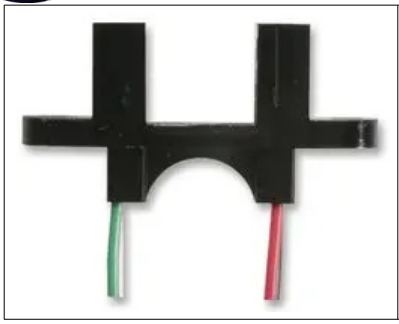


Mise en œuvre de la fourche optique – Mesure de vitesse

Anthony Juton, janvier 2023

Pour mesurer la vitesse de la voiture, on utilise une fourche optique montée sur la roue dentée de l'arbre de transmission.
Un tour de la roue dentée correspond à 79 mm de la voiture au sol.
Il y a 16 fentes sur la roue dentée. Donc l'écart angulaire entre 2 fentes correspond à 4,9375 mm.
Le signal issue de la fourche optique a l'allure suivante :

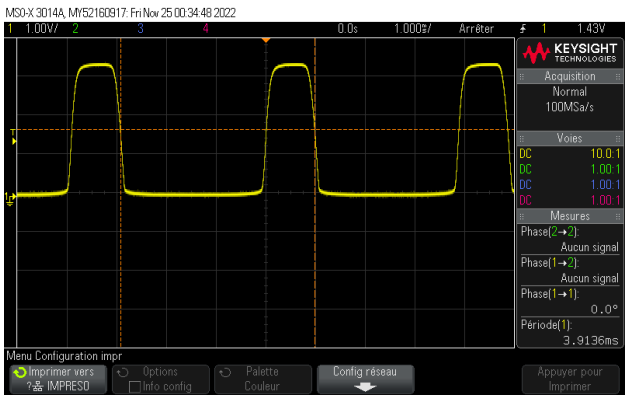


Figure 1: Acquisition signal fourche à faible vitesse

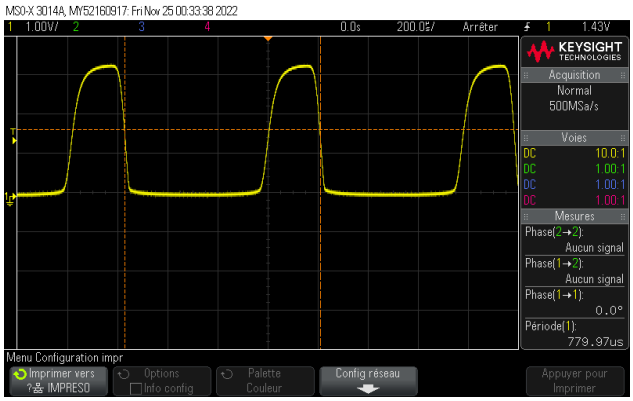


Figure 2: Acquisition signal fourche à haute vitesse

Sur la Figure 1, on mesure un écart de 3,91 ms entre 2 fronts descendants, ce qui correspond théoriquement à 1,26 $m.s^{-1}$.
Sur la Figure 2, on mesure un écart de 780 µs entre 2 fronts descendants, ce qui correspond théoriquement à 6,3 $m.s^{-1}$.

L'objet de ce guide est de mettre en œuvre une mesure précise de la durée s'écoulant entre 2 passages de fentes dans la fourche. Pour cela on utilise le mode capture du timer 2 du microcontrôleur STM32L432KC.

1 Configuration du périphérique

On utilise ici l'environnement de développement gratuit, multiplateforme basé sur Eclipse STM32CubeIDE. On peut partir du projet avec les réglages par défaut pour la carte STM32L432KC.

Le schéma de la carte Hat de la voiture test CoVAPSy *Hat_CoVAPSy_v1re2.pdf* indique que la fourche est reliée à la broche A0 de la carte Nucléo, ce qui correspond à la broche PA0 du microcontrôleur.

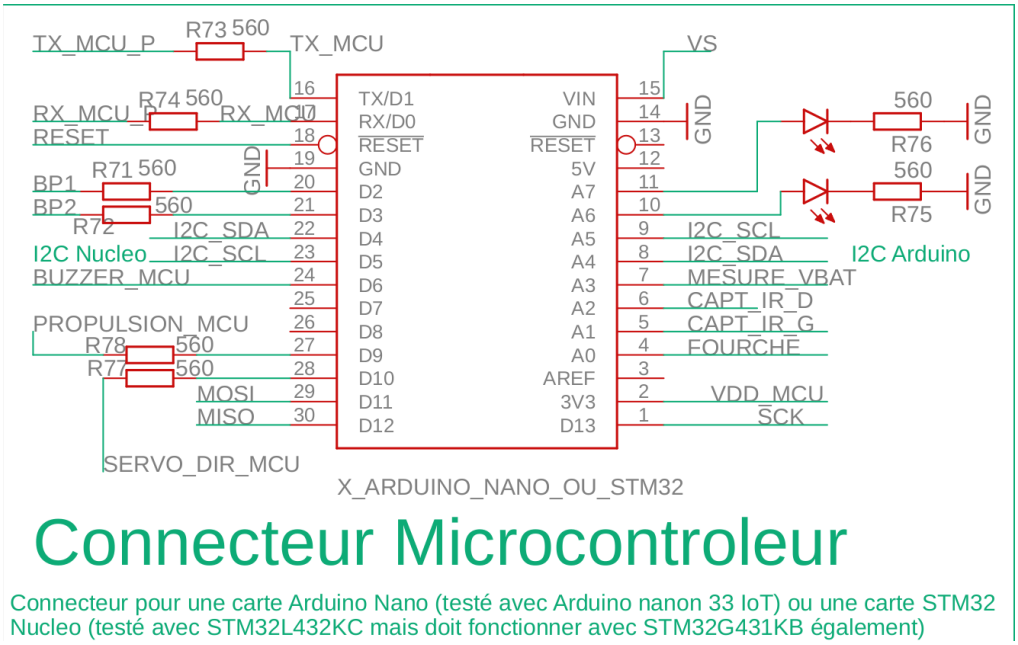


Figure 3: Extrait du schéma de la carte Hat_CoVAPSy_v1re2

1.1 Configuration de l'horloge

Dans STM32CubeIDE, la fenêtre de configuration des horloges indique, avec les réglages par défaut, une horloge à 32 MHz pour les Timers.

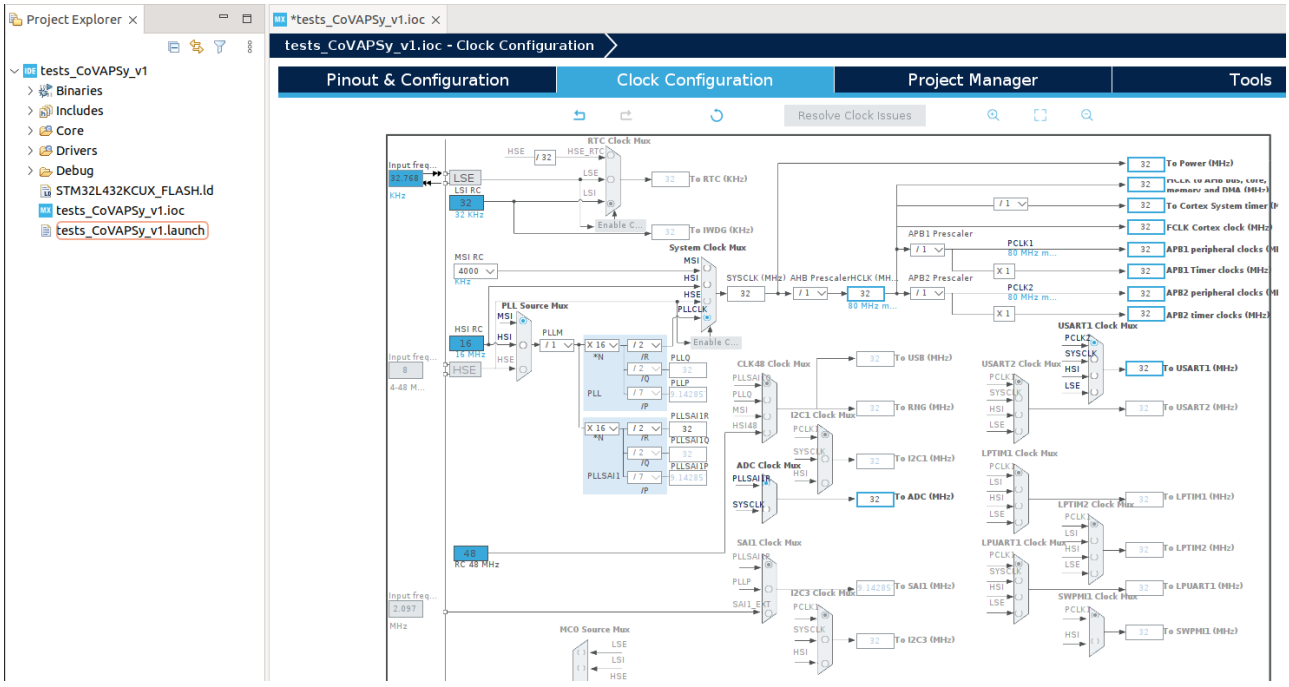
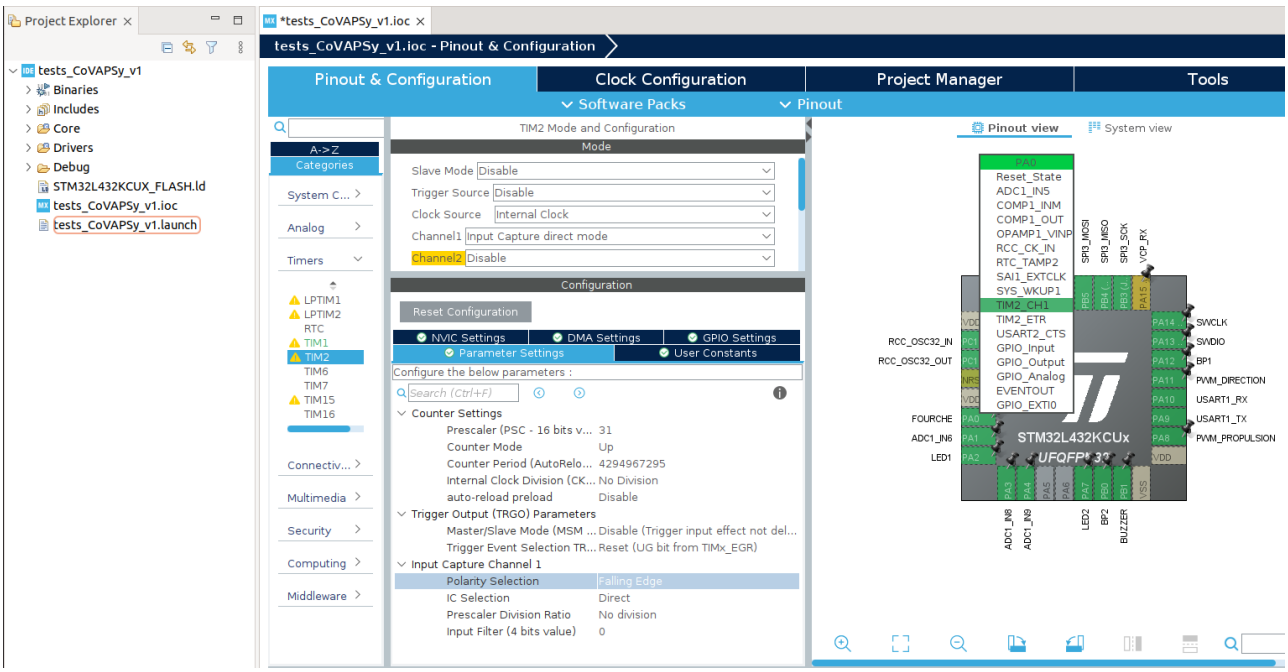


Figure 4: fenêtre STM32CubeIDE de configuration des horloges

1.2 Configuration des entrées/sorties

On configure la broche PA0 comme entrée capture du timer 2 Channel 1, en utilisant d'une part le schéma *Pinout View* et d'autre part le volet de configuration des périphériques. Dans ce volet, on peut choisir un préscalaire de 31 (pour obtenir une horloge du timer à 1 MHz) et un mode de capture sur front descendant.



Pour stocker la mesure de la durée à chaque front descendant, on active l'interruption dans l'onglet *NVIC Settings* :

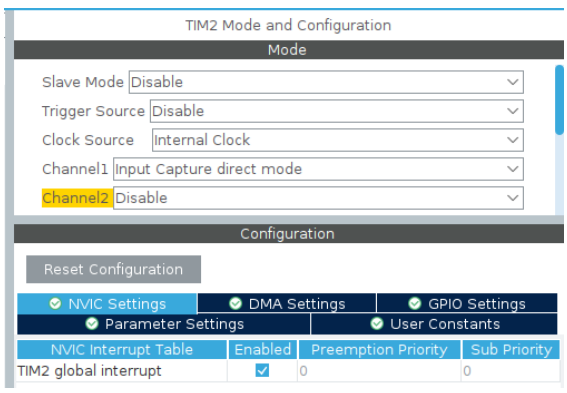


Figure 5: Activation de l'interruption

1.3 Génération automatique du code

Une fois la configuration terminée, on génère le code associé : *Project > Generate Code*.

2 Code

L'échange entre la fonction d'interruption et le programme principal se fait via une variable globale *vitesse_m_s*. L'interruption étant prioritaire sur le programme principal, elle ne peut être interrompue pendant l'écriture sur la variable *vitesse_m_s*. Il faut veiller dans les programmes utilisant cette variable vitesse à lire la valeur une seule fois par boucle et sur un seul cycle instruction, pour éviter que la fonction d'interruption ne la modifie entre 2 usages dans la même boucle.

On place en macro définition la distance sur un tout d'axe de transmission pour éventuellement le préciser plus tard.

```
/* USER CODE BEGIN PD */
#define DISTANCE_1_TOUR_AXE_TRANSMISSION_MM 79
/* USER CODE END PD */

/* USER CODE BEGIN PV */
float vitesse_mesuree_m_s = 0;
/* USER CODE END PV */
```

2.1 Démarrage de l'interruption

Dans le programme principal, on démarre le timer 2 et active les interruptions :

```
MX_TIM2_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim2);
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
```

2.2 La fonction d'interruption

Dans la fonction d'interruption déclenchée par le front descendant sur le signal de la fourche, on récupère la valeur du timer et on lui soustrait la valeur précédente pour avoir la durée de l'intervalle. Attention, pour conserver sa valeur d'un lancement d'interruption à l'autre, la variable mesure_precedente_us doit avoir l'attribut *static*.

```
/* USER CODE BEGIN 4 */
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef* htim)
{
	static uint32_t mesure_precedente_us=0 ;
	uint32_t mesure_us ;
	mesure_us = __HAL_TIM_GET_COMPARE(&htim2, TIM_CHANNEL_1); // ou TIM2->CCR1
	vitesse_mesuree_m_s = DISTANCE_1_TOUR_AXE_TRANSMISSION_MM *1000 / 16.0 /
(mesure_us - mesure_precedente_us);
	mesure_precedente_us = mesure_us;
}
/* USER CODE END 4 */
```

3 Visualisation du fonctionnement

STM21CubeIDE permet de visualiser l'évolution en temps réel de quelques variables globales via la fenêtre *Live Expression* de la perspective *Debug*.

Un court programme de modification de la vitesse avec les boutons permet alors de faire des tests sur table (les roues dans le vide !) et de visualiser la vitesse mesurée.

mise_en_oeuvre_composants_CoVAPSy_v1.ioc | main.c

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim2);
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_4);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    bp1 = HAL_GPIO_ReadPin(BP1_GPIO_Port, BP1_Pin);
    bp2 = HAL_GPIO_ReadPin(BP2_GPIO_Port, BP2_Pin);
    if((bp1 == BP_ENFONCE) && (bp1_old == BP_RELACHE))
    {
        HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
        if(dc_propulsion < 1700)
                dc_propulsion += 10;
        if(dc_direction < (BUTEE_DROITE-10))
            dc_direction += 10;
        __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, dc_propulsion);
        __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, dc_direction);
    }
    if((bp2 == BP_ENFONCE) && (bp2_old == BP_RELACHE))
    {
        HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
        if(dc_propulsion > 1500)
            dc_propulsion -= 10;
        if(dc_direction > (BUTEE_GAUCHE+10))
            dc_direction -= 10;
        __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, dc_propulsion);
        __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_4, dc_direction);
    }

    bp1_old = bp1;
    bp2_old = bp2;
  /* USER CODE END WHILE */
```

Variabl | Breakp | Expres | Regist | Live Ex | SFRs

Expression	Type	Value
vitesse_mesuree_m_s	float	1.88095236
Add new expression		

4 Amélioration du programme

La précision de l'intervalle angulaire entre 2 fentes n'étant pas parfait, une moyenne sur plusieurs mesures peut être intéressante. Le nombre de mesures pertinentes dépend de la vitesse et attention à la première mesure après un redémarrage.

Une fois la vitesse mesurée suffisamment précisément, il est possible de faire un asservissement de vitesse…

Bon courage