

Campus Félix Leclerc

**Rapport final**

Projet synthèse en TSO

*(247-601-HU)*

Session Automne 2016

Benoît Houle

Présenté à

Bilal Manaï

Technologies du génie électrique

9 juillet 2020

# Introduction

Pour le projet synthèse du DEC de la technique en systèmes ordinés, il faut réaliser un projet de façon individuel. Une partie de la session d’automne est alloué pour la planification et la majorité de la session d’hivers est dédié pour la réalisation. Il n’y a pas de limite pour les méthodes utilisées, tant qu’elles sont sécuritaires et que le résultat diffère le moins possible de l’objectif final. Ce projet représente une synthèse des connaissances acquises.

Mon projet est d’adapter le contrôleur d’une machine CNC pour en faire un graveur de circuit intégré. Le tout va permettre à un utilisateur d’envoyer un dessin sous le format d’un fichier Gerber et de le voir se faire imprimer.

Grâce à un crayon à encre magnétique, le graveur va pouvoir faire plusieurs types de dessin, par exemple relier des lumières sur du linge sans avoir de fils encombrant.

Ce projet se différencie principalement par le type de dessin réalisable. Ce qui est possible de dessiner est un fichier Gerber, donc un dessin cartographique, qui possède des coordonnées horizontales et verticales.

Il se différencie aussi d’une simple machine à dessin ou d’un simple graveur avec sa capacité à se connecter à un réseau et d’être disponible à travers une page web.

Ce rapport commence par la production du périphérique réalisé pour remplacer l’ancien contrôleur. Il y a ensuite la programmation utilisé pour faire fonctionner ce périphérique. Exceptionnellement, une section expliquant l’impact du COVID-19 est présente. La dernière section inclue une analyse critique du projet.

Table des matières

[Introduction 1](#_Toc45173035)

[Diagramme schématique 3](#_Toc45173036)

[Diagramme matriciel 4](#_Toc45173037)

[Coût du projet 5](#_Toc45173038)

[Programmation 6](#_Toc45173039)

[Organigramme 6](#_Toc45173040)

[Contrôleur 6](#_Toc45173041)

[Serveur 7](#_Toc45173043)

[UML 8](#_Toc45173044)

[Pseudo code 9](#_Toc45173045)

[Serveur 9](#_Toc45173046)

[Contrôleur 10](#_Toc45173047)

[Listage 11](#_Toc45173048)

[Serveur 11](#_Toc45173049)

[Contrôleur 24](#_Toc45173050)

[Discussion 34](#_Toc45173051)

[Impact du COVID-19 34](#_Toc45173052)

[Problèmes rencontrés 34](#_Toc45173053)

[Les DRV8834 34](#_Toc45173054)

[La vitesse des moteurs 35](#_Toc45173055)

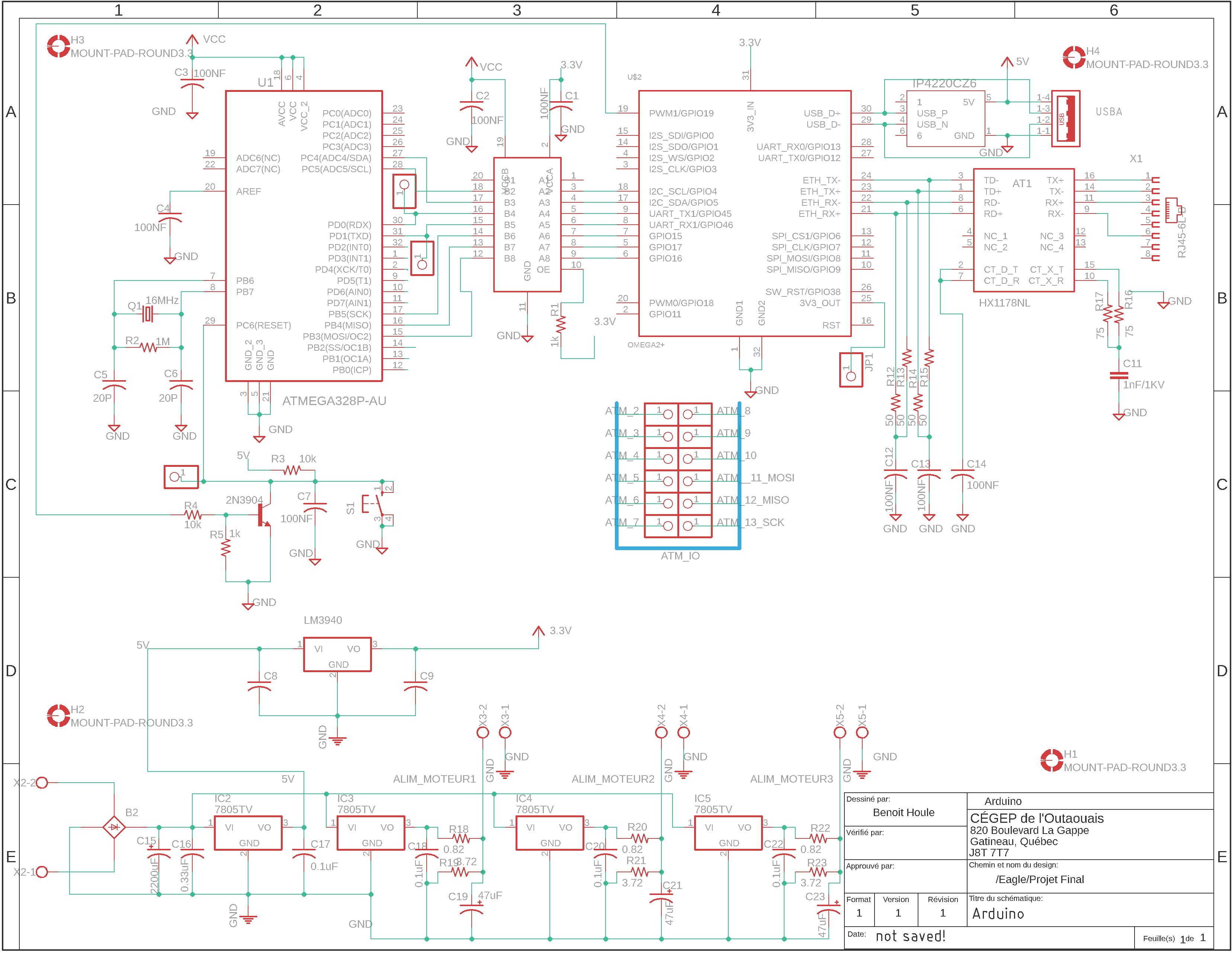
[Définition d’un constructeur « overloaded » 35](#_Toc45173056)

[Tests effectués 35](#_Toc45173057)

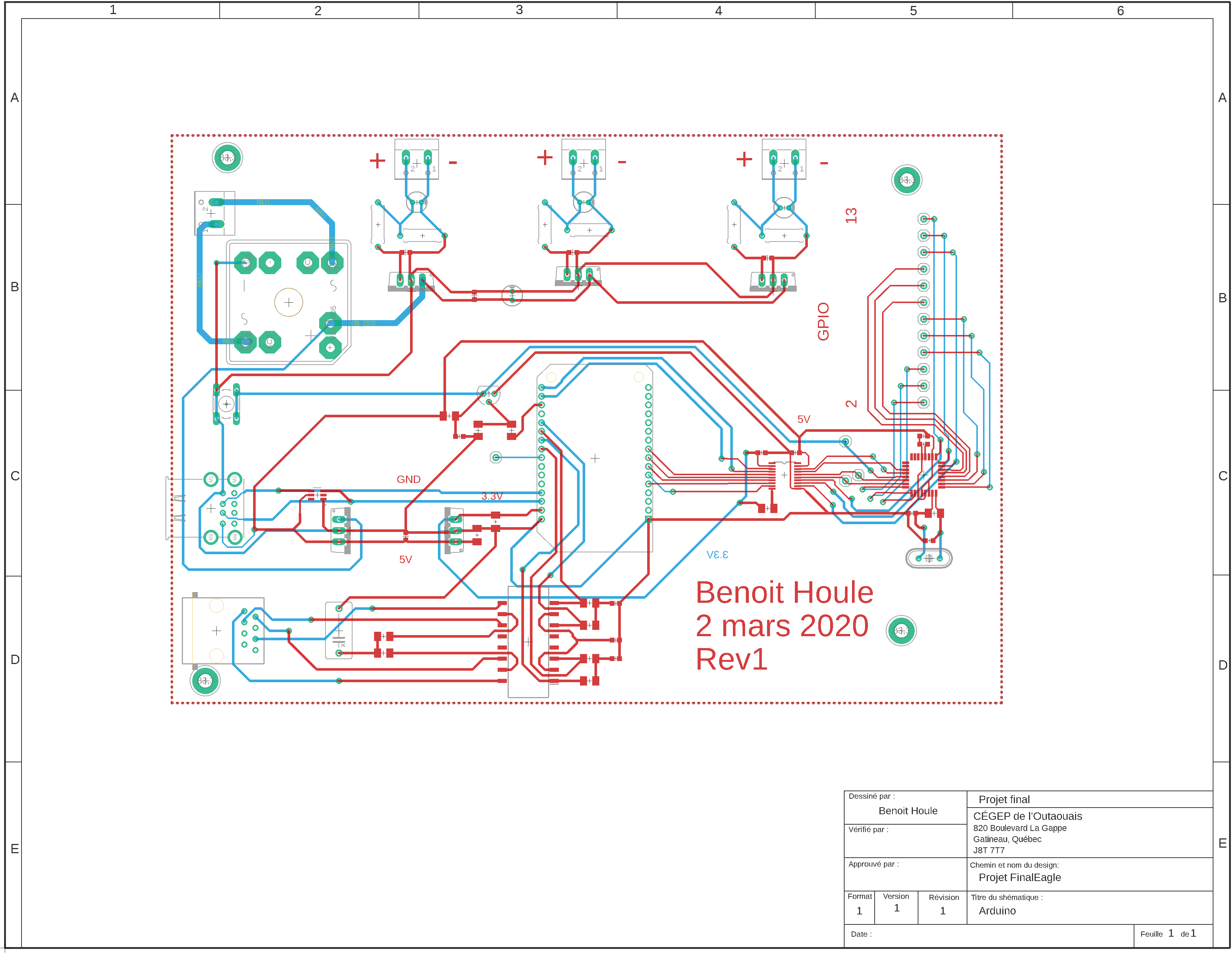
[Améliorations possibles 36](#_Toc45173058)

[Conclusion 37](#_Toc45173059)

# Diagramme schématique



# Diagramme matriciel



# Coût du projet

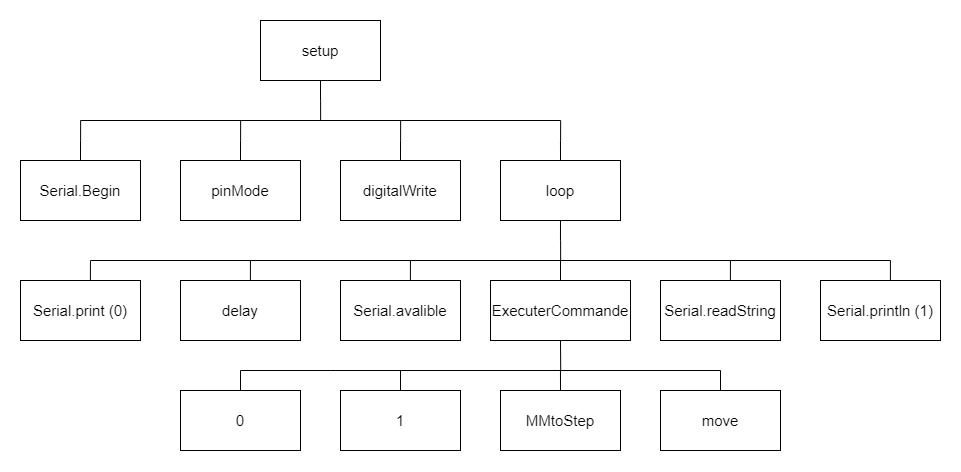
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pièce | Numéro de pièce | Empaquetage | Description | Prix (CAD$) |
| HX1178NL | AT1 | SOP254P1283X647-16N | Transformateur de signal 10/100Base-T | 10.1 |
| IP4220CZ6 | IP4220CZ6 | DIODE\_SOT457\_1P7X3P1-M | - | 0.56 |
| CRYSTALHC49S | Q1 | HC49/S | 16MHz CRYSTAL | 0.55 |
| OMEGA2+ | O1 | OMEGA2 | Onion Omega2+ | 15.16 |
| ATMEGA328P-AU | U1 | QFP80P900X900X120-32N | 8-bit Microcontrôleur | 2.76 |
| USB\_2P85-32003-00X-1 | USB | USB2 | Prise USB | 2.39 |
| TXS0108E | U2 | TSSOP20 | 8-BIT Traducteur de tension bidirectionnel | 0.73 |
| RJ45-6L-B | X1 | RJ45-NO-SHIELD | Port Ethernet | 10.32 |
| 7805TV | IC2,IC3,IC4,IC5 | TO220V | - | 8.2 |
| 7824TV | LM3940 | TO220V | - | 2.07 |
| Condensateurs & résistances | Valeur | Empaquetage | |  |
| C1, C2, C3, C4, C7, C12, C13, C14 | 100NF | C0603K | | 0.73 |
| C5, C6 | 20P | C0603K | | 0.9 |
| C11 | 1nF/1KV | XC10B6 | | 0.54 |
| C15 | 2200uF | SANYO-OSCON | | 1.16 |
| C16 | 0.33uF | C0603K | | 0.52 |
| C17, C18, C20, C22 | 0.1uF | C0603K | | 0.66 |
| C19, C21, C23 | 47uF | SANYO-OSCON | | 0.99 |
| R1 | 1k | M1206 | | 0.18 |
| R2 | 1M | M1206 | | 0.19 |
| R3, R4 | 10k | M1206 | | 0.36 |
| R5 | 1k | M3216 | | 2.23 |
| R12, R13, R14, R15 | 50 | M1206 | | 3.85 |
| R16, R17 | 75 | M1206 | | 0.66 |
| R18, R20, R22 | 0.82 | 0309/10 | | 1.54 |
| R19, R21, R23 | 3.72 | 0309/10 | | 1.28 |
|  |  |  | Total en dollars Canadien | 68.63 |

# Programmation

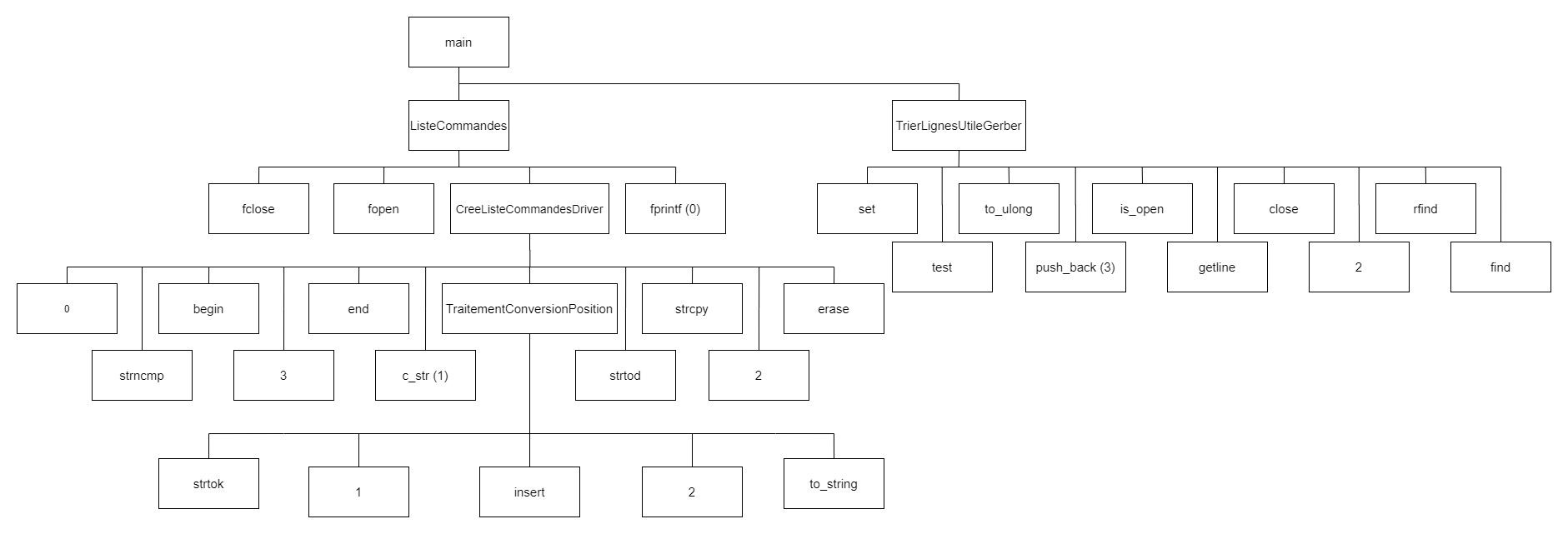
## Organigramme

### Contrôleur

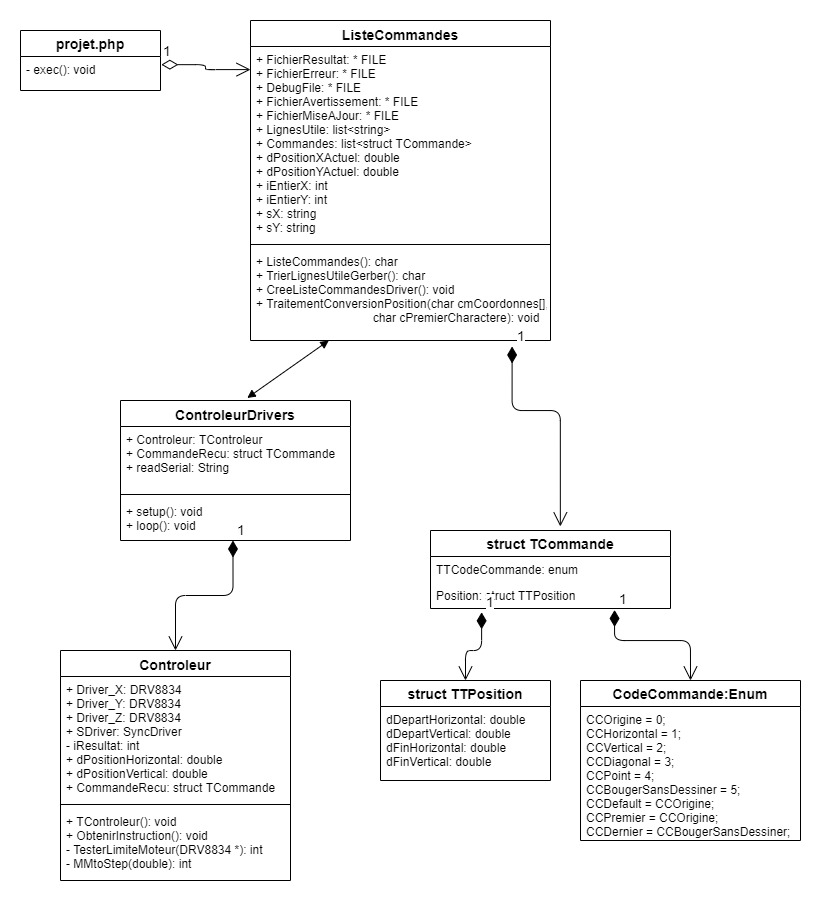
### 



### Serveur



## UML



## Pseudo code

### Serveur

Initialiser les modules utilisés.

Démarrer le serveur.

Quand un client demande la page pour upload un fichier

Envoyer la page pour upload un fichier.

Quand un client demande la page pour voir les mises à jour

Envoyer la page pour voir les mise à jour.

Quand un client demande la page pour crée un utilisateur

Envoyer la page pour crée un utilisateur.

Quand un client upload un fichier

Obtenir le nom d’utilisateur et le mot de passe qui est avec le fichier.

Si le chemin vers le fichier et la combinaison du nom et du mot de passe est valide,

Exécuter le programme pour obtenir la chaine de commande.

Ajouter le résultat a la liste d’attente pour faire le dessin.

Retourner une validation au client.  
 Sinon

Retourner un message d’erreur au client.

Quand un client demande les mises à jour

Calculer le nombre de dessin total.

S’il y a au moins un dessin à faire

Envoyer au contrôleur la prochaine commande à faire

Retourner les mises à jour au client.   
 Sinon

Retourner un message d’erreur au client.

Quand un client demande de crée un utilisateur

Si le nom et le mot de passe est valide

Créer un fichier avec le nom d’utilisateur et le mot de passe comme nom

Si le fichier c’est bien crée

Retourner une validation et une confirmation au client.

Sinon

Retourner un message d’erreur au client.

Sinon

Retourner un message d’erreur au client.

### Contrôleur

*Code principale :*

Initialiser les GPIOs utilisés.

Commencer la communication sériel.

Exécuter la fonction pour tester la limite des moteurs.

Faire une boucle infinie

Quand de l’information est reçu par le port sériel

Traduire l’information reçu en commande.

Exécuter la commande reçu.

Fin de la boule infinie.

*Code pour exécuter une commande :*

Si une coordonnée dépasse sa limite

Réduire la coordonnée à la limite.

Mettre un code d’erreur dans la valeur retourné.

S’il faut faire un retour à l’origine

Calculer les déplacements nécessaires.

Déplacer les moteurs.

Sinon s’il faut dessiner un point

Calculer les déplacements nécessaires.

Déplacer les moteurs.

Descendre le crayon.

Remonter le crayon.

Sinon

Si on a besoin de déplacer les moteurs avant de dessiner

Calculer les déplacements nécessaires.

Déplacer les moteurs.

S’il faut dessiner une ligne

Calculer les déplacements nécessaires.

Descendre le crayon.

Déplacer les moteurs.

Remonter le crayon.

Retourner les erreurs a la fonction appelante.

## Listage

### Serveur

#### Main.cpp

/\*! @file main.cpp

@version 1.00

@brief Description général de la fonction du programme.

@internal

projet Gerber (Gerber)

@note Note au programmeur.

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-05-16

@author Benoit Houle

@note Entrée du code.

<hr width="100%" height="5">

\*/

//\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using namespace std;

#include "ListeCommandes.h"

#include <cstdlib>

#include <cstdio>

#include <string>

//\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*! @brief Description générale de la fonction du programme.

@param argc Nombre d'arguments sur la ligne de commande.

@param argv Matrice de pointeurs vers les chaînes de caractères de chaque argument sur la ligne de commande.

@return Code d'erreur.

@retval EXIT\_SUCCESS(0) Aucune erreur.

@retval EXIT\_FAILURE(1) Échec.

@internal

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-05-16

@author Benoit Houle

@note Entrée du code.

<hr width="100%" height="5">

\*/

int main( int argc, char\*\* argv )

{

if ( argc == 2 )

{

ListeCommandes( argv[1] );

}

else cerr << "<p>Erreur dans le nom du fichier.</p>";

return EXIT\_SUCCESS;

}

#### ListeCommandes.h

/\*! @file ListeCommandes.h

@version 1.00

@brief Déclaration pour pouvoir contrôler des moteurs.

@internal

projet Gerber (Gerber)

@note Note au programmeur.

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-05-16

@author Benoit Houle

@note Entrée du code.

<hr width="100%" height="5">

\*/

//\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <string>

#include <bitset>

#include <cstdlib>

#include <cstdio>

#include <iostream>

#include <fstream>

#include <cstring>

#include <list>

#ifndef LISTECOMMANDES\_H

#define LISTECOMMANDES\_H

#ifdef \_\_cplusplus

extern "C"

{

#endif

#include "Commande.hpp"

#define MILIM 0

#define POUCE 1

char ListeCommandes( string FichierGerberAOuvrir );

char TrierLignesUtileGerber( string FichierGerberAOuvrir );

void CreeListeCommandesDriver( );

void TraitementConversionPosition( char cmCoordonnes[], char cPremierCharactere );

#ifdef \_\_cplusplus

}

#endif

#endif /\* LISTECOMMANDES\_H \*/

#### ListeCommandes.cpp

/\*! @file ListeCommandes.cpp

@version 1.00

@brief Implémentation pour pouvoir contrôler des moteurs.

@internal

projet Gerber (Gerber)

@note Note au programmeur.

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-05-16

@author Benoit Houle

@note Entrée originale du code.

<hr width="100%" height="5">

\*/

//\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using namespace std;

#include "ListeCommandes.h"

FILE \* FichierResultat;

FILE \* DebugFile;

list<string> LignesUtile;

list<struct TCommande> Commandes;

double dPositionXActuel = 0.0;

double dPositionYActuel = 0.0;

//Précision des coordonées

int iEntierX = 0;

int iEntierY = 0;

//Coordoné horizontal avant d'etre transtypé en double

string sX;

//Coordoné vertical avant d'etre transtypé en double

string sY;

//\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*! @brief Fonction principale.

@return Code d'erreur

@retval 0000 0001 Le type d'unité est précisé

@retval 0000 0010 La précision des coordonées

@retval 0000 0100 Si la fin du fichier est valide

@retval 0000 1000 ERREUR - Plus d'un type d'unité est précisés

@retval 0001 0000 ERREUR - Plus d'une précision pour les coordonées est précisés

@retval 0010 0000 ERREUR - Il y a plus d'une fin de fichier

@retval 0100 0000 ERREUR - Le fichier Gerber n'a pas ouvert

@internal

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-05-16

@author Benoit Houle

@note Entrée du code.

<hr width="100%" height="5">

\*/

char ListeCommandes ( string FichierGerberAOuvrir )

{

FichierResultat = fopen ( "/Gerber/FichierResultat", "w+" );

DebugFile = fopen ( "/Gerber/DebugFile", "w+" );

char cResultat = TrierLignesUtileGerber ( FichierGerberAOuvrir );

//Si tous les éléments néssécaires sont présent

if ( cResultat & 0b0000'0001 &&

cResultat & 0b0000'0010 &&

cResultat & 0b0000'0100 &&

!( cResultat & 0b0100'0000 ) )

{

CreeListeCommandesDriver( );

list<struct TCommande>::iterator IteratorCommande = Commandes.begin( );

for ( IteratorCommande ; IteratorCommande != --Commandes.end( ); IteratorCommande++ )

{

if ( IteratorCommande->CodeCommande == CCDiagonal )

{

fprintf( FichierResultat, "Ligne diagonal : ( %.2f , %.2f ) vers ( %.2f , %.2f ).\n",

IteratorCommande->Position.dDepartHorizontal,

IteratorCommande->Position.dDepartVertical,

IteratorCommande->Position.dFinHorizontal,

IteratorCommande->Position.dFinVertical

);

fprintf( DebugFile, "Ligne diagonal : ( %.2f , %.2f ) vers ( %.2f , %.2f ).\n",

IteratorCommande->Position.dDepartHorizontal,

IteratorCommande->Position.dDepartVertical,

IteratorCommande->Position.dFinHorizontal,

IteratorCommande->Position.dFinVertical

);

cout << "3(";

}

else if ( IteratorCommande->CodeCommande == CCPoint )

{

fprintf( FichierResultat, "Point : ( %.2f , %.2f ).\n",

IteratorCommande->Position.dDepartHorizontal,

IteratorCommande->Position.dDepartVertical,

IteratorCommande->Position.dFinHorizontal,

IteratorCommande->Position.dFinVertical

);

fprintf( DebugFile, "Point : ( %.2f , %.2f ).\n",

IteratorCommande->Position.dDepartHorizontal,

IteratorCommande->Position.dDepartVertical,

IteratorCommande->Position.dFinHorizontal,

IteratorCommande->Position.dFinVertical

);

cout << "4(";

}

else

{

fprintf( FichierResultat, "Retourner a l'origine\n" );

cout << "0(";

}

cout << IteratorCommande->Position.dDepartHorizontal << "," << IteratorCommande->Position.dDepartVertical << "," << IteratorCommande->Position.dFinHorizontal << "," << IteratorCommande->Position.dFinVertical << ")" << endl;

}

fprintf( FichierResultat, "Retourner a l'origine\n" );

cout << "0(0,0,0,0)";

}

else

{

//-------------------------

//Enregistrement des erreurs.

//-------------------------

cout << "<ul>";

if ( !( cResultat & 0b0000'0001 ) )

{

//fprintf( FichierErreur, "<li>Le type d'unité n'est pas précisé.</li>\n" );

cout << "<li>Le type d'unité n'est pas précisé.</li>";

}

if ( !( cResultat & 0b0000'00010 ) )

{

//fprintf( FichierErreur, "<li>La précision des coordonées n'est pas précisé.</li>\n" );

cout << "<li>La précision des coordonées n'est pas précisé.</li>";

}

if ( !( cResultat & 0b0000'0100 ) )

{

//fprintf( FichierErreur, "<li>Il n'y a pas de fin de fichier.</li>\n" );

cout << "<li>Il n'y a pas de fin de fichier.</li>";

}

if ( cResultat & 0b0100'0000 )

{

//fprintf( FichierErreur, "<li>Le fichier n'a pas pu être ouvert.</li>\n" );

cout << "<li>Le fichier n'a pas pu être ouvert.</li>";

}

cout << "</ul>";

}

fclose ( DebugFile );

fclose ( FichierResultat );

//Retourner le code d'erreur à la routine appelante.

return cResultat;

}

/\*! @brief Prend tous les lignes utiles du fichier Gerber.

@return Code d'erreur

@retval 0000 0001 Le type d'unité est précisé

@retval 0000 0010 La précision des coordonées

@retval 0000 0100 S'il y a une fin de fichier

@retval 0000 1000 ERREUR - Plus d'un type d'unité est précisés

@retval 0001 0000 ERREUR - Plus d'une précision pour les coordonées est précisés

@retval 0010 0000 ERREUR - Il y a plus d'une fin de fichier

@retval 0100 0000 ERREUR - Le fichier Gerber n'a pas ouvert

@internal

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-05-16

@author Benoit Houle

@note Entrée du code.

<hr width="100%" height="5">

\*/

char TrierLignesUtileGerber ( string FichierGerberAOuvrir )

{

//Fichier gerber a lire

ifstream FichierGerber ( FichierGerberAOuvrir.c\_str( ) );

/\*Pour avoir un fichier valide, on doit avoir au minimum 3 lignes :

- MOMM ou MOIN pour l'unité des dimention

- FSLAX\*\*Y\*\* pour la précision des dimentions

- M02 pour la fin du fichier

S'il manque une de ces trois lignes, le fichier est invalide.

\*/

bitset<8> cCodeErreur;

if ( FichierGerber.is\_open( ) )

{

string LigneLu;

//-------------------------

//Lire tous les lignes du ficher et placer ceux qui sont utilse dans un liste

//-------------------------

while ( getline ( FichierGerber, LigneLu ) )

{

if ( LigneLu.size ( ) > 0 )

{

//Pour savoir si on bouge les moteurs

if ( LigneLu.rfind ( "D01\*" ) != -1 || LigneLu.rfind ( "D02\*" ) != -1 )

{

LignesUtile.push\_back ( LigneLu );

}

//Pour savoir si les dimentions sont en milimètres ou en pouces.

else if ( LigneLu.find ( "%MOMM" ) != -1 || LigneLu.find ( "%MOIN" ) != -1 )

{

//Si je connais déjà la précision

if ( cCodeErreur.test( 0 ) )

{

cCodeErreur.set( 3 );

}

else

{

LignesUtile.push\_back ( LigneLu );

cCodeErreur.set( 0 );

}

}

//Pour savoir la précision utilisé

else if ( LigneLu.find ( "%FSLAX" ) != -1 )

{

if ( cCodeErreur.test( 1 ) )

{

cCodeErreur.set( 4 );

}

else

{

LignesUtile.push\_back ( LigneLu );

cCodeErreur.set( 1 );

}

}

//Pour savoir si c'est la fin du fichier

else if ( LigneLu.find ( "M02\*" ) != -1 )

{

if ( cCodeErreur.test( 2 ) )

{

cCodeErreur.set( 5 );

}

else

{

LignesUtile.push\_back ( LigneLu );

cCodeErreur.set( 2 );

}

}

}

}

//Fermer le fichier

FichierGerber.close ( );

}

else cCodeErreur.set( 6 );

return cCodeErreur.to\_ulong( );

}

/\*! @brief Crée la liste de commande pour contrôler les drivers des moteurs.

@internal

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-05-16

@author Benoit Houle

@note Entrée du code.

<hr width="100%" height="5">

\*/

void CreeListeCommandesDriver ( )

{

//Type d'unité choisi. Vu que je n'ai pas la machine, je ne peu pas faire de calibration, donc je n'utilise pas différente unité.

bool Unite = MILIM;

//utilisé pour pouvoir modifier une chaine non modifiable (const)

char cmCoordonnes[256];

//L'ittérateur pour traverser dans la liste de ligne utile

list<string>::iterator IteratorLigne = LignesUtile.begin( );

//La commande à ajouter a la liste de commande

struct TCommande CommandeAAjouter;

//------------------------------------------

//Choisir si on utilise des milimètres ou des pouces

//-----------------------------------------

while ( IteratorLigne != --LignesUtile.end( ) )

{

if ( strncmp ( IteratorLigne->c\_str( ) , "%MOMM%", 5 ) == 0 )

{

fprintf( DebugFile, "Ligne MO enlever :%s\n", IteratorLigne->c\_str( ) );

LignesUtile.erase( IteratorLigne ) ;

IteratorLigne = --LignesUtile.end( );

}

else if ( strncmp ( IteratorLigne->c\_str( ) , "%MOIN%", 5 ) == 0 )

{

fprintf( DebugFile, "Ligne MO enlever :%s\n", IteratorLigne->c\_str( ) );

bool Unite = POUCE;

LignesUtile.erase( IteratorLigne ) ;

IteratorLigne = --LignesUtile.end( );

}

else IteratorLigne++;

}

//------------------------------------------

//Calculer la précision

//-----------------------------------------

IteratorLigne = LignesUtile.begin( );

while ( IteratorLigne != --LignesUtile.end( ) )

{

if ( strncmp ( IteratorLigne->c\_str( ) , "%FSLA", 5 ) == 0 )

{

iEntierX = IteratorLigne->c\_str( )[6] - '0';

iEntierY = IteratorLigne->c\_str( )[9] - '0';

fprintf( DebugFile, "Ligne enlever :%s\n", IteratorLigne->c\_str( ) );

LignesUtile.erase( IteratorLigne ) ;

IteratorLigne = --LignesUtile.end( );

}

else IteratorLigne++;

}

fprintf( DebugFile, "---------------------------\n" );

for ( list<string>::iterator Iterator = LignesUtile.begin( ); Iterator != --LignesUtile.end( ); Iterator++ )

{

fprintf( DebugFile, "%s\n", Iterator->c\_str( ) );

}

fprintf( DebugFile, "---------------------------\n" );

//------------------------------------------

//Mettre tous les déplacement des moteurs dans la liste de commande

//-----------------------------------------

for ( IteratorLigne = LignesUtile.begin( ); IteratorLigne != --LignesUtile.end( ); IteratorLigne++ )

{

fprintf( DebugFile, "%s\n", IteratorLigne->c\_str( ) );

strcpy( cmCoordonnes, IteratorLigne->c\_str( ) );

TraitementConversionPosition( cmCoordonnes, IteratorLigne->c\_str( )[0] );

fprintf( DebugFile, "D02 || D01 : %c || %c\n", IteratorLigne->c\_str( )[IteratorLigne->size( ) - 3], IteratorLigne->c\_str( )[IteratorLigne->size( ) - 2] );

//------------------------------------------

//Si les moteurs sont déjà a la bonne place

//-----------------------------------------

if ( IteratorLigne->c\_str( )[IteratorLigne->size( ) - 3] == '1' || IteratorLigne->c\_str( )[IteratorLigne->size( ) - 2] == '1' )

{

CommandeAAjouter.Position.dDepartHorizontal = dPositionXActuel;

CommandeAAjouter.Position.dDepartVertical = dPositionYActuel;

fprintf( DebugFile, "Debut----- \nsX : %f\nsY : %f\n", dPositionXActuel, dPositionYActuel );

dPositionXActuel = strtod( sX.c\_str( ), NULL );

dPositionYActuel = strtod( sY.c\_str( ), NULL );

fprintf( DebugFile, "Fin------- \nsX : %s\nsY : %s\n", sX.c\_str( ), sY.c\_str( ) );

CommandeAAjouter.Position.dFinHorizontal = dPositionXActuel;

CommandeAAjouter.Position.dFinVertical = dPositionYActuel;

}

//------------------------------------------

//Si on doit déplacer les moteurs avant de dessiner

//-----------------------------------------

else if ( IteratorLigne->c\_str( )[IteratorLigne->size( ) - 3] == '2' || IteratorLigne->c\_str( )[IteratorLigne->size( ) - 2] == '2' )

{

CommandeAAjouter.Position.dDepartHorizontal = strtod( sX.c\_str( ), NULL );

CommandeAAjouter.Position.dDepartVertical = strtod( sY.c\_str( ), NULL );

fprintf( DebugFile, "Debut----- \nsX : %s\nsY : %s\n", sX.c\_str( ), sY.c\_str( ) );

IteratorLigne++;

strcpy( cmCoordonnes, IteratorLigne->c\_str( ) );

TraitementConversionPosition( cmCoordonnes, IteratorLigne->c\_str( )[0] );

fprintf( DebugFile, "Fin------- \nsX : %s\nsY : %s\n", sX.c\_str( ), sY.c\_str( ) );

dPositionXActuel = strtod( sX.c\_str( ), NULL );

dPositionYActuel = strtod( sY.c\_str( ), NULL );

CommandeAAjouter.Position.dFinHorizontal = dPositionXActuel;

CommandeAAjouter.Position.dFinVertical = dPositionYActuel;

}

//------------------------------------------

//Choisir le code de la commande

//-----------------------------------------

if ( CommandeAAjouter.Position.dDepartHorizontal != CommandeAAjouter.Position.dFinHorizontal || CommandeAAjouter.Position.dDepartVertical != CommandeAAjouter.Position.dFinVertical )

CommandeAAjouter.CodeCommande = CCDiagonal;

else if ( CommandeAAjouter.Position.dDepartHorizontal == CommandeAAjouter.Position.dFinHorizontal && CommandeAAjouter.Position.dDepartVertical == CommandeAAjouter.Position.dFinVertical )

CommandeAAjouter.CodeCommande = CCPoint;

else

CommandeAAjouter.CodeCommande = CCOrigine;

Commandes.push\_back( CommandeAAjouter );

fprintf( DebugFile, "---------------------------\n" );

}

//------------------------------------------

//Reset la position des moteurs a l'origine.

//-----------------------------------------

CommandeAAjouter.Position.dDepartHorizontal = 0;

CommandeAAjouter.Position.dDepartVertical = 0;

CommandeAAjouter.Position.dFinHorizontal = 0;

CommandeAAjouter.Position.dFinVertical = 0;

CommandeAAjouter.CodeCommande = CCOrigine;

Commandes.push\_back( CommandeAAjouter );

}

/\*! @brief Traite une commande pour en séparer et traiter 2 coordonées (x ,y )

@param cmCoordonnes Ligne utilisé pour la modification

@param cPremierCharactere Premier caractère de la ligne

@internal

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-05-16

@author Benoit Houle

@note Entrée du code.

<hr width="100%" height="5">

\*/

void TraitementConversionPosition( char cmCoordonnes[], char cPremierCharactere )

{

sX = strtok ( cmCoordonnes, "XYD" );

sY = strtok ( NULL, "XYD" );

//------------------------------------------

//S'il y a uniquement une coordonée a traiter

//-----------------------------------------

if ( sY.c\_str( )[2] == '\*' )

{

if ( cPremierCharactere == 'X' )

{

sY = to\_string( dPositionYActuel );

if ( sX.size( ) > iEntierX )

{

sX.insert( iEntierX, "." );

}

}

else

{

sY = sX;

sX = to\_string( dPositionXActuel );

if ( sY.size( ) > iEntierY )

{

sY.insert( iEntierY, "." );

}

}

}

else

{

if ( sX.size( ) > iEntierX )

{

sX.insert( iEntierX, "." );

}

if ( sY.size( ) > iEntierY )

{

sY.insert( iEntierY, "." );

}

}

}

#### Commande.hpp

/\*! @file Commande.hpp

@version 1.00

@brief Déclaration pour une commande.

@internal

projet Graveur de circuit imprimé (PCB) sur machine CNC

@note Note au programmeur.

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-03-03

@author Benoit Houle

@note Création du code.

<hr width="100%" height="5">

\*/

//\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef COMMANDE\_HPP

#define COMMANDE\_HPP

#ifdef \_\_cplusplus

extern "C"

{

#endif

//\*\* ==========================================================================

//\*\* Insérer ici les déclarations.

//\*\* ==========================================================================

//! Les différentes commandes disponibles

enum TCodeCommande

{

//! Retourner a l'origine

CCOrigine = 0,

//! Executer un mouvement horizontal

CCHorizontal = 1,

//! Executer un mouvement vertical

CCVertical = 2,

//! Executer un mouvement diagonal

CCDiagonal = 3,

//! Dessiner un point

CCPoint = 4,

//! Bouger les moteurs sans dessiner

CCBougerSansDessiner = 5,

CCDefault = CCOrigine,

CCPremier = CCOrigine,

CCDernier = CCBougerSansDessiner,

} ;

//! La position de départ et de fin des moteurs pour la commande

struct TPosition

{

double dDepartHorizontal;

double dDepartVertical;

double dFinHorizontal;

double dFinVertical;

} ;

//! Tous l'information nécesaire pour executer une commande

struct TCommande

{

enum TCodeCommande CodeCommande;

struct TPosition Position;

} ;

#ifdef \_\_cplusplus

}

#endif

#endif /\* COMMANDE\_HPP \*/

### Contrôleur

#### ControleurDrivers.ino

/\*! @file ControleurDrivers.ino

@version 1.00

@brief Utilisation de la class pour controler des moteurs

@internal

projet Graveur de circuit imprimé (PCB) sur machine CNC

@note Note au programmeur.

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-03-02

@author Benoit Houle

@note Entrée originale du code.

<hr width="100%" height="5">

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "Controleur.hpp"

// Contrôleur pour commander les 3 drivers.

TControleur Controleur;

//Information recu

String readSerial;

//Commande recu

struct TCommande CommandeRecu;

void setup() {

Serial.begin(19200);

CommandeRecu.CodeCommande = CCDiagonal;

CommandeRecu.Position.dDepartHorizontal = 0;

CommandeRecu.Position.dDepartVertical = 0;

CommandeRecu.Position.dFinHorizontal = 0;

CommandeRecu.Position.dFinVertical = 0;

pinMode(LIMITE\_X, INPUT\_PULLUP);

pinMode(LIMITE\_Y, INPUT\_PULLUP);

pinMode(LIMITE\_Z, INPUT\_PULLUP);

pinMode(LED\_BUILTIN, OUTPUT);

digitalWrite(LED\_BUILTIN, LOW);

if (Controleur.TesterLimiteMoteurs())

{

Serial.println("Un ou des moteurs n'ont pas touché leur limite.");

}

}

void loop() {

delay(100);

// Attendre de recevoir de l'information

if (Serial.available() > 0) {

readSerial = Serial.readString();

char InfoRecu[100];

strcpy(InfoRecu, readSerial.c\_str());

char \* pch;

pch = strtok (InfoRecu, "(,)");

String SCodeCommande = pch;

CommandeRecu.CodeCommande = SCodeCommande.toInt();

pch = strtok (NULL, "(,)");

String SDepartHorizontal = pch;

CommandeRecu.Position.dDepartHorizontal = SDepartHorizontal.toFloat();

pch = strtok (NULL, "(,)");

String SDepartVertical = pch;

CommandeRecu.Position.dDepartVertical = SDepartVertical.toFloat();

pch = strtok (NULL, "(,)");

String SFinHorizontal = pch;

CommandeRecu.Position.dFinHorizontal = SFinHorizontal.toFloat();

pch = strtok (NULL, "(,)");

String SFinVertical = pch;

CommandeRecu.Position.dFinVertical = SFinVertical.toFloat();

Serial.print("exec:");

Serial.println(Controleur.ExecuterCommande(&CommandeRecu));

}

}

#### Controleur.hpp

/\*! @file Controleur.hpp

@version 1.00

@brief Déclaration pour un contrôleur de moteurs

@internal

projet Graveur de circuit imprimé (PCB) sur machine CNC

@note Fait pour un Arduino.

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-03-02

@author Benoit Houle

@note Entrée du code.

<hr width="100%" height="5">

\*/

//\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef CONTROLEUR\_HPP

#define CONTROLEUR\_HPP

#ifdef \_\_cplusplus

extern "C"

{

#endif

#include "Commande.hpp"

#include "SyncDriver.h"

#include "DRV8834.h"

// Motor steps per revolution. Most steppers are 200 steps or 1.8 degrees/step

#define MOTOR\_STEPS 200

// Target RPM for cruise speed

#define RPM 60

// Acceleration and deceleration values are always in FULL steps / s^2

#define MOTOR\_ACCEL 2000

#define MOTOR\_DECEL 1000

// Microstepping mode. If you hardwired it to save pins, set to the same value here.

#define MICROSTEPS 1

// Faire tourner un moteur pendant 60 secondes. (MOTOR\_STEPS\*MICROSTEPS\*RPM)

#define TESTER\_LIMITE\_STEPS (MOTOR\_STEPS\*MICROSTEPS\*RPM)

// Pin pour le moteur horizontal

#define DIR\_X 2

#define STEP\_X 3

// Pin pour le moteur vertical

#define DIR\_Y 4

#define STEP\_Y 5

// Pin pour le moteur de l'axe des Z

#define DIR\_Z 6

#define STEP\_Z 7

// Pin pour les boutons de limite

#define LIMITE\_X 8

#define LIMITE\_Y 9

#define LIMITE\_Z 10

class TControleur

{

public:

//Constructeur par défaut

TControleur();

//! La commande a effectuer

struct TCommande CommandeRecu;

//! Execute une commande

int ExecuterCommande(struct TCommande \* Commande);

//! Teste la limite d'un moteur

int TesterLimiteMoteurs();

private:

//! Le résultat de la commande. Si elle s'Est bien effectuer ou non.

int iResultat;

//! Convertie un nombre de millimetre en nombre de step pour le moteur.

long MMtoStep(double dMM);

int iLimite\_X;

int iLimite\_Y;

int iLimite\_Z;

double dPositionHorizontal;

double dPositionVertical;

} ;

#ifdef \_\_cplusplus

}

#endif

#endif /\* CONTROLEUR\_HPP \*/

#### Controleur.cpp

/\*! @file Controleur.cpp

@version 1.00

@brief Implémentation de la class pour controler des moteurs

@internal

projet Graveur de circuit imprimé (PCB) sur machine CNC

@note Note au programmeur.

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-03-02

@author Benoit Houle

@note Entrée originale du code.

@date 2020-7-07

@author Benoit Houle

@note Ajout de TesterLimiteMoteurs().

<hr width="100%" height="5">

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <Arduino.h>

#include "Controleur.hpp"

DRV8834 Driver\_X(MOTOR\_STEPS, DIR\_X, STEP\_X);

DRV8834 Driver\_Y(MOTOR\_STEPS, DIR\_Y, STEP\_Y);

DRV8834 Driver\_Z(MOTOR\_STEPS, DIR\_Z, STEP\_Z);

MultiDriver MultiDrivers(Driver\_X, Driver\_Y);

TControleur::TControleur() {

// Activer les moteurs

Driver\_X.begin(RPM, MICROSTEPS);

Driver\_Y.begin(RPM, MICROSTEPS);

Driver\_Z.begin(RPM, MICROSTEPS);

Driver\_X.setSpeedProfile(Driver\_X.LINEAR\_SPEED, MOTOR\_ACCEL, MOTOR\_DECEL);

Driver\_Y.setSpeedProfile(Driver\_Y.LINEAR\_SPEED, MOTOR\_ACCEL, MOTOR\_DECEL);

Driver\_Z.setSpeedProfile(Driver\_Z.LINEAR\_SPEED, MOTOR\_ACCEL, MOTOR\_DECEL);

int iLimite\_X = 0;

int iLimite\_Y = 0;

int iLimite\_Z = 0;

double dPositionHorizontal = 0.0;

double dPositionVertical = 0.0;

}

/\*! @brief Teste la limite d'un moteur

@return Nombre de moteur qui n'ont pas touché la limite.

@retval 0 Tous les moteurs ont touché leur limite.

@retval 1 1 moteur n'a pas touché sa limite

@retval 2 2 moteurs n'ont pas touchés leur limite

@retval 3 3 moteurs n'ont pas touchés leur limite

@internal

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-03-02

@author Potato

@note Entrée du code.

<hr width="100%" height="5">

\*/

int TControleur::TesterLimiteMoteurs() {

int iCodeErreur = 0;

//------------------------------------

//Tester le moteur X.

//------------------------------------

Serial.print("Limite moteur X - Bouton jaune : ");

Driver\_X.startMove(TESTER\_LIMITE\_STEPS);

for (iLimite\_X = 0; digitalRead(LIMITE\_X) && iLimite\_X < TESTER\_LIMITE\_STEPS ; iLimite\_X++)

{

Driver\_X.nextAction();

}

Driver\_X.stop();

if (iLimite\_X >= TESTER\_LIMITE\_STEPS) iCodeErreur++;

Serial.println(iLimite\_X);

delay(250);

//------------------------------------

//Tester le moteur Y.

//------------------------------------

Serial.print("Limite moteur Y - Bouton Blanc : ");

Driver\_Y.startMove(TESTER\_LIMITE\_STEPS);

for (iLimite\_Y = 0; digitalRead(LIMITE\_Y) && iLimite\_Y < TESTER\_LIMITE\_STEPS ; iLimite\_Y++)

{

Driver\_Y.nextAction();

}

Driver\_Y.stop();

if (iLimite\_Y >= TESTER\_LIMITE\_STEPS) iCodeErreur++;

Serial.println(iLimite\_Y);

delay(250);

//------------------------------------

//Tester le moteur Z.

//------------------------------------

Serial.print("Limite moteur Z - Bouton Bleu : ");

Driver\_Z.startMove(TESTER\_LIMITE\_STEPS);

for (iLimite\_Z = 0; digitalRead(LIMITE\_Z) && iLimite\_Z < TESTER\_LIMITE\_STEPS ; iLimite\_Z++)

{

Driver\_Z.nextAction();

}

Driver\_Z.stop();

if (iLimite\_Z >= TESTER\_LIMITE\_STEPS) iCodeErreur++;

Serial.println(iLimite\_Z);

delay(250);

//------------------------------------

//Replacer les 3 moteurs.

//------------------------------------

Driver\_X.move(-iLimite\_X);

delay(250);

Driver\_Y.move(-iLimite\_Y);

delay(250);

Driver\_Z.move(-iLimite\_Z);

delay(250);

// Retourner le code d'erreur à la routine appelante.

return iCodeErreur;

}

/\*! @brief Executer une commande

@param Commande La commande a executer

@internal

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-03-02

@author Potato

@note Entrée du code.

<hr width="100%" height="5">

\*/

int TControleur::ExecuterCommande(struct TCommande \* Commande) {

int iCodeErreur = 0;

double dMouvementHorizontal = 0;

double dMouvementVertical = 0;

//------------------------------------

//Appliquer les limites.

//------------------------------------

if (Commande->Position.dDepartHorizontal > (double)iLimite\_X)

{

Commande->Position.dDepartHorizontal = (double)iLimite\_X;

iCodeErreur++;

}

if (Commande->Position.dDepartVertical > (double)iLimite\_Y)

{

Commande->Position.dDepartVertical = (double)iLimite\_Y;

iCodeErreur++;

}

if (Commande->Position.dFinHorizontal > (double)iLimite\_X)

{

Commande->Position.dFinHorizontal = (double)iLimite\_X;

iCodeErreur++;

}

if (Commande->Position.dFinVertical > (double)iLimite\_Y)

{

Commande->Position.dFinVertical = (double)iLimite\_Y;

iCodeErreur++;

}

//--------------------------------------

//Executer la commande selon son code.

//--------------------------------------

if (Commande->CodeCommande == CCOrigine)

{

dMouvementHorizontal = -dPositionHorizontal;

dMouvementVertical = -dPositionVertical;

MultiDrivers.move(MMtoStep(dMouvementHorizontal), MMtoStep(dMouvementVertical));

delay(1000);

dPositionHorizontal = 0;

dPositionVertical = 0;

}

else if (Commande->CodeCommande == CCPoint)

{

dMouvementHorizontal = Commande->Position.dDepartHorizontal - dPositionHorizontal;

dMouvementVertical = Commande->Position.dDepartVertical - dPositionVertical;

MultiDrivers.move(MMtoStep(dMouvementHorizontal), MMtoStep(dMouvementVertical));

Driver\_Z.move(25);

delay(1000);

Driver\_Z.move(-25);

delay(1000);

dPositionHorizontal = Commande->Position.dDepartHorizontal;

dPositionVertical = Commande->Position.dDepartVertical;

}

else

{

//Si on a besoin de déplacer les moteurs avant de dessiner

if (dPositionHorizontal != Commande->Position.dDepartHorizontal || dPositionVertical != Commande->Position.dDepartVertical || Commande->CodeCommande == CCBougerSansDessiner)

{

dMouvementHorizontal = Commande->Position.dDepartHorizontal - dPositionHorizontal;

dMouvementVertical = Commande->Position.dDepartVertical - dPositionVertical;

delay(1000);

MultiDrivers.move(MMtoStep(dMouvementHorizontal), MMtoStep(dMouvementVertical));

delay(1000);

if (Commande->CodeCommande == CCBougerSansDessiner)

{

dPositionHorizontal = Commande->Position.dFinHorizontal;

dPositionVertical = Commande->Position.dFinVertical;

}

else

{

dPositionHorizontal = Commande->Position.dDepartHorizontal;

dPositionVertical = Commande->Position.dDepartVertical;

}

}

//Dessiner une ligne

if (Commande->CodeCommande == CCDiagonal)

{

dMouvementHorizontal = Commande->Position.dFinHorizontal - Commande->Position.dDepartHorizontal;

dMouvementVertical = Commande->Position.dFinVertical - Commande->Position.dDepartVertical;

Driver\_Z.move(25);

delay(1000);

MultiDrivers.move(MMtoStep(dMouvementHorizontal), MMtoStep(dMouvementVertical));

delay(1000);

Driver\_Z.move(-25);

delay(1000);

dPositionHorizontal = Commande->Position.dFinHorizontal;

dPositionVertical = Commande->Position.dFinVertical;

}

}

// Retourner le code d'erreur à la routine appelante.

return iCodeErreur;

}

/\*! @brief Calcule le nombre de step pou un nombre de millimètre

@param dMM Le nombre de millimètre

@return Nombre de step

@internal

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-03-02

@author Potato

@note En raison du COVID, cette fonction n'est pas utile.

<hr width="100%" height="5">

\*/

long TControleur::MMtoStep(double dMM)

{

long lVariableTemporaire = (long)dMM;

return lVariableTemporaire;

}

#### Commande.hpp

/\*! @file Commande.hpp

@version 1.00

@brief Déclaration pour une commande.

@internal

projet Graveur de circuit imprimé (PCB) sur machine CNC

@note Note au programmeur.

<hr width="100%" height="5">

<b>Historique</b>

<hr width="100%" height="1">

@date 2020-03-03

@author Benoit Houle

@note Création du code.

<hr width="100%" height="5">

\*/

//\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef COMMANDE\_HPP

#define COMMANDE\_HPP

#ifdef \_\_cplusplus

extern "C"

{

#endif

//! Les différentes commandes disponibles

enum TCodeCommande

{

//! Retourner a l'origine

CCOrigine = 0,

//! Executer un mouvement horizontal

CCHorizontal = 1,

//! Executer un mouvement vertical

CCVertical = 2,

//! Executer un mouvement diagonal

CCDiagonal = 3,

//! Dessiner un point

CCPoint = 4,

//! Bouger les moteurs sans dessiner

CCBougerSansDessiner = 5,

CCDefault = CCOrigine,

CCPremier = CCOrigine,

CCDernier = CCBougerSansDessiner,

} ;

//! La position de départ et de fin des moteurs pour la commande

struct TPosition

{

double dDepartHorizontal;

double dDepartVertical;

double dFinHorizontal;

double dFinVertical;

} ;

//! Tous l'information nécesaire pour executer une commande

struct TCommande

{

enum TCodeCommande CodeCommande;

struct TPosition Position;

} ;

#ifdef \_\_cplusplus

}

#endif

#endif /\* COMMANDE\_HPP \*/

# Discussion

## Impact du COVID-19

Cette année est très spéciale en raison d’une pandémie qui résulte en une quarantaine. En raison du manque d’équipements, la soudure de mon PCB ne peut pas être faite. De plus, je n’ai pas de machine complète. L’objectif de calibrer les moteurs est donc réalisés avec des boutons. Finalement, il est impossible de tester quand le crayon touche la feuille sans s’endommager, donc cet objectif est abandonné.

Personnellement, la situation pour travailler chez moi est loin d’être idéal. Un membre de ma famille travaille dans le salon et fait souvent des appels, ma chambre est très mal climatisée, donc il fait souvent 26 celsius ou 27 celsius et m’ont TDAH sont des exemples de choses qui mon compliqué la tâche.

## Problèmes rencontrés

### Les DRV8834

Le premier problème que j’ai rencontré est le dysfonctionnement du DRV8834. Au moment de l’étude de faisabilité, je n’ai pas eu le temps de le tester. J’avais testé le fonctionnement du moteur pour être certain de comprendre comment il fonctionne. Durant la pause entre les sessions, j’ai tenté de la faire fonctionner chez moi. J’ai tout essayé, les branchements minimaux, le branché au complet, des alimentations différentes (à l’intérieur de ses limites), mais tous ces tests n’ont rien donné. J’ai même lu la datasheet au complet deux ou trois fois, ça ne m’a pas donné d’autres idées. Une fois le projet commencé, avec l’aide d’un professeur (Guy), on a essayé de trouver le problème, mais sans succès. Quelques jours plus tard, on a continué à chercher, mais rien n'a changé. La semaine d’après, Guy m’a dit de refaire les tests. Avec exactement la même configuration, le DRV8834 fonctionne bien.

Pour trouver le problème, nous avons fait plusieurs tests. On a commencé par se rafraîchir la mémoire sur l'information qu'on avait. Ensuite, j’ai refait les branchements de A à Z, mais rien n'a changé. On a ensuite imprimé les pages utiles du datasheet et nous avons commencé à numéroter les pins et à analyser le circuit interne. Avec ce travail, on a découvert qu’une pin (VREFO) devait être à 2V, mais elle est au GND. Cette pin est nécessaire pour limiter le courant. Sans elle, il n'y a pas de courant qui peut sortir des broches de sortie, donc la chip ne fonctionne pas.

Pour essayer de régler le problème, j'ai dessoudé le potentiomètre où était branché VREFO pour m'assurer que ce ne soit pas un problème de mauvaise soudure. Les tests n'ont pas donné de résultat différent. Comme dernière tentative, j'ai pris la dernière pièce, encore dans son sac de plastique, et j'ai soudé des headers pour être certain qu'il n'y ait pas de problème de contact. Malheureusement, quand on a recommencé nos tests, les résultats étaient les mêmes. C’est la semaine d’après que j’ai refait les tests avec exactement les même configurations et que le DRV8834 fonctionne bien.

### La vitesse des moteurs

Le prochain problème survenu est immédiatement après le dernier. Quand les moteurs bougeaient, on aurait dit qu'ils restaient pris sur quelque chose et n'avançaient presque plus. Un professeur (Denis) m'a dit de m'assurer que les moteurs n'allaient pas trop vite. Ça m’a fait penser que les DRV8834 pour mes moteurs sont capables d'aller à la vitesse demandée et plus sans problème, mais je n'avais pas pensé aux moteurs. Après avoir réduit la vitesse et vu que le problème n’apparait pas, j'ai compris où était le problème. Les moteurs ne pouvaient pas aller aussi vite que demandé parce que la polarité d’un moteur stepper prend un certain temps à s’inverser et je lui demandais d’aller trop vite.

### Définition d’un constructeur « overloaded »

Dans le code de l’Arduino, j’utilise la classe DRV8834 pour contrôler un moteur, donc j’ai besoin de 3 objets DRV8834 dans ma classe. Par contre, dans sa librairie, il y a 4 constructeurs. Quand vient le temps de compiler, le compilateur ne sait pas quel constructeur du DRV8834 utiliser puisque je ne peux pas donner d’arguments dans la définition de ma classe. Après avoir tout essayé et fait des heures de recherche, j’ai tout simplement mis 3 DRV8834 comme variable globale dans ma classe.

## Tests effectués

Dans mon code pour faire une chaine de commande, mon premier test est de vérifier si le fichier a été ouvert correctement. Ensuite, j’évalue chaques lignes pour voir si elle sont utiles. Après cette évaluation, je test tous les lignes utiles pour vérifier si tous les éléments obligatoires sont présents. Le prochain test est de vérifier si tout les fichiers utilisés comme sortie d’information de mon programme, tels que FichierResultat ou FichierMiseAJour, se sont bien ouvert pour y déposer un résultat.

La dernière série de teste est dans mon contrôleur. Je commence par évaluer l’information reçu pour être certain qu’elle est valide et pour savoir quel fonction utiliser. Ensuite, je test le code de la commande reçu pour s’avoir quel type de mouvement exécuter. Finalement, le je test le résultat de la fonction pour savoir quoi envoyé au serveur.

## Améliorations possibles

Plusieurs améliorations sont possibles. Il serais possible de mettre une caméra pour filmer le progrès du dessin. De cette façon, non seulement l’utilisateur peut voir son dessin en temps réel, mais le projet est plus sécuritaire.

Il pourrait aussi y avoir une boite en plastique transparent qui couvre le tout pour permettre la vision, mais qui empêche n’importe qui d’endommager le matériel ou de se blesser. De plus, si cette boite de plastique est verrouillée et que seulement l’utilisateur qui a fait imprimer le dessin par la page web peut le déverrouiller, personne ne peut voler le contenu.

Le crayon magnétique pourrait aussi être remplacé par un tube avec une tête de crayon qui mène vers un pot d’encre magnétique. De cette façon, il n’y a pas de crayon à changer ni à acheter.

Le programme pourrait aussi être poursuivi pour incorporer des lignes courbées ce qui complèterait les mouvements possibles. Avec cet ajout, il serait possible de faire des macro dans un fichier Gerber, donc d’inclure les macro dans le traducteur pour compléter tout ce qui peut être dans un fichier Gerber.

# Conclusion

Mon projet synthèse s’est généralement bien déroulé. Il m’a appris beaucoup sur plusieurs sujets tels que les moteurs stepper, la fabrication de PCB et les serveurs web. À l’exception de quelques petits détails, j’ai pu compléter mon projet. Tout fonctionne et il est possible d’imprimer un fichier à partir d’un client web.

Par contre, le serveur ne peut pas recevoir ce que le contrôleur lui envoie. En conséquence, il envoie les commandes uniquement lorsque l’utilisateur clique sur un bouton.

Le dépôt du fichier fonctionne très bien et affiche un message pour informer l’utilisateur de l’état du dépôt ou de l’invalidité de son authentification.

Les mises à jour fonctionnent tout aussi bien. Quand un client demande les mises à jour, la page web affiche tous les dessins et leur progression.

Le programme qui construit une chaine de commande à partir d’un fichier Gerber fonctionne très bien. Il est en mesure le lire un fichier Gerber et en sortir une chaine de commande sans problème. Si le fichier n’existe pas ou s’il y a des complications, il met une phrase indiquant les complications pour que la page web puisse les afficher.

L’objectif 2 : << Obtenir la calibration de l’ensemble 3 moteurs + stylo par rapport à un point de référence avec une précision de +/- 1mm >> et l’objectif 4 : << Être capable de savoir quand le crayon touche la feuille sans s’endommager pour dessiner >> ne sont plus d’actualité en raison du COVID.

Pour finir, même si plusieurs choses ne se sont pas bien déroulées, j’ai vraiment aimé faire ce projet. Il m’a permis d’apprendre énormément sur tous les aspects enseignés depuis notre première année. J’ai aussi eu l’opportunité de comprendre les faiblesses dans ma gestion du temps. J’ai tendance à passer des heures sur quelque chose d’insignifiant et ensuite manquer de temps. J’ai hâte de faire d’autres projets, autant au niveau personnel que professionnel.