Perifèrics
Practica 1

Angel Farre – angel.farre
Enric Gutiérrez – enric.gutierrez

INDEX

1. Descripció de l'arquitectura del programa

Inicialment les configuracions dels diferents mòduls emprats a la practica son cridats en el main del programa. Aquests mòduls son el **timer2**, el **timer3**, el **timer4** i el **EXTI**, els quals explicarem amb mes detall en els punts Diagrama de mòduls i Descripció dels mòduls del controlador.

El timer 2 l'hem emprat per comptar temps i poder així determinar si les pulsacions que feia el usuari amb el boto eren de categoria curta (menor a dos segons) o be de categoria llarga (majors a dos segons).

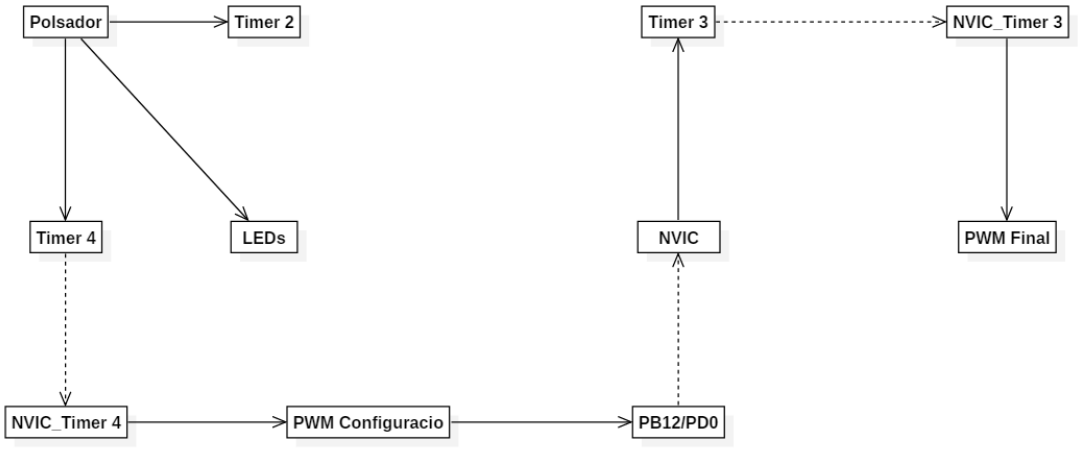
Aquesta informació ens permetia saber quin dels dos PWM possibles a configurar per l'usuari havíem de modificar.

Per al primer PWM (de 10ms), s'ha fet us del timer 4. Aquest timer es de tipus General-Purpose i ens permet configurar-lo de diverses maneres. Per facilitar la realització del PWM, s'ha configurat en mode PWM (`TIM_OCMode_PWM1`). D'aquesta forma simplement s'ha d'anar modificant el paràmetre pulse per a modificar el duty cicle.

De forma anàloga al timer 4, hem configurat el timer 3 per a produir el PWM de sortida a 100us, el qual dependrà directament del valor que disposi el PWM del timer 4 en el seu canal corresponent.

2. Diagrama de mòduls

En la Il·lustració 1 – Diagrama general dels mòduls emprats en la pràcticasegüent, es pot observar les relacions existents entre els diferents mòduls. Addicionalment, es mostra a continuació de la imatge quina es la configuració de cada mòdul per a esclarir el seu funcionament:



Il·lustració 1 – Diagrama general dels mòduls emprats en la pràctica

- Timer 2: Free running timer per controlar el temps que l'usuari prem el boto.
    - Configuració:

```
TIM_TimeBaseInitTypeDef SetupTimer;
/* Enable timer 2, using the Reset and Clock Control register */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
SetupTimer.TIM_Prescaler = 0x0000;
SetupTimer.TIM_CounterMode = TIM_CounterMode_Up;
SetupTimer.TIM_Period = 0xFFFFFFFF;
SetupTimer.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseInit(TIM2, &SetupTimer);
TIM_Cmd(TIM2, ENABLE); /* start counting by enabling CEN in CR1 */
```

- Timer 3: Timer configurat en mode PWM.
    - Configuració Timer:

```
// Timer initialization struct
TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;

TIM_TimeBaseInitStruct.TIM_Prescaler = 1;
TIM_TimeBaseInitStruct.TIM_Period = 1450;
TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;

// Initialize TIM3
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStruct);
```

- Configuració mode PWM:

```
// PWM initialization struct
TIM_OCInitTypeDef TIM_OCInitStruct;

// Common PWM settings
TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;

TIM_OCInitStruct.TIM_Pulse = 660;

TIM_OC1Init(TIM3, &TIM_OCInitStruct); //PA6
TIM_OC2Init(TIM3, &TIM_OCInitStruct); //PA7

TIM_OC1PreloadConfig(TIM3, TIM_OCPreload_Enable);
TIM_OC2PreloadConfig(TIM3, TIM_OCPreload_Enable);
```

- Timer 4
  - Configuració Timer:

```
// Timer initialization struct
TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;

TIM_TimeBaseInitStruct.TIM_Prescaler = 33;
TIM_TimeBaseInitStruct.TIM_Period = 8479;
TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;

// Initialize TIM4
TIM_TimeBaseInit(TIM4, &TIM_TimeBaseInitStruct);
```

  - Configuració mode PWM:

```
// PWM initialization struct
TIM_OCInitTypeDef TIM_OCInitStruct;

// Common PWM settings
TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;

TIM_OCInitStruct.TIM_Pulse = 1247;

TIM_OC1Init(TIM4, &TIM_OCInitStruct);
TIM_OC2Init(TIM4, &TIM_OCInitStruct);
TIM_OC3Init(TIM4, &TIM_OCInitStruct);// PD14
TIM_OC4Init(TIM4, &TIM_OCInitStruct);// PD13

TIM_OC1PreloadConfig(TIM4, TIM_OCPreload_Enable);
TIM_OC2PreloadConfig(TIM4, TIM_OCPreload_Enable);
TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);
TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);
```

- PD0:
    - Configuració EXTI:

```
/* PD0 is connected to EXTI_Line0 */
EXTI_InitStruct.EXTI_Line = EXTI_Line0;
/* Enable interrupt */
EXTI_InitStruct.EXTI_LineCmd = ENABLE;
/* Interrupt mode */
EXTI_InitStruct.EXTI_Mode = EXTI_Mode_Interrupt;
/* Triggers on rising and falling edge */
EXTI_InitStruct.EXTI_Trigger = EXTI_Trigger_Rising_Falling;
/* Add to EXTI */
EXTI_Init(&EXTI_InitStruct);
```

    - Configuració NVIC:

```
/* PD0 is connected to EXTI_Line0, which has EXTI0_IRQn vector */
NVIC_InitStruct.NVIC_IRQChannel = EXTI0_IRQn;
/* Set priority */
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0x00;
/* Set sub priority */
NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0x00;
/* Enable interrupt */
NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
/* Add to NVIC */
NVIC_Init(&NVIC_InitStruct);
```

- PB12:
    - Configuració EXTI:

```
/* PB12 is connected to EXTI_Line12 */
EXTI_InitStruct.EXTI_Line = EXTI_Line12;
/* Enable interrupt */
EXTI_InitStruct.EXTI_LineCmd = ENABLE;
/* Interrupt mode */
EXTI_InitStruct.EXTI_Mode = EXTI_Mode_Interrupt;
/* Triggers on rising and falling edge */
EXTI_InitStruct.EXTI_Trigger = EXTI_Trigger_Rising_Falling;
/* Add to EXTI */
EXTI_Init(&EXTI_InitStruct);
```

    - Configuració NVIC:

```
// PB12 is connected to EXTI_Line12, which has EXTI15_10_IRQn vector
NVIC_InitStruct.NVIC_IRQChannel = EXTI15_10_IRQn;
/* Set priority */
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0x00;
/* Set sub priority */
NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0x01;
/* Enable interrupt */
NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
/* Add to NVIC */
NVIC_Init(&NVIC_InitStruct);
```

3. Descripció dels mòduls del controlador

Tal i com hem mostrat en la figura Il·lustració 1 – Diagrama general dels mòduls emprats en la pràctica, el funcionament de la practica es el següent:

- Timer 2:
  Timer configurat com a free running timer sense interrupció. En el main, dins del bucle infinit, es consulta si el polsador esta premut consultant el pin adient. Si esta premut, s'agafa el temps de referencia actual del timer 2 i ens enbuclem de nou fins que el usuari deixa de prémer el boto.

  Al deixar de prémer el boto, agafem de nou la referencia actual del timer 2 i es fa la diferencia amb el moment en que l'usuari ha premut el boto inicialment.

  Amb això podem saber el temps que l'usuari ha tingut premut el boto i actuar de forma corresponent.

  Si la pulsació es llarga, ens trobarem en el mode configuració d'un dels dos PWM inicials de configuració. Si la pulsació es curta, voldrà dir que volem modificar la configuració actual del motor (o PWM) en el que ens trobem.

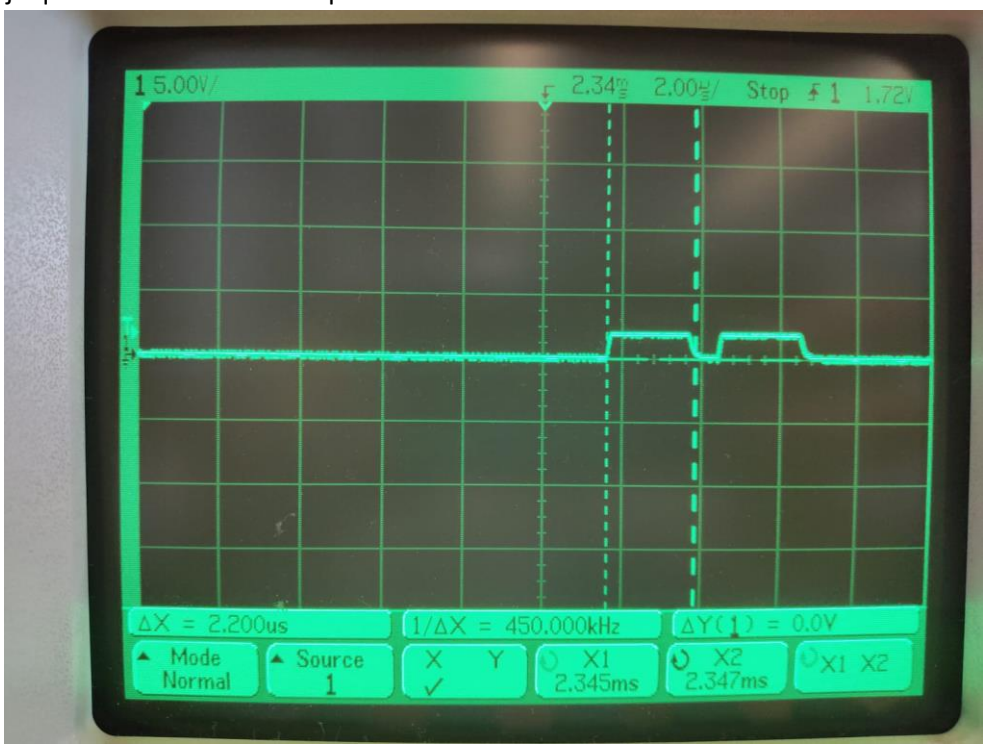
- Timer 4:
  Timer configurat de forma que generi pels pins de sortida un pols PWM. Aquest PWM tindrà un període de 10ms. Inicialment, els PWM es troben en el que es el equivalent del estat repòs, es a dir, un duty cicle del 15% que es correspon a un temps a 1 de 1.5 ms. Aquests PWM poden variar des de velocitat màxima enrere (duty del 7% o 0.7ms a 1) fins a màxima velocitat endavant (duty cicle del 23% o 2.3ms a 1).
  Aquest duty cicle varia de forma cíclica en base a les pulsacions curtes que hagi introduït el usuari, però nomes modifica el duty cicle d'un dels dos canals amb els que treballa el timer, deixant el altre PWM amb un duty cicle igual però compartint ambdós el mateix període.
  D'aquesta forma ens estalviem el tenir que configurar un timer per a cada PWM.

- Timer 3:
  El timer 3 esta configurat de forma anàloga al timer 4, sent la diferencia que el període d'aquest timer es de 100us.
  El funcionament es senzill. Simplement rep com a paràmetre d'entrada el temps que els dos PWM del timer 4 estan a 1, i en base a aquest temps, modifica els dos canals on es mostren els PWM de sortida.
  Així doncs, en aquests timer, el temps en repòs consistiria en un duty cicle del 50%, menters que si estiguéssim anant endavant a màxima velocitat, el duty seria del 100%. Anar enrere a màxima velocitat suposaria un duty del 0%. El output es proporcional a la velocitat a la que es desitja anar.

- EXTI (EXTernal Interrupts):
  Per a realitzar la comunicació entre els PWM de configuració inicials, i els PWM finals, s'han implementat EXTIs en els pins que els comuniquen. D'aquesta forma, podem generar una interrupció a cada flanc de pujada i de baixada que genera el PWM inicial.
  Al capturar les interrupcions generades pels pins, podem ajustar els valors dels PWM finals amb els valors dels PWM inicials comunicats amb les interrupcions

4. Resultat de les qüestions plantejades

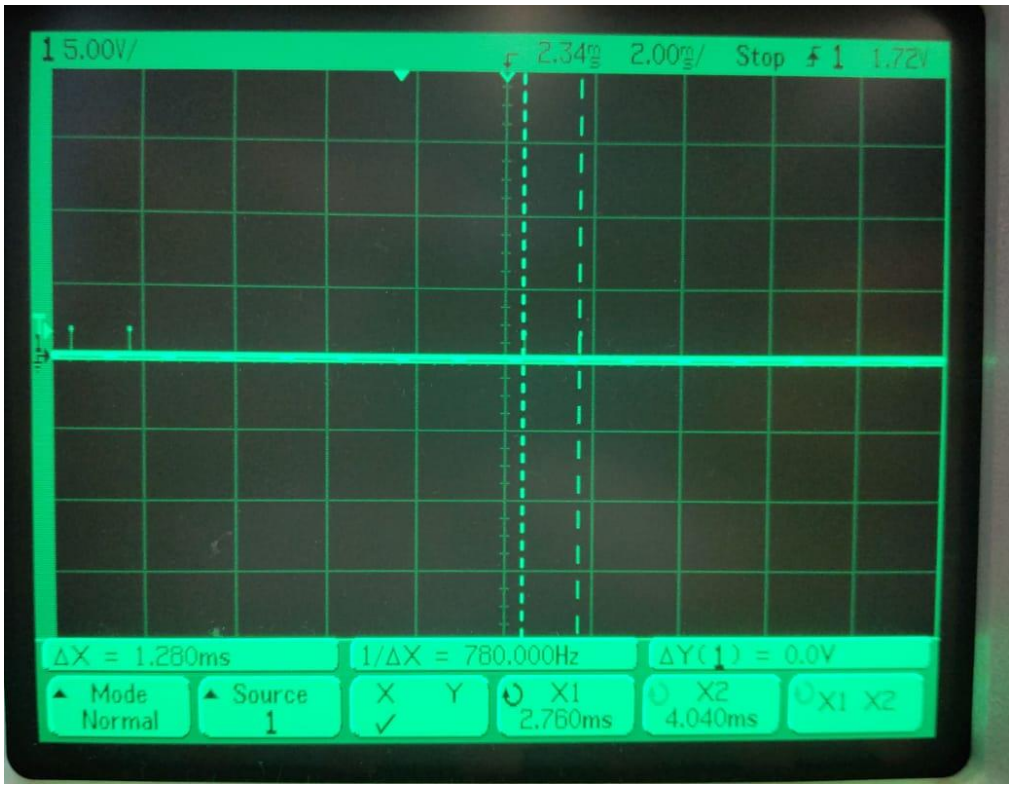
- Durada de RSI dels pins PB12 i PD0: Per ambdós pins la durada de la RSI es la mateixa, ja que fan les mateixes tasques. La durada es de 2.2us



Il·lustració 2 – Durada de la RSI mostrada en l'oscil·loscopi

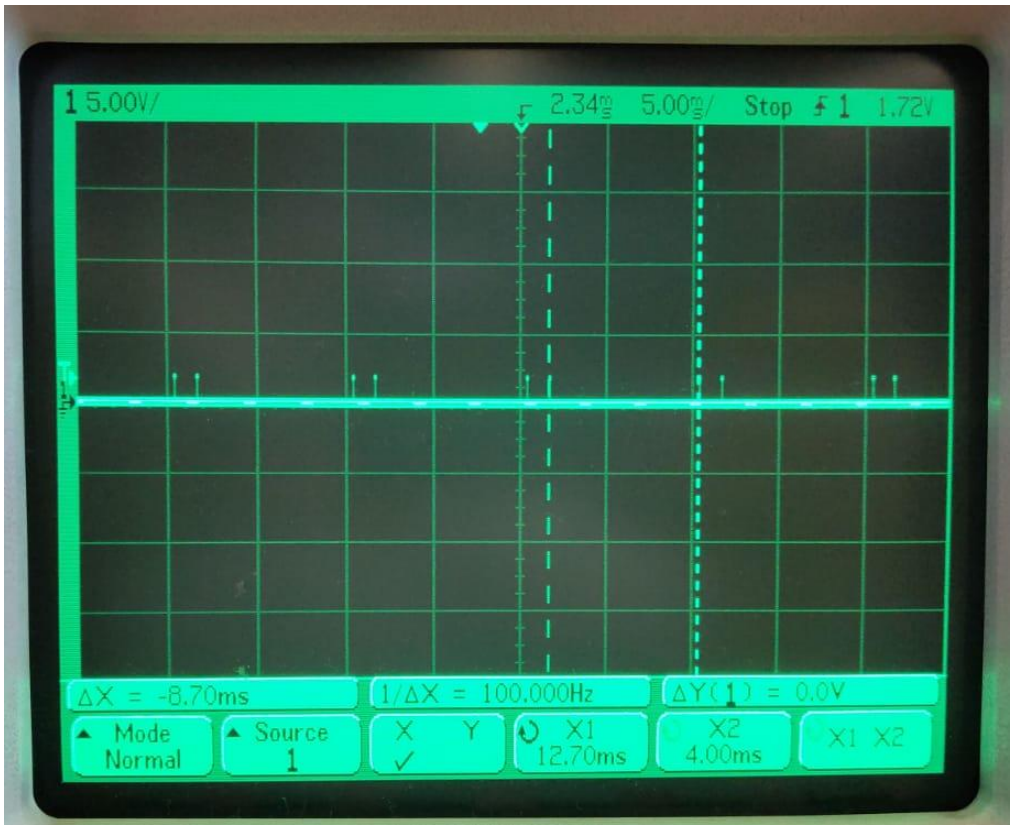
- Latència de les RSI: El temps necessari des de la petició de RSI fins al inici de la execució de la mateixa es d'uns 650ns.

- El error dependrà del moment en el que saltin les interrupcions de PD0 i PB12, ja que també tenim corrent les interrupcions dels timers pel la qual cosa, si es dona la interrupció en un moment que no hi ha una altra interrupció del timer, hi ha un temps de 1.28ms per a una interrupció de 2.2us, sent la possibilitat de que vulguem fer una interrupció just en el moment en el que no la podem fer de $2.2us \div 1.28ms = 1.71 * 10^{-3}$.



Il·lustració 3 – Interval curt entre dos RSI

Per una altra banda, si s'estan generant les interrupcions dels timers, tardarem mes a poder generar la nostra RSI, concretament 8.7ms, sent la possibilitat de voler fer la nostra RSI en un moment en el que no puguem de $2.2us \div 8.7ms = 252.87 * 10^{-6}$



- El codi te configurat les interrupcions dels pins PD0 com a màxima prioritat

```
/* Set priority */
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0x00;
/* Set sub priority */
NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0x00;
```

  Per contra, el pin PB12 esta configurat com a máxima prioritat tambe, pero la subprioritat esta un nivel per sota de la del pin PD0. Amb això fem que al saltar les interrupcions dels dos timers (que salten a l'hora si els dos PWM inicials están configurats iguals), no hi hagi cap problema amb el solapament o assignació de prioritats de forma dinámica al estar executant el codi.

  Per últim, tot I que el funcionament es correcte, aquesta prioritat podria no ser màxima, ja que dependrà del nivell de resposta que li vulguem donar al sistema. No es un paràmetre crític per al funcionament general.

- L'ample de banda es calcula de la següent forma:

$$BW = Cavg * Trsi$$

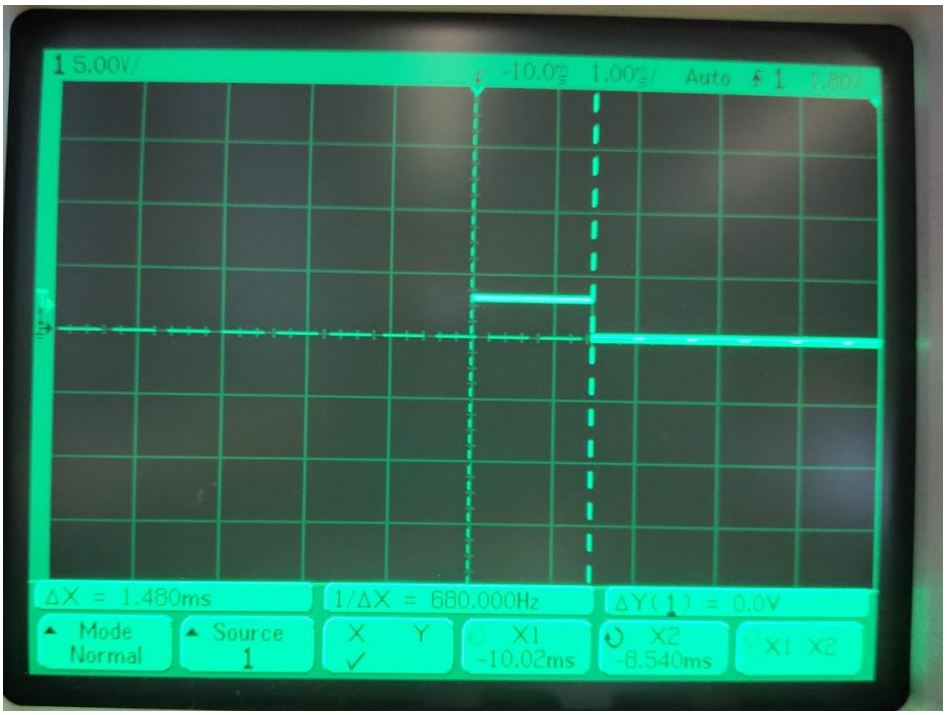
  En un segon es donen 200 interrupcions i cada una té una durada de 2.2us, sent el

Il·lustració 4 – Interval llarg entre dos RSI

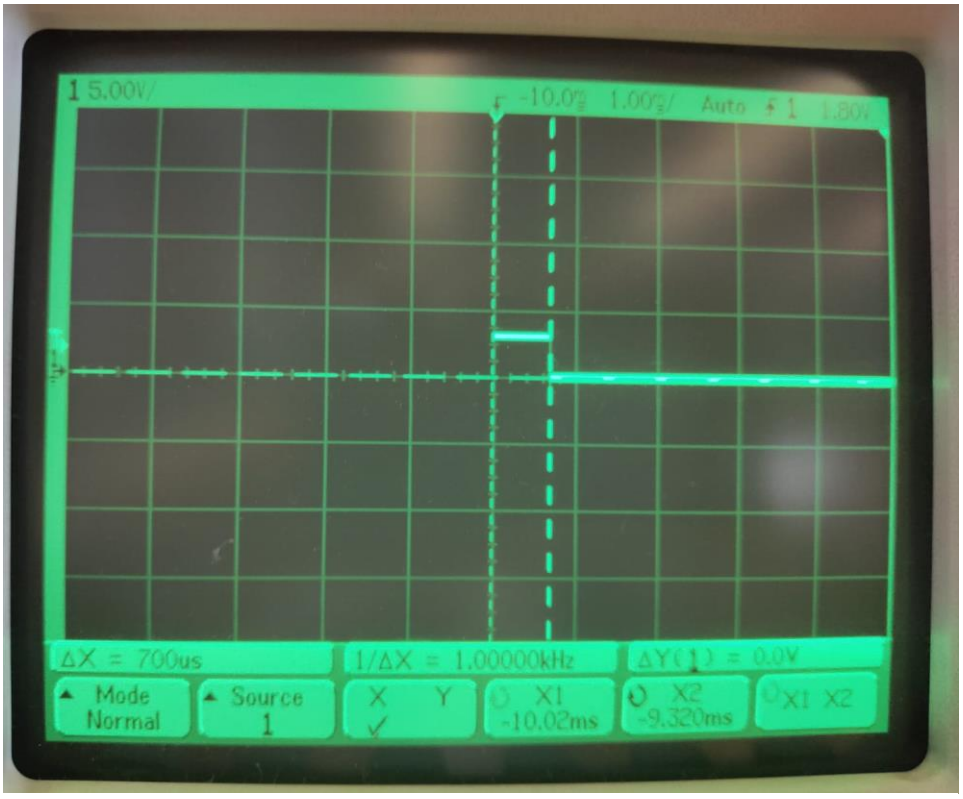
  resultat total (200* 2.2ns) * 100 = **0.044 %**

5. Captures de pantalla i formes d'ona

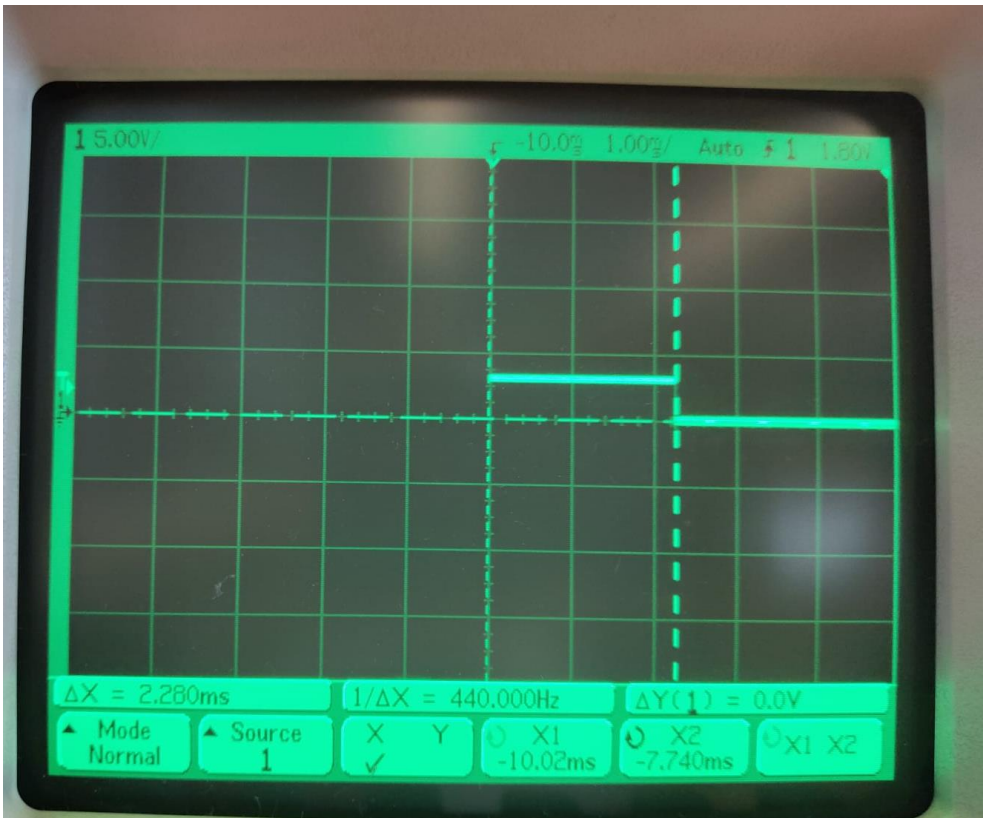
A continuació es mostren en ordre, els estats de repòs, velocitat mínima endavant i velocitat màxima enrere per al PWM inicial de configuració, seguit del estat en repòs del PWM final. Les dues velocitats mencionades anteriorment, el en PWM final corresponen a la línia a 1 i la línia a 0 corresponentment.



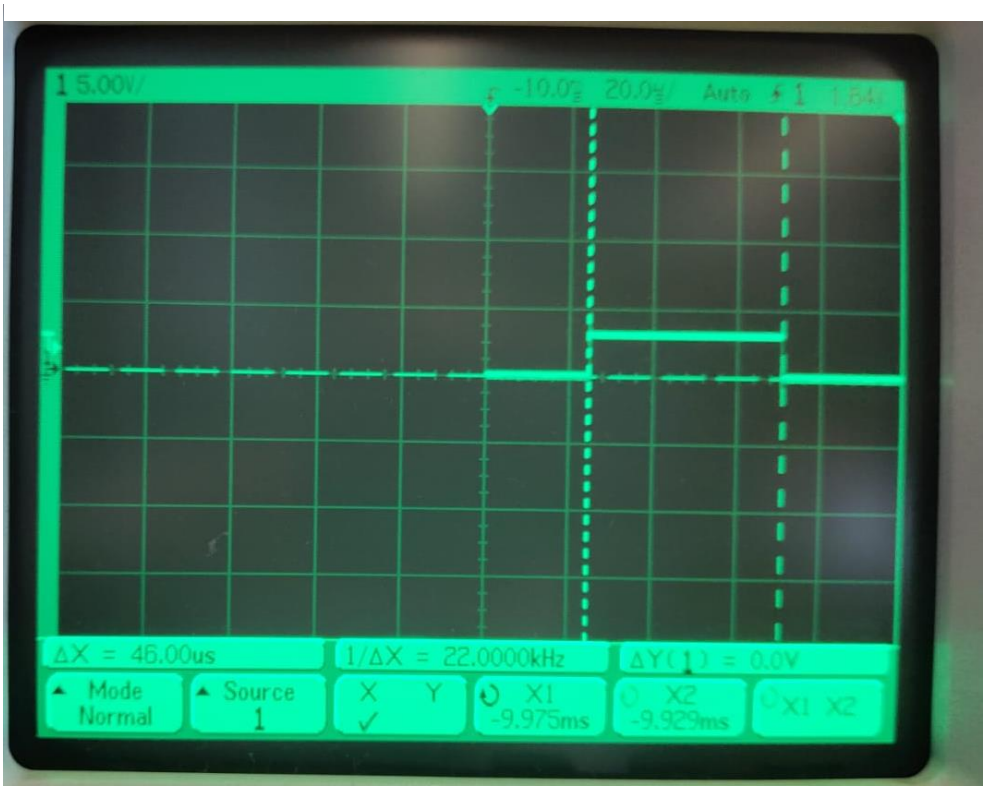
Il·lustració 5 – Estat de repòs del PWM inicial de configuracó



Il·lustració 6 – Velocitat mínima endavant del PWM inicial de configuració



Il·lustració 7 – Velocitat màxima enrere del PWM inicial de configuració



Il·lustració 8 – Estat de repós en el PWM final

6. Conclusions

La realització d'aquesta primera pràctica ens ha servit per a poder començar a aprendre i entendre un nou tipus de placa com és la STM32F429I-Discovery. Hem pogut observar que es tracta d'un element que ens ofereix un ampli ventall de possibilitats. En aquest cas, gràcies a que cada timer disposa d'un nombre de canals es pot realitzar més d'una operació treballant amb aquest mateix timer, quelcom nou que no havíem vist fins ara.

Ja que es tracta d'un nou component hem hagut de buscar molta informació en el seu datasheet el qual ens ha ajudat a aprendre moltes funcions i capacitats que té. D'aquesta manera, podem dir que aquesta primera pràctica l'hem trobat molt útil a nivell d'introducció a la STM32F429I-Discovery ja que els requisits no eren novedosos, però si el entorn de treball, ja que inclús s'ha fet us del IDE de STMicroelectronics.

També ha estat molt útil per a repassar conceptes de com treballar tant amb interrupcions de elements tant interns com externs com per exemple els timers, pins, etc. A part, hem pogut recordar la generació de PWM's de diferents duty cycles a partir d'aquestes interrupcions.