



ECOLE  
**POLYTECHNIQUE**  
DE BRUXELLES

COMPUTING PROJECT  
PROJ-H402

---

## RAPPORT DE PROJET

---

*Auteur*  
Haythem BOUGHARDAIN

*Superviseurs*  
Quentin DELHAYE  
François QUITIN



ANNÉE ACADÉMIQUE 2021–2022

## **TABLE DES MATIÈRES**

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Choix technologiques</b>	<b>1</b>
2.1	Tirage du fil	1
2.2	Coupe du fil	3
2.3	Code et circuit	5
<b>3</b>	<b>Duplication du prototype</b>	<b>6</b>
<b>4</b>	<b>Utilisation du prototype</b>	<b>7</b>
<b>5</b>	<b>Améliorations éventuelles du prototype</b>	<b>8</b>
<b>6</b>	<b>Conclusion</b>	<b>8</b>
<b>A</b>	<b>Code</b>	<b>9</b>

## 1 INTRODUCTION

---

Dans le cours de COMPUTING PROJECT, il est demandé aux étudiants de choisir un sujet parmi une liste imposée, et de mener à bien le projet qui en découle. Dans le cadre de ce rapport, le projet choisi est celui d'un prototype permettant la coupe automatique de fil électrique, à travers l'utilisation d'un microcontrôleur ainsi que d'un système d'alimentation et de coupe de fil. Ce rapport va ainsi passer sur les choix technologiques qui ont été effectués durant l'année académique, la manière par laquelle le prototype final pourra être dupliqué, comment celui-ci pourra être utilisé, et enfin les améliorations éventuelles qui pourraient lui être apportées.

## 2 CHOIX TECHNOLOGIQUES

---

Avant de se lancer dans la décision des choix technologiques du prototype, il est important d'imaginer les parties principales du système qui sera mis en place. Concrètement, il est nécessaire de gérer en premier lieu un mécanisme qui pourra tirer le fil électrique à partir d'une bobine, et diriger ce fil vers un autre mécanisme servant à le couper. C'est pourquoi le projet est ainsi séparé en deux parties principales : le tirage, et la coupe du fil électrique.

### 2.1 Tirage du fil

---

Pour tirer le fil, le système imaginé au début du projet était très proche de celui utilisé dans les imprimantes 3D pour le même objectif : un moteur pas-à-pas NEMA 17 est monté d'un mécanisme d'extrusion permettant d'utiliser le mouvement rotatif de la tige pour faire coulisser le fil électrique le long d'un écrou. Un exemple de celui-ci peut être observé dans la Figure 1 ci-dessous :



FIGURE 1 – Exemple de mécanisme d'extrusion sur un moteur NEMA 17

Ainsi, plusieurs choix sont nécessaires avant d'aboutir à un système abouti : il faut d'abord choisir un moteur pas-à-pas adéquat, ainsi qu'un driver de moteur pas-à-pas compatible, et enfin un mécanisme d'extrusion.

Le choix du moteur s'est porté sur un moteur pas-à-pas de la gamme NEMA 17 mentionnée plus haut, en particulier le 42SH38-4A (Figure 2). Celui-ci peut fournir un couple de 0,36Nm, et demande une tension d'alimentation de 2,8V ; raisons pour lesquelles il a été

choisi étant donné qu'un grand couple n'est pas nécessaire dans le mécanisme d'extrusion présenté.



FIGURE 2 – Moteur pas-à-pas 42SH38-4A

Un driver pour moteur pas-à-pas est aussi nécessaire afin de créer le lien entre ce dernier et le microcontrôleur. Ici, c'est le L293D qui est choisi, qui est simplement un composant servant de pont H, permettant de contrôler le moteur pas-à-pas utilisé.



FIGURE 3 – Driver pour moteur pas-à-pas L293D

Enfin, le mécanisme d'extrusion est la dernière pièce nécessaire pour compléter la première partie du projet. En premier lieu, celui-ci avait été modélisé en 3D pour être imprimé et assemblé par la suite. Malheureusement, étant donné un manque d'expérience avec l'impression 3D ainsi que la précision requise pour chacune des pièces, cette idée a dû être abandonnée. Ainsi, des recherches ont été effectuées pour trouver un mécanisme d'extrusion directement prêt à l'achat en ligne ; le choix de celui-ci s'est finalement porté sur l'extrudeuse Bowden de la marque Redrex (Figure 4).

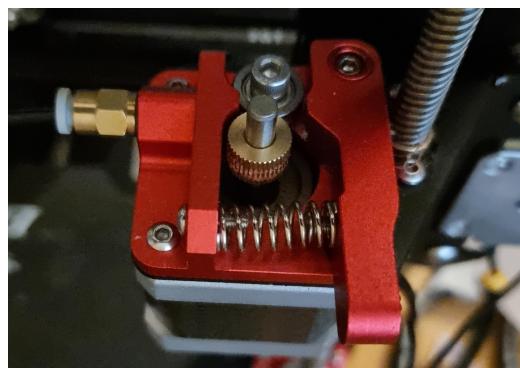


FIGURE 4 – Extrudeuse Bowden de la marque Redrex équipée sur un moteur pas-à-pas NEMA 17

En dernier lieu, un servomoteur est utilisé afin de diriger le fil dans une direction voulue (afin d'éventuellement permettre le choix d'une coupe ou d'un dénudage du fil). Celui-ci est un simple SG90, qui est peu cher et simple d'utilisation. Au final, le servomoteur se trouve dans le prototype, mais n'est pas utilisé étant donné des complications au niveau du dénudage du fil (expliquées dans la prochaine partie du rapport).

## 2.2 Coupe du fil

Afin de couper le fil tiré de la manière décrite plus haut, il a été nécessaire de trouver un mécanisme adéquat pour cette partie du prototype. Dans le cas de ce projet, celui-ci se rapproche du système bielle-manivelle : concrètement, le mouvement de rotation d'un moteur pas-à-pas est converti à l'aide de quelques pièces en un mouvement de translation, permettant de contrôler l'ouverture et la fermeture d'une pince pour couper le fil. Ce système convient au prototype et au type du projet pour plusieurs raisons : il est simple à modéliser, ne requiert pas beaucoup de pièces pour être fonctionnel, et, tenant compte d'une bonne construction du mécanisme, il est fiable.

Le premier choix à faire était celui de la pince. En effet, avant l'achat de celle-ci, il était important de garder en tête le fait qu'idéalement, le prototype devrait pouvoir couper ou dénuder du fil. C'est pourquoi le choix de la pince s'est porté sur une pince à dénuder multi-fonctions : celle choisie porte en effet des indentations de différentes tailles, permettant notamment le dénudage de fils de certains diamètres, ce qui peut être remarqué dans la Figure 5. Un test en magasin avec le fil fourni pour le projet a été fait pour vérifier la compatibilité de la pince.



FIGURE 5 – Pince à dénuder multi-fonctions

En ce qui concerne le moteur pas-à-pas choisi, étant donné qu'un code était fonctionnel pour un autre moteur pas-à-pas de la même gamme dans le prototype et qu'un driver compatible avait aussi été trouvé plus tôt, un NEMA 17 a été choisi, plus particulièrement le 17HS19-2004S1. Il possède des dimensions similaires au 42SH38-4A, mais sa principale différence vient du couple de 0,59Nm qu'il peut fournir. Ceci est important étant donné qu'ici, le couple sera utile pour fournir un réel mouvement dans un système mécanique, contrairement au 42SH38-4A qui était simplement nécessaire pour tirer du fil. Une autre caractéristique importante fut la forme de la tige du moteur : en effet, dans le cas du 17HS19-2004S1, celle-ci est un cylindre tronqué sur sa hauteur. Ceci permet de créer une pièce pouvant s'insérer de manière beaucoup plus stable sur la tige du moteur, ce qui est beaucoup plus difficile sur une tige parfaitement cylindrique, et qui est nécessaire dans le cas du système bielle-manivelle employé ici. Le 17HS19-2004S1 peut être visualisé dans la Figure 6.

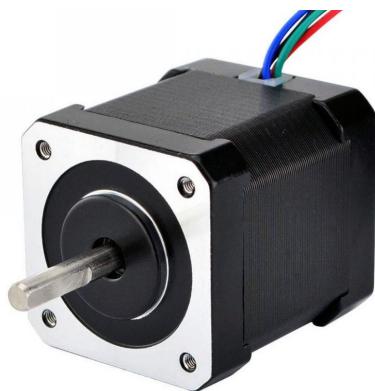


FIGURE 6 – Moteur pas-à-pas 17HS19-2004S1

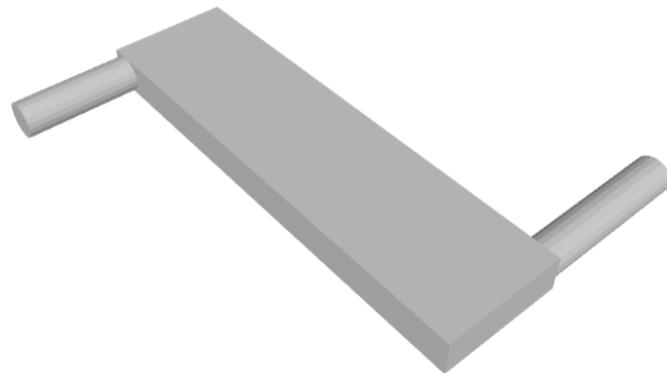
Ensuite, pour relier le moteur pas-à-pas à la pince, le système bielle-manivelle a dû être modélisé. Ici, il prend forme à l'aide de trois pièces modélisées en 3D :

- Un support placé selon la tige du moteur pas-à-pas, avec une indentation circulaire (permettant l'insertion de la manivelle).
- Une manivelle en forme de tige surmontée de deux appuis à chacune de ses extrémités.
- Un support placé sur une branche de la pince à dénuder, avec une indentation circulaire (permettant l'insertion de la manivelle).

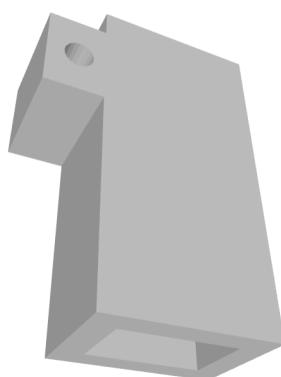
Ces trois pièces peuvent être visualisées séparément dans la Figure 7.



(a) Support du moteur pas-à-pas



(b) Manivelle



(c) Support de la pince

FIGURE 7 – Pièces modélisées en 3D formant le système bielle-manivelle

Lorsque ces pièces sont assemblées, le système bielle-manivelle présenté à la Figure 8 est obtenu. Ainsi, le support placé sur le moteur pas-à-pas est fixe dans l'espace, mais peut tourner, ce qui entraîne la manivelle liée au support placé sur la pince. Celle-ci étant elle aussi fixée, ceci permet d'assurer que le support placé ne bougera que verticalement, ouvrant ou fermant ainsi la pince.

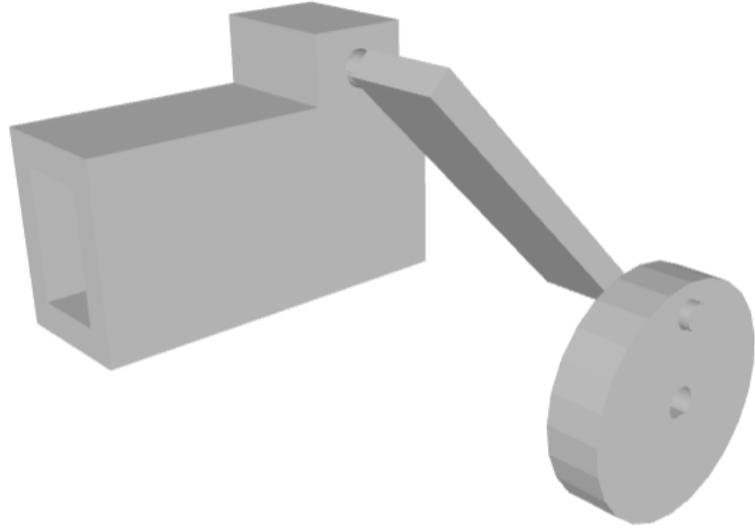


FIGURE 8 – Système bielle-manivelle

Enfin, le système de coupe de fil obtenu a pu être testé et fonctionne convenablement pour ce but. Malheureusement, l'objectif de pouvoir dénuder du fil n'a pas pu être réalisé : en effet, pour que ceci fonctionne, la position de sortie du fil doit être assez précise que pour assurer que celui-ci passera par l'indentation adéquate de la pince à dénuder. Lorsque ce n'est pas le cas, ceci bloque le système bielle-manivelle mentionné, et empêche donc le prototype de fonctionner par la suite.

### 2.3 Code et circuit

Le prototype utilise un microcontrôleur fourni par l'ULB pour faire fonctionner le programme principal ; *PSoC Creator* est utilisé pour générer le code en C. Le code créé est surtout utile pour pouvoir gérer les moteurs pas-à-pas dans le prototype ainsi que le servomoteur.

Pour les moteurs pas-à-pas, la librairie Arduino pour ce genre de moteurs a été étudiée afin de comprendre la manière de les contrôler (<https://github.com/arduino-libraries/Stepper>). Ceci a permis de savoir quelle tension logique appliquer aux broches du moteur pas-à-pas, ainsi que de connaître le sens de ces configurations afin de générer une rotation. Trois fonctions sont donc définies pour chacun des deux moteurs pas-à-pas (d'où la distinction entre `feeder` et `cutter`) : une fonction sert à faire tourner le moteur d'un pas, un autre fait tourner le moteur d'un nombre de pas spécifié en paramètre, et la dernière permet de faire un certain nombre de tours dans le cas du moteur `cutter` ou une certaine distance en mm dans le cas du moteur `feeder` (dans ce dernier cas, il a été mesuré expérimentalement qu'un tour du moteur de tirage faisait avancer 31 mm de fil). Dans ces fonctions, un Timer est utilisé afin de gérer la fréquence à laquelle les pas sont faits par les moteurs pas-à-pas ; par défaut, cette fréquence vaut 100Hz, donc un moteur pas-à-pas peut faire 100 pas par seconde.

Pour le servomoteur, une simple fonction `toggle_servo_position()` est utilisée pour définir si la position du servomoteur devrait permettre la coupe (SERVO\_CUT) ou le dénudage (SERVO\_STRIP) du fil. Pour cela, un *pulse-width modulator* (PWM) est utilisé.

Le `main()` contient les initialisations des composants, ainsi qu'une boucle infinie du fonctionnement du programme : celui-ci fait simplement tourner le moteur de tirage du fil d'une certaine longueur définie dans le code, puis le moteur de coupe fait une rotation qui permet de couper le fil.

Le code source du projet, ainsi que les ressources telles que les fichiers nécessaires à PSoC Creator ainsi que les fichiers `.stl` pour les pièces modélisées en 3D se trouvent tous sur le dépôt GitHub suivant : <https://github.com/hboughardain/automatic-wire-cutter>. Le code peut aussi se trouver plus bas dans le rapport au Listing 1.

### 3 DUPLICATION DU PROTOTYPE

---

En ce qui concerne la duplication du prototype, il est en premier lieu important d'énoncer une liste complète de tous les composants utilisés :

- un microcontrôleur (fourni par l'ULB dans ce cas) ;
- un moteur pas-à-pas 42SH38-4A ;
- un moteur pas-à-pas 17HS19-2004S1 (ainsi qu'un support de montage permettant de tenir le moteur horizontalement) ;
- deux drivers pour moteur pas-à-pas L293D ;
- un kit d'extrusion de la marque Redrex adapté à un moteur pas-à-pas NEMA 17 ;
- un servomoteur SG90 ;
- deux breadboards ;
- du fil électrique ;
- une pince à dénuder multi-fonctions (achetée via Cotubex) ;
- les pièces à imprimer en 3D mentionnées plus haut.

Le prototype final obtenu peut être visualisé ci-dessous :

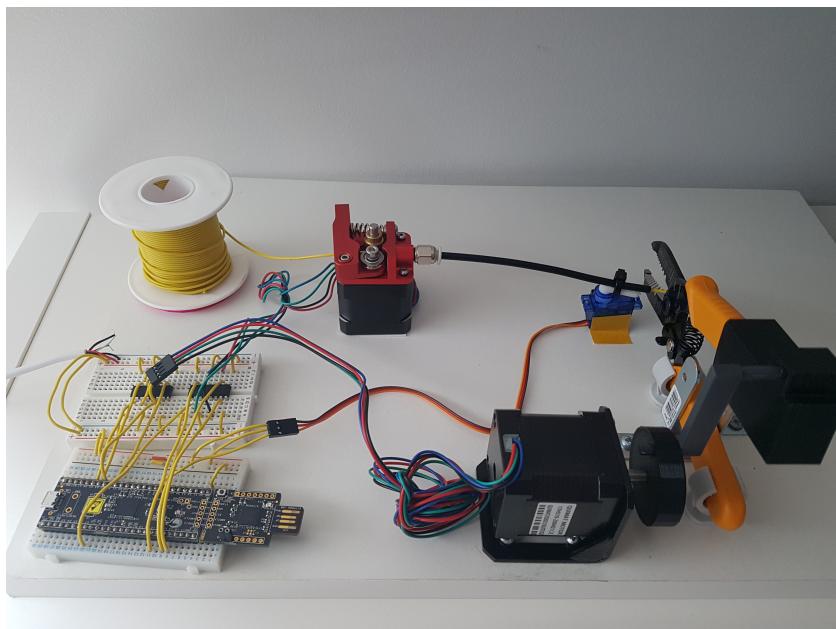


FIGURE 9 – Prototype final

Le circuit électrique se compose notamment du microcontrôleur connecté aux deux drivers de moteur pas-à-pas, eux-mêmes connectés aux moteurs de coupe et de tirage selon les connexions décrites dans les datasheets des composants.

Une bobine de fil électrique est liée au premier moteur pas-à-pas qui se charge du tirage ; le fil passe dans un tube dirigé par un servomoteur et est amené vers la pince à dénuder. Cette dernière est liée au deuxième moteur pas-à-pas par un système de bielle-manivelle construit à l'aide de pièces imprimées en 3D. Le tout tient sur une planche en bois.

À part les pièces mentionnées dans la liste plus haut, des pièces sont nécessaires pour construire le petit dispositif permettant de tenir la pince à dénuder verticalement. Ici, ce sont simplement des supports de construction utilisés : deux de forme circulaire pour assurer que la pince tient verticalement, et deux en forme d'angle perpendiculaire pour assurer que la pince ne se déloge pas horizontalement.

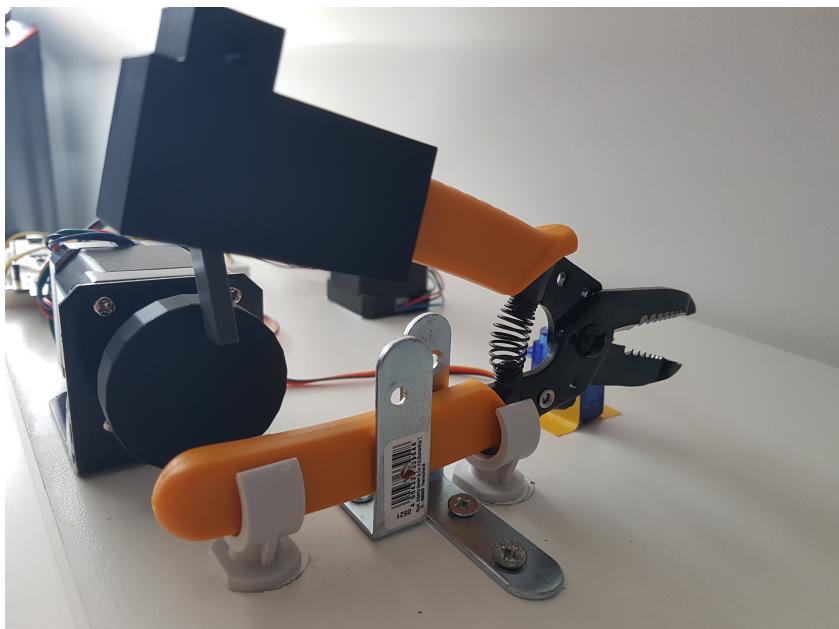


FIGURE 10 – Point de vue rapproché sur la pince à dénuder et le système bielle-manivelle

## 4 UTILISATION DU PROTOTYPE

Pour utiliser le prototype, il faut en premier lieu brancher celui-ci à deux sources d'alimentation :

- sur le breadboard dans les colonnes dédiées à l'alimentation de 5V et la masse ;
- sur le microcontrôleur via une des entrées USB présentes (soit le Micro-USB, soit l'USB-A).

Lorsque le prototype est alimenté, celui-ci démarre automatiquement, et tirera et coupera du fil électrique fourni par la bobine de manière continue jusqu'à ce que l'alimentation du prototype soit interrompue manuellement.

Éventuellement, il est aussi possible de modifier le code du microcontrôleur en utilisant PSoC Creator ; notamment en modifiant l'argument de `turn_feeder_motor()` dans le `main()`, étant donné que celui-ci indique la longueur en mm du fil qui sera tiré. Ainsi, en modifiant ce paramètre, il est possible d'avoir des longueurs de fil coupé de longueur arbitraire.

## **5 AMÉLIORATIONS ÉVENTUELLES DU PROTOTYPE**

---

Enfin, même si le prototype final obtenu permet de réaliser la tâche élémentaire de couper du fil automatiquement, celui-ci est loin d'être parfait, et pourrait bien entendu recevoir de nombreuses améliorations.

La plus évidente serait de pouvoir permettre la fonction de dénudage, comme cela avait été prévu au début du projet. À l'instant actuel, cette fonction n'a pas été activée dans le code du microcontrôleur, car elle ne complète sa tâche que rarement, étant donné la position très précise que le fil doit avoir à la sortie du tube issu du moteur à tirer. Une amélioration dans ce cas-ci serait éventuellement de choisir un tube plus fin par lequel passerait le fil et de rapprocher à l'extrême la pince à dénuder de sa sortie pour essayer d'assurer une précision maximale, et permettre la fonction de dénudage.

Si ceci est réalisable, il est alors aussi plausible de penser à une adaptation des épaisseurs de fils à dénuder. En effet, la pince à dénuder multi-fonctions choisie possède des indentations de nombreuses tailles pour différents types de fils. S'il est possible d'atteindre une précision permettant le dénudage d'une certaine épaisseur de fil, alors il devrait pouvoir en découler l'option de choisir ce paramètre avant de lancer l'exécution du prototype.

Enfin, une idée qui avait été discutée au début du projet fut celle de l'ajout d'un composant avec une interface graphique, permettant une interaction en temps réel avec l'utilisateur lors du fonctionnement du prototype. Ce composant pourrait par exemple prendre la forme d'un petit écran tactile où s'afficheraient les paramètres tels que la longueur du fil à couper dans l'état actuel du prototype, mais aussi la longueur à dénuder et l'épaisseur du fil traité dans le cas des améliorations discutées plus haut.

## **6 CONCLUSION**

---

En conclusion, ce projet a permis aux étudiants concernés de s'immiscer dans l'ensemble des domaines de l'informatique, de l'électronique et de l'électromécanique dans le but de réaliser un produit permettant la coupe automatique de fil électrique. En effet, celui-ci est passé par diverses étapes telles que la modélisation et l'impression de pièces en 3D, la programmation d'un microcontrôleur, et l'assemblage manuel d'un prototype. Le résultat final permet la coupe de fil électrique de manière automatisée, simple et précise. Malgré le fait que des améliorations soient bien entendu possibles, telles que le choix du dénudage du fil, l'objectif principal du projet a pu être atteint.

## A CODE

---

```
1 #include "project.h"
2
3 #define STEP_ANGLE 1.8           // value of the step angle from the stepper
4   motor
5 #define PERIOD 2                // desired period for a full rotation of
6   stepper motor shaft
7 #define SERVO_CUT 1750          // PWM value to put servomotor in position
8   to cut wire
9 #define SERVO_STRIP 2600         // PWM value to put servomotor in position
10  to strip wire
11
12 int servo_position = SERVO_CUT;
13
14 /*
15 ** Moves the feeder motor by a single step.
16 ** The sequence of control signals for 4 control wires is as follows:
17 /**
18 ** Step P1  P2  P3  P4
19 ** 1    1    0    1    0
20 ** 2    1    0    0    1
21 ** 3    0    1    0    1
22 ** 4    0    1    1    0
23 /**
24 ** (inspired by Arduino stepper motor library
25 ** https://github.com/arduino-libraries/Stepper/blob/master/src/Stepper.cpp)
26 */
27 void step_feeder_motor(int step) {
28     switch (step) {
29         case 0:
30             FeederPin1_Write(1);
31             FeederPin2_Write(0);
32             FeederPin3_Write(1);
33             FeederPin4_Write(0);
34             break;
35         case 1:
36             FeederPin1_Write(1);
37             FeederPin2_Write(0);
38             FeederPin3_Write(0);
39             FeederPin4_Write(1);
40             break;
41         case 2:
42             FeederPin1_Write(0);
43             FeederPin2_Write(1);
44             FeederPin3_Write(0);
45             FeederPin4_Write(1);
46             break;
47         case 3:
48             FeederPin1_Write(0);
49             FeederPin2_Write(1);
50             FeederPin3_Write(1);
51             FeederPin4_Write(0);
52             break;
53     }
54 }
```

```

52  ** Moves the feeder motor by the specified number of steps
53  ** This method uses a timer to create a big enough delay between steps
54  ** The timer has a 1 kHz frequency; by using counters, this allows to
55  ** modulate the frequency as wanted. By default, the frequency is 100 Hz.
56 */
57 void feeder_motor(int steps_to_move) {
58     int steps_left = steps_to_move;
59     int counter = 0;
60     while (steps_left > 0) {
61         if ((StepperMotorTimer_STATUS_TC &
62              StepperMotorTimer_ReadStatusRegister()) != 0) {
63             counter++;
64             if (counter == 5 * PERIOD) {
65                 step_feeder_motor((steps_to_move - steps_left) % 4);
66                 steps_left--;
67                 counter = 0;
68             }
69         }
70     }
71 }
72 /*
73 ** Moves the feeder motor by the specified length (in mm)
74 ** by using the measured step angle.
75 ** This was measured by testing and concluding that
76 ** one rotation of the feeder motor yields 31 mm of wire.
77 */
78 void turn_feeder_motor(int length) {
79     feeder_motor((360 / STEP_ANGLE) * length / 31);
80 }
81
82 /*
83 ** Moves the cutter motor by a single step.
84 ** The sequence of control signals for 4 control wires is as follows:
85 /**
86 ** Step P1  P2  P3  P4
87 ** 1   1   0   1   0
88 ** 2   1   0   0   1
89 ** 3   0   1   0   1
90 ** 4   0   1   1   0
91 ** (inspired by Arduino stepper motor library
92 ** https://github.com/arduino-libraries/Stepper/blob/master/src/Stepper.cpp)
93 */
94 void step_cutter_motor(int step) {
95     switch (step) {
96         case 0:
97             CutterPin1_Write(1);
98             CutterPin2_Write(0);
99             CutterPin3_Write(1);
100            CutterPin4_Write(0);
101            break;
102        case 1:
103            CutterPin1_Write(1);
104            CutterPin2_Write(0);
105            CutterPin3_Write(0);
106            CutterPin4_Write(1);
107            break;
108        case 2:

```

```

109     CutterPin1_Write(0);
110     CutterPin2_Write(1);
111     CutterPin3_Write(0);
112     CutterPin4_Write(1);
113     break;
114 case 3:
115     CutterPin1_Write(0);
116     CutterPin2_Write(1);
117     CutterPin3_Write(1);
118     CutterPin4_Write(0);
119     break;
120 }
121 }
122 /*
123 ** Moves the cutter motor by the specified number of steps
124 ** This method uses a timer to create a big enough delay between steps
125 ** The timer has a 1 kHz frequency; by using counters, this allows to
126 ** modulate the frequency as wanted. By default, the frequency is 100 Hz.
127 */
128 void cutter_motor(int steps_to_move) {
129     int steps_left = steps_to_move;
130     int counter = 0;
131     while (steps_left > 0) {
132         if ((StepperMotorTimer_STATUS_TC &
133             StepperMotorTimer_ReadStatusRegister()) != 0) {
134             counter++;
135             if (counter == 5 * PERIOD) {
136                 step_cutter_motor((steps_to_move - steps_left) % 4);
137                 steps_left--;
138                 counter = 0;
139             }
140         }
141     }
142 }
143 /*
144 ** Moves the cutter motor by the specified number of rotations
145 ** by using the measured step angle.
146 */
147 void turn_cutter_motor(int rotations) {
148     cutter_motor((360 / STEP_ANGLE) * rotations);
149 }
150 /*
151 ** Changes the servomotor's position between cutting or stripping stance.
152 */
153 void toggle_servo_position() {
154     if (servo_position == SERVO_CUT) {
155         servo_position = SERVO_STRIP;
156     } else if (servo_position == SERVO_STRIP) {
157         servo_position = SERVO_CUT;
158     }
159     PWM_WriteCompare(servo_position);
160     CyDelay(100);
161 }
162 int main(void) {

```

```
166  
167     CyGlobalIntEnable;  
168     StepperMotorTimer_Start();  
169     PWM_Start();  
170     PWM_WriteCompare(SERVO_CUT);  
171  
172     for(;;) {  
173         CyDelay(1000);  
174         turn_feeder_motor(50);  
175         CyDelay(100);  
176         turn_cutter_motor(1);  
177         CyDelay(3000);  
178     }  
179 }
```

Listing 1 – Code source programmé dans le microcontrôleur