תיעוד 24.5.21:

אחרי ששינינו את מימוש הUSART כך שנדפיס כל שורה פעם אחת לTeraTerm,

בדיקות שנרצה לעשות:

1. נבדוק את הפונקציונליות של הטיימרים והalert handler בלי הevent queue
2. לרוקן את הפסיקה של טיימר 3, ומפעילות את טיימר 3 ואת alert handler (בלי הevent queue) כך נבדוק אם יש בעיה בפונקציונליות של ה set וdone או בהתנגשות הטיימרים.
3. להשתמש בTimer done timer3 וב3 set timer אבל בdone
4. לא להשתמש בפונקציונליות של set timout timer3 וtimeout done timer3 אלא ב 3 timeout רגיל.

תחילה נשלח 10 התראות ונבדוק כמה זמן לוקח להודעה ממוצעת להישלח+ נבדוק איפה כל התראה נתקעת+ כמה ניסיונות לוקח לה להישלח:

שימוש בAlert handler 1 (נותן 3 ניסיונות לrecord alert):

main שלנו נראה כך:

```
Enable_sensor();
While(1)
{
	alert_Handler();
	delay_with_timer2(5);
	write_usart2((uint8_t*)("\r\n_____________\r\n"));
}
```

בדיקה 1:
לקח 2 ניסיונות, בניסיון הראשון נתקע בחיבור ל firebase, בשורה:
AT+CIPSTART="SSL"," firebase URL",443
בניסיון השני זה עבד.
זמן סכ"ה: 1:37 דקות

בדיקה 2:
לקח ניסיון אחד, לא נתקע בכלל
זמן סכ"ה: 00:10 דקות

בדיקה 3:
לקח 2 ניסיונות, בניסיון הראשון נתקע בחיבור ל firebase, בשורה:
AT+CIPSTART="SSL"," firebase URL",443
ב1:05 דקות התחיל הניסיון השני, והוא הצליח תוך 10 שניות.
זמן סכ"ה: 1:10 דקות

בדיקה 4:
לקח 2 ניסיונות, בניסיון הראשון נתקע בחיבור ל firebase, בשורה:
AT+CIPSTART="SSL"," firebase URL",443
ב1:08 דקות התחיל הניסיון השני, והוא הצליח תוך 7 שניות.
זמן סכ"ה: 1:15 דקות

ניסיון 5:

לקח 2 ניסיונות, בניסיון הראשון נתקע בחיבור ל firebase, בשורה:
AT+CIPSTART="SSL"," firebase URL",443
ב1:07 דקות התחיל הניסיון השני, והוא הצליח תוך 6 שניות.
זמן סכ"ה: 1:13 דקות
----------עד כה לחצנו על כפתור reset (כפתור יום הדין) בין בדיקה לבדיקה.

מהניסיון הבא לא נלחץ על reset:

ניסיון 6:
לקח 3 ניסיונות, בניסיון הראשון נתקע בחיבור ל firebase, בשורה:
AT+CIPSTART="SSL"," firebase URL",443
ב1:05 דקות התחיל הניסיון השני, שנתקע בחיבור ל firebase, בשורה:
AT+CIPSTART="SSL"," firebase URL",443
ב2:10 דקות התחיל הניסיון השלישי,והוא הצליח תוך 12 שניות.
זמן סכ"ה: 2:22 דקות

ניסיון 7:
לקח 3 ניסיונות, בניסיון הראשון נתקע בחיבור ל firebase, בשורה:
AT+CIPSTART="SSL"," firebase URL",443
ב1:50 דקות התחיל הניסיון השני, שנתקע בחיבור ל firebase, בשורה:
AT+CIPSTART="SSL"," firebase URL",443
ב2:50 היה Buffer over flow בגלל הreset
התחיל הניסיון השלישי, וקיבלנו מענה ב3:07 דקות
זמן סכ"ה: 3:07 דקות

ניסיון 8:
לקח ניסיון 1 לקח 6 שניות.

ניסיון 9:
לקח ניסיון 1, נתקענו בשורה של ההתחברות לרשת:
AT+CWJAP=”*****”
זמן סכ"ה: 1:05 דקות

ניסיון 10:
לקח ניסיון 1, נתקענו בRST במשך דקה,
ואז תוך 5 שניות ההתראה נשלחה.
זמן סכ"ה: 1:05 דקות

*** שמנו לב שבין סיום שליחה אחד לניסיון שליחה הבא לוקח יותר מ5 שניות (למרות שבmain שמנו דילוי של 5 שניות בין alert_Handler אחד לשני, היה מצב של 50 שניות הפרש בין השליחות)

***אנחנו תוהות האם הסיבה היא הAT+RST

<u>נבדוק 10 ניסיונות נוספים בלי הAT+RST:</u>

ניסיון	שליחה 1	שליחה 2	שליחה 3	הצלחה/כשלון
0	01:05	02:07	03:27	הצלחה

	CIPSTART	CWJAP		
1	00:20			הצלחה
	CWMODE			
2	00:07			הצלחה
3	1:00	1:07		הצלחה
	CIPSTART			
4	00:06			הצלחה
5	00:07			הצלחה
6	01:00	01:11		הצלחה
	CIPSTART	חזר פעמיים על CWJAP		
7	01:03			הצלחה
	CWJAP (היה CWJAP פעמיים)			
8	01:00			הצלחה
	CWJAP			
9	01:50			הצלחה
	CWJAP (היה CWJAP פעמיים) ב01:09 נתקענו ב CLOSE			
10	02:00	04:00	07:00	נכשלה
	CWJAP	CWJAP ב03:05 נתקענו ב CIPSTART	CWJAP ב05:00 נתקענו ב CIPSTART	

--> גם אחרי שנטרלנו את הAT+RST שמנו לב שבין סיום שליחה אחד לניסיון שליחה הבא לוקח יותר מהדיליי של טיימר 2 שהגדרנו בmain (לא בדקנו בדיוק, אבל במקום 10 שניות לקח בערך 50 שניות)

-->נראה שהסיבה להשהיה המוגזמת בין ניסיון לניסיון לא הייתה בגלל הAT+RST

***שמנו לב שtimer4 חיכה יותר זמן ממה שהגדרנו בtimeout (לא בדקנו בדיוק, אבל במקום 10 שניות לקח בערך 50 שניות)

***נתנו לCWJAP timeout של 6 שניות, ו2 ניסיונות. וממה שראינו, בפועל אבן ניתנו לCWJAP 2 ניסיונות, אבל כל ניסיון היה הרבה יותר מ6 שניות(~סדר גודל של 30-50 שניות).

<u>נבדוק מה זמן הדיליי בין סיום הAlert_Handler להתחלת הAlert_Handler הבאה:</u>

(ציפייה: 10 שניות בין Alert_Handlerאחד לשנייה)

<u>מטרת הטסט</u>: לראות אם ההשהיה הלא רצויה בין ניסיון לניסיון מתרחשת בAlert_Handler או בטיימר 2:

<u>הmain נראה כך</u>:
(לא השתמשנו בAT+RST)

```
Enable_sensor();
```

```
While(1)
{
	alert_Handler();
	write_usart2((uint8_t*)("\r\n***************\r\n"));
	delay_with_timer2(10);
	write_usart2((uint8_t*)("\r\n***************\r\n"));
}
```

ניסיון	טיימר 4 CWJAP (אמור להיות 00:06)	טיימר 4 CIPSTART (אמור להיות 00:30)	השהיה של טיימר 2 (אמור להיות 00:10)
1	לא בדקנו	לא בדקנו	00:10
2 (השליחה הזו הייתה נוראית במיוחד, לא בדקנו, אבל נראה שטיימר 4 עשה בעיות)	לא בדקנו	לא בדקנו	00:50
3	00:50	לא בדקנו	00:57

-->אפשר לראות שיש לנו בעיה עם הטיימרים. לא ברור איפה ולמה.

***הערה: צריך לבדוק את האפשרות שאנחנו בטעות נתנו לפסיקות מסוימות קדימות על פני פסיקות אחרות. (למרות שלא השתמשנו בחיישן, הפסיקות שלו היו מאופשרות, בדיליי של טיימר 2 הפסיקות שלו היו מאופשרות, ובדיליי של טיימר 4 הפסיקות שלו היו מאופשרות, והפסיקות של הטיימרים לא התרחשו בו זמנית. והפסיקה של Rx הייתה מאופשרת לכל אורך הדרך)

-->שללנו את האפשרות שההשהיה הארוכה בין הניסיונות היא בגלל הalert_Handler (לא כולל הפסיקה של טיימר 4)

<u>בדיקה הבאה:</u>
השארנו את כל הפסיקות והinitים כמו בבדיקה הקודמת, אבל הפעם לא נשלח התראות (לא נקרא בmain ל;alert_Handler) ולכן לא יהיה שימוש בטיימר 4.

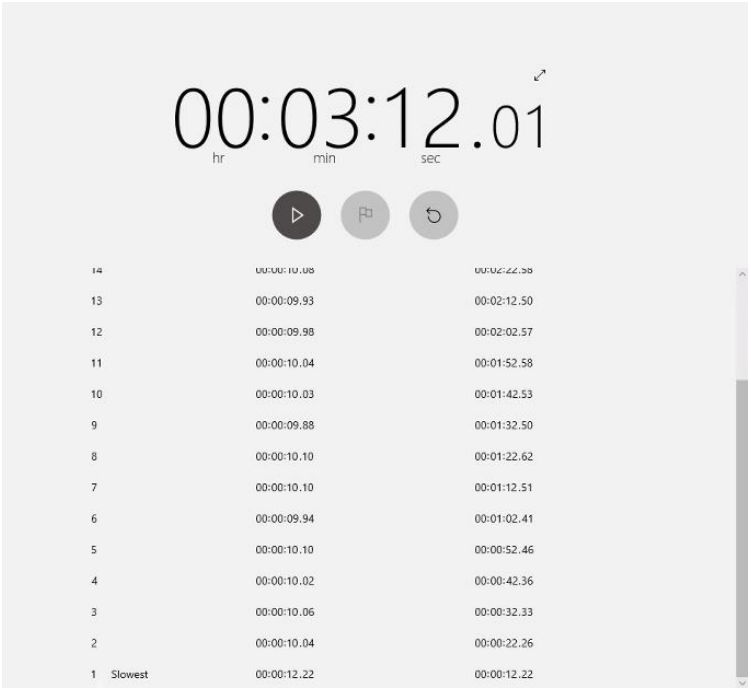
<u>מטרת הטסט</u>: לבדוק אם טיימר 2 יחכה זמן גדול משכתבנו בדיליי (במקרה שלנו יותר 10 שניות. [נקווה שלא])

נשים בהערה את ;(Enable_sensor כך שה main שלנו יראה כך:

הmain שלנו כעת:

```
While(1)
{
	delay_with_timer2(10);
	write_usart2((uint8_t*)("\r\n***************\r\n"));
	delay_with_timer2(10);
	write_usart2((uint8_t*)("\r\n$$$$$$$$$$$$$\r\n"));

}
```



10<-- שניות בין הדפסה להדפסה

נחזיר את הEnable_sensor כך שהmain שלנו יראה בעת כך:

```
Enable_sensor();
While(1)
{
	delay_with_timer2(10);
	write_usart2((uint8_t*)("\r\n***************\r\n"));
	delay_with_timer2(10);
	write_usart2((uint8_t*)("\r\n$$$$$$$$$$$$$\r\n"));

}
```

ניסיון בלי תזוזה:

ניסיון עם תזוזה:

-->בין הדפסה להדפסה עברה ~דקה.

נקודה למחשבה: זה הtimeout שקבענו לטיימר 3 בפסיקה של החיישן

ניסיון בלי תזוזה בכלל! (החיישן בתוך קופסא):

-->הפרש של ~10 שניות בין הדפסה להדפסה

- עד עכשיו תוכן הפסיקה של החיישן היה:

```
void EXTI4_IRQHandler(void)
{
	if(timeout_done_timer3())
	{
		EXTI->PR |= 0x00000010;
		add_event(alert_Handler);
		set_timeout_timer3(60); //60 seconds = 1 minute
	}
}
```

-->אנחנו כן משתמשות בפונקציונליות הset וdone של טיימר 3.

נשים את הפונקציונליות הזו בהערה, כך שהפסקה של החיישן תראה כך:

```
void EXTI4_IRQHandler(void)
{
	EXTI->PR |= 0x00000010;
	add_event(alert_Handler);
}
```

ניסיון עם תזוזה: (המון תזוזה)

10~<-- שניות בין הדפסה להדפסה

בדיקה הבאה: ניצור פונקציונליות שמשתמשת ב2 טיימרים בו זמנית

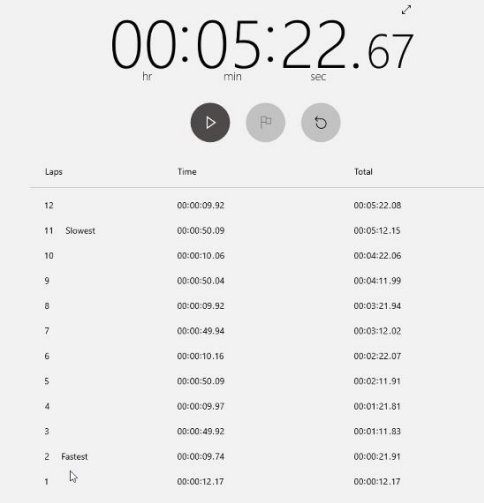
(נכניס רק את הinitים ההכרחיים.)

main יראה כך:

```
Int void main(void){
USART2_init(); // for debugging
                // init_queue();
                // init_sensor_with_interrupt(); // sensor interrupts are not inabled
                init_timer2(); // for monitoring switch state.
                init_timer3(); // for sensor delay
                // init_timer4(); // for ESP8266 timeout
                USART1_init(); // for ESP8266
                //init_i2c1();
                write_usart2((uint8_t*)("\r\n_________________\r\n"));//For test
                // enable_sensor(); // FOR TESTING
                while(1)
                {

                        if(timeout_done_timer3())
                        {
                                set_timeout_timer3(60); //60 seconds = 1 minute
                                                        delay_with_timer2(10);
                                write_usart2((uint8_t*)("\r\n#################\r\n"));
                                delay_with_timer2(10);
                                write_usart2((uint8_t*)("\r\n~~~~~~~~~~~~~~~~\r\n"));
                        }
}
```

נבדוק זמנים. נרצה הדפסות במרווחים של 10,10,40 שניות



בתוצאה ראינו מרווחי הדפסות של 10,10,50,10,50,10,50,10,50... הדבר הגיוני בהתחשב בmain.

<u>בדיקה באה: הרבה תזוזה+אפשור פסיקת חיישן</u>

<u>מטרה של הטסט</u>: לבדוק האם כאשר יש הרבה תזוזה עובדרת דקה וחצי בין הדפסה להדפסה

פסיקת החיישן תראה כך:

```
void EXTI4_IRQHandler(void)
{
	if(timeout_done_timer3())
	{
		EXTI->PR |= 0x00000010;
		add_event(alert_Handler);
		set_timeout_timer3(90);
	}
}
```

המain יראה כך:

```
Int void main(void){
USART2_init(); // for debugging
 init_queue();
init_sensor_with_interrupt(); // sensor interrupts are not inabled
init_timer2(); // for monitoring switch state.
init_timer3(); // for sensor delay
init_timer4(); // for ESP8266 timeout
USART1_init(); // for ESP8266
write_usart2((uint8_t*)("\r\n_____\r\n"));//For test
enable_sensor(); // FOR TESTING
while(1)
{
		delay_with_timer2(10);
		write_usart2((uint8_t*)("\r\n################\r\n"));
		delay_with_timer2(10);
		write_usart2((uint8_t*)("\r\n????????????????\r\n"));}
 }
```

תוצאה:

-->הדפסות בהפרשים של 90 שניות! כמו מה שקבענו בפסיקה של החיישן.

כמות התזוזה לא היה גדולה למשך כל הבדיקה, מספיק היה לזוז מעט.

היינו מצפות שההדפסות יתרחשו בmain כל 10 שניות (כמו שכתבנו בדיליי של טיימר 2 בmain), בלי קשר לטיימר 3 בפסיקת החיישן. בפועל מי שקבע את קצב ההדפסות הוא טיימר 3 בפסיקת החיישן.

בדיקה נוספת: בinit של טיימר 2 ו3 נשים בהערה את השורות:

```
NVIC_SetPriorityGrouping(7); //This should disable interrupt nesting(priority wont be not allowed)//->MABY IT'S THE DEFAULT

NVIC_SetPriority(TIM2_IRQn,0); //set all interrupt priority to zero //
                                so that no preemption occurs.//->MABY IT'S THE DEFAULT
```

נבדוק מה תוצאות הטסט זהה לקודם שעשינו (רק עם timeout של 60 שניות בפסיקה של החיישן ולא 90 שניות):

```
void EXTI4_IRQHandler(void)
{
        if(timeout_done_timer3())
        {
                EXTI->PR |= 0x00000010;
                add_event(alert_Handler);
                set_timeout_timer3(60);
        }
}
```

main יראה כך:

```
Int void main(void){
USART2_init(); // for debugging
 init_queue();
init_sensor_with_interrupt(); // sensor interrupts are not inabled
init_timer2(); // for monitoring switch state.
init_timer3(); // for sensor delay
init_timer4(); // for ESP8266 timeout
USART1_init(); // for ESP8266
write_usart2((uint8_t*)("\r\n_____\r\n"));//For test
enable_sensor(); // FOR TESTING
while(1)
{
                delay_with_timer2(10);
                write_usart2((uint8_t*)("\r\n################\r\n"));
                delay_with_timer2(10);
                write_usart2((uint8_t*)("\r\n????????????????\r\n"));}
 }
```

תוצאות: לא עזר, הזמן בין הדפסה להדפסה היה הזמן שהגדרנו בפסיקת החיישן בtimer3.

הוצאנו את השורות האלו מהערה.

בדיקה נוספת:

נגדיר את הזמן בtimeout של טיימר 3 בפסיקת החיישן ל10 שניות, בעוד שאת הדיליי בmain נגידר 60 שניות:

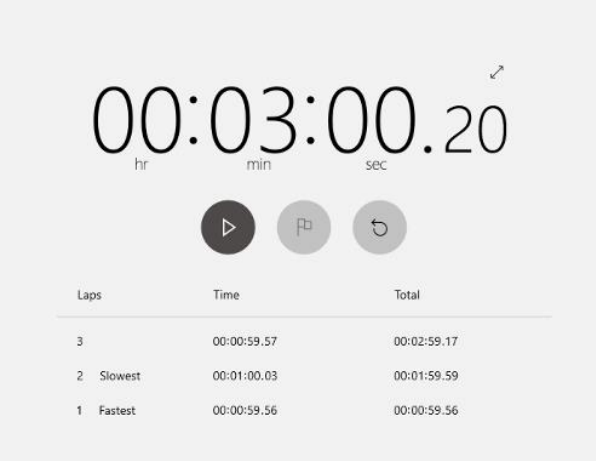
אם ההשארה שלנו שטיימר 3 בפסיקת החיישן דורס את טיימר 2 בmain, בין הדפסה להדפסה יעברו 10 שניות

```
void EXTI4_IRQHandler(void)
{
	if(timeout_done_timer3())
	{
		EXTI->PR |= 0x00000010;
		add_event(alert_Handler);
		set_timeout_timer3(10);
	}
}
```

main יראה כך:

```
Int void main(void){
USART2_init(); // for debugging
 init_queue();
init_sensor_with_interrupt(); // sensor interrupts are not inabled
init_timer2(); // for monitoring switch state.
init_timer3(); // for sensor delay
init_timer4(); // for ESP8266 timeout
USART1_init(); // for ESP8266
write_usart2((uint8_t*)("\r\n_____\r\n"));//For test
enable_sensor(); // FOR TESTING
while(1)
{
		delay_with_timer2(60);
		write_usart2((uint8_t*)("\r\n################\r\n"));
		delay_with_timer2(60);
		write_usart2((uint8_t*)("\r\n??????????????\r\n"));}
 }
```

תוצאה:



--->הפרש הזמן בין ההדפסות הוא 60 שניות! ולא 10!

אולי אנחנו מצביעות על אותן הכתובות כשאנחנו משתמשים בטיימר 2,3,4? ואז אנחנו דורסות כל פעם את תוכן הטיימר?

נגדיר עוד 2 structים:

```
typedef struct timer2{

        uint32_t countTicks;
        uint32_t targetTick;

}TIMER2;

typedef struct timer3{

        uint32_t countTicks;
        uint32_t targetTick;

}TIMER3;

typedef struct timer{

        uint32_t countTicks;
        uint32_t targetTick;

{TIMER;
```

ונגדיר את הטיימרים עם structים שונים.

ראינו שזה לא משפיע, לכן מספיק struct אחד.

<u>בדיקה הבאה:</u>

נכניס הדפסות בפסיקות של טיימר 2 ו 3 (מחוץ לif שבפסיקה), ונוודא שאנחנו נכנסות לשתי הפסיקות הללו.

ה main נראה כך:

```
enable_sensor(); // FOR TESTING while(1) {        delay_with_timer2(60);
        delay_with_timer2(10);
        write_usart2((uint8_t*)("\r\n##################\r\n"));
        delay_with_timer2(10);
        write_usart2((uint8_t*)("\r\n??????????????????\r\n"));}
 }
```

בהדפסות ראינו שהודפס מה שכתבנו בפסיקה של טיימר 2 וגם מה שכתבנו בטיימר 3.

-->בלומר אנחנו כן נכנסות לפסיקות.

<u>בדיקה הבאה:</u>

נכניס הדפסות בפסיקות של טיימר 2 ו 3 (בתוך הif שבפסיקה), ונוודא שאנחנו נכנסות לשתי הפסיקות הללו.

בהדפסות ראינו שהודפס מה שכתבנו בפסיקה של טיימר 2 וגם מה שכתבנו בטיימר 3.

-->בלומר אנחנו כן נכנסות לפסיקות.

<u>בדיקה הבאה:</u>

בנוסף להדפסות בפסיקות של טיימרים 2,3 נדפיס גם בפסיקה של החיישן.

שמנו לב לתבניתיות, שבמידה ואנחנו נכנסות לפסיקת החיישן, מה שהדפסנו בmain יכתב אחרי שאנחנו נכנסות

<u>בדיקה הבאה:</u>

אנחנו נכנסות לפסיקה של החיישן הרבה מאוד פעמים! הסיבה לכך היא שאנחנו מכבות את דגל הפסיקה של החיישן רק אחרי שהtimeout3 כבר סיים (כשהוא done)! כלומר רק אחרי 60 שניות או 90 שניות (תלוי מה הגדרנו בset_timeout_timer!)

נעביר את כיבוי הדגל למחוץ לif.

מצב קודם:

```
if(timeout_done_timer3())
{
	write_usart2((uint8_t*)("\r\nSENSOR_IF\r\n"));
	EXTI->PR |= 0x00000010; //reset flag by writing 1 to bit 4 (reference manual 10.3.6)
	add_event(alert_Handler);
	set_timeout_timer3(60); //60 seconds = 1 minute

}
```

נעביר למצב הבא:

```
if(timeout_done_timer3())
{
	write_usart2((uint8_t*)("\r\nSENSOR_IF\r\n"));
	add_event(alert_Handler);
	set_timeout_timer3(60); //60 seconds = 1 minute
}
	EXTI->PR |= 0x00000010; //reset flag by writing 1 to bit 4 (reference manual 10.3.6)
```

<u>תוצאת ההרצה:</u>

בין הדפסה להדפסה עברו 10 שניות!!!!

אחרי הגילוי המרעיש נבנה את הmain הבא: (החיישן מחובר, אבל אנחנו לא ממשות עם הevent quequeה)

```
Int void main(void){
USART2_init(); // for debugging
 init_queue();
init_sensor_with_interrupt(); // sensor interrupts are not inabled
init_timer2(); // for monitoring switch state.
init_timer3(); // for sensor delay
init_timer4(); // for ESP8266 timeout
USART1_init(); // for ESP8266
write_usart2((uint8_t*)("\r\n_____\r\n"));//For test
enable_sensor(); // FOR TESTING
while(1)
{
      alert_Handler();
      delay_with_timer2(10);
  }
}
```

<u>תוצאות ההרצה:</u>

ניתן לראות שבין שליחה אחת לשניה יש דיליי של 10 שניות (כפי שקבענו בדיליי עם טיימר 2 בmain)!

בדיקת הtimeout של CIPSTART:

אנחנו אכן רואות שאנחנו מחכות כ30 שניות בין שליחת CIPSTART אחת לשניה (כפי שקבענו בtimeout)

<u>צריך</u>:
לתקן פונקציונליות של control state.