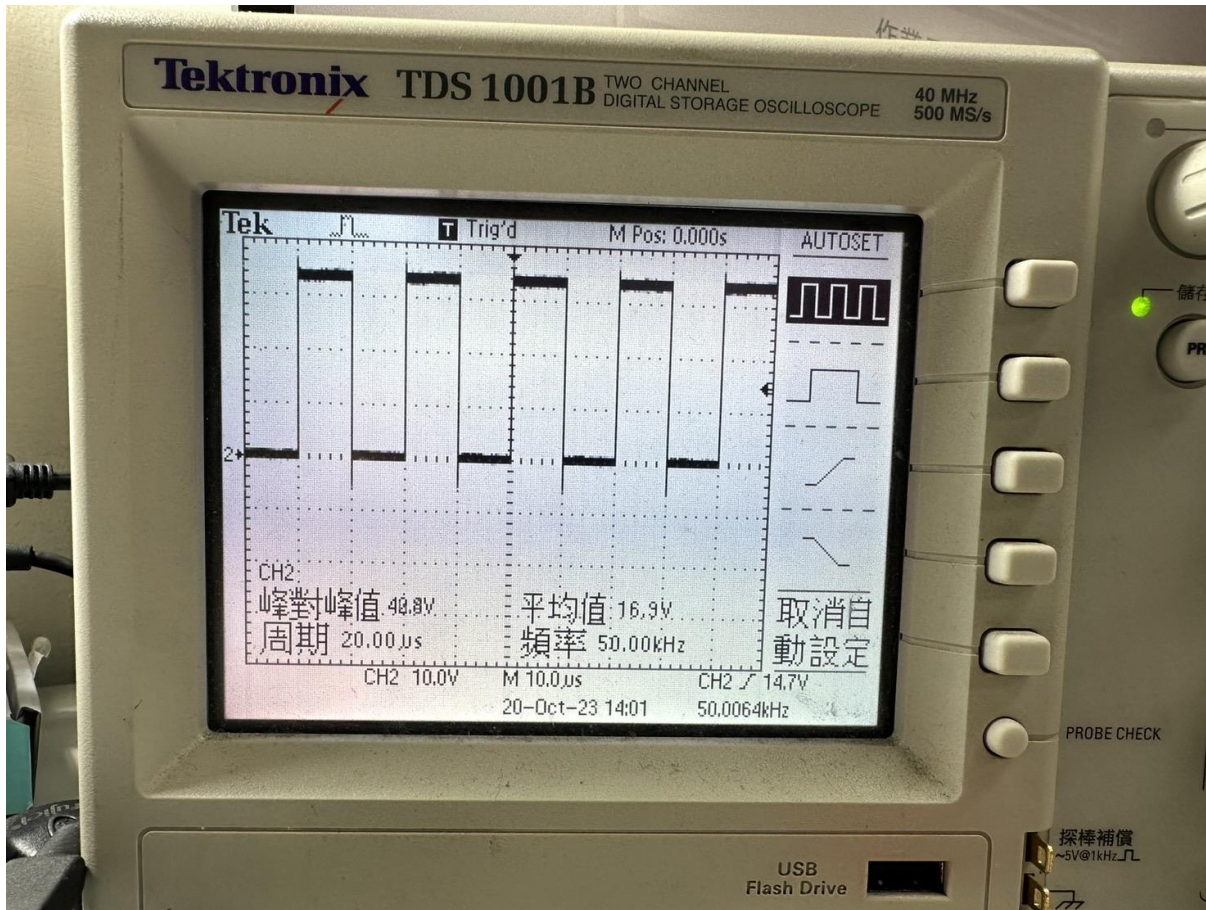
77 #define PWM_DUTY_0      (0)
78 #define PWM_DUTY_25    (HTCFG_PWM_TM_RELOAD * 0.25)
79 #define PWM_DUTY_50    (HTCFG_PWM_TM_RELOAD * 0.5)
80 #define PWM_DUTY_75    (HTCFG_PWM_TM_RELOAD * 0.75)
81 #define PWM_DUTY_100   (0xFFFF)
82
83 #define PWM_FREQ_50K    (HTCFG_PWM_TM_PCLK / HTCFG_PWM_TM_PRESCALER / 50000)
84 #define PWM_FREQ_40K    (HTCFG_PWM_TM_PCLK / HTCFG_PWM_TM_PRESCALER / 40000)
85 #define PWM_FREQ_12K    (HTCFG_PWM_TM_PCLK / HTCFG_PWM_TM_PRESCALER / 12000)
86
87
88 /* Settings -----*/
89 #define HTCFG_PWM_TM_PRESCALER    (1) // 1 ~ 65535
90 #define HTCFG_PWM_FREQ_HZ        (50000)
91 #define HTCFG_PWM_IDLE_STATE     (0) // 0: 0 duty, 1: 100 duty
92
93 #define HTCFG_PWM_TM_RELOAD      (HTCFG_PWM_TM_PCLK / HTCFG_PWM_TM_PRESCALER / HTCFG_PWM_FREQ_HZ)
94 #if (HTCFG_PWM_TM_RELOAD > 65536)
95 #error "HTCFG_PWM_TM_RELOAD out of range! Should be less than or equal to 65536."
96 #endif
97
98
99
100 #define HTCFG_PWM_TM_PCLK        (LIBCFG_MAX_SPEED)
101
102
103 #define LIBCFG_MAX_SPEED         (48000000)

```

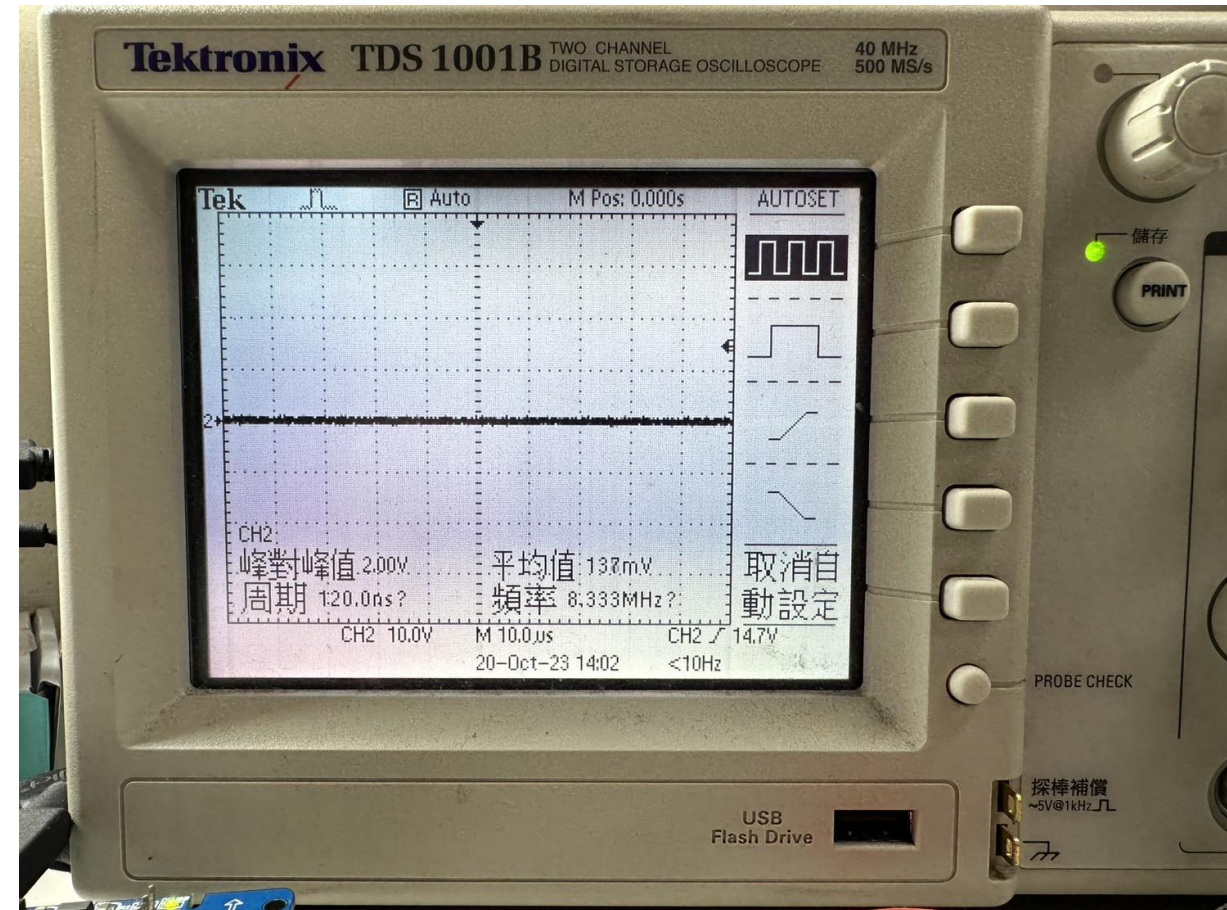
$$\begin{aligned} TM_RELOAD &= CRR = 48000000 / 1 / 50000 = 960 < 65535 \\ T &= 1/f = 1/50000 = 0.0002s = 20\mu s \end{aligned}$$

PS: Since GPTM is a 16-bit up/down counter, if CRR exceeds 16 bits, it cannot be displayed correctly.

Show on the oscilloscope

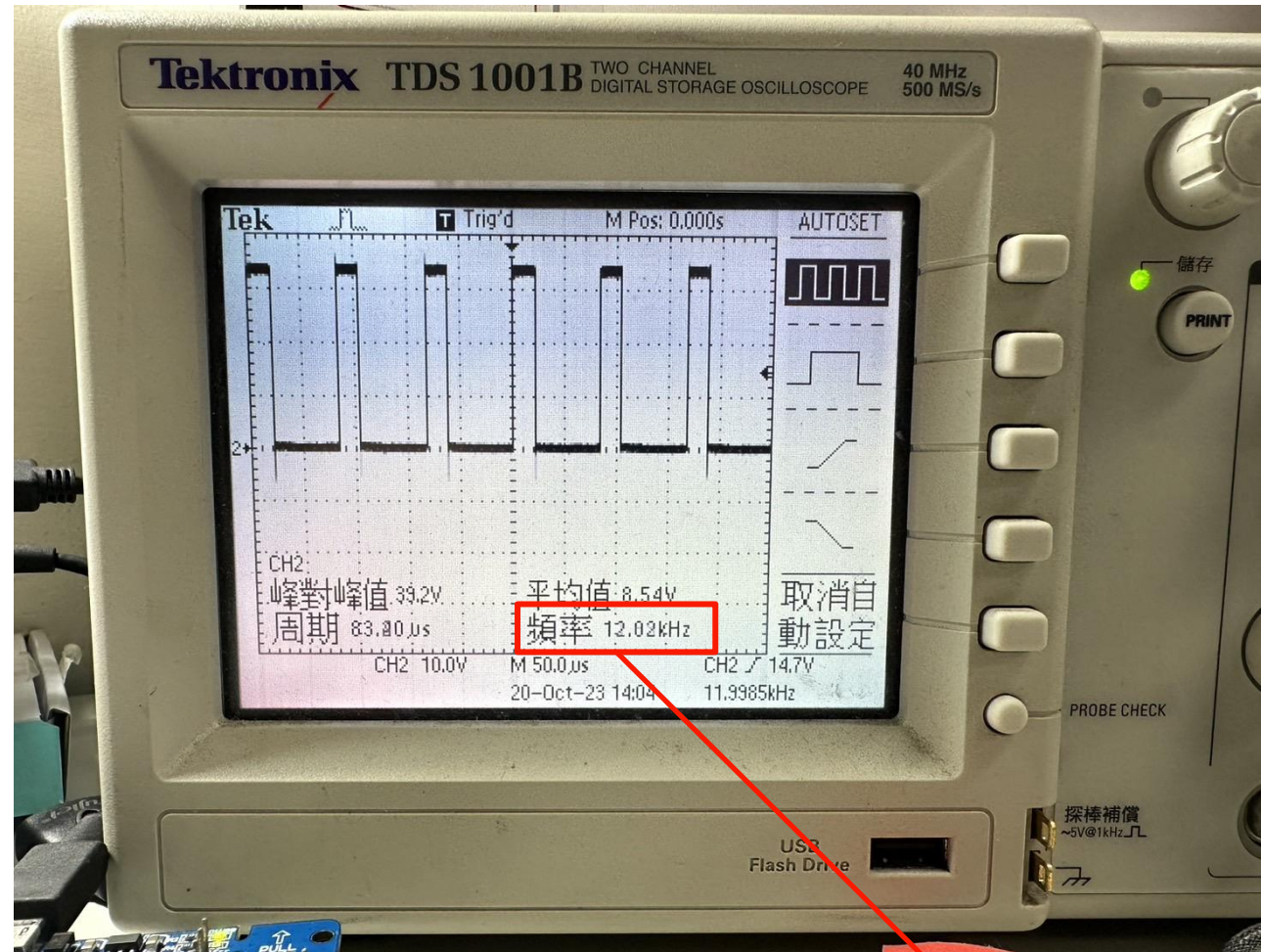


➤ PWM_UpdateDuty(PWM_CH0, PWM_DUTY_50);



➤ PWM_UpdateDuty(PWM_CH0, PWM_DUTY_0);

Show on the oscilloscope



➤ PWM_UpdateDuty(PWM_CH0, PWM_FREQ_12K * 0.25);

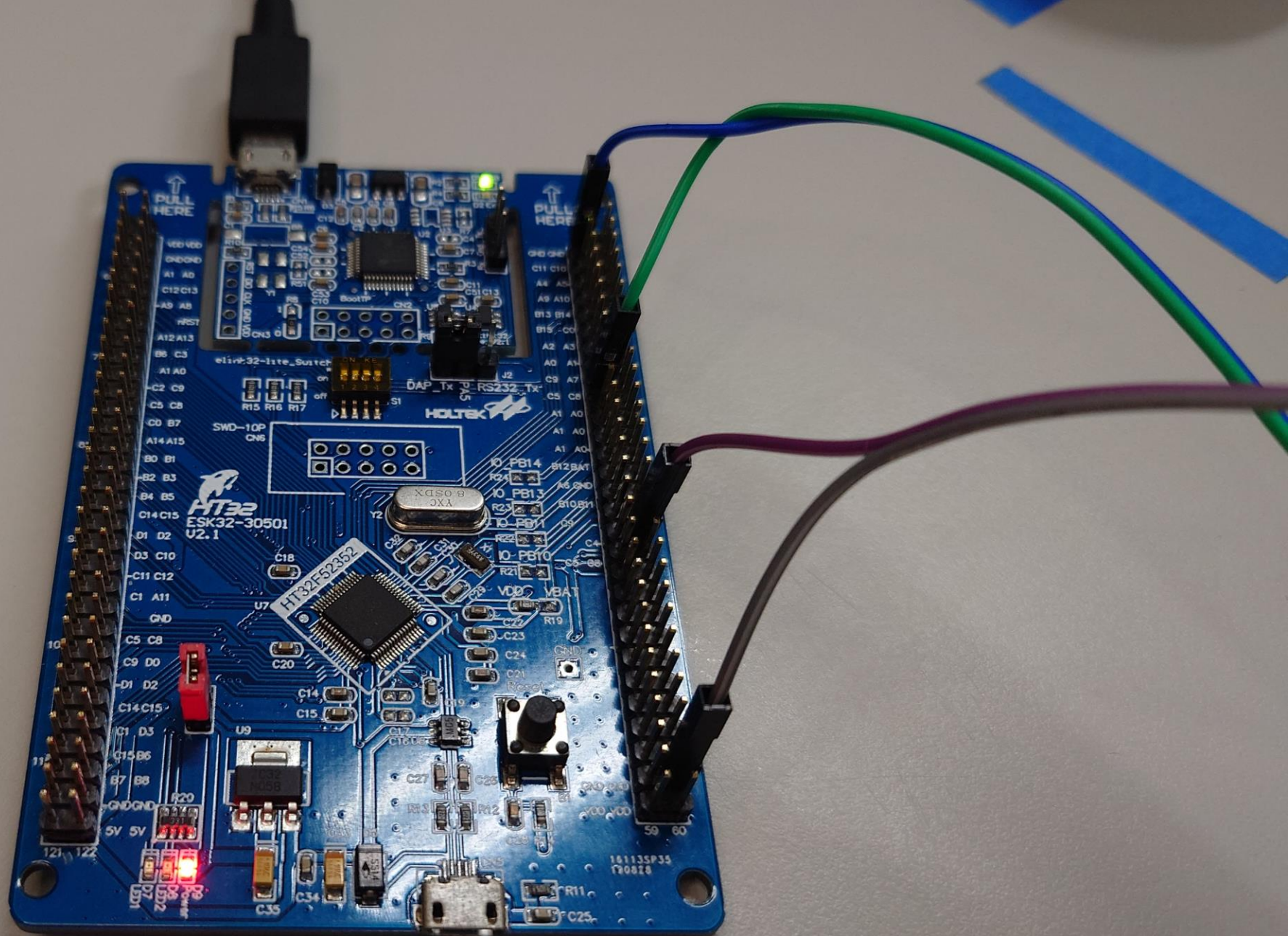
Homework W7-1.

[https://github.com/CYCU-AIoT-System-
Lab/Microcontroller-
Experiment/blob/main/w7/TM-PWM-
Experiment_Steps.md](https://github.com/CYCU-AIoT-System-Lab/Microcontroller-Experiment/blob/main/w7/TM-PWM-Experiment_Steps.md)

Execute the example and show on oscilloscope

- Objective: Observe output signal with oscilloscope and explain the code.
- Hint:
 1. Use key "F12" to locate specified pins and connect them.
 2. Increase the default delay time for result observation.
 3. Connect PWM **CH2** and **CH3** to the oscilloscope.

☆ PS. Please record.



The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large red speech bubble is centered on the page, with the text 'Homework W7-2' written inside in white.

Homework W7-2

Control servo motor (SG90)

- Objective: Modify code and connect **CH2** to oscilloscope, and use **CH3** to control SG90.
- Hint:
 1. Connect **CH2** to oscilloscope.
 2. Connect **CH3** to servo motor SG90, and rotate it to 0, 45, 90, 135, and 180 degrees angle.
 3. Edit code in files: main.c & PWM.h.

☆ PS. Please record.

SG90

Datasheet: <https://reurl.cc/6DX3pV>

Feature:

```
38  /* Settings -----*/
39  #define HTCFG_PWM_TM_PRESCALER          // 1 ~ 65535
40  #define HTCFG_PWM_FREQ_HZ              (50)
41  #define HTCFG_PWM_IDLE_STATE           (0) // 0: 0 duty, 1: 100 duty
42
43  #define HTCFG_PWM_TM_RELOAD             (HTCFG_PWM_TM_PCLK / HTCFG_PWM_TM_PRESCALER / HTCFG_PWM_FREQ_HZ)
44  #if (HTCFG_PWM_TM_RELOAD > 65536)
45  #error "HTCFG_PWM_TM_RELOAD out of range! Should be less than or equal to 65536."
46  #endif
```

【引線接法】

- 棕色：GND
- 紅色：VCC 3~7.2V (建議5V)
- 橙色：控制訊號

其工作原理是：

控制信號由接收機的通道進入信號調製晶片，獲得直流偏置電壓。它內部有一個基準電路，產生週期為20ms，寬度為1.5ms的基準信號，將獲得的直流偏置電壓與電位器的電壓比較，獲得電壓差輸出。最後，電壓差的正負輸出到電機驅動晶片決定電機的正反轉。當電機轉速一定時，通過級聯減速齒輪帶動電位器旋轉，使得電壓差為0，電機停止轉動。當然我們可以不用去瞭解它的具體工作原理，知道它的控制原理就夠了。就象我們使用電晶體一樣，知道可以拿它來做開關管或放大管就行了，至於管內的電子具體怎麼流動是可以完全不用去考慮的。

舵機的控制：

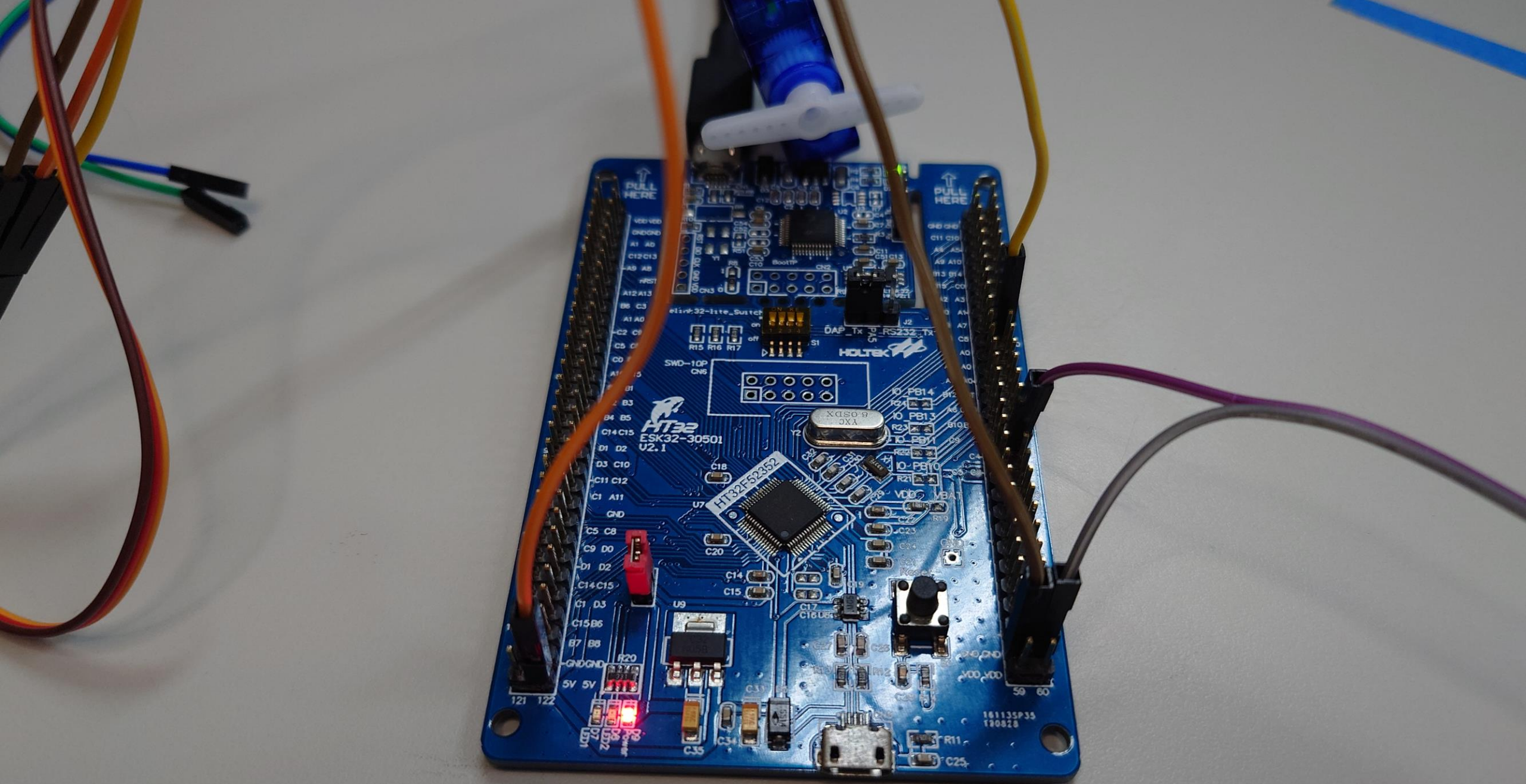
舵機的控制一般需要一個20ms左右的時基脈衝，該脈衝的高電平部分一般為0.5ms~2.5ms範圍內的角度控制脈衝部分。以180度角度伺服為例，那麼對應的控制關係是這樣的：

0.5ms-----	0度；	$0^\circ: 0.5/20=0.025$
1.0ms-----	45度；	$45^\circ: 1.0/20=0.05$
1.5ms-----	90度；	$90^\circ: 1.5/20=0.075$
2.0ms-----	135度；	$135^\circ: 2.0/20=0.1$
2.5ms-----	180度；	$180^\circ: 2.5/20=0.125$

```
#define PWM_DUTY_0          (HTCFG_PWM_TM_RELOAD * 0.025)
#define PWM_DUTY_25        (HTCFG_PWM_TM_RELOAD * )
#define PWM_DUTY_50        (HTCFG_PWM_TM_RELOAD * )
#define PWM_DUTY_75        (HTCFG_PWM_TM_RELOAD * )
#define PWM_DUTY_100       (HTCFG_PWM_TM_RELOAD * )
```

這只是一種參考數值，具體的參數，請參見舵機的技术參數。

$$f = 1/T = 1/20\text{ms} = 50\text{Hz} (= \text{HTCFG_PWM_FREQ_HZ})$$
$$\text{TM_RELOAD} = \text{CRR} = 48000000/50/? < 65535$$



The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. A large red speech bubble is centered on the page, pointing downwards.

Homework W7-3

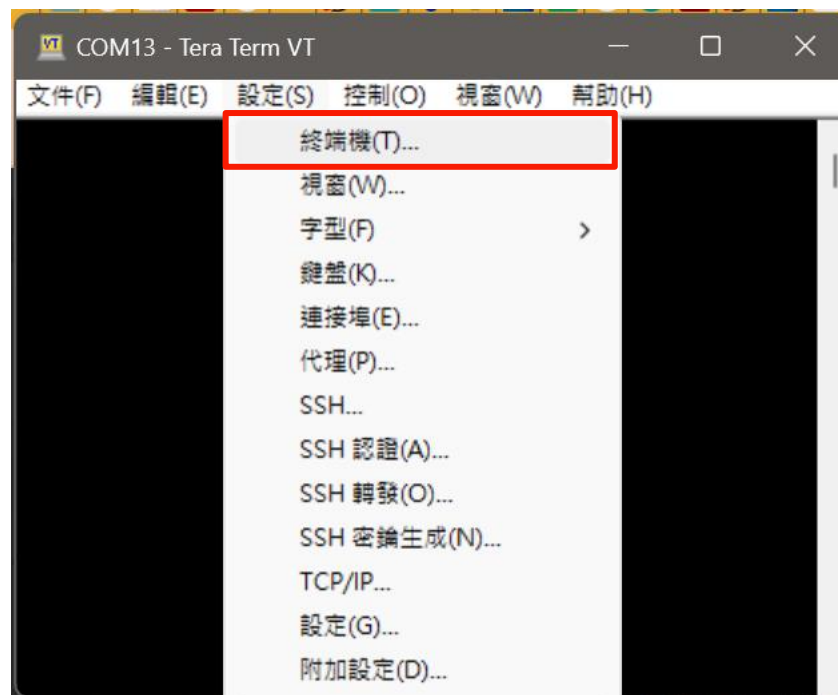
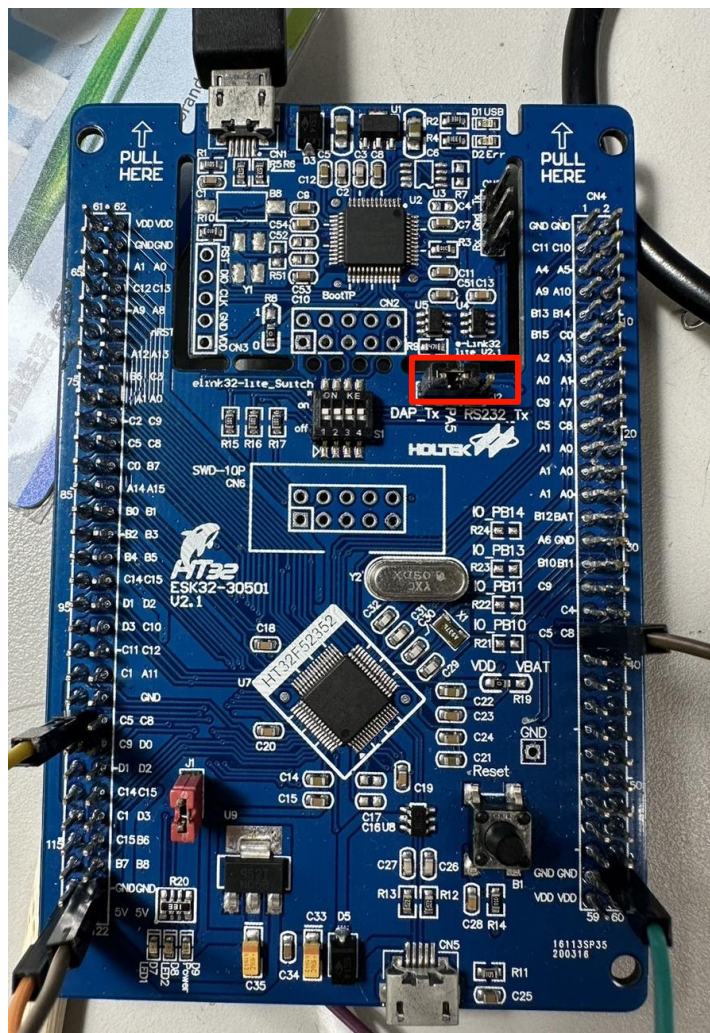
Control the motor angle by input

- Objective: Use laptop keyboard (Tera Term) input Duty Cycle to control SG90.
- Hint:
 1. Connect **CH2** to oscilloscope.
 2. Connect **CH3** to servo motor SG90, and use user input to rotate it to 0, 45, 90, 135, and 180 degrees angle.
 3. Edit code.

☆ PS. Please record.

Step1:

Check if the jumper is connected to DAP_TX and PA5.



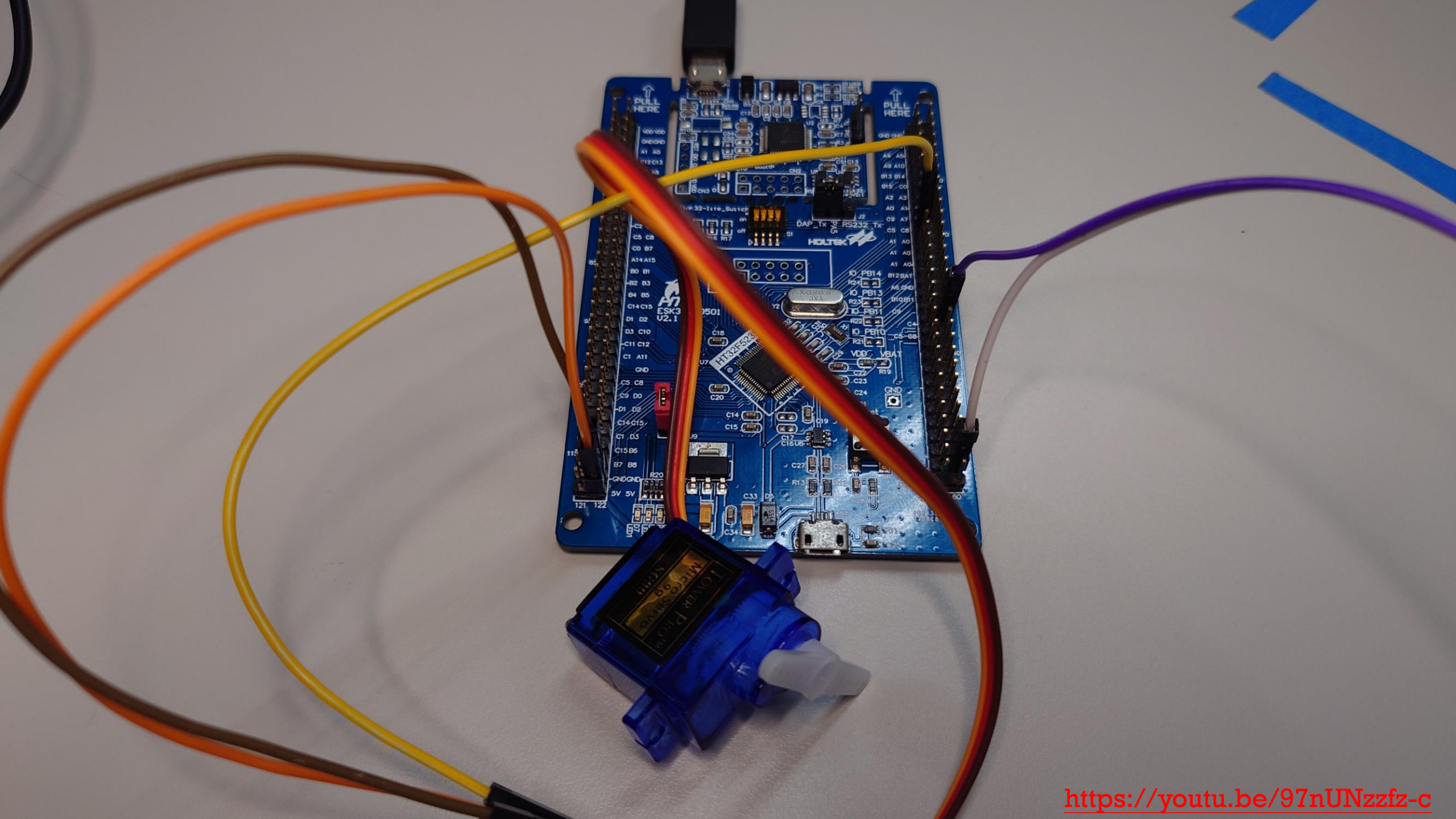
Enable local echo in the terminal settings.

Step2:

- Add code to the appropriate location and annotate or remove unnecessary code.
- PS. input is floating number

```
float input;  
RETARGET_Configuration();  
PWM_Init();
```

```
while (1)  
{  
    printf("Please input ...\n\r");  
    scanf("%f", &input);  
    printf("the input is %f\r\n", input);  
  
    PWM_UpdateDuty(PWM_CH2, HTCFG_PWM_TM_RELOAD * input);  
    PWM_UpdateDuty(PWM_CH3, HTCFG_PWM_TM_RELOAD * input);  
    PWM_Cmd(ENABLE);  
  
    Delay(20000000);  
}
```

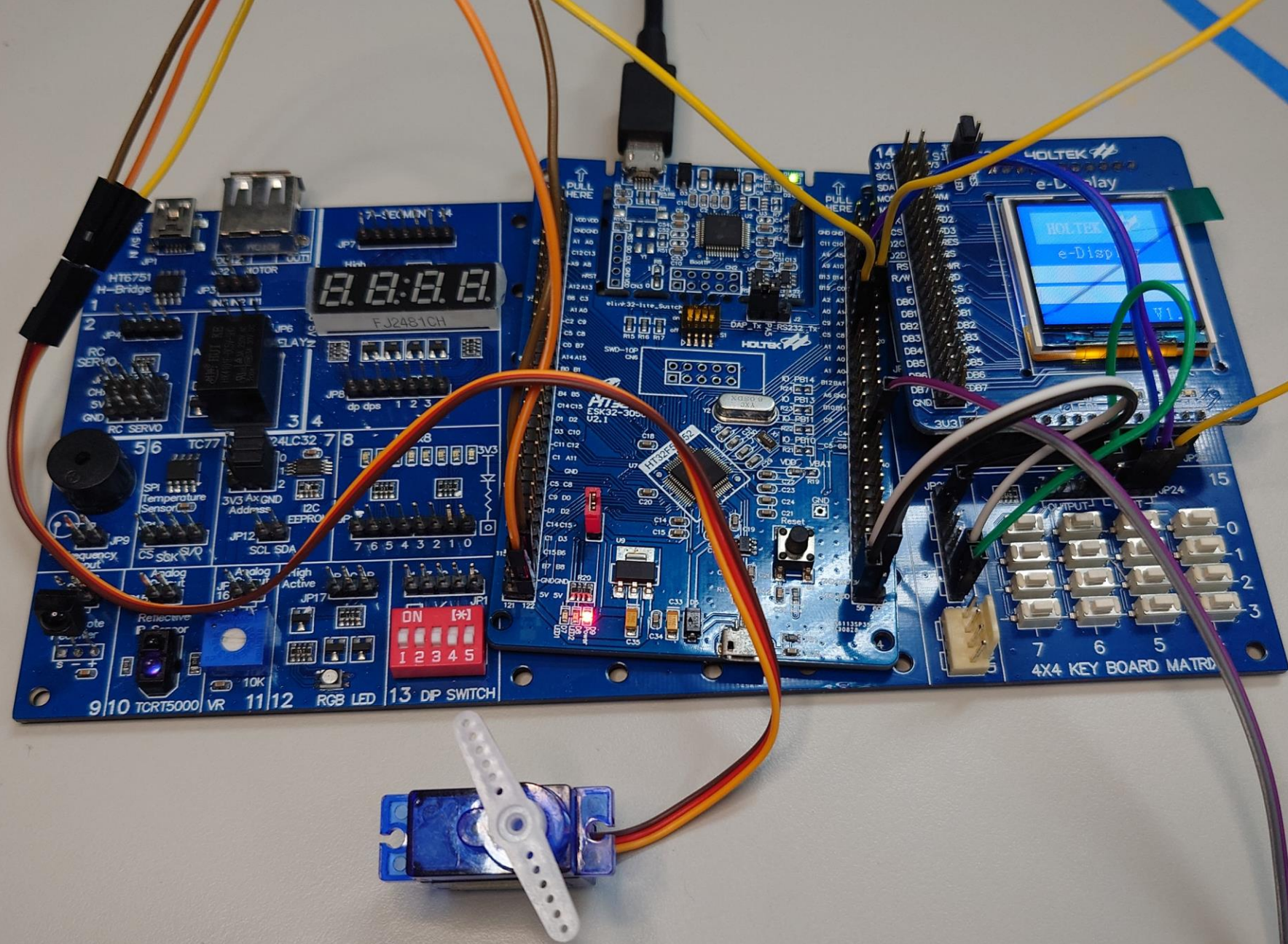
The background features a series of concentric circles in light gray, some solid and some dashed, creating a ripple effect. In the center, there is a red speech bubble with a white outline and a small tail pointing downwards.

Homework W7-4 Bonus Question

Bonus: Use buttons to control the motor.

- Objective: Use three buttons to control SG90 to rotate to 0, 90, and 180 degrees.
- Hint:
 1. Connect **CH2** to oscilloscope.
 2. Connect **CH3** to servo motor SG90.
 3. Use pin **B1**, **B2**, **B3** to take GPIO input.
 4. Refer to week 2 example GPIO/InputOutput.

☆ PS. Please record and explain the code.





Class
Dismissed