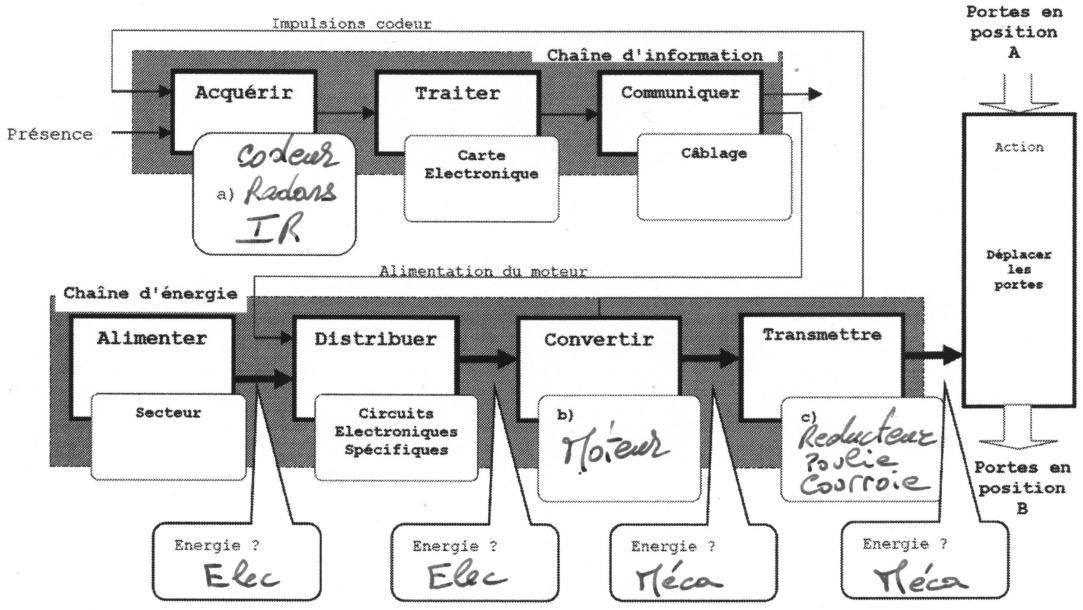
Document réponse 1 – Étude fonctionnelle

Q1.



Q2.

$d = k_2 \alpha_2$; $\alpha_2 = k_1 \alpha_M$; $N = k_3 \alpha_M$ (2)

$d = k_1 k_2 \alpha_M$ (1)

$\alpha_M = \frac{d}{k_1 k_2}$ (1) ; $\alpha_M = \frac{N}{k_3}$ (2)

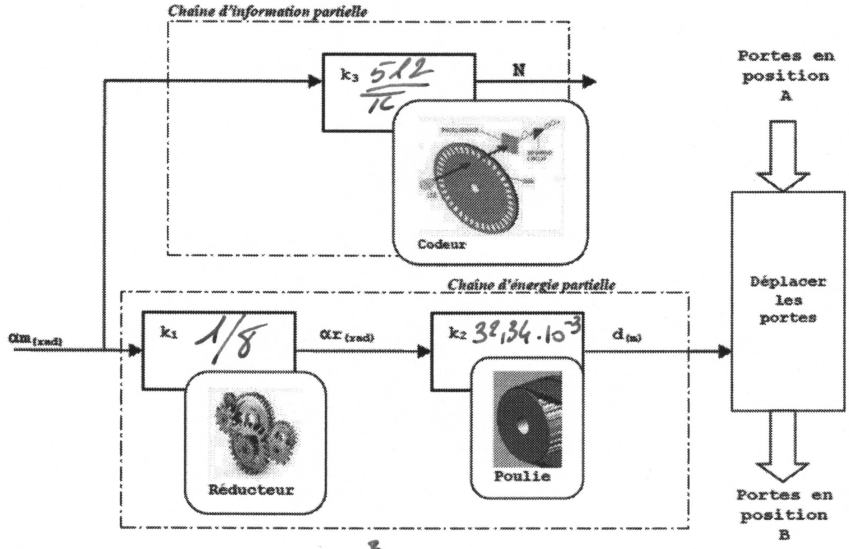
$$d_{(m)} = \frac{k_1 . k_2}{k_3} N$$

Q3.

$k_1 = 1/8$ (valeur du "reducteur" en Annexe 1)

$k_2 = R = \frac{D}{2} = 32{,}34 \times 10^{-3}$ m

$k_3 = \frac{N}{2\pi} = \frac{1024}{2\pi} = \frac{512}{\pi}$



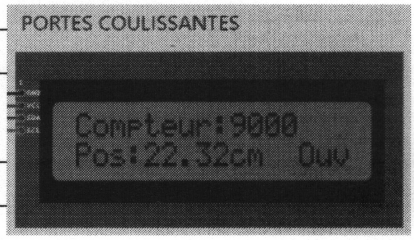
Q4.

$$k = \frac{k_1 k_2}{k_3} = \frac{\frac{32{,}34 \times 10^{-3}}{8}}{\frac{512}{\pi}} = \frac{32{,}34 \times 10^{-3}}{8} \times \frac{\pi}{512} = 24{,}8 . 10^{-6}$$

$$d(m) = 24{,}8 . 10^{-6} N$$

Document réponse 2 – Programmation

b. R1, R2 : resistances de Pull-up.
Elles forcent l'entrée du µc a l'état 1 en l'absence de signal.



c. A <=> CLK B <=> DT

d. Sens 1 : Counterclockwise (Anti-horaire)
Sens 2 : clockwise (horaire)
Sens 1 et sens 2 font référence a l'axe du moteur

e. SW : Contact ac fermeture
Rôle : délivre un etat logique ("1" ou "0")

g.
```
// Variables globales
int compteur = 0;
bool sens = LOW; // LOW ou HIGH
float position_1 = 0.0, position = 1.0;
```

f.
```
void setup() {
  // Configuration des broches du codeur
  pinMode(CODEUR_CLK, INPUT );
  pinMode(CODEUR_DT, INPUT );
  pinMode(CODEUR_SW, INPUT_PULLUP );
```

h.
```
// Sous-programme (service)d'interruption
// Déclenché par un front descendant du signal A
// On incrémentera et décrémentera le compteur par 1000 pour avoir des résultats réalistes
en cm.
void lectureCodeur() {
  bool etatLogiqueCODEUR_DT = digitalRead (CODEUR_DT); // Lecture du signal DT
  if (etatLogiqueCODEUR_DT == HIGH) { // si Rotation Horaire alors
    compteur = compteur + 1 ; // Incrémentation de compteur
    sens = HIGH ;//+1000 pour la simulation
  }
  else { // si Rotation Anti-Horaire alors
    compteur = compteur - 1 ; // Décrémentation de compteur
    sens = LOW ; //-1000 pour la simulation
  }
}
```

i.

```
// Sous-programmes et fonctions
//-------------------------------------------------------------
// Lecture de la variable compteur, désactivation des interruptions.
// Ainsi, on est sûr que la valeur contenue dans compteur
// ne changera pas pendant qu'on la lit.
int lireCompteur() {
  int valeurCompteur;
  noInterrupts(); // début protection de la variable compteur
  valeurCompteur = compteur;
  interrupts(); // fin protection de la variable compteur
  return valeurCompteur;
}
```

j.

```
int lireSens() {
  // A compléter
  int valeurSens;
  noInterrupts()
  valeurSens = sens
  interrupts();
  return valeurSens
}
```

k.

```
void razCompteur() {
  noInterrupts();
  compteur = 0;
  interrupts();
}
```

l. Imprimer et joindre la copie du code de **loop()**

m. Testez le programme.

Appel prof

Validation du travail **Oui** ☐ **Partiel** ☐ **Non** ☐