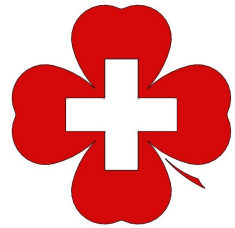
ENSEA
Ecole Nationale Supérieure de l'Electronique et de ses Applications



Master's 1 Project Presentation

Development of a DRO and DIVIDER System

Interns: Selim Farci, AbdelKader CHAMBI

Company Supervisor: Mr. Michaël EYRAUD

University Supervisor: Mr. CHIPI

Sommaire

Introduction

Contexte et objectif du projet

Ce système DRO et DIVISER vise à faciliter l'utilisation des systèmes horlogers industriels tout en préservant les anciennes machines industrielles. Ces machines, très recherchées par les artisans horlogers, conservent leur aspect vintage car ce sont des objets de collection de grande valeur. N'oubliez pas que le prix et la crédibilité industrielle sont des facteurs importants pour le développement de ce projet.

- Le système DRO doit afficher et modifier les valeurs fournies par des capteurs et des échelles linéaires à partir d'un contrôleur à écran tactile d'au moins 7".
- Le système DIVIDER est un système complémentaire du système DRO, il doit contrôler 2 moteurs pas à pas pour effectuer diverses fonctions.

Le système DRO

Le système DRO sera le centre de contrôle et le panneau d'interface utilisateur avec écran, contrôle et interrupteurs, il doit fonctionner de manière autonome.

- Affichage avec interface graphique : nombre d'axes, boutons de fonction, information sur le moteur de broche.
- Fonctions : ZERO, Ø / R, + / -, vitesse de la broche, couple de la broche, retour au menu.

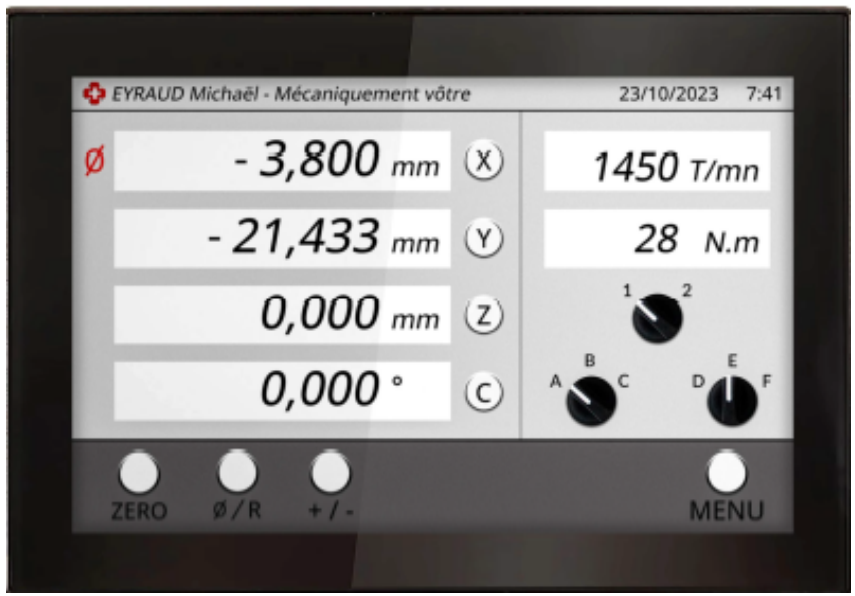


Figure: Interface graphique du système DRO

Le système DIVIDER

Le système DIVIDER complémente le système DRO en permettant le contrôle de 2 moteurs pas à pas pour des fonctions variées.

- Cycle d'avance linéaire : rotation du moteur pas à pas No. 1.
- Indexation angulaire : rotation du moteur pas à pas No. 2.

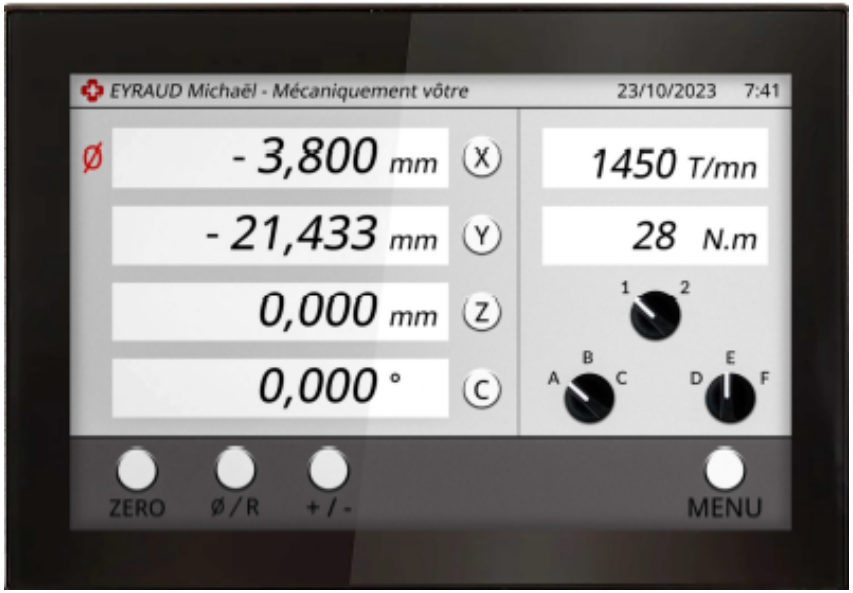


Figure: Interface graphique du système DIVIDER

Cahier des charges

Diagramme de cas d'utilisation

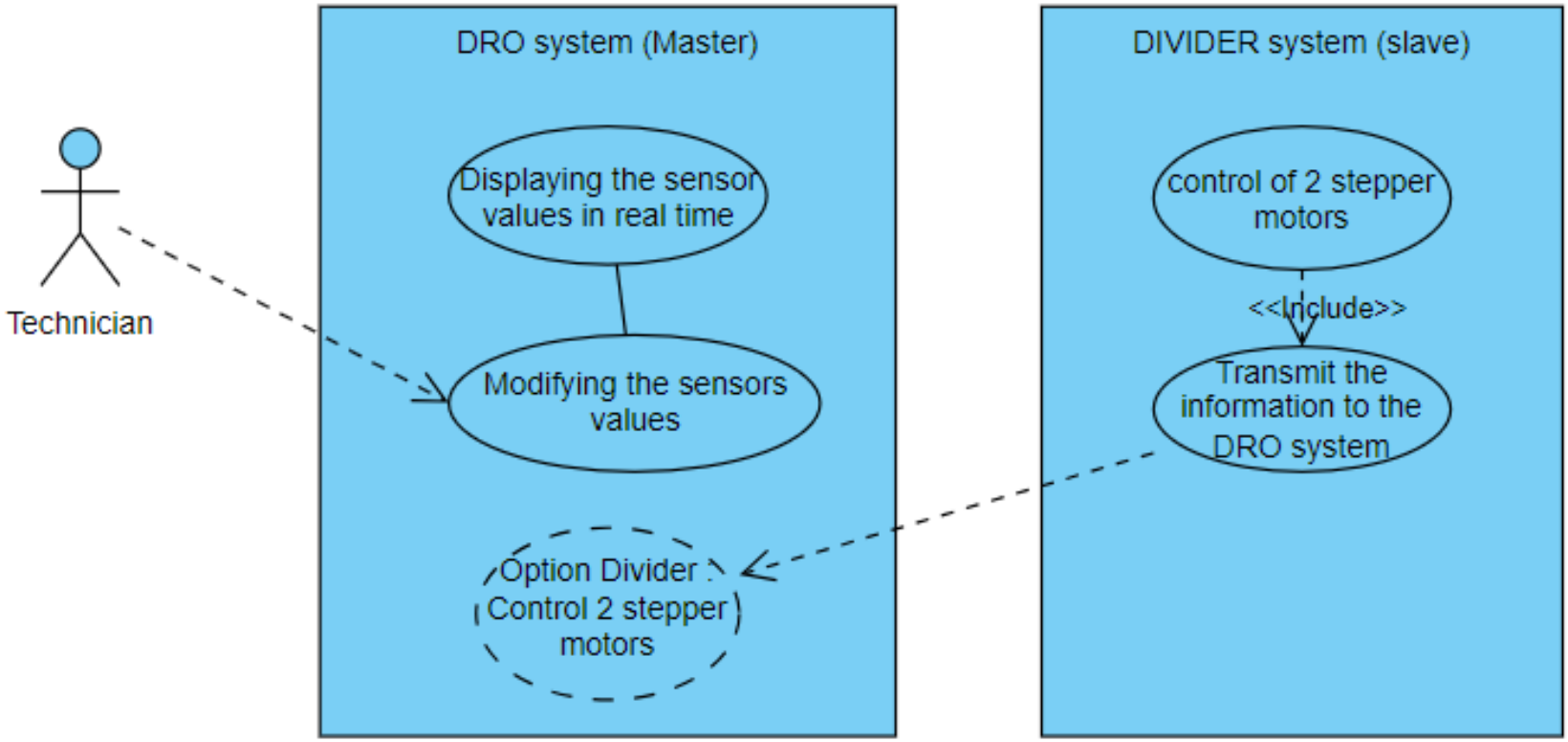


Figure: Diagramme de cas d'utilisation

Diagrammes de exigences du DRO

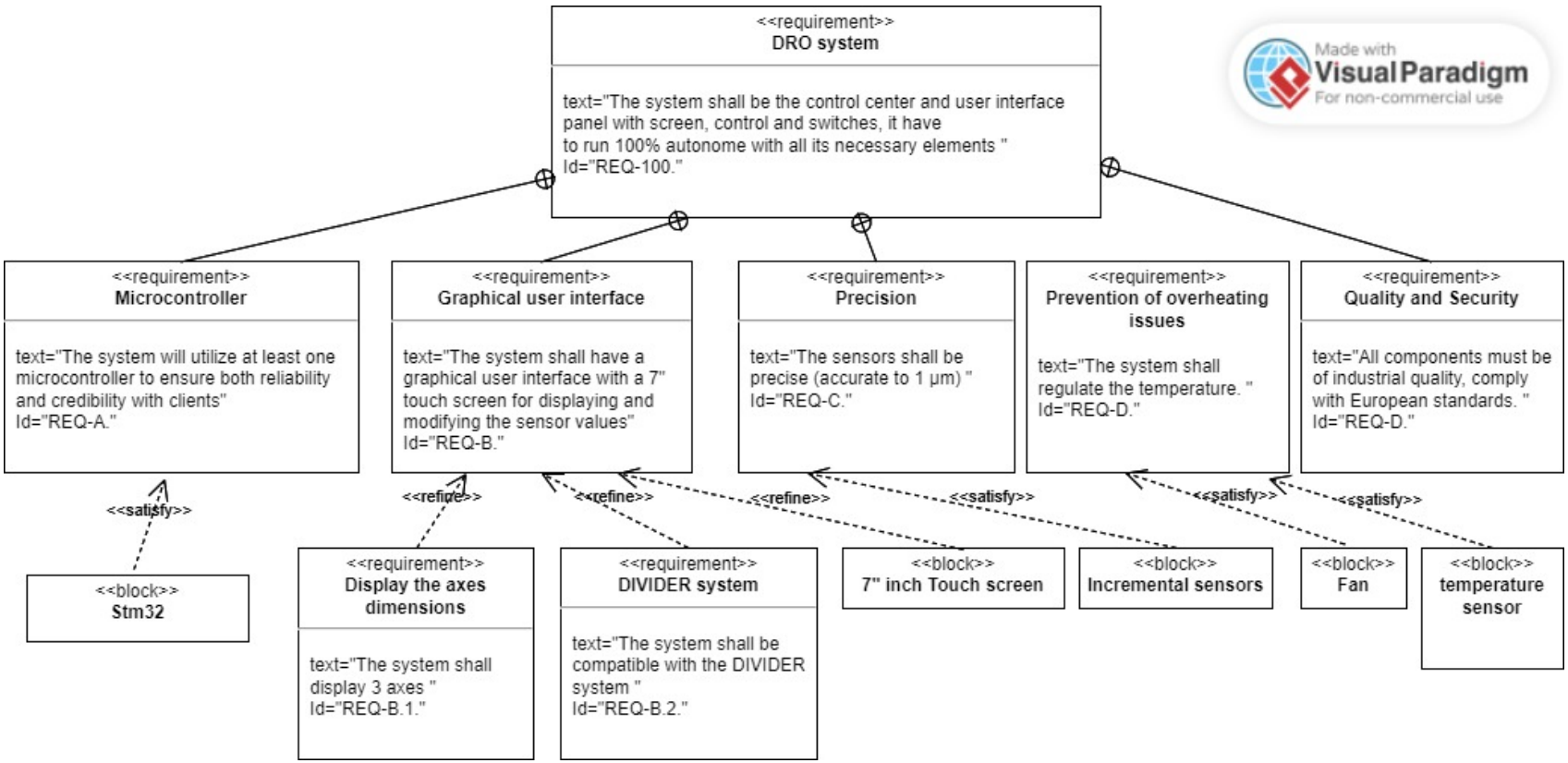


Figure: Diagramme d'exigences du système DRO

Diagrammes d'exigences du système DIVIDER

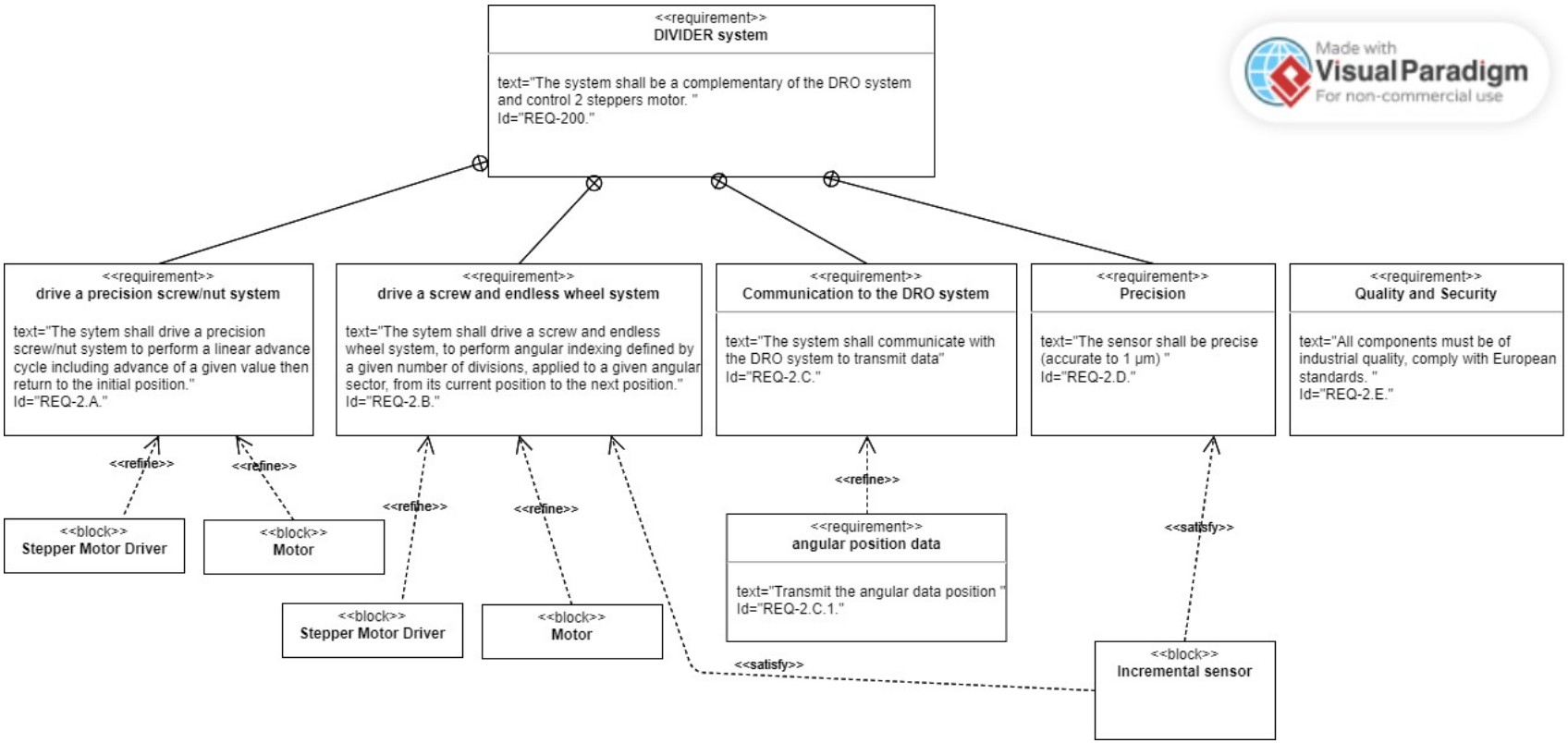


Figure: Diagramme de exigences du système DIVIDER

Choix des composants

Choix du microcontrôleur

Nous avons comparé différentes technologies pour choisir la meilleure solution pour notre projet.

Critères	PLC	STM32	Raspberry Pi
Microcontrôleur	Siemens S7	STM32 Series	Raspberry Pi series
Langage de programmation	Ladder Logic	C, C++	Python, C++
Environnement adapté	Industriel	Industriel	Éducation
Type d'écran	HMI Panels	TFT LCD	HDMI
Prix	Élevé	Modéré	Modéré

Table: Comparaison des microcontrôleurs

Choix de l'écran

Solution	Écran	Détails du produit	Prix
Sol 1	Waveshare 7"	1024x600, HDMI/USB	70 dollars
Sol 2	Riverdi 7"	Haute qualité, STM32 intégré	200 dollars
Sol 3	Generic 7" TFT	800x480, SPI	35 dollars

Table: Comparaison des écrans

Le choix s'est porté sur la solution Riverdi en raison de sa qualité et de son intégration facile avec le microcontrôleur STM32.

Visualisation de l'écran Riverdi 7 pouces

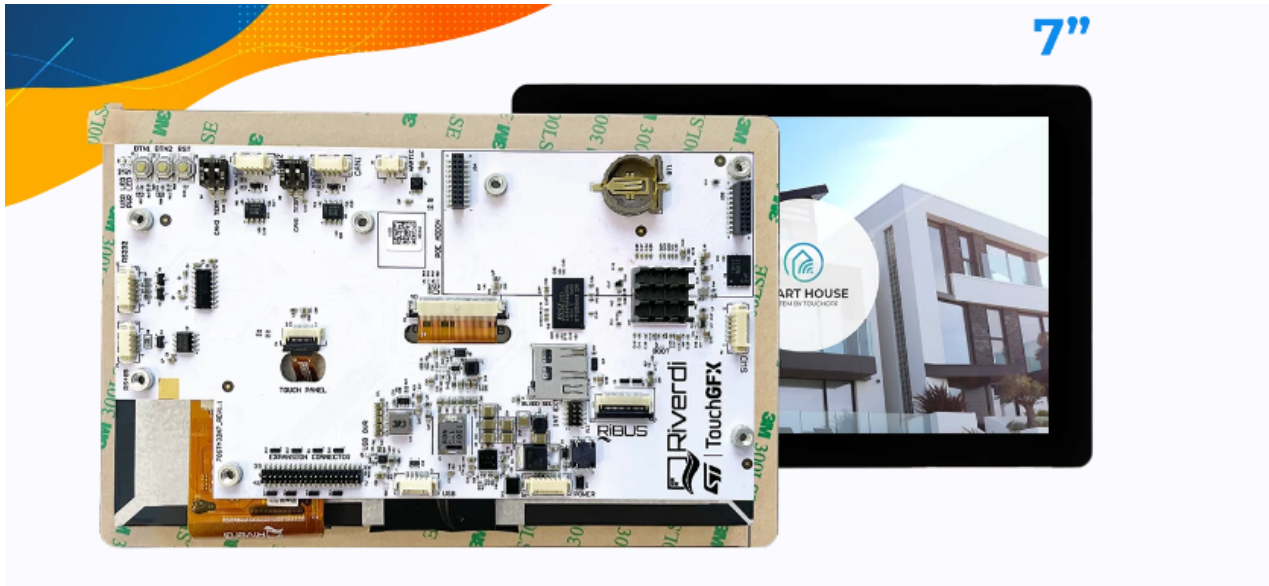


Figure: Ecran Riverdi (ref : RVT70HSSNWC00-B)

Caractéristiques techniques de l'écran

L'écran du Riverdi possède 40 broches qui peuvent être utilisées. Ces broches proviennent du STM32H7 (le STM inclus dans le Riverdi). Ces broches disposent des interfaces suivantes :
- 2 x I2C
- 1 x UART
- 1 x USART
- 1 x SPI
- 1 x USB
- 7 x PWMs
- 2 x DACs (Digital-to-analog)
- 2 x ADCs (Analog-to-digital)

On utilisera une des deux communication I2C pour le projet.

Choix de l'écran

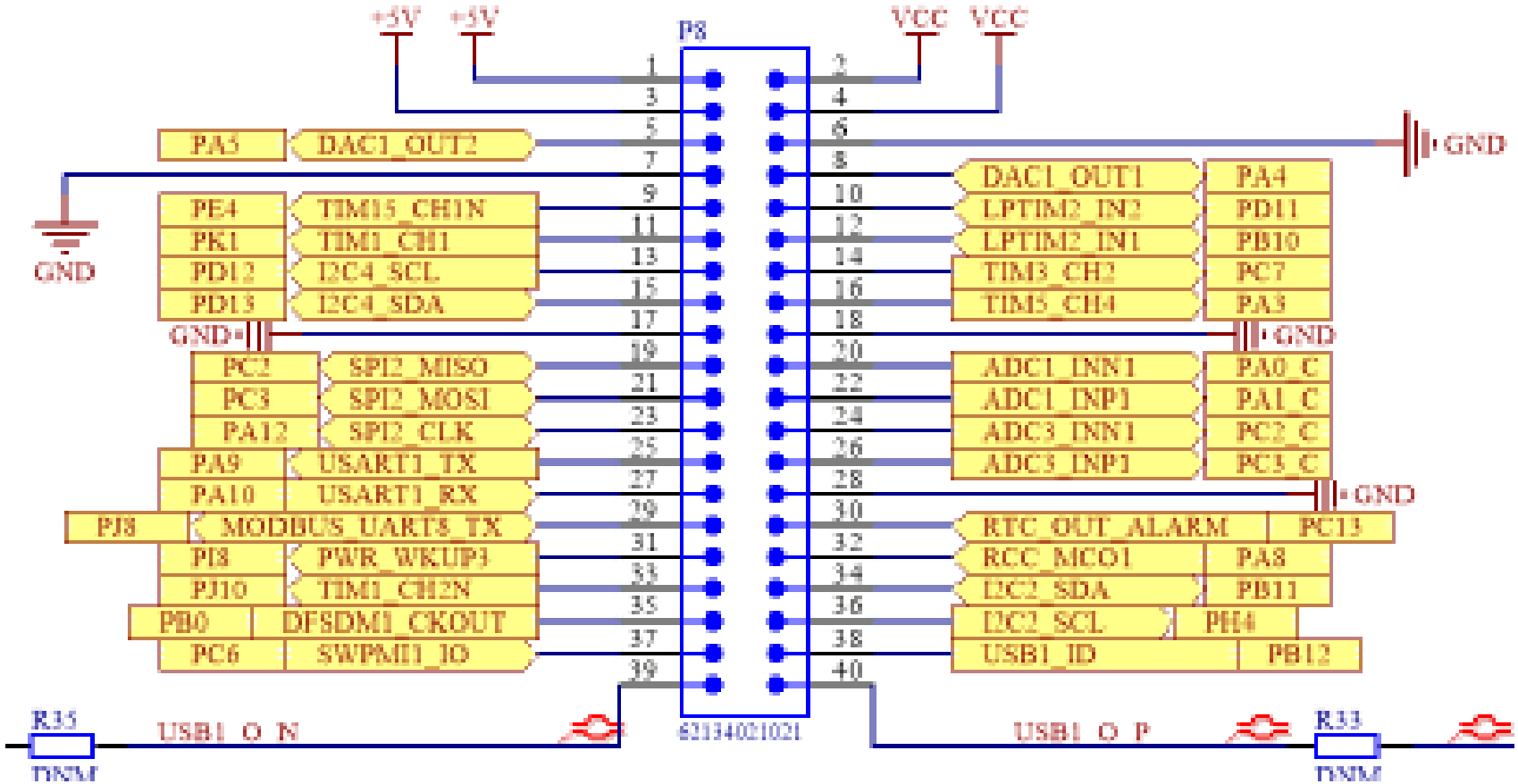
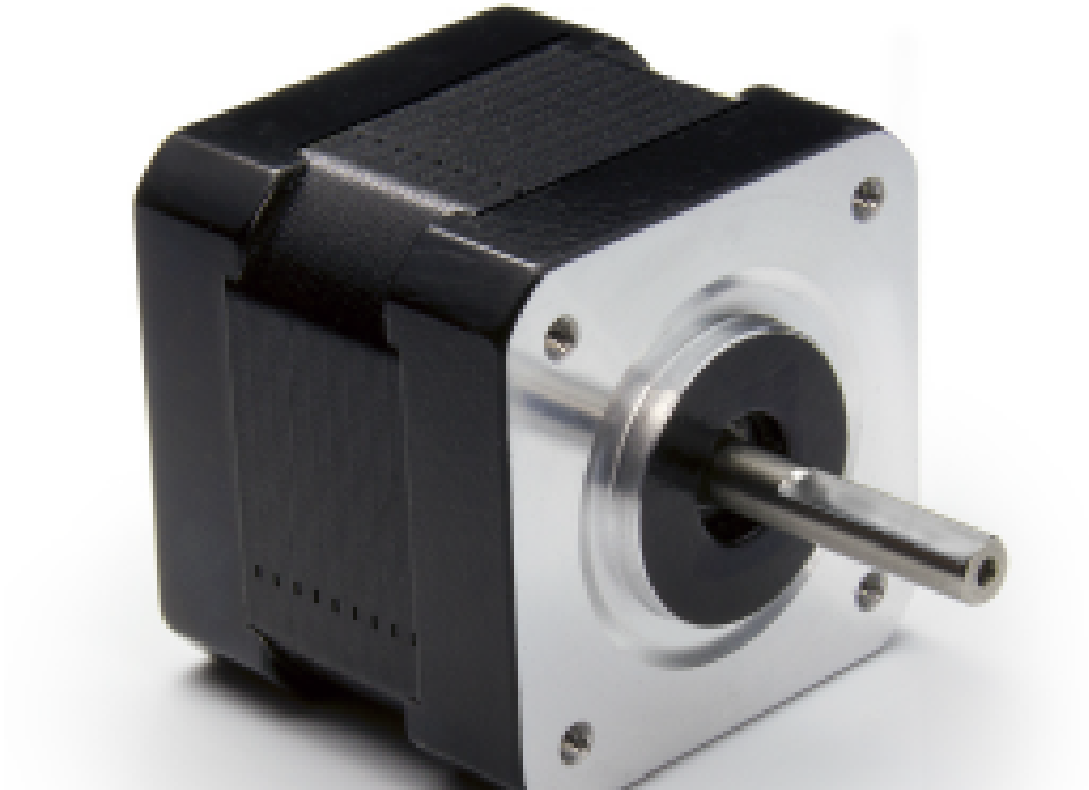


Figure: Riverdi screen (ref : RVT70HSSNWC00-B)

L'ecran doit s'alimenter avec une tension comprise entre 8 et 48V.

Choix du Moteur

Selon les spécifications, le moteur pas à pas doit être de type Nema 17 avec un couple de 0,44 N.m et une bonne précision de l'angle de pas (nous utiliserons 0,9°). Nous avons choisi le moteur WO-4209L-01P Lin Engineering en respectant les exigences. Le courant de sortie est de 1,7A/phase et la tension d'alimentation est de 24V (nous aurons besoin de ces informations pour le pilote).



Caractéristique technique du moteur

0.4

Part Number	**WO-4209L-01P**
Step Angle	0.9°
Frame Size	NEMA 17
Body Length (Dim. A)	1.89 in (48 mm)
Current	1.7 Amps/Phase
Holding Torque	62 oz-in (0.44 Nm)
Resistance	1.9 Ohms/Phase
Rotor Inertia	0.37 oz-in^2
Number of Leads	4
Connection	Bipolar
Weight	0.7 lbs (0.32 kg)

Choix du driver

- Driver R701P pour contrôler les moteurs pas à pas.

R701P

MICROSTEPPING DRIVER

FEATURES & BENEFITS

REPLACES R701 MICROSTEPPING DRIVE

- 10 microstepping driver
- Common Ground or Common + 5 Volts Input Option Available
- Optically isolated Step, Direction, and Disable/Enable inputs
- Automatic Current Reduction
- Adjustable trimpot for noise and vibration reduction
- Operates from 18 to 80 VDC
- Selectable Driver Peak Current Ranges: 0 to 7 Amps
- Low Power Dissipation from 1 to 12 Watts (1 to 7 Amps)
- Excellent sinusoidal current waveform for smooth operation
- Low current ripple for low noise
- Low Cost
- High Efficiency

Figure: Alimentation Mean Well

Caractéristique technique du driver

Pin #	Color
1	Power Ground
2	+18 to 80 VDC
3	A Phase
4	A Bar Phase
5	B Phase
6	B Bar Phase
7	Disable Input
8	Direction Input
9	Step Input
10	+5 VDC
11	Current Set
12	Current Set

Figure: Driver outputs

Caractéristique technique du driver

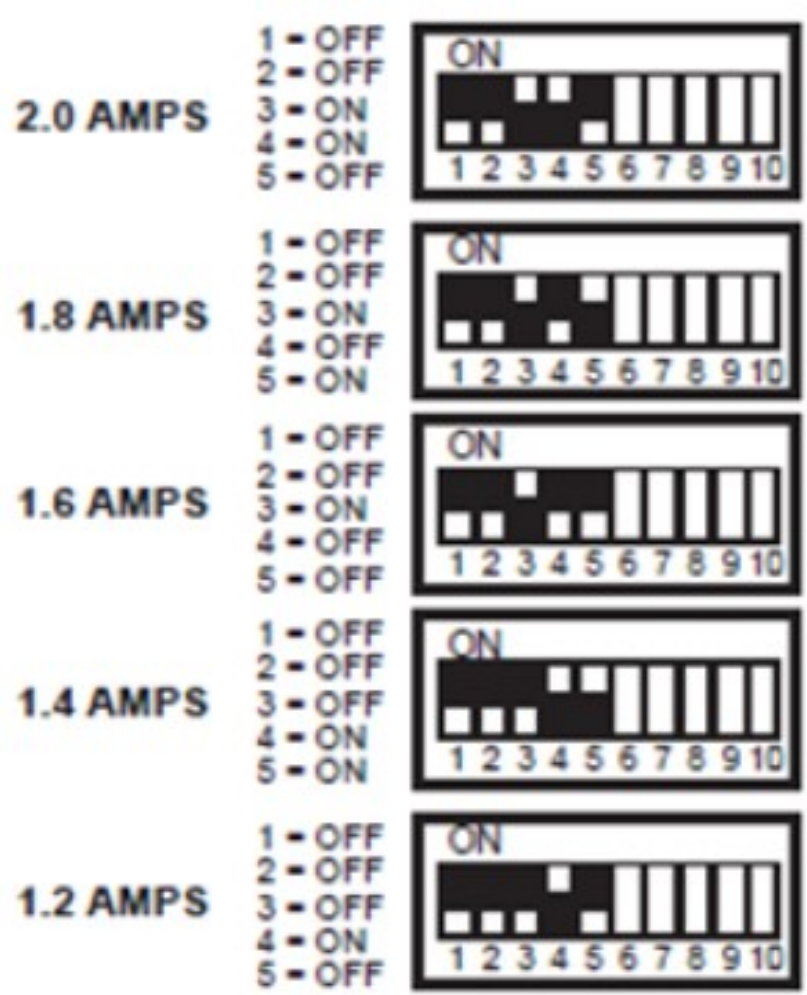


Figure: Driver option set switches

Caractéristique technique du driver

Nous avons décidé d'utiliser un microcontrôleur STM32 NUCLEO (STM32F411xE) car, après avoir calculé le nombre d'entrées/sorties utilisées par les différents composants, nous nous sommes aperçus que le nombre de pins du Riverdi était insuffisante.pour l'ensemble du système.

Cela rend également assez indépendant le système DIVIDER.

Nous avons choisis ce microcontroller pour sa petite taille.

stm32F411xE

Figure: stm32F411xE

Caractéristique technique du driver

Ce microcontrôleur possède des broches d'interface I2C pour communiquer avec le STM32H7 (inclus sur la carte Riverdi). Les broches PB6/PB10/PA8 sont destinées aux signaux SCL et PB7/PB9/PB4 aux signaux SDA.

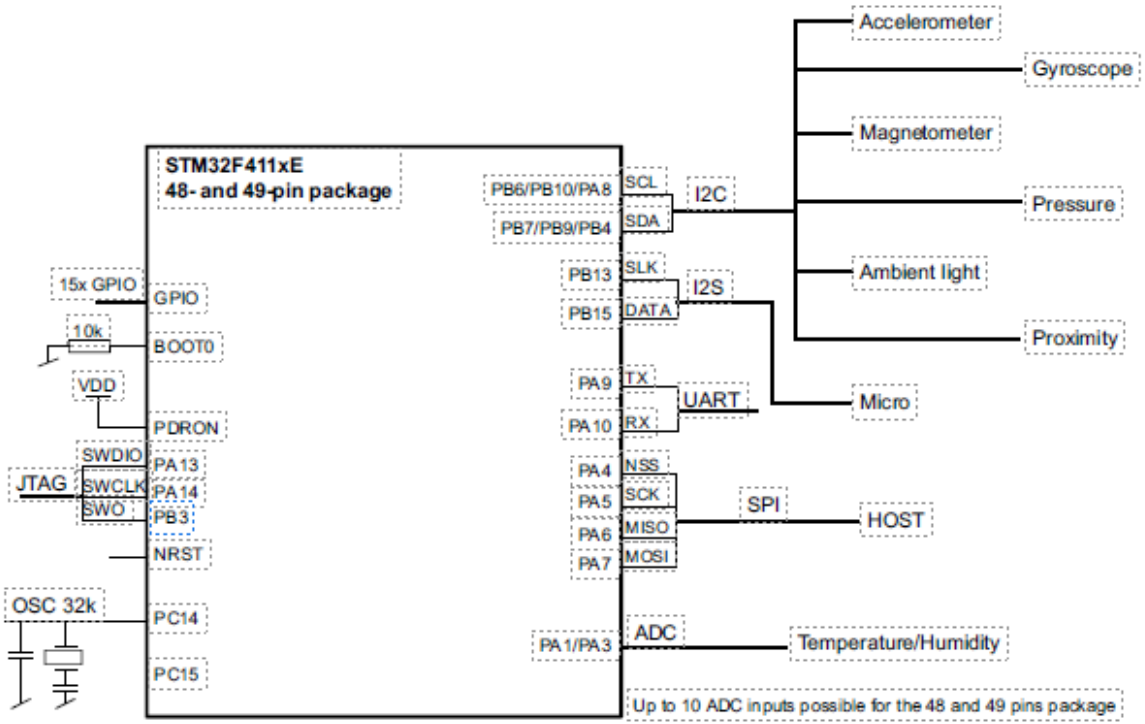


Figure: stm32F4 I2C pins

Choix de l'alimentation

L'alimentation alimentera le DRO et le DIVIDER en 24V.

Nous devons identifier la consommation de puissance et de courant de chaque composant à partir de leurs fiches techniques.

Nous devons identifier la consommation de puissance et de courant de chaque composant à partir de leurs fiches techniques.

La consommation électrique maximale de notre système (si tout fonctionne simultanément à pleine puissance) est de 105W avec un courant de 5,1A.

Nous avons choisi l'alimentation EDR-150-24 Mean Well, qui prend une entrée de 230VAC et fournit une tension de sortie de 24VDC.La puissance maximale que cette alimentation peut fournir est de 156 W, soit 1,5 fois plus que nécessaire. fois plus que nécessaire, et elle peut délivrer jusqu'à 6,5A (avec une marge de 1,5A).

Alimentation EDR-150-24 Mean Well

Figure: Power supply

Alimentation EDR-150-24 Mean Well

MODEL		EDR-150-24
OUTPUT	DC VOLTAGE	24V
	RATED CURRENT	6.5A / 230VAC 5.2A / 115VAC
	CURRENT RANGE	0 ~ 6.5A / 230VAC 0 ~ 5.2A / 115VAC
	RATED POWER	156W / 230VAC 125W / 115VAC
	RIPPLE & NOISE (max.) Note.2	150mVp-p
	VOLTAGE ADJ. RANGE	24 ~ 28V
	VOLTAGE TOLERANCE Note.3	±1.0%
	LINE REGULATION	±0.5%
	LOAD REGULATION	±1.0%
	SETUP, RISE TIME	1500ms, 60ms/230VAC 3000ms, 60ms/115VAC at full load
	HOLD UP TIME (Typ.)	16ms/230VAC 10ms/115VAC at full load
INPUT	VOLTAGE RANGE Note.6	90 ~ 264VAC 127 ~ 370VDC [DC input operation possible by connecting AC/L(+), AC/N(-)]
	FREQUENCY RANGE	47 ~ 63Hz
	EFFICIENCY (Typ.)	87%
	AC CURRENT (Typ.)	2.6A/115VAC 1.7A/230VAC
	INRUSH CURRENT (Typ.)	20A/115VAC 35A/230VAC
	LEAKAGE CURRENT	<1mA / 240VAC

Figure: Power supply specification

Schémas électriques

Schéma électrique du système DRO

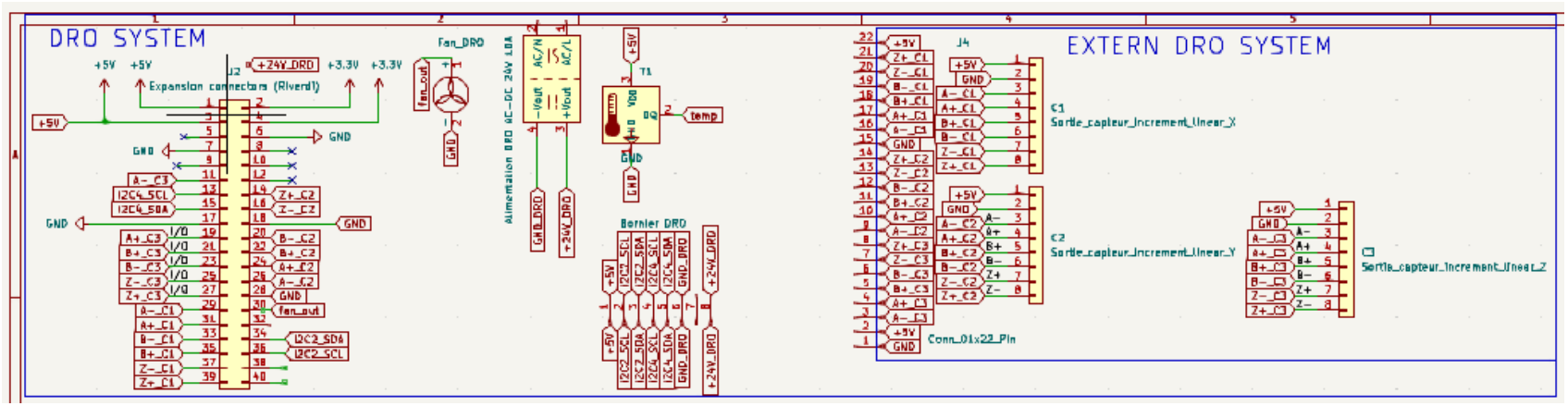


Figure: Schéma électrique du système DRO

Schéma électrique du système DIVIDER

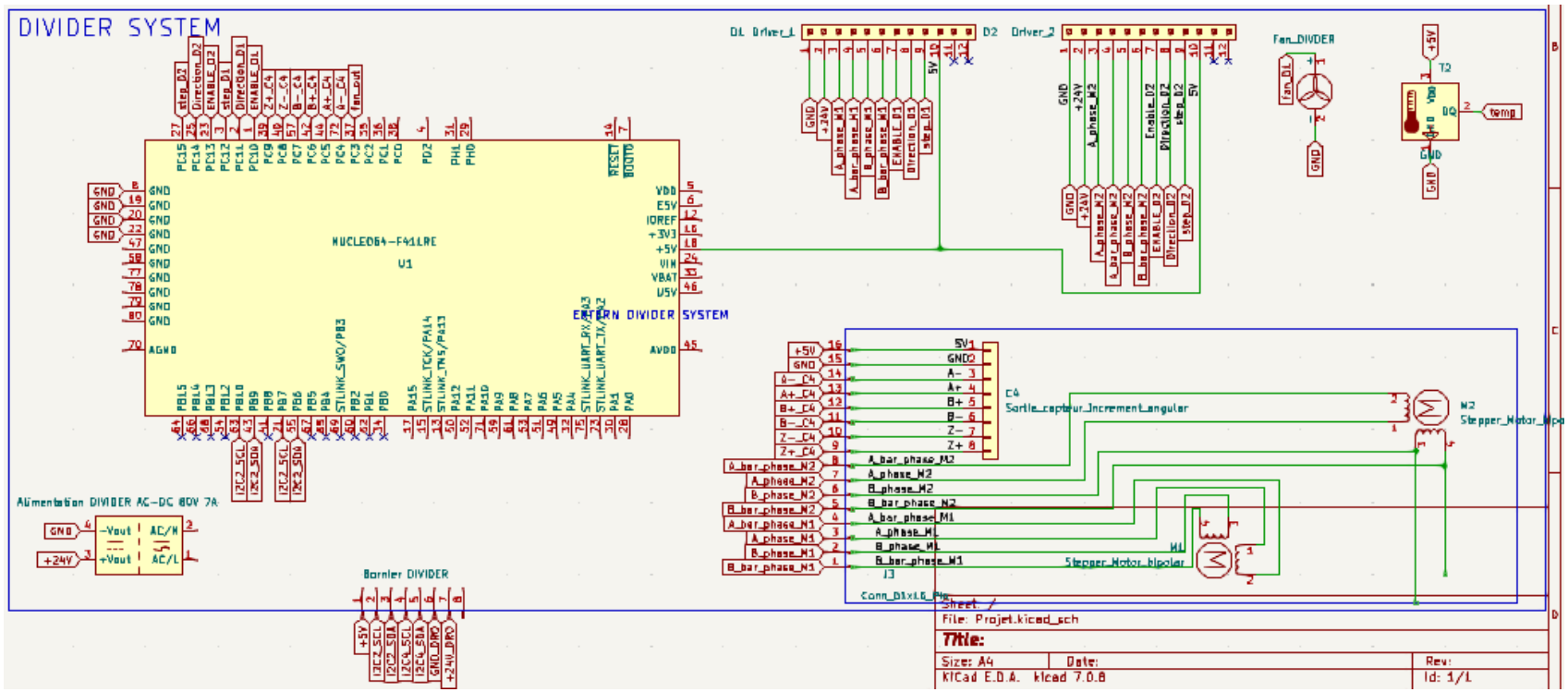
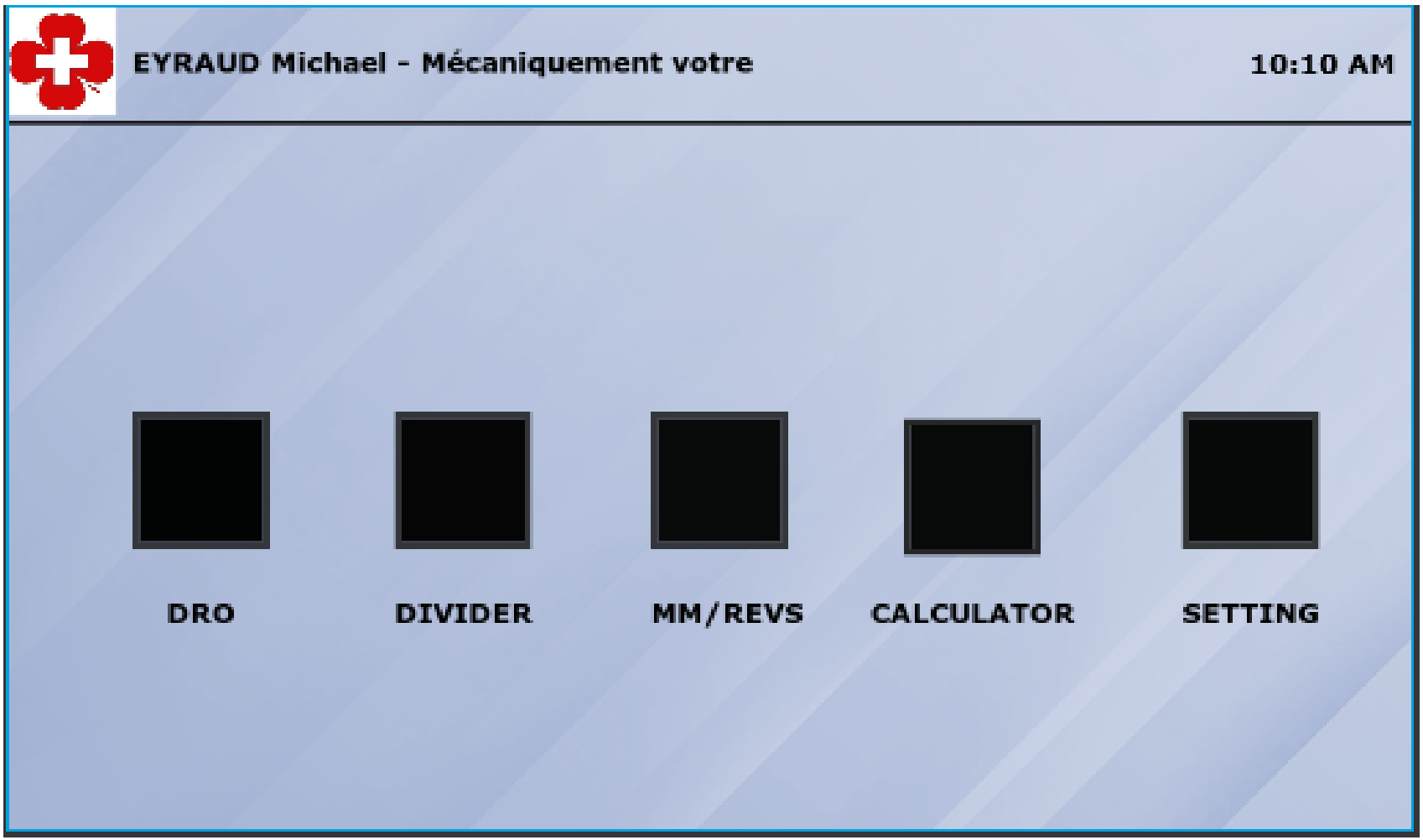


Figure: Schéma électrique du système DIVIDER

Interface Graphique (sous TouchGFX)

Interface graphique MENU

On réalise sur TouchGFX l'interface graphique, puis on ajoutera le code des fonctions par la suite (le code de chaque prototypes en 1 dossier de code unique). Cette interface respecte le cahier des charges du client.



Interface graphique DRO

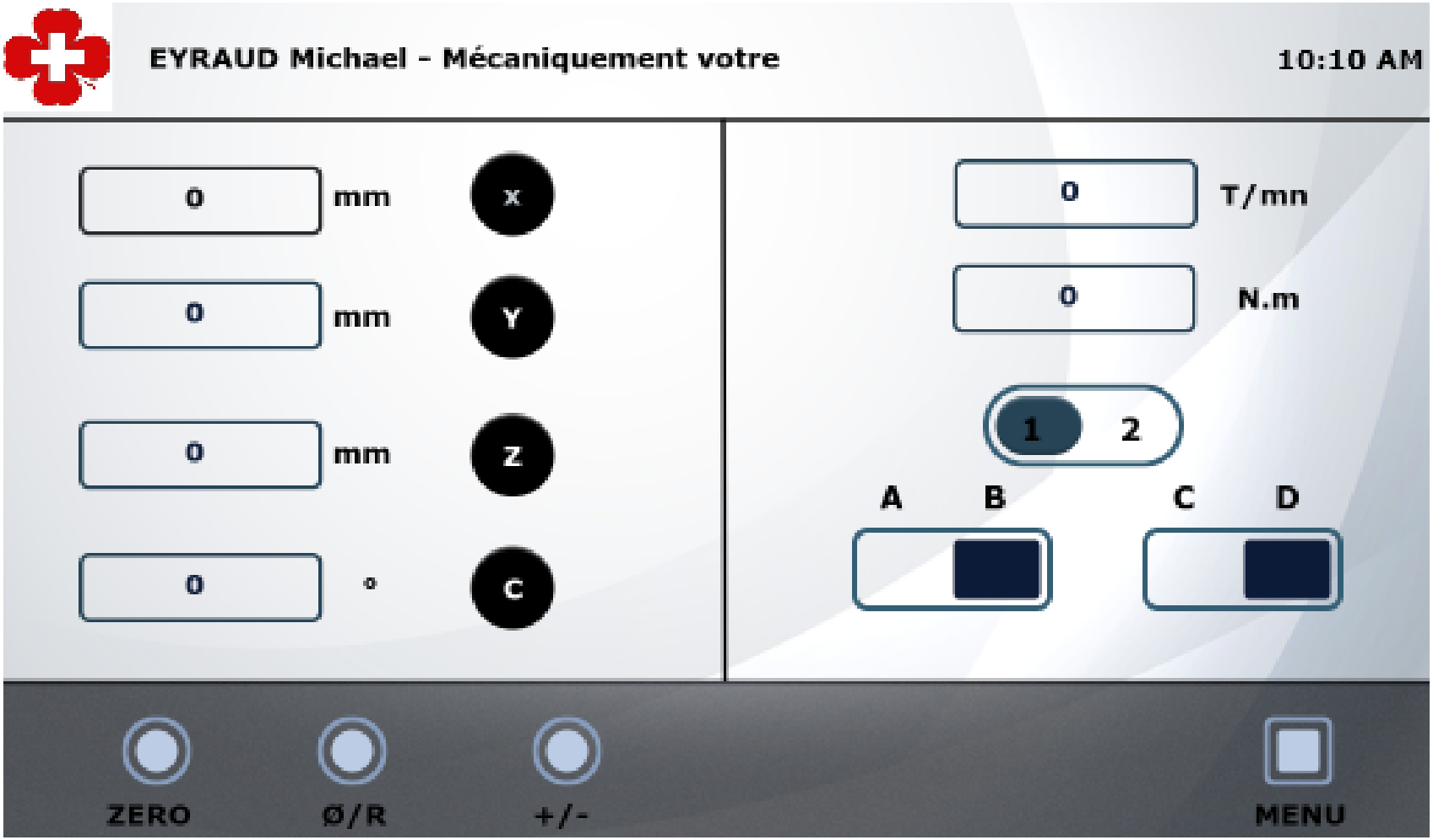


Figure: Interface graphique du DRO

Interface graphique DIVIDER

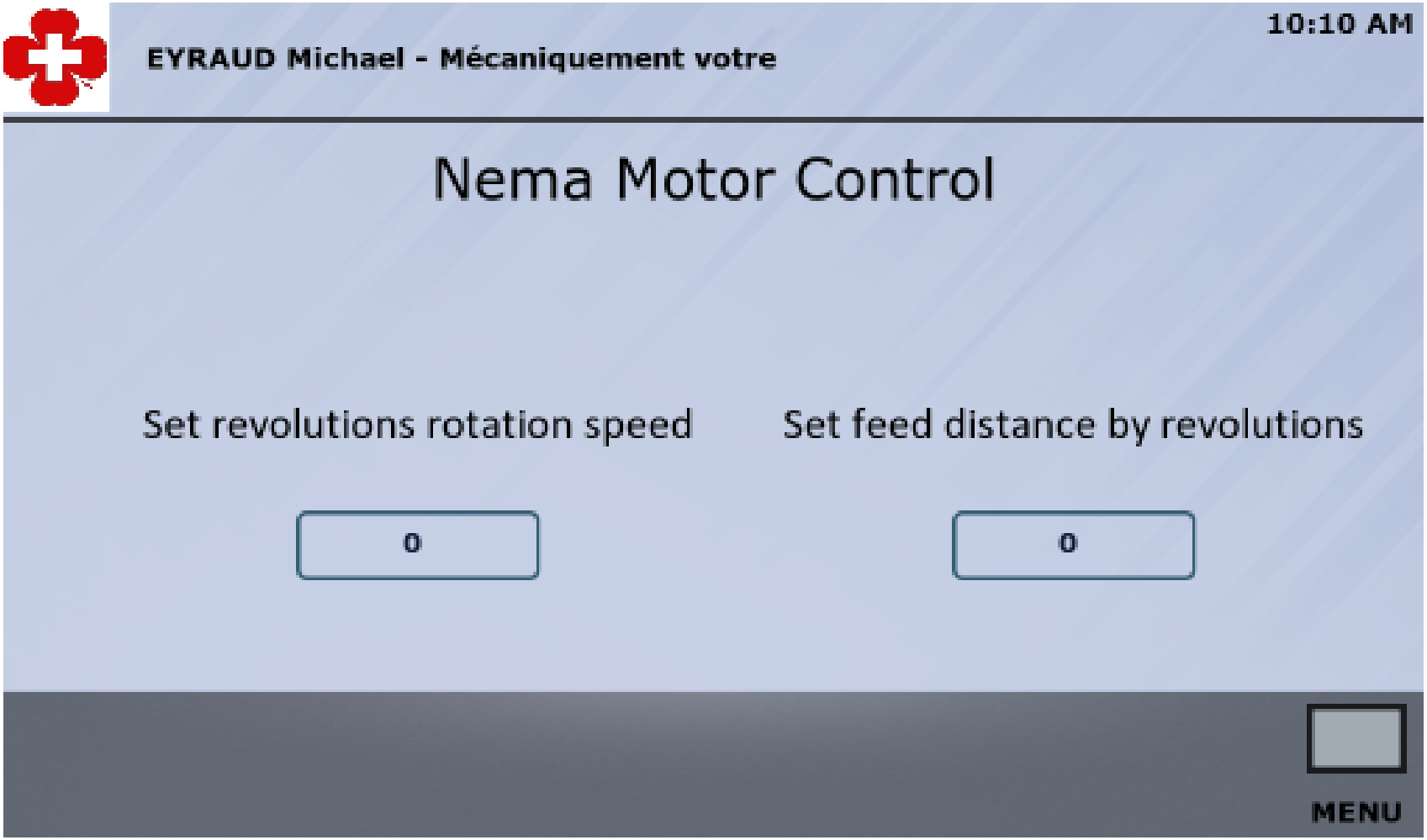


Figure: Interface graphique du DIVIDER

Interface graphique MMREVS

Figure: Interface graphique du MMREVS

Développement du prototype I2C

Communication entre deux STM32

Il existe différentes façons de communiquer entre deux microcontrôleurs :

- UART (USART)

- SPI (Serial Pe- ripheral Interface)

- I2C (Inter - Integreted Circuit)

Nous comparons les différentes façons de communiquer avec les deux cartes.

Criteria	UART	SPI	I2C
Advantages	Simplicity of coding, commonly used, fewer pins required than SPI.	Duplex communication, speed, flexibility, no collision, practical for displaying sensor information.	Uses only 2 wires, supports multiple masters, ACK/NACK bits, widely used.
Disadvantages	No acknowledgment bits, slower than SPI.	Requires more pins, limited to short distances, no acknowledgment protocols, often a single master, compatibility issues with different operation modes.	Slower than SPI, protocol overhead.

Table 3: Comparison between the different existing technologies with sufficient industrial credibility

Figure: I2C Master/Slave

Principe de fonctionnemnt

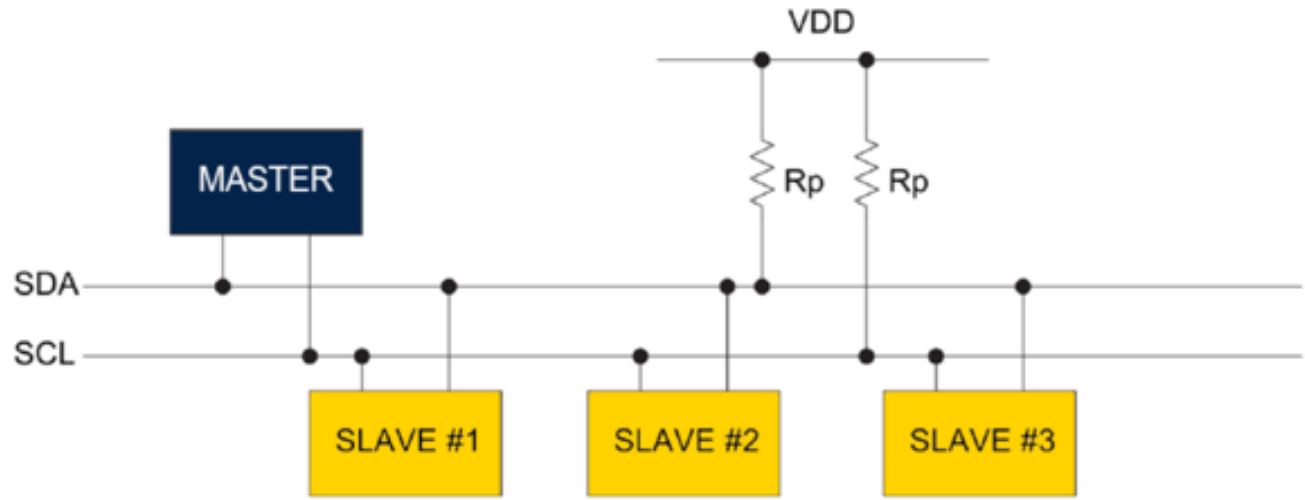


Figure: Principe de fonctionnement de L'I2C

[b]

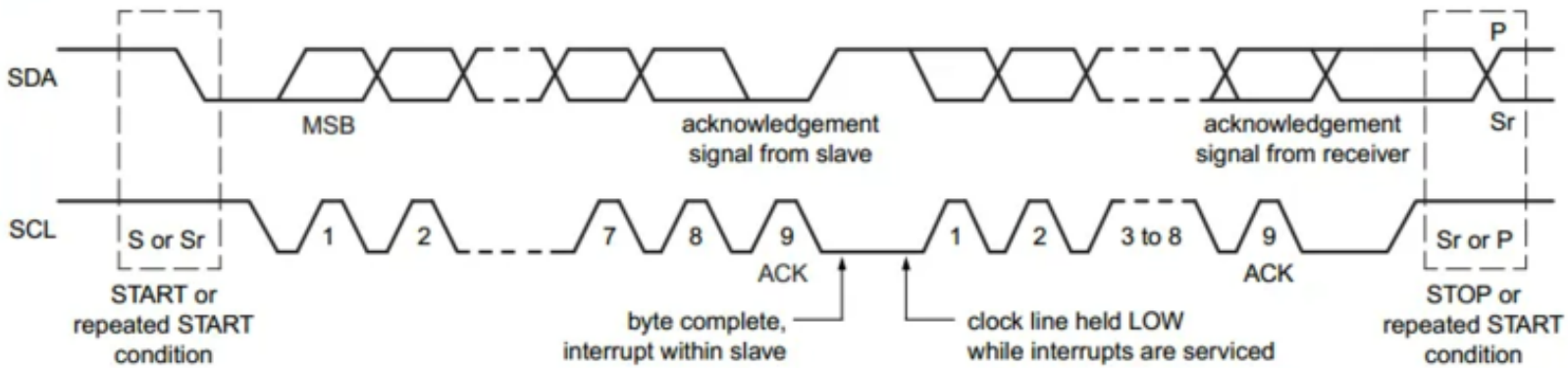


Figure: Chronogramme de SDA et SCL

Buffers communication I2C

```
uint8_t data[] = "Hello Slave!";
```

Figure: Buffer que l'ont envoie au slave

[b]

```
/* Private variables ---------- */
I2C_HandleTypeDef hi2c1;

/* USER CODE BEGIN PV */
uint8_t receivedData[20];
/* Private variables */
#define SLAVE_ADDRESS 0x08
/* USER CODE END PV */
```

Figure: Code de transmission (master)

Code de transmission

```
/* USER CODE BEGIN 3 */
  HAL_I2C_Master_Transmit(&hi2c1, SLAVE_ADDRESS << 1, data, sizeof(data), HAL_MAX_DELAY);
  HAL_Delay(1000);
```

Figure: Transmission de données (Master)

[b]

```
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
    HAL_I2C_Slave_Receive(&hi2c1, receivedData, sizeof(receivedData), HAL_MAX_DELAY);
}
```

Figure: Reception de données (slave)

Configuration I2C pour le Master et le slave

On fait un adressage pour le slave mais pas pour le master.

```
 USER CODE END I2C1_Init 1 */
.2c1.Instance = I2C1;
.2c1.Init.Timing = 0x00707CBB;
.2c1.Init.OwnAddress1 = 0;
.2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
.2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
.2c1.Init.OwnAddress2 = 0;
.2c1.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
.2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
.2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
 (HAL_I2C_Init(&hi2c1) != HAL_OK)
```

Figure: Configuration Master

```
hi2c1.Instance = I2C1;
hi2c1.Init.ClockSpeed = 100000;
hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
hi2c1.Init.OwnAddress1 = 36;
hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
hi2c1.Init.OwnAddress2 = 0;
hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
```

Debugger

Développement du prototype moteurs

Génération des signaux pour les moteurs

Nous générons un code qui contrôlera les drivers. Pour cela, nous allons générer des signaux de commandes "STEP" (signal PWM), "ENABLE" et "DIRECTION". Nous utilisons le bloc du registre pour créer le signal PWM.
Nous devons tout d'abord connaitre certaines formules.

$$\text{TIM CLOCK} = \frac{\text{APB TIM CLOCK}}{\text{PRESCALAR}}$$

$$\text{FREQUENCY} = \frac{\text{TIM CLOCK}}{\text{ARR}}$$

$$\text{DUTY \%} = \frac{\text{CCRx}}{\text{ARR}} \times 100$$

Figure: Formules des timers

Configuration des horloges

TIM1 est lié à APB2. On identifie : APB TIM clock = 100 MHz.

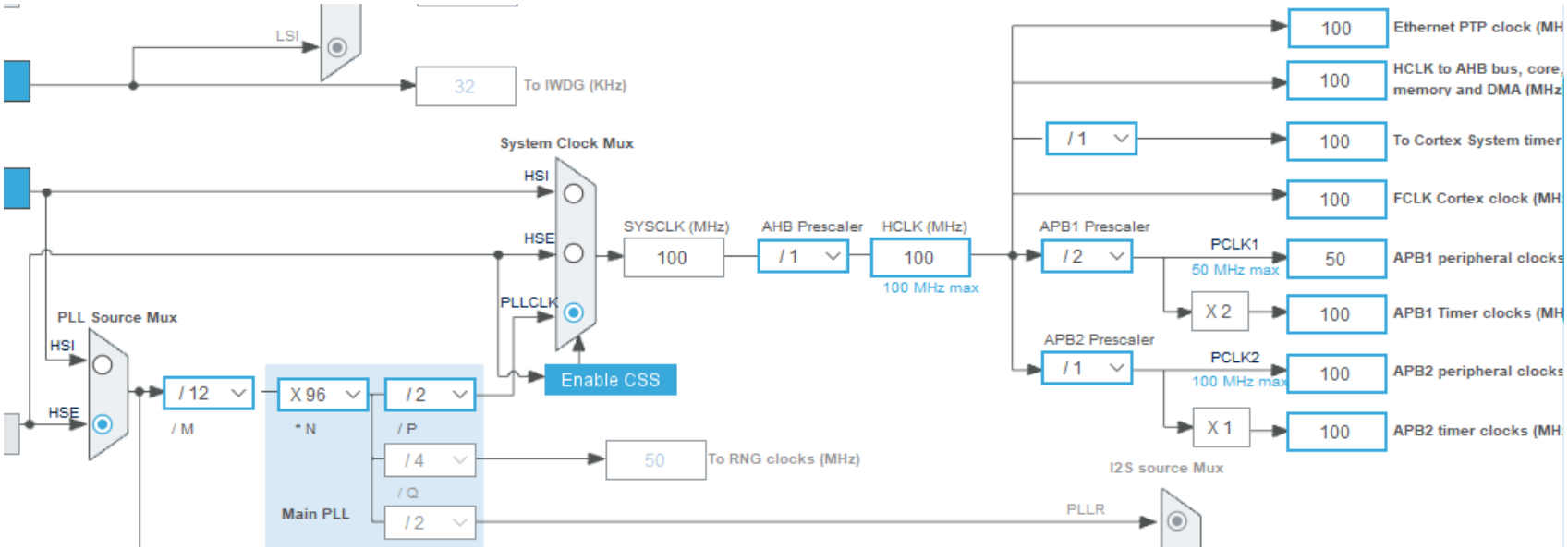


Figure: Configuration de l'horloge

Configuration de TIMER 1

TIM clock = APB TIM clock / PRESCALER
TIM clock = 100 Mhz / 100 = 1 Mhz Donc on paramètre
PRESCALER à : 100-1 (= 99) Frequency = 1 Mhz / ARR
Frequency = 1Mhz / 100 = 10 kHz

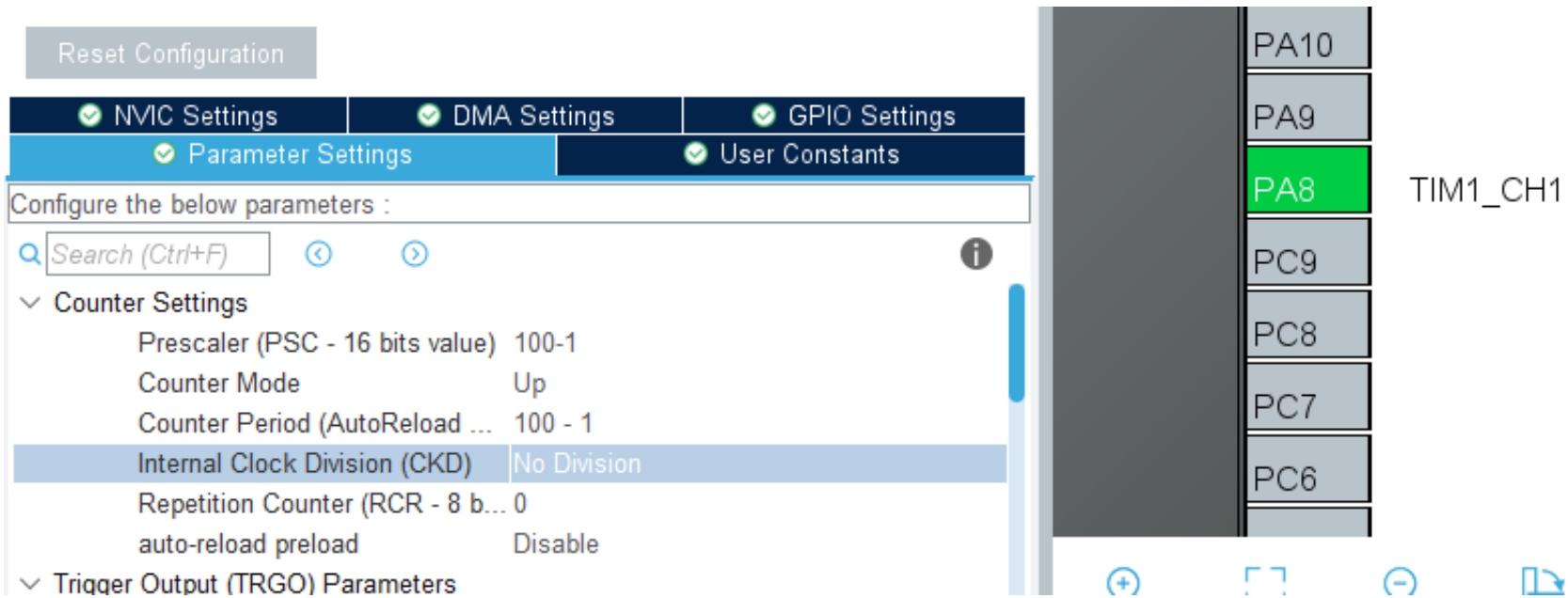


Figure: Configuration du TIMER 1

Plus de détails sur le PRESCALER

14.4.11 TIM1 and TIM8 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in "reset mode").

Figure: Formules des timers

Visualisation du signal PWM "STEP"

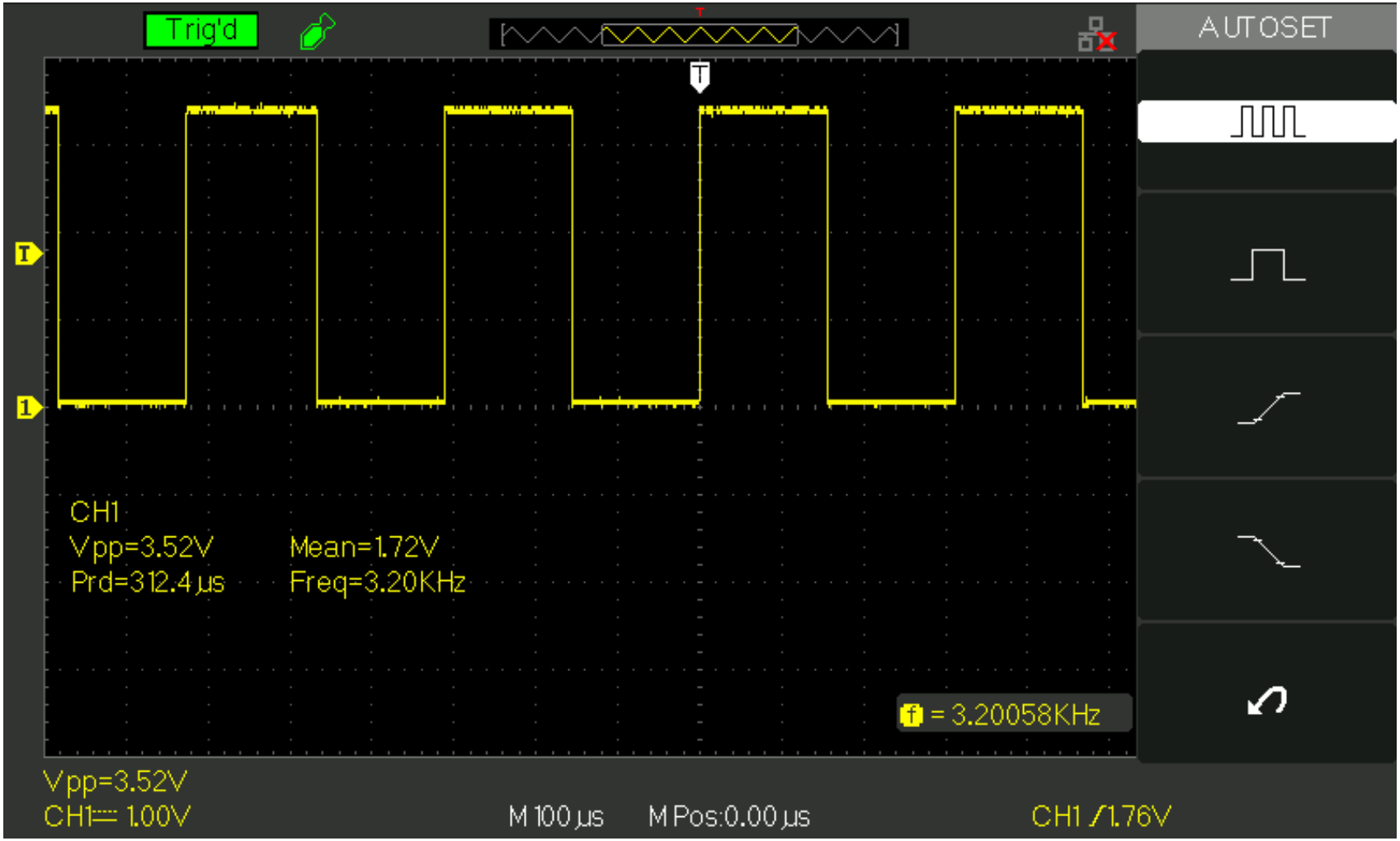


Figure: Visualisation de "STEP"

On génère également les signaux "ENABLE" et "DIRECTION" qui sont des signaux binaires.

Test du prototype moteur

Prototype capteur à Incrementation

Les signaux de commandes

Tous les signaux ont un signal complémentaire afin de réduire les interférences électromagnétiques (EMI) et d'améliorer la qualité de la communication et la robustesse du signal, en particulier sur les longs câbles.

- Signal A : Le signal A+ est un signal carré qui change d'état (haut/bas) à intervalles réguliers lorsque le codeur tourne. le codeur tourne.
- Signal B : le signal B+ est un signal carré déphasée de 90 degrés par rapport au signal A+. Le déphasage de 90 degrés entre A+ et B+ est utilisé pour déterminer le sens de rotation. Si A+ précède B+, la rotation se fait dans un sens, et si B+ précède A+, la rotation se fait dans le sens opposé.
- Signal Z : Le signal Z+ est un signal carré qui génère une seule impulsion par rotation complète du codeur.
  Le signal d'index (Z+) est utilisé pour fournir une référence absolue à la position, ce qui permet de recalibrer ou de synchroniser la position avec le codeur.

Conclusion et suite du projet