

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTRE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE
SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté des Sciences et de la technologie

Département d'Electronique

Mémoire

Présenté pour obtenir

LE DIPLOME DE LICENCE

FILIERE : **Electronique**

Spécialité : Industries électroniques

Par

- **Zidane Iyad**
- **Sareh Hamdi**
- **Bourkaib Radhwane**

Intitulé

Contrôle de la vitesse d'un moteur DC avec LabVIEW et Arduino

Evalué le :

Par la commission d'évaluation composée de :

<i>Nom & Prénom</i>	<i>Grade</i>	<i>Qualité</i>	<i>Etablissement</i>
<i>M.</i>	<i>MCB</i>	<i>Président</i>	<i>Univ-BBA</i>
<i>Mme. Laouamri Asma</i>	<i>MAA</i>	<i>Encadreur</i>	<i>Univ-BBA</i>
<i>M.</i>	<i>....</i>	<i>Examineur</i>	<i>Univ-BBA</i>

Année Universitaire 2020/2021

***R**emerciements*

Avant tout, nous remercions Allah le tout puissant, de nous avoir donné la fois et la puissance d'arriver à ce jour -là ou nous sommes arrivées aujourd'hui.

*Nos remerciements s'adressent à notre encadreur madame **Laouamri Asma** Qui nous a accompagnés tout au long de ce travail avec ses conseils.*

*Un grand merci à **Abderrahim Mehiris Moussa** Pour son aide et ses conseils en programmation.*

Nos sincères remerciements vont aussi aux membres de jury à ceux qui corrigeront cette mémoire

Enfin, nous remercions tous ceux qui nous ont aidés de près ou de loin à mener à bien ce mémoire.

Résumé :

Ce projet vise à contrôler la vitesse d'un moteur à courant continu à l'aide d'Arduino et d'une interface de contrôle sous Labview.

Premièrement, une partie théorique et introductive sur les outils utilisés dans notre projet est exposée. Puis, une partie pratique explique la réalisation de la carte électronique, la préparation de l'interface de contrôle, aussi, dans cette partie on donne la méthode de liaison de ces deux réalisations. A la fin, on expose les résultats de simulation dans différents états.

Abstract:

This project aims to control the speed of a DC motor using Arduino and a control interface under Labview.

First, a theoretical and introductory part on the tools used in our project is exposed. Then, a practical part explains the realization of the electronic card, the preparation of the control interface, also, in this part we give the method of linking these two realizations. At the end, the simulation results are displayed in different states.

الملخص :

يهدف هذا المشروع إلى التحكم في سرعة محرك DC باستخدام Arduino وواجهة تحكم ضمن Labview.

أولاً، يتم عرض جزء نظري وتمهيدي حول الأدوات المستخدمة في مشروعنا. ثم يشرح الجزء العملي تحقيق البطاقة الإلكترونية، وإعداد واجهة التحكم، كما نقدم في هذا الجزء طريقة ربط هذين الإنجازين. في النهاية، يتم عرض نتائج المحاكاة في حالات مختلفة.

SOMMAIRE

Introduction général.....	01
---------------------------	----

I. PARTIE THEORIQUE

I. 1.Présentation De L'Arduino

1.1. Définition Arduino.....	04
1.2. Le But d'utilisation.....	04
1.3. Les Composants de l'Arduino.....	04
1.4. Les Modèles d'Arduino.....	06
1.4.1 Arduino Uno.....	06
1.4.2. Arduino Nano.....	06
1.4.3 Arduino Leonardo.....	06
1.4.4. Arduino Mega 2560.....	07
1.5. Caractéristique technique de la carte Arduino Mega 2560.....	07

I.2. Commande D'un Moteur à Courant Continu Par PWM

2.1. Introduction.....	10
2.2. Le moteur DC.....	10
2.3. Commande d'un moteur à courant continu par Arduino.....	10
2.3.1. PWM (modulation width pulse).....	11
2.3.2. Le principe du PWM.....	11
2.3.3. Les pôles PWM dans l'arduino.....	11
2.3.4. Les Avantages de PWM.....	12
2.4. Le rapport cyclique.....	12
2.4.1. Différents cas possible du rapport cyclique.....	13
2.5.Control de la vitesse du moteur.....	13

2.6. Mesure du signal PWM.....	14
--------------------------------	----

I.3. Simulation Graphique Avec LabView

Introduction	16
3.1. Utilité de LabVIEW.....	16
3.2. Description de la simulation.....	16
3.3. Simulation dans LabVIEW.....	17
3.4. Les représentations graphiques des données numériques.....	18
3.4.1 Les graphes	18
3.4.2. Les graphes déroulants.....	18
3.4.3. Les graphes XY	19
Conclusion.....	19

II. PARTIE PRATIQUE

II.1 Conception De La Carte Sur Proteus

1.1. Introductin.....	22
1.2. Les Composantes	22
1.3. Schéma électronique.....	23
1.4. Description du fonctionnement.....	24
1.4.1. Partie de contrôle.....	24
1.4.2. Partie traitement et connexion.....	25
1.4.3. Partie Puissance.....	25
1.4.4. Partie d'affichage.....	26
1.4.5. Algorithme de travail de l'Arduino.....	26
1.5. Résultats de test.....	26

II.2. Réalisation De L'interface De Contrôle

2.1 Introduction.....	31
2.2. L'interface graphique.....	31

2.3. Liaison de l'interface de contrôle avec la carte électronique.....	33
2.4. Liaison virtuelle.....	33
2.4.1 Définition de VSPE	33
2.4.2 Identification des pôles de connexion.....	34

II.3. Simulation De La Carte Avec L'interface De Contrôle

3.1 Introduction.....	37
3.2 Configuration de Proteus	37
3.3 Configuration de LabView.....	38
3.4 Résultats de simulation.....	39
Conclusion.....	42
Conclusion Générale.....	43

Bibliographie

Liste des Figures :

Figure I.1 : Un afficheur LCD 2X16 avec une carte Arduino.....	04
Figure I.2 : Connecteur d'alimentation et de communication.....	05
Figure I.3 : Exemple d'une utilisation d'arduinoUno.....	05
Figure I.4 : Microcontrôleur ATMEGA328P.....	05
Figure I.5 : Oscillateur (quartz).....	05
Figure I.6 : Arduino Uno.....	06
Figure I.7 : Arduino Nano.....	06
Figure I.8 : Arduino Leonardo.....	06
Figure I.9 : Description de la carte Arduino MEGA 2560.....	07
Figure I.10 : Types de moteurs.....	10
Figure I.11 : Un moteur DC.....	10
Figure I.12 : Les ports sortis PWM dans Arduino Uno.....	11
Figure I.13 : Le rapport cyclique.....	12
Figure I.14 : Schéma électrique du contrôle de vitesse d'un moteur DC.....	13
Figure I.15 : Face avant d'un instrument virtuel qui acquérir un signal.....	16
Figure 1.16 : Graphe déroulant.....	18
Figure II.1 : Schéma électronique.....	24
Figure II.2 : Liaison du capteur LM35 à l'Arduino.....	25
Figure II.3 : Liaison de la résistance Variable à l'Arduino.....	25
Figure II.4 : Le bouton poussoir.....	25
Figure II.5 : Algorithme de travail.....	27
Figure II.6 : Circuit électronique au démarrage.....	28
Figure II.7 : Circuit électronique lors du changement de valeur du rapport cyclique.....	29
Figure II.8 : Circuit électronique avec une température supérieure à 60.....	29

Figure II.9: Diagramme de l'interface de contrôle.....	32
Figure II.10 : la face avant de l'interface de contrôle.....	32
Figure II.11 : La configuration des pins.....	33
Figure II.12: L'interface utilisateur du programme.....	34
Figure II.13 : Option de lien dans le Programme.....	35
Figure II.14 : Identification des pôles.....	35
Figure II.15 : Les pôles utilisés dans La connexion.....	35
Figure II.16 : Schéma électronique après reconfiguration.....	37
Figure II.17: La configuration de ports COM.....	38
Figure II.18 : Sélection des COM dans l'interface de contrôle.....	38
Figure II.19 : Démarrage de la simulation.....	39
Figure II 20 : Changement de la vitesse de moteur.....	40
Figure II.21 : les différentes mesures dans le cas de dépassement de température.....	40
Figure II.22 : Changement méthode de contrôle de la vitesse de moteur.....	41
Figure II.23 : Le changement de la vitesse de moteur avec le proteus.....	41

Liste des tableaux :

Tableau I.1 : Les broches PWM dans chaque carte Arduino.....	11
Tableau II.1 : Les éléments électroniques utilisés dans la simulation.....	23

Introduction générale

Dans ce projet, nous étudions comment contrôler la vitesse d'un moteur à courant continu basé sur Arduino et Labview, Avec surveillance de la température du moteur.

Pour la réalisation de cette interface, on utilise le logiciel Labview. Ce dernier nous facilite la modélisation des parties de contrôle et de mesure des différents paramètres du moteur (vitesse, température...). On utilise également Proteus pour réaliser et simuler notre carte électronique. Sous cet outil, l'Arduino est programmé par le langage C.

Le lien entre la carte électronique et l'interface de contrôle se fait par le port série RS232 et l'outil VSPE (Virtual Serial Port Emulator).

Afin d'effectuer cette réalisation, le travail est divisé en deux parties principales : théorique et pratique.

Dans la partie théorique, on expose, tout d'abord, une étude sur l'Arduino , puis on présente la commande de moteur DC par la PWM et on termine par une illustration de quelques notions de base de la simulation graphique par le Labview.

La partie pratique comporte trois sous parties, la première explique la conception de la carte électronique et la deuxième détaille la méthode et les étapes de réalisation de l'interface graphique. Dans la troisième sous partie, la simulation des deux parties de commande est exposée, dans plusieurs états.

On termine ce mémoire par une conclusion générale, dont on mentionne les choses qu'on a appris pendant la réalisation de ce projet.

PARTIE THEORIQUE

1.1. Présentation De L'Arduino

1.1. Définition d'Arduino

C'est une petite carte programmable qui peut apprendre à effectuer des tâches en fonction d'un programme écrit d'avance. Elle est au format carte bancaire (légèrement plus petite que le Raspberry pi) [1]

1.2. But d'utilisation

L'Arduino permet d'apprendre l'électronique, le codage et le fonctionnement des objets courants. Il existe même des interfaces de programmation par blocs pour permettre aux enfants de s'initier à l'Arduino aussi facilement qu'avec Scratch.

Il permet d'apprendre, de comprendre et même de se tromper sans pour autant prendre de gros risques car les tensions utilisées ne sont pas dangereuses pour l'homme et le matériel utilisé n'est pas très cher à remplacer. [1] **La Figure I.1** montre un petit montage électronique commandé par l'Arduino.

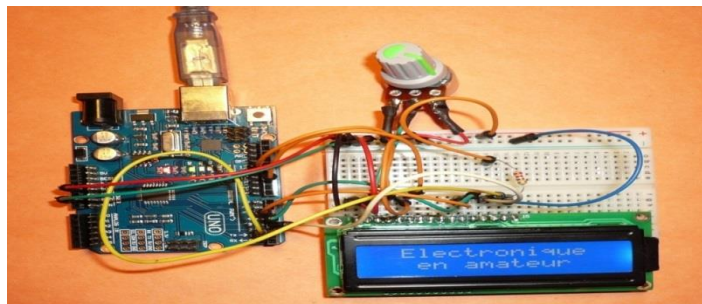


Figure I.1 : Un afficheur LCD 2X16 avec une carte Arduino

1.3. Composants de l'Arduino : [1]

La carte Arduino se compose essentiellement des éléments suivants :

- **Port de Communication et d'Alimentation** : la carte possède une prise USB (USB B, mini USB ou micro USB) qui permet sa programmation en la reliant à un ordinateur. Elle peut être alimentée par sa prise USB (5V) (figure 1.2) ou par la prise Jack (7-12V). [1]
- **Connecteurs d'entrées/sorties** : Les connecteurs d'entrées et de sorties permettent à l'Arduino de communiquer avec des composants électroniques. Prenant l'exemple de la figure 1.3, dont on branche une LED entre la borne n°9 de l'Arduino et la masse. Ensuite, on écrit un programme qui demande à l'Arduino d'allumer la LED (en envoyant du courant à la borne n°9). On peut aussi, demander à l'Arduino d'allumer la LED pendant une seconde, puis de

l'éteindre pendant également une seconde et de recommencer. Ainsi, on obtiendra un clignotement.

Par la suite, on peut étendre le programme en ajoutant des LED ou en utilisant d'autres éléments comme des boutons ou des capteurs.

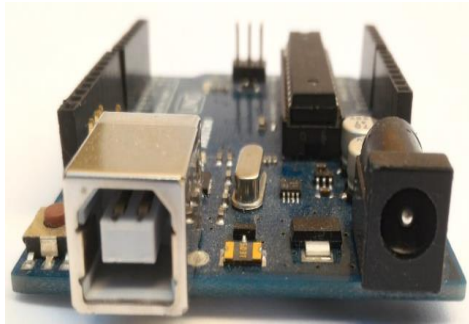


Figure I.2 : Connecteur d'alimentation et De communication

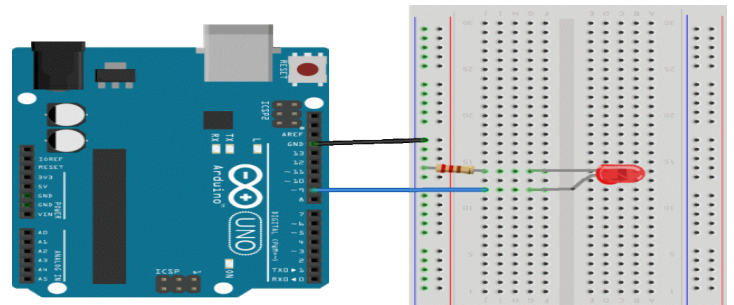


Figure I.3 : Exemple d'une utilisation d'Arduino Uno

- **Microcontrôleur :** C'est le cerveau de l'Arduino, c'est un peu comme un ordinateur miniature. Il intègre à lui seul le processeur, la mémoire vive (RAM), la mémoire de stockage (FLASH et EEPROM).



Figure I.4 : Microcontrôleur ATMEGA328P

- **Oscillateur ou quartz à 16 MHz :** Il a été ajouté afin d'augmenter la fréquence de l'Arduino UNO. Sans cet élément le microcontrôleur (ATMEGA328P) ne fonctionnerait qu'à 8 MHz. Et donc l'Arduino serait 2 fois moins rapide.



Figure I.5 : Oscillateur (quartz)

1.4. Modèles d'Arduino

1.4.1 Arduino Uno : Ce type représente une version standard et il est le plus utilisé dans les applications électroniques. Mais, son microcontrôleur (ATMega328P) ne gère pas la communication USB, il doit donc utiliser une puce secondaire à savoir : atmega16u2, CH340, CH341 ou FTDI.

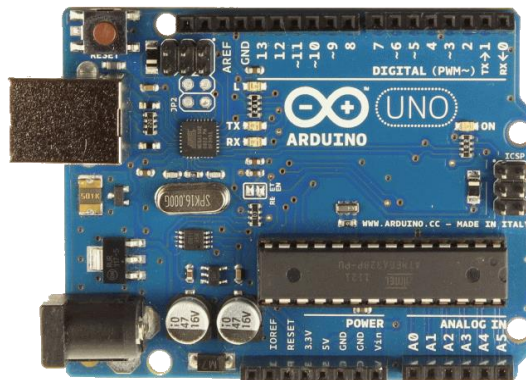


Figure I.6 : La carte d'Arduino Uno

1.4.2 Arduino Nano : C'est la version miniature de l'Arduino UNO (même microcontrôleur), spécialement adaptée aux breadboard ou à une intégration dans un petit boîtier.



Figure I.7 : Arduino Nano

1.4.3 Arduino Leonardo : Une évolution de l'Arduino Uno, il possède le même format, mais son microcontrôleur (ATMega32U4) gère directement l'USB, ce qui lui permet d'être reconnu comme un périphérique USB (clavier, souris, manette de jeu) par l'ordinateur. Cependant l'Arduino Uno reste indétrônable et n'a pas été remplacé par le Leonardo. [1]

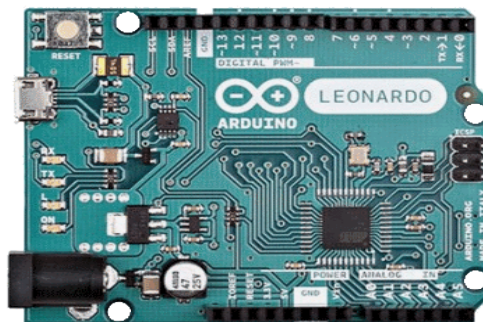


Figure I.8: Arduino Leonardo

1.4.4 Arduino Mega 2560 : La carte Arduino Mega 2560 est une carte à microcontrôleur basée sur un ATmega2560. Cette carte dispose de :

- 54 broches numériques d'entrées/sorties (dont 14 peuvent être utilisées en sorties PWM).
- 16 entrées analogiques (qui peuvent être utilisées en broches entrées/sorties numériques).
- 4 UART (port série matériel).
- un quartz de 16Mhz.
- une connexion USB.
- un connecteur d'alimentation jack.
- un connecteur ICSP (programmation "in-circuit").
- un bouton de réinitialisation (reset).

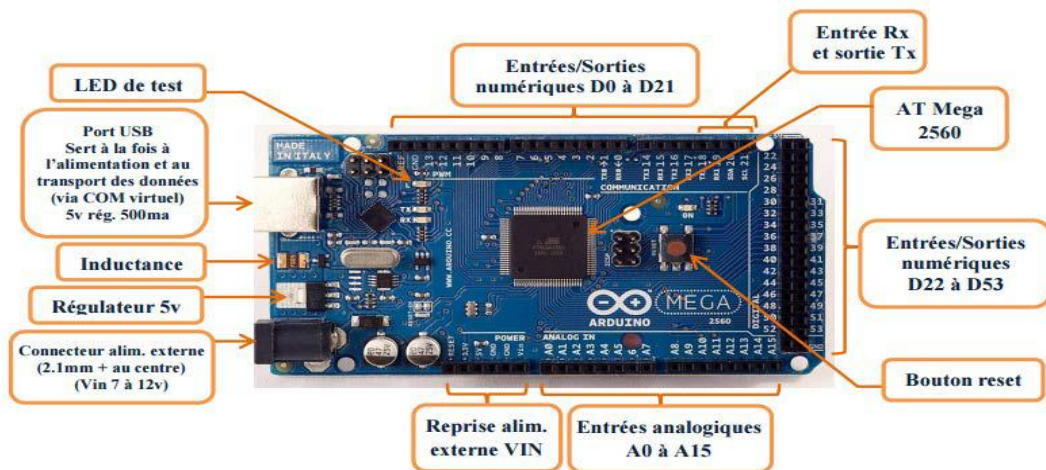


Figure I.9 : Description de la carte Arduino MEGA 2560

1.5. Caractéristique technique de la carte Arduino Mega 2560 [3]

Un module Arduino est généralement construit autour d'un microcontrôleur ATMEL AVR, et de composants complémentaires qui facilitent la programmation et l'interfaçage avec d'autres circuits. Chaque module possède au moins un régulateur linéaire 5V et un oscillateur à quartz 16 MHz (ou un résonateur céramique dans certains modèles). Le microcontrôleur est préprogrammé avec un bootloader de façon à ce qu'un programmeur dédié ne soit pas nécessaire.

Le Microcontrôleur ATmega2560 : c'est un circuit intégré qui rassemble sur une puce plusieurs éléments complexes dans un espace réduit. Il s'occupe de tout ce qui est calculs, exécution des instructions du programme et gestion des ports d'entrée/sortie.

Les Mémoires : L'ATmega 2560 à 256Ko de mémoire FLASH pour stocker le programme (dont 8Ko également utilisés par le bootloader), également 8 ko de mémoire SRAM (volatile) et 4Ko d'EEPROM (non volatile - mémoire qui peut être lue à l'aide de la librairie EEPROM).

Entrées et sorties numériques : Chacune des 54 broches numériques de la carte Mega peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique. Ces broches fonctionnent en 5V. Chaque broche peut fournir ou recevoir un maximum de 40mA d'intensité et dispose d'une résistance interne de "rappel au plus" (pull-up) (déconnectée par défaut) de 20-50 KOhms. Cette résistance interne s'active sur une broche en entrée à l'aide de l'instruction digitalWrite (broche, HIGH).

De plus, certaines broches ont des fonctions spécialisées :

- **Communication Série :** les broches du port série sont : Serial : 0 (RX) et 1 (TX) ; Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). [4]
- **Interruptions Externes :** Broches : 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19(interrupt 4), 20 (interrupt 3), et 21 (interrupt 2). Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur. [4]
- **Impulsion PWM** (largeur d'impulsion modulée): Broches 0 à 13. Elles fournissent une impulsion PWM 8-bits à l'aide de l'instruction analog Write. [4]
- **SPI** (Interface Série Périphérique) : Broches 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Ces broches supportent la communication SPI (Interface Série Périphérique). [4]
- **I2C :** Broches 20 (SDA) et 21 (SCL). Elles Supportent les communications de protocole I2C.
- **LED :** Broche 13. Il y a une LED incluse dans la carte connectée à la broche 13. [3]

***I.2. Commande D'un Moteur à Courant
Continu Par PWM***

2.1 Introduction

Un moteur électrique est une machine qui convertit l'énergie électrique en énergie mécanique ou l'énergie mécanique en énergie électrique, et il est divisé en plusieurs types y compris servomoteur, moteur pas à pas, et **le moteur DC**. [4]



Figure I.10 : Types de moteurs

2.2. Moteur à courant continu ou à courant direct (DC)

Il représente le type de moteur électrique le plus simple et le plus utilisé dans les systèmes automatiques, qui nécessitent une variation précise de la vitesse de rotation. Il est utilisé aussi, dans les applications industrielles à cause de son simple principe de fonctionnement et de sa commande facile. Ces moteurs possèdent deux fils, et nécessitent une source de tension continue. Alors qu'en dirigeant une tension sur ces deux fils, le moteur tourne dans un certain sens, et lorsque les pôles sont inversés, il tourne dans le sens opposé. L'effort d'alimentation se situe entre 6V et 12V [4]



Figure I.11 : Un moteur DC

2.3. Commande d'un moteur à courant continu par Arduino

Il existe plusieurs méthodes pour contrôler la vitesse d'un moteur à courant continu, mais l'une des plus simples est d'appliquer Signal **PWM** (modulation width pulse) sur le moteur.

2.3.1. Le signal PWM (modulation width pulse) : Le signal **PWM** est un acronyme anglais qui signifie Pulse Width Modulation, ou modulation de largeur d'impulsion (**MLI**) en français. La modulation de largeur d'impulsion (PWM) est une technique permettant d'obtenir des résultats analogiques par des moyens numériques. La commande numérique est utilisée pour créer une onde carrée activée et désactivée. Ce modèle marche-arrêt peut simuler les tensions entre le Vcc complet de la carte (par exemple, 5V sur la Uno et Méga) et l'arrêt (0V). La durée du temps d'activation est appelée largeur d'impulsion. Pour obtenir différentes valeurs analogiques, nous pouvons changer ou modifier cette largeur d'impulsion. [5] [6]

2.3.2. Le principe du PWM : L'objectif est de réduire la puissance moyenne délivrée d'une sortie digitale (HIGH ou LOW) en modulant les impulsions du signal. L'objectif est d'avoir un pseudo sorti analogique pouvant pendre 256 valeurs (0 à 255). [5]

Le signal **PWM** est utilisé par exemple pour contrôler la luminosité d'une LED, changer la couleur d'une LED RGB ou encore piloter la vitesse d'un moteur. [5]

2.3.3. Les pôles PWM dans l'Arduino : Pour générer un signal **PWM** avec Arduino, il faut utiliser une broche compatible avec cette modulation du signal. L'Arduino a de nombreux pôles qui sont utilisés comme entrée et sortie, et leur nombre varie en fonction du type d'Arduino [7]

Dans le tableau suivant, un résumé des broches **PWM** utilisées dans chaque carte Arduino :

Type ou modèle d'Arduino	Broches disponibles PWM
Uno	3, 5, 6, 9, 10 et 11
Pro Mini	3, 5, 6, 9, 10 et 11
Nano	3, 5, 6, 9, 10 et 11
Leonardo	3, 5, 6, 9, 10, 11 et 13
Due	2 à 13
Zéro	2 à 13
Méga	2 à 13 et 44 à 46

Tableau I.1 : Les broches **PWM** dans chaque carte Arduino. [7]

Les pôles **PWM** peuvent être identifiés par la barre oblique devant le nom du pôle, comme dans la figure suivant :

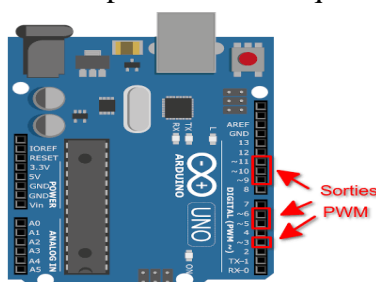


Figure I.12 : Les ports sorties **PWM** dans Arduino Uno

2.3.4 Les Avantages de PWM : deux avantages ont lieu et qui sont :

- Le moteur peut être piloté par la sortie numérique d'un Arduino.
- Amélioration considérable du rendement énergétique.

2.4. Le rapport cyclique

Le rapport cyclique désigne pour un phénomène périodique, le ratio entre la durée du phénomène sur une période de temps et la durée de cette même période. On parle souvent de rapport cyclique lorsqu'on a un signal rectangulaire.

$$\text{Le rapport cyclique} = \frac{T_{\text{on}}}{T} = T_{\text{on}} \times f$$

Tel que : T_{on} : temps à l'état on « High » dans une période du signal

T : période du signal

f : fréquence du signal

Le rapport cyclique pourra varier de 0% (Tension nulle 0V) jusqu'à 100% (Tension égale à 5V). En revanche, au niveau Arduino, il n'attribuera pas une valeur comprise entre 0 et 100 mais une valeur scalaire entre 0 et 255 (8 bits).

En faisant une règle ou une loi, il est possible de déterminer pour chaque entier (**PWM**) compris entre 0 et 255 la valeur du rapport cyclique ou la tension moyenne en sortie de l'Arduino :

$$\text{rapport cyclique} = \left(\frac{\text{PWM}}{255} \times 100\% \right)$$

$$\text{Tension} = \text{rapport cyclique} \times 5V$$

La figure suivante montre différents cas possible du rapport cyclique :

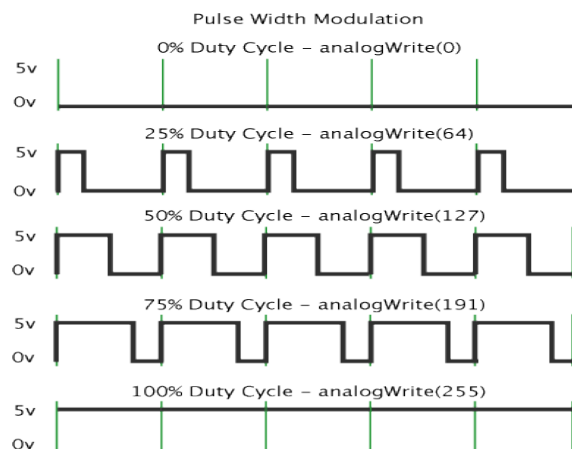


Figure I.13 : Le rapport cyclique

2.4.1 Différents cas possible du rapport cyclique :

PWM = 0	→ rapport cyclique = 0%	→ tension = 0V
PWM = 33	→ rapport cyclique = 10%	→ tension = 0.5V
PWM = 64	→ rapport cyclique = 25%	→ tension = 1.25V
PWM = 127	→ rapport cyclique = 50%	→ tension = 2.5V
PWM = 191	→ rapport cyclique = 75%	→ tension = 3.75V
PWM = 255	→ rapport cyclique = 100%	→ tension = 5V [6]

2.5. Control de la vitesse du moteur

Pour faire varier la vitesse d'un moteur à courant continu, il faut faire varier la tension d'alimentation aux bornes du moteur. Mais l'Arduino ne sait délivrer que des tensions de 5V sur ses broches avec un courant beaucoup trop faible pour alimenter un moteur. La solution est faire appel à la **PWM** et d'amplifier le signal par un transistor. [8]

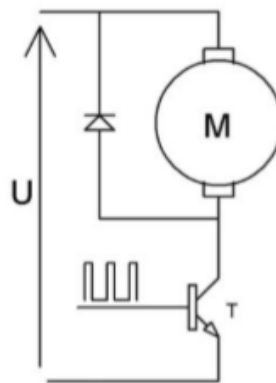


Figure I.14 : Schéma électrique du contrôle de vitesse d'un moteur DC

Dans l'Arduino, grâce à sa fonction "analog Write", un signal carré est créé pour basculer entre le niveau bas (LOW = 0V) et le niveau haut (HIGH = + 5V), avec une fréquence fixe (environ 490 ou 980 Hz). Son cycle de service peut être modifié, c'est-à-dire le temps de haut niveau avec le temps de cycle du signal. [5]

Ce signal est généré sur la broche Arduino étiquetée PWM, grâce à la fonction "analog Write" qui spécifie le rapport cyclique à utiliser. La valeur est comprise entre 0 (cycle d'utilisation égal à 0 %) et 255 (cycle d'utilisation égal à 100 %) et toute valeur intermédiaire peut être utilisée (par exemple, le rapport cyclique 191 est égal à 75 %). Le signal est amplifié par un transistor, et parce qu'il fonctionne en commutation, sa consommation électrique est quasi nulle. Le moteur qui reçoit ce type de signal est toujours alimenté par sa tension

d'alimentation nominale, son couple est donc maximal. D'autre part, le moteur traite ce signal (signal PWM) comme la tension et le courant moyens égaux à la tension d'alimentation (5V ou 3,3V) multipliée par le rapport cyclique ; par conséquent, il ne tourne pas aussi vite. Pour modifier la vitesse du moteur, il vous suffit de modifier le rapport cyclique du signal PWM fourni par Arduino. [8]

2.6. Mesure du signal PWM :

Pour afficher le signal **PWM** et le changement du **rapport cyclique** un oscilloscope analogique connecté à une sortie numérique configurée dans un **PWM** peut être utilisé.

I.3. Simulation Graphique Avec LabView

Introduction

Les programmes de logiciel LabVIEW sont appelés instruments virtuels ou VIs car leur apparence et leurs fonctions sont similaires à celles d'instruments réels, tels que les oscilloscopes et les multimètres. Le logiciel LabVIEW comprend une large gamme d'outils d'acquisition, d'analyse, d'affichage et enregistrement des données. Ainsi que, des outils pour développement des programmes [9]

3.1 Utilité de LabView

Dans LabVIEW, on peut créer une interface utilisateur ou une face-avant avec des commandes et des indicateurs. Les commandes sont des boutons, des cadrans et d'autres mécanismes d'entrée. Les indicateurs sont des graphiques, des LED et d'autres outils affichage de sorties. Après avoir construit la face-avant, on peut utiliser des VIs et des structures pour ajouter du code afin de contrôler les objets de la face-avant. Le diagramme contient ce code [9]

On peut utiliser LabVIEW, aussi, pour communiquer avec le matériel, afin d'acquérir de données ou d'images via les périphériques de contrôle d'axe et les instruments GPIB, PXI, VXI, RS232 et RS485. [9]

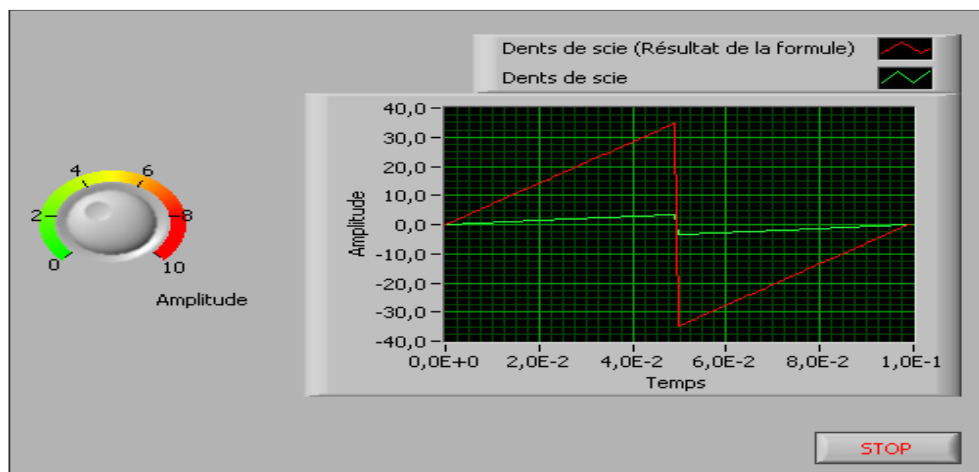


Figure I.15 : Face avant d'un instrument virtuel qui acquérir un signal

3.2. Description de la simulation

La simulation est le processus d'utilisation d'un logiciel pour recréer et analyser un comportement dynamique du système. Vous pouvez utiliser le processus de simulation pour réduire les coûts de développement de produits en accélérant le développement de produits.

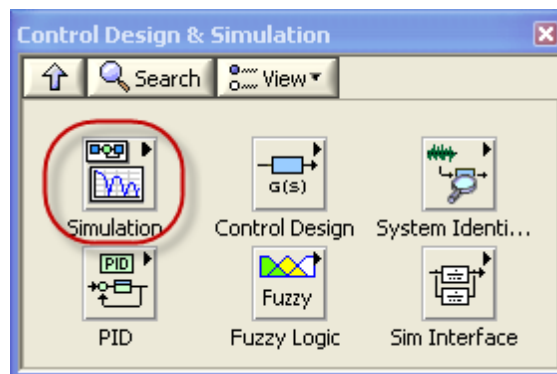
Vous pouvez également utiliser le processus de simulation pour mieux comprendre le comportement des systèmes dynamiques qui ne peuvent pas être facilement reproduits en laboratoire.

Par exemple, par rapport à la construction, aux tests et à la reconstruction d'un vrai moteur à réaction, un moteur à réaction simulé peut économiser du temps, de la main-d'œuvre et de l'argent.

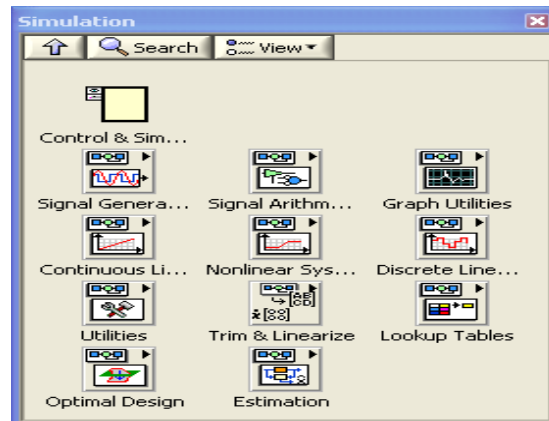
On peut exploiter le module LabVIEW Control Design and Simulation' pour simuler des systèmes dynamiques ou des composants d'un système dynamique, comme par exemple, le matériel des contrôleurs, des actionneurs et des capteurs. Un système dynamique peut être modélisé par une équation différentielle qui décrit son comportement [10].

3.3 Simulation avec LabVIEW

Utilisez les VIs et les fonctions de simulation pour créer des applications de simulation dans LabVIEW. Dans la palette Control Design & Simulation, nous avons la sous-palette Simulation :



La palette Simulation dans LabVIEW est représentée dans la figure suivante:



Les principales fonctionnalités de la palette Simulation sont :

- Boucle de contrôle et de simulation - Vous devez placer toutes les fonctions de simulation dans la boucle de contrôle et de simulation ou le sous-système de simulation.
- Fonction de système linéaire continu-Utilisez la fonction de système linéaire continu pour représenter le système linéaire continu de l'équation différentielle sur la simulation graphique.
- Fonction arithmétique du signal - Utilisez des fonctions arithmétiques du signal pour effectuer des opérations arithmétiques sur les signaux dans les systèmes analogiques. [11]

3.4 Représentations graphiques des données numériques :

Dans LabVIEW, il y a 3 types de représentations graphiques : les graphes déroulants, les graphes et les graphes XY.

3.4.1 Les graphes : ils permettent l'affichage des résultats de simulation en fonction du temps à la fin de l'exécution d'un instrument virtuel.

3.4.2 Les graphes déroulants : via le menu Palette des commandes/Moderne/Graphe, on lance un graphe déroulant montré sur la figure 3.2. Ce graphe nous permet d'afficher une variable, en temps réel, au court du temps en choisissant l'exécution en mode continu pour visualiser les résultats de simulation.

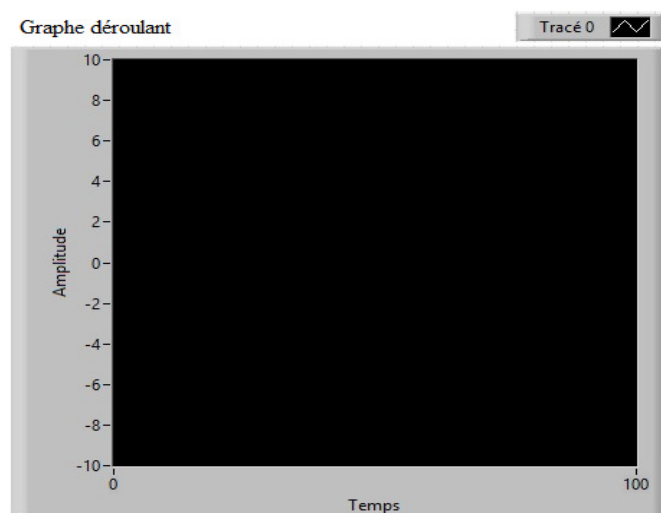


Figure I.16 : Graphe déroulant

3.4.3 Les graphes XY : ils permettent de tracer une quantité en fonction d'une autre et ils n'acceptent que des tableaux en entrée. La première entrée est sauvegardée sous le tableau X et la deuxième entrée est enregistrée sous le tableau Y.

II. PARTIE PRATIQUE










*II.1 Conception De La Carte Sur
Proteus*

1.1 Introduction

Afin de réaliser et simuler notre carte électronique, on a choisi le logiciel le plus utilisé à cause de sa simplicité 'PROTEUS'. Dans les parties ci-dessous, on présente les détails techniques de cette réalisation.

1.2. Composants de la carte

Le tableau suivant regroupe les éléments électroniques utilisés dans notre réalisation :

Composant	Nombre demandé	Modèle électronique
1. Arduino Méga	1	
2. Oscilloscope	1	
3. Moteur	2	
4. Optocoupleur	1	
5. Diode	1	
6. Led	2	
7. Mosfet	1	
8. LM35	1	
9. Résistance Variable (100 kOhm)	1	



10. Bouton-poussoir	1	
11. Résistance	2	

Tableau II.1 : Les éléments électroniques utilisés dans la simulation

1.3. Schéma électronique :

Notre schéma électronique montré par la **Figure II.1**, est divisé en quatre parties, à savoir :

- Partie de contrôle
- Partie traitement et connexion
- Partie puissance
- Partie d'affichage

Les différents éléments sont connectés dans le circuit électronique en fonction des étapes suivantes :

- Dans la partie commande, la résistance variable et le capteur LM35 sont connectés à Vcc et GND, tandis que leur troisième électrode est connectée aux entrées analogiques de l'ArduinoA0 et A1 respectivement. Le bouton poussoir est connecté à GND et sa deuxième électrode est connectée en parallèle avec l'électrode de réinitialisation sur l'Arduino et une résistance de 10 kOhm, cette dernière connectée à VCC.
- La partie puissance se compose de deux parties. La première partie représente le circuit de protection et d'isolation entre la faible et la grande puissance. Ce circuit connecte une des pins de la photodiode de l'opto-coupleur et une led à la sortie 9 (sortie PWM) et les autres pins sont connectés à la masse. L'opto-coupleur est connecté à la fois à la base de MOSFET et à la résistance de 10 kOhm. Tandis que, la source de MOSFET est liée au moteur principal. Une source d'alimentation (générateur de courant) est ajoutée en parallèle avec le transistor de l'optocoupleur et le drain MOSFET. Pour protéger le moteur des courants rétrogrades, une diode est insérée. La deuxième partie représente le circuit de refroidissement du moteur principal. Elle contient un moteur qui entraîne un ventilateur. Ce moteur est lié entre le pôle 6 de l'Arduino (sortie numérique) et une lampe connectée à la masse.

- Pour l’affichage, un oscilloscope est utilisé pour montrer les changements du signal de sortie de la broche 9 de l’Arduino. Dans le circuit électronique, l’oscilloscope sera connecté directement à la sortie 9.
- Le traitement et la liaison entre les différentes parties de notre carte sont assurés par l’Arduino.

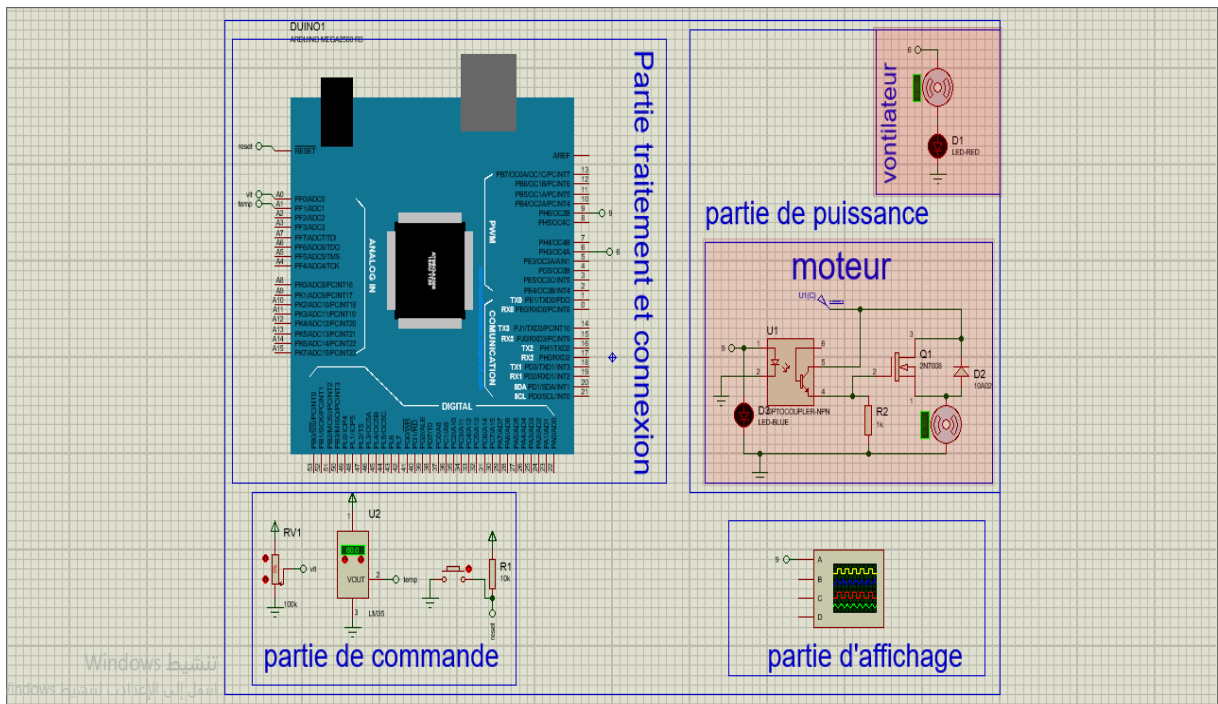


Figure II.1 : Schéma électronique

1.4. Description du fonctionnement

1.4.1. Partie de contrôle : La partie contrôle est chargée d’envoyer les informations sur la température du moteur le démarrage de fonctionnement à l’Arduino. Elle contient un capteur de température LM 35, bouton Poussoir et une résistance variable.

- **Capteur LM 35 :** il s'agit d'un capteur de température avec une tension de sortie linéairement proportionnelle à la température en degrés Celsius. Il mesure la température de l'environnement extérieur du moteur et envoie ces mesures à l'entrée (A1) de l'Arduino (**Figure II.2**).
- **Résistance Variable :** Une résistance variable est une résistance tripolaire qui agit comme un diviseur de tension et est très utilisée dans les appareils électroniques. Dans ce circuit, elle servira à envoyer différentes valeurs de tension à l'entrée (A0) de l’Arduino (**Figure II.3**).

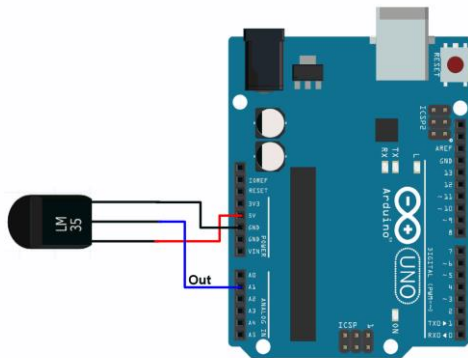


Figure II.2 : Liaison du capteur LM35 à l'Arduino

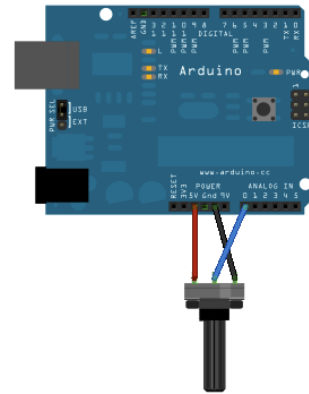
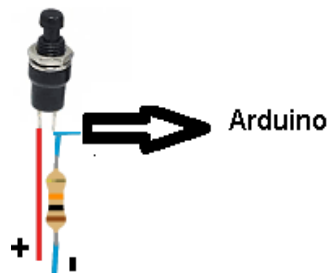


Figure II.3 : Liaison de la résistance Variable à l'Arduino

- **Bouton Poussoir :** Le bouton-poussoir est considéré comme un interrupteur. Dans ce circuit, le bouton contrôle la réinitialisation de la carte Arduino, il est connecté à l'électrode de reset.

Figure II.4 : Le bouton poussoir



1.4.2. Partie de traitement et de connexion : l'Arduino est programmé pour effectuer les tâches suivantes:

- Recevoir les différentes valeurs de résistances variables puis les traiter et les envoyer via le pôle 9 (sortie PWM) afin de contrôler la vitesse du moteur.
- Recevoir les valeurs de LM35, puis les traiter et les comparer à une température de 60 degrés, et au cas où elle dépasse cette valeur, le moteur du ventilateur refroidit le moteur principal.
- Si le bouton-poussoir est enfoncé, l'Arduino se réinitialise.

1.4.3. Partie de puissance : Premièrement, le moteur principal est commandé de la manière suivante :

l'opto-coupleur reçoit les signaux PWM de la broche 9 de l'Arduino (sortie PWM) et contrôle le MOSFET, qui à son tour contrôle la tension traversant le moteur.

Deuxièmement, le moteur du ventilateur est un dispositif de sécurité qui réduit la température du moteur pour assurer son bon fonctionnement. Le moteur du ventilateur reçoit des informations

et des commandes de fonctionnement via la sortie digital 6 de l'Arduino, où les valeurs sont soit High (on) ou Low (off) en fonction de la température du capteur thermique LM35.

1.4.4. Partie d'affichage : Dans cette partie, les modifications apportées à la sortie 9 de l'Arduino seront affichées et les changements de cycle de service seront surveillés en utilisant l'oscilloscope.

1.4.5. Algorithme de travail de l'Arduino : la **Figure II.5** donne l'organigramme de programmation de l'Arduino extraire du fonctionnement suivant :

La longueur de la largeur PWM est contrôlée manuellement via l'entrée analogique A0 sur l'Arduino via une résistance variable, puis traitée et transmise via la sortie 9 (sortie PWM). L'appareil contrôle la vitesse du moteur en fonction du cycle de service reçu de l'Arduino.

Afin de protéger d'avantage le moteur et d'assurer son fonctionnement normal, un système externe de surveillance de la température du moteur a été créé et le LM35 est connecté à l'entrée analogique A1 de l'Arduino. Il surveille la température externe du moteur, puis l'Arduino compare la température à 60°C. Si la température est supérieure, l'Arduino envoie la commande de démarrage, mais si elle est inférieure, une commande d'arrêt est envoyée au moteur du ventilateur, tout au long de la sortie digital 6.

Afin d'afficher et de surveiller le changement de vitesse du moteur, l'entrée de l'oscilloscope est connectée à la broche Arduino 9, où l'oscilloscope affiche clairement le rapport cyclique du moteur de commande.

1.5. Résultats de test

Après avoir écrit le code Arduino et fait une simulation sur le Proteus, quelques résultats ont été notés qui peuvent être résumés dans les points suivants :

- A l'état initial, avant le démarrage de la simulation, L'OSCILLOSCOPE est masqué et tous les composants électroniques sont à l'état mort.
- Au début du processus de simulation, un panneau d'oscilloscope est affiché, qui montre avec précision les changements dans le signal PWM.

La Figure II.6 montre l'état PWM dans son état initial (le rapport cyclique est 0). Il montre également que la température est de 60 degrés.

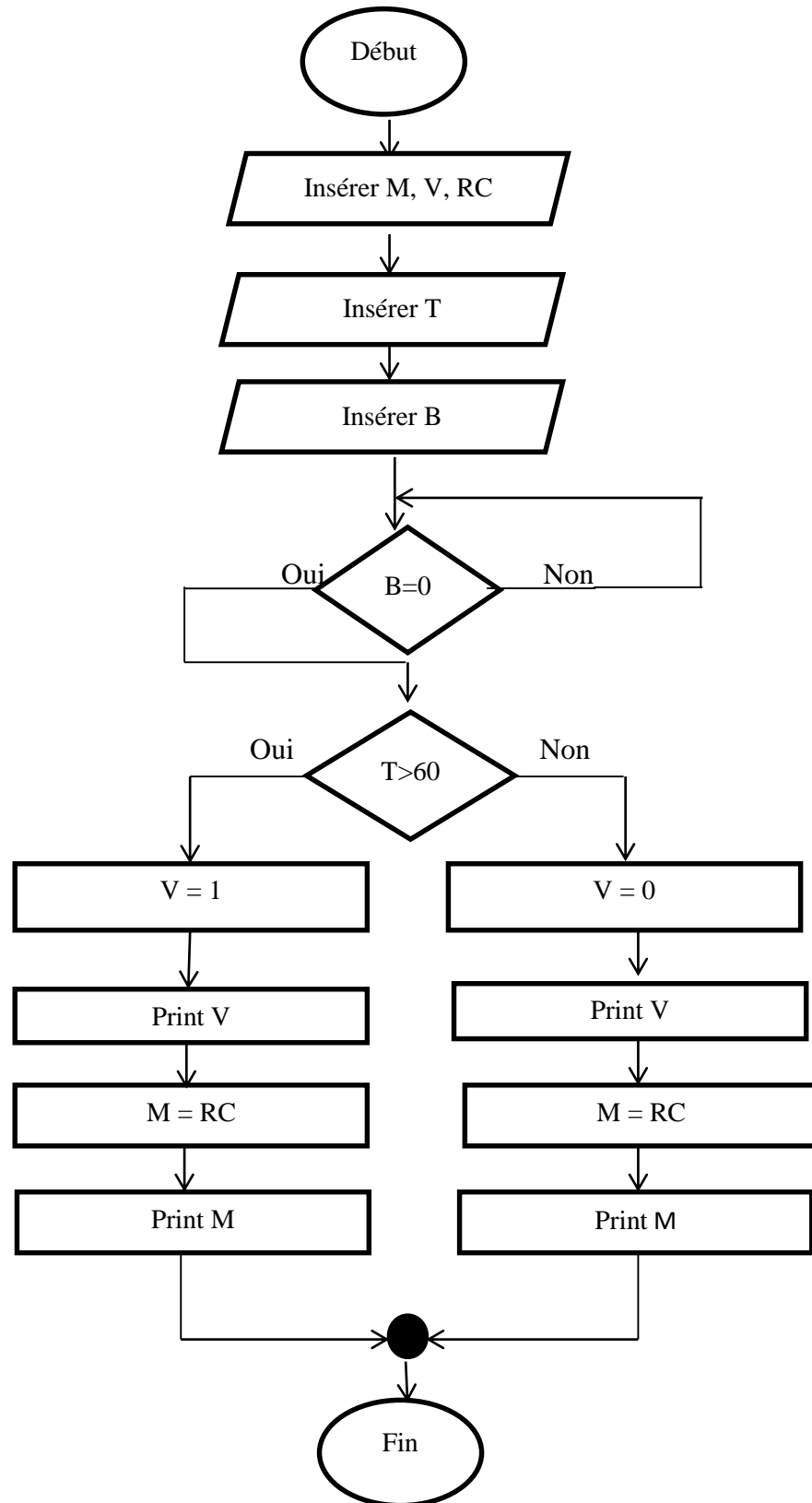


Figure II.5 : Algorithme de travail

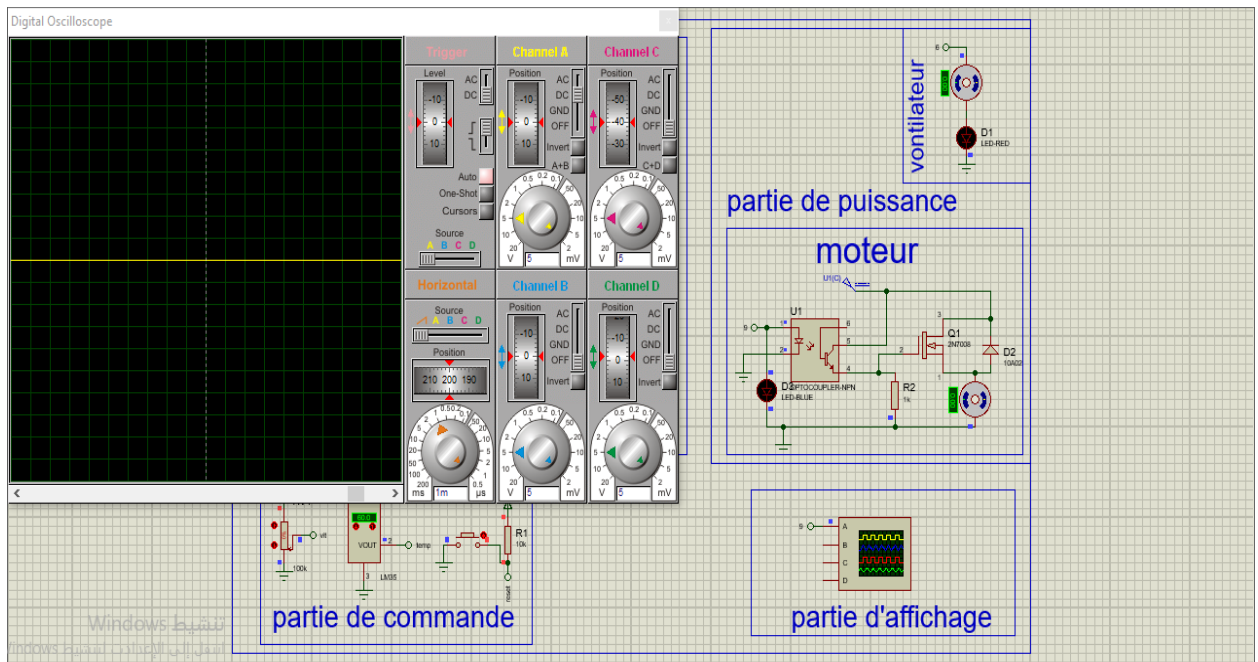


Figure II.6 : Circuit électronique au démarrage

- En manipulant la valeur de la résistance variable, on remarque que le rapport cyclique se change en relations directe avec la valeur de la résistance, où il est:

$$\text{rapport cyclique} = \frac{R}{R_T} \times 100\% = \frac{V_R}{V} \times 100\%$$

Telque :

R : La valeur de la résistance variable actuelle .

R_T : La valeur totale de la résistance variable (100 kOhm).

V_R : La tension de la résistance variable actuelle.

V : la tension totale de l'entrée de la broche analogique de l'Arduino.

La Figure II.7 montre le résultat de simulation de la carte avec une valeur de résistance variable de 67 kOhms et un rapport cyclique de 0.67, ceci est observé dans l'oscilloscope. Il est également à noter que la lampe et le moteur du ventilateur sont en position d'arrêt, car la température n'a pas encore dépassé les 60 C°.

On remarque sur la Figure II.8 que la lampe clignote et le moteur du ventilateur tourne car on a augmenté la température à une valeur supérieure à 60 C°. On observe aussi, que la rapport cuclique égal à 1et par conséquent, le moteur tourne avec sa valeur maximale.

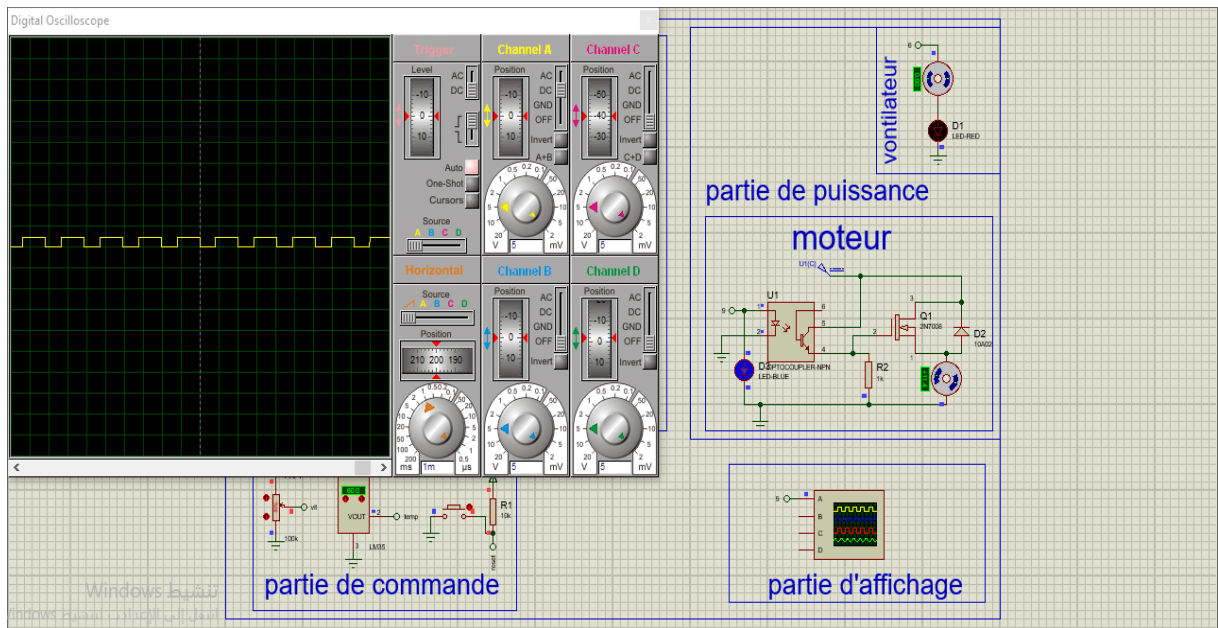


Figure II.7 : Circuit électronique lors du changement de valeur du rapport cyclique

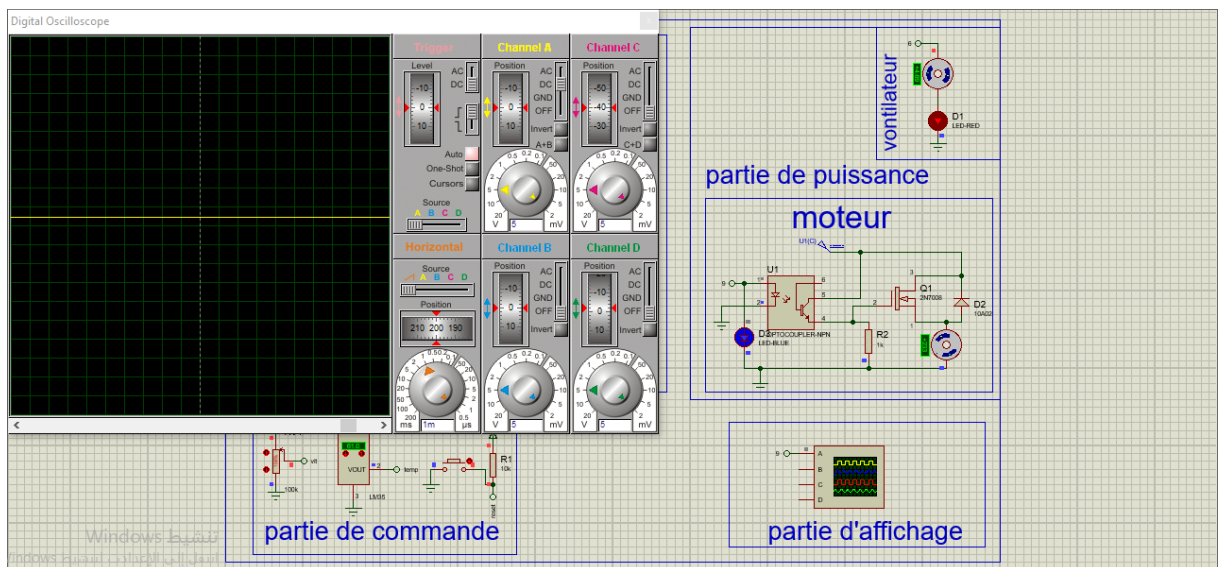


Figure II.8 : Circuit électronique avec un rapport cyclique maximale et une température supérieure à 60

II.2. Réalisation De L'interface De Contrôle

2.1 Introduction :

Afin de contrôler le moteur, on réalise une interface graphique sous LabView. Les icônes de l'instrument Input/Output : visa clear, read, write, close' sont utilisées. Il s'agit de permettre la réception et le traitement des données au fur et à mesure de leur envoi.

2.2 L'interface graphique :

On explique dans les sections ci-dessous les étapes de réalisation de diagramme et de la face avant de notre interface de contrôle.

2.2.1 Diagramme : la **Figure 2.1** représente le digramme ou le fonctionnement interne du contrôleur. Ce diagramme assure les tâches suivantes :

- Pour envoyer les informations de contrôle : On met d'abord une icône '*visa clear*' liée à une entrée de contrôle '*visa resource name*' et aux deux icônes '*visa write*' chargées de l'envoi des données. L'une d'eux est connectée à un contrôleur numérique (il peut être considéré comme l'élément qui contrôle la vitesse du moteur), tandis que la deuxième icône est connectée à une lettre fixe (par exemple la lettre 'v'). Le rôle de cette dernière est la séparation des numéros envoyés de la première icône '*visa write*'. Le processus d'envoi des données de contrôle du régime moteur est scellé en liant les deux icônes '*visa write*' à deux icônes '*visa close*'.
- Pour recevoir les informations de contrôle : De la même manière, une icône '*visa clear*' est associée à une entrée de contrôle '*visa resource name*' et à une icône '*visa read*' qui permet de lire les données reçues. Ainsi, le processus se termine par un lien avec l'icône '*visa close*'.
- Un interrupteur a été ajouté au diagramme, dont le but est de commuter et de choisir entre le contrôle de la vitesse du moteur via la résistance variable du Proteus (en sélectionnant la lettre A) ou via le contrôleur numérique de Labview(en en sélectionnant la lettre M).

2.2.2 Face avant : la face avant de la **Figure 2.2** permet le contrôle graphique direct de notre application, via des boutons et des afficheurs comme suit :

- Le bouton rotatif envoi la valeur de la tension du moteur et l'autre bouton permet l'arrêt de la simulation.

- La valeur de la PWM transmise par l'Arduino est affichée sur un graphe en fonction du temps. Les changements de la température du moteur sont indiqués par un thermomètre et sont affichés aussi, dans un graphe. Une led est allumée en rouge, si la température dépasse 60 degrés.
- Un interrupteur électrique a été ajouté afin de choisir librement entre utiliser le Proteus ou le Labview.

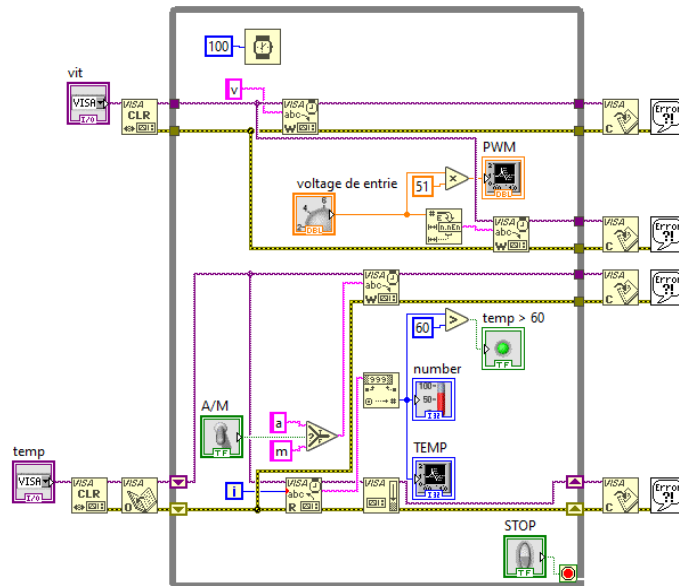


Figure II.9: Diagramme de l'interface de contrôle

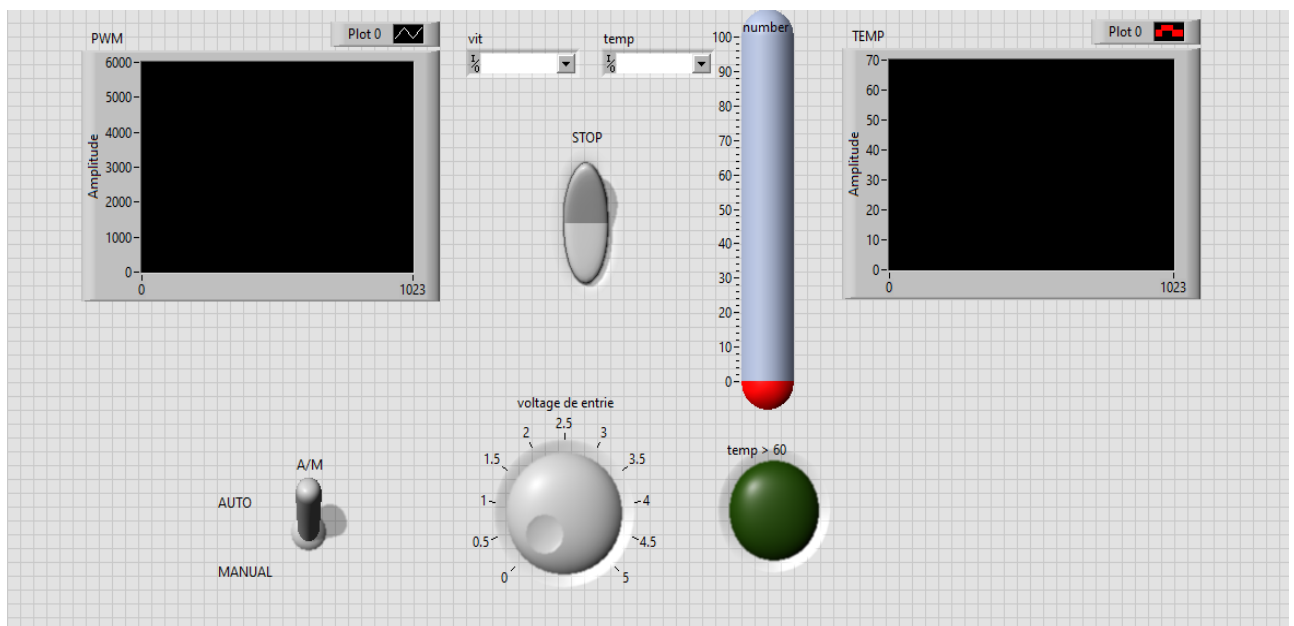


Figure II.10 : la face avant de l'interface de contrôle

2.3. Liaison de l'interface de contrôle avec la carte électronique :

La communication entre l'interface de contrôle et la carte électronique est réalisée par le port série RS232 à 9 broches. Le brochage et la configuration du port sont donnés à l'annexe A. une liste des signaux utilisés par ce port est donnée à l'annexe B.

Les Pins de port série utilisés dans notre application sont comme suit : dans le RS232 Rx et Tx, et dans le arduino 0, 1, 17 et 18

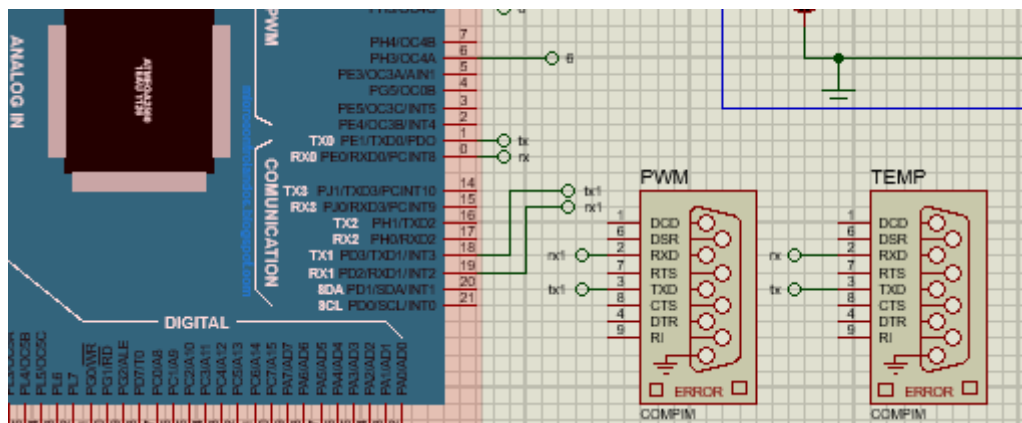


Figure II.11 : La configuration des pins

2.4. Liaison virtuelle :

Pour qu'il y a eu une communication entre le Proteus et le logiciel LabView et pour qu'on puisse exécuter notre simulation, il faut qu'on réalise une liaison virtuelle par un émulateur appelé 'VSPE'.

2.4.1 Définition de VSPE : (Free Virtual Serial Port Emulator) est un programme qui permet aux ingénieurs informatiques et aux développeurs de créer, tester et déboguer des applications qui utilisent le port série, son interface dans l'est montrée **Figure II.12**

Il permet la création de dispositifs virtuels pour envoyer et recevoir des données. VPSE vous permet de partager les données physiques de plusieurs applications, d'exposer des ports au réseau local, de créer des ports parallèles virtuels, etc.

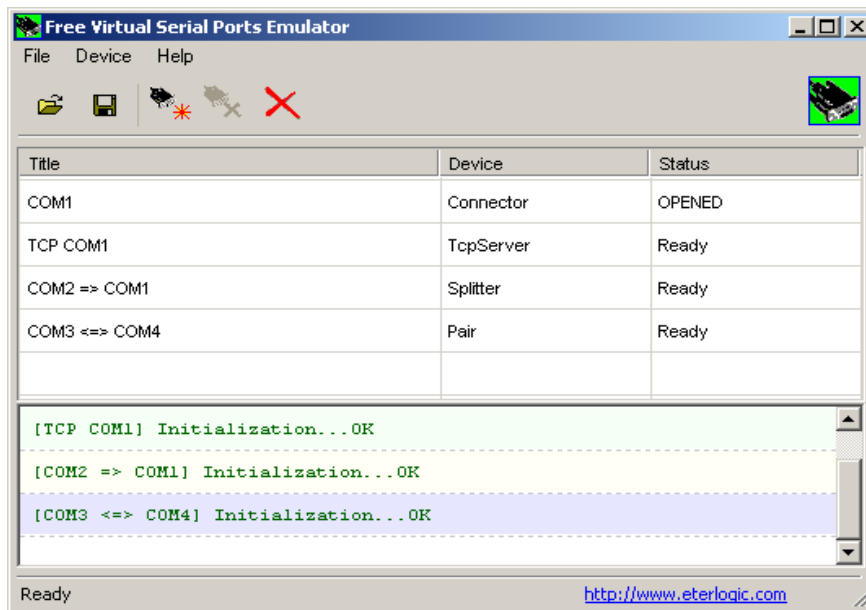


Figure II.12: L'interface utilisateur du programme.

Aujourd'hui, les ports séries ne sont plus largement utilisés. Raison c'est simple : c'est une chose très ancienne. Habituellement, les périphériques matériels modernes sont connectés à l'ordinateur via un port USB haut débit, et parfois leur propre pilote Usb To Com est installé. Le port série n'est pas installé dans la plupart de nouveaux ordinateurs portables.

2.4.2 Identification des pôles de connexion : dans l'objectif, de fournir un échange de données entre les deux utilitaires utilisés dans notre projet, on a réalisé une interface virtuelle simple en procédant comme suit :

On configure les pôles virtuels chargés de transmettre et de recevoir des données. Pour cela, on ouvre le programme VSPE, puis on clique sur ("Create new device") et on choisit l'option de connexion paire, comme il est indiqué à la **Figure II.13**. Puis, on choisit le nombre de pôles utilisés dans la connexion (**Figure II.14**). Dans notre cas, on a besoin de deux fils de connexion RS232, par conséquent, on définit quatre pôles : COM2 avec COM1 et COM4 avec COM3 (**Figure II.15**). L'un des d'eux fils sera utilisé pour envoyer les données de contrôle de vitesse du moteur via l'interface graphique, et l'autre sera utilisé pour recevoir la température externe du moteur.

Figure II.13 : Option de lien dans le Programme

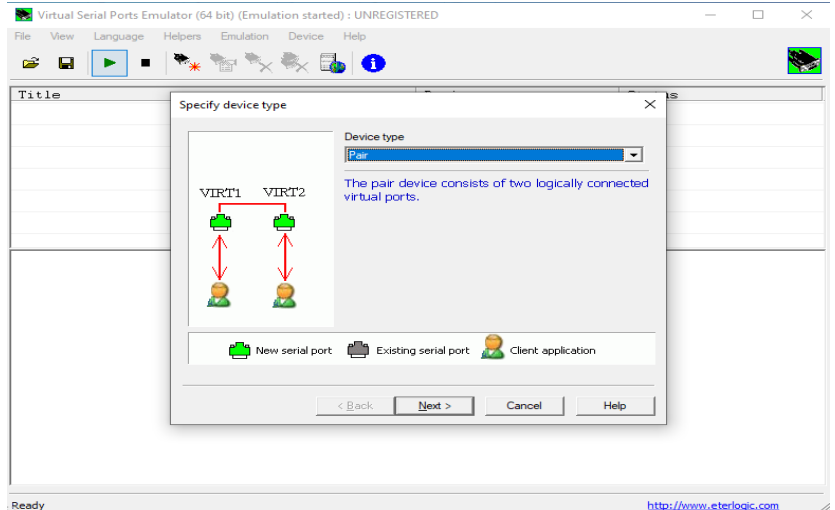


Figure II.14 : Identification des pôles

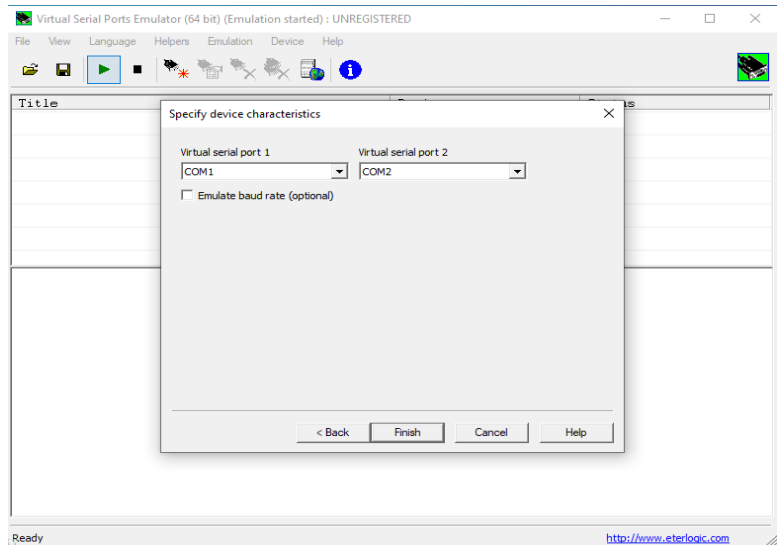
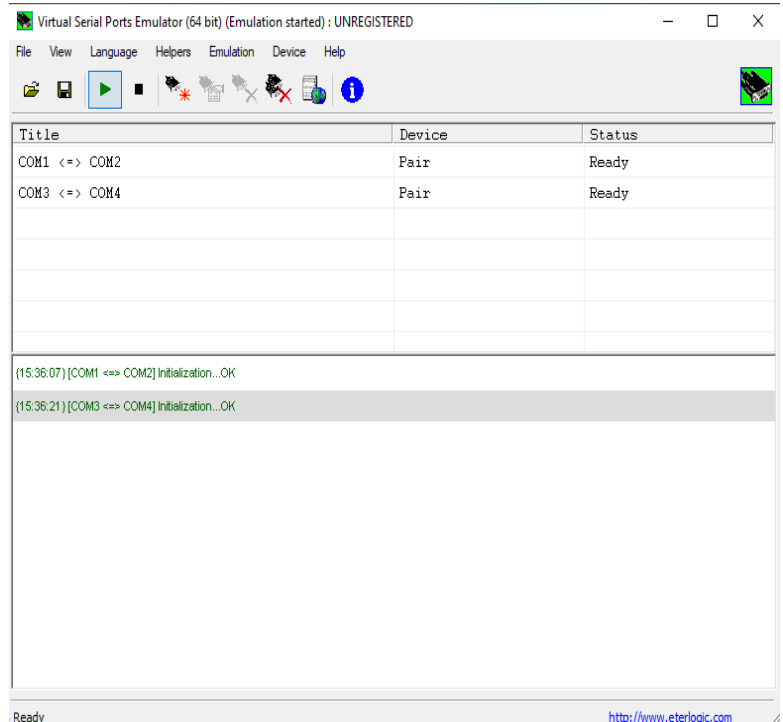


Figure II.15 : Les pôles utilisés dans La connexion



II.3. Simulation De La Carte Avec L'interface De Contrôle

3.1 Introduction :

Avant l'exécution de la simulation et puisque on a créé une interface de communication virtuelle, on doit configurer le Proteus et le LabView.

3.2 Configuration de Proteus :

L'Ajout de 2 ports COM virtuels (COMPIM) dans la carte électronique, comme il est indiqué dans la **figure II.16**. Le premier port est lié aux sorties 0 et 1 de l'Arduino (Tx et Rx), et le deuxième port est lié aux sorties 18 et 19 de l'Arduino (Tx1 et Rx1).

Afin d'afficher les données transmises et reçues, nous utilisons deux "terminaux virtuels", dont l'un est connecté à la sortie 1 pour afficher les informations de température envoyées au Labview, et l'autre est connecté à la sortie 18 pour connaître la valeur de la PWM reçu.

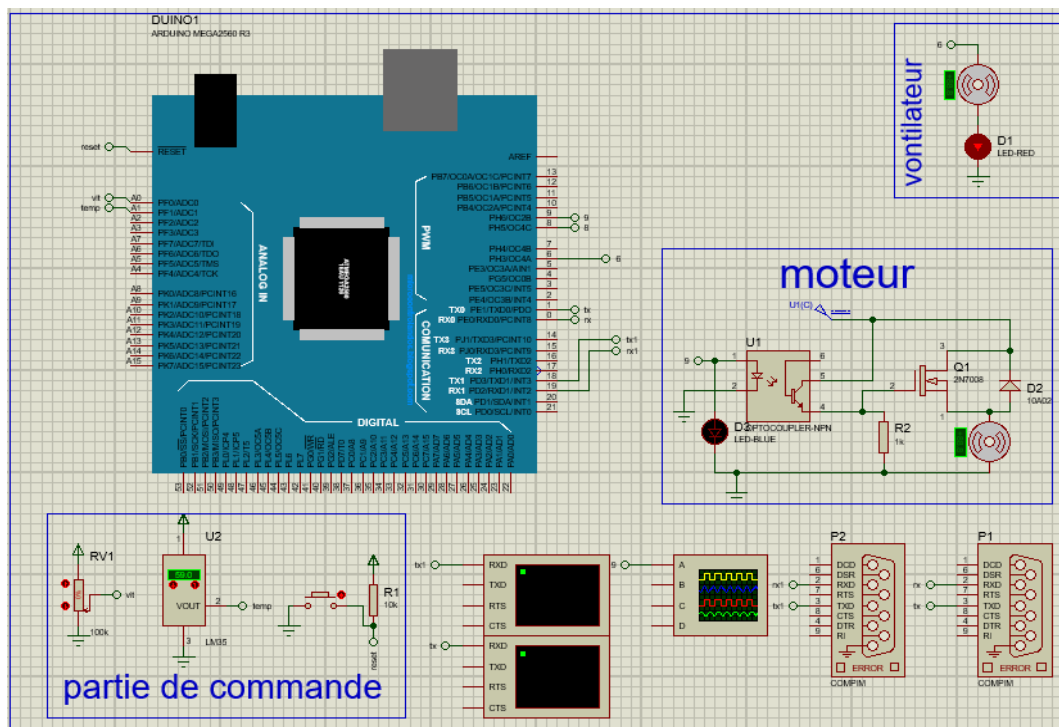


Figure II.16 : Schéma électronique après reconfiguration

Après avoir effectué ces modifications, on prépare les paramètres de COMPIM. Tout d'abord, choisissez le nom du port, puis sélectionnez le port actuel (Physical port), COM1 pour envoyer les valeurs de température et COM3 pour recevoir les valeurs PWM, Mettez le nombre 9600 dans chacun des « Physical baud rate » et « Virtual baud rate », comme il est illustré par la **Figure II.17**.

3.3 Configuration de LabView :

Maintenant, on spécifie les ports dans le "visa resource name". Le port COM2 est spécifié pour afficher les valeurs de température reçues, tandis que COM4 est spécifié pour envoyer les valeurs PWM. Bien sûr dans le VSPE, le port COM2 est lié au port COM1 et le port COM4 est lié au port COM3.

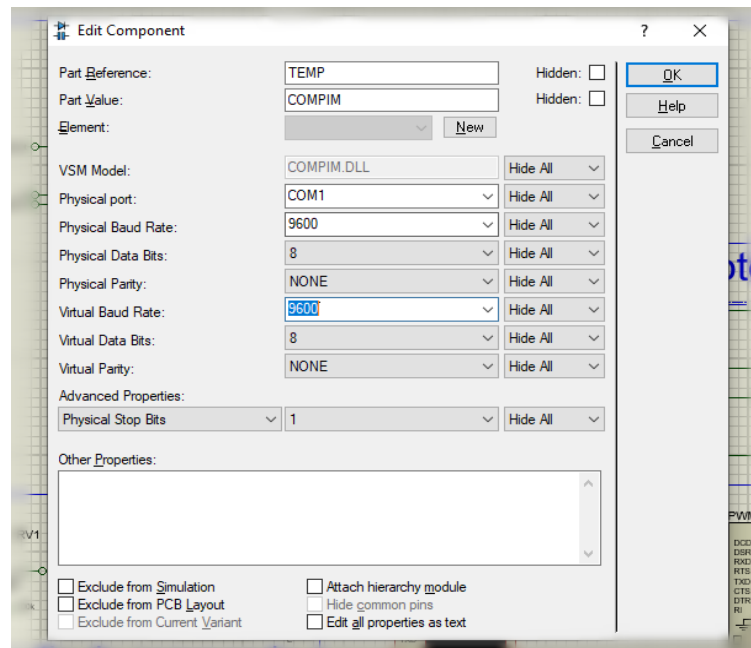


Figure II.17: La configuration de ports COM

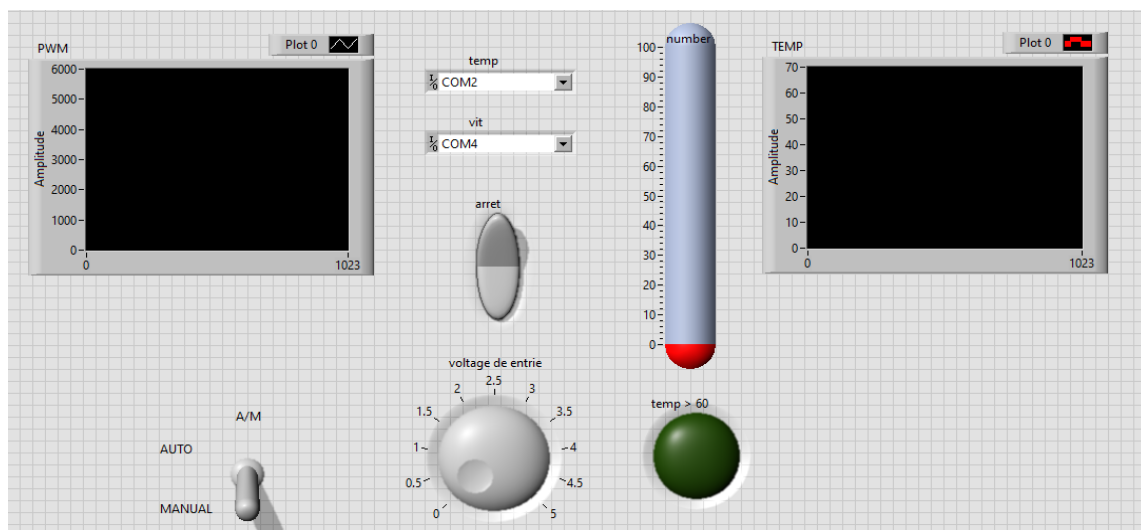


Figure II.18 : Sélection des COM dans l'interface de contrôle

3.4 Résultats de simulation :

Au démarrage de la simulation, l'interrupteur doit être en position "auto" pour permettre le pilotage du moteur via l'interface de commande. Ensuite, la valeur de température sera affichée immédiatement (dans cet exemple, elle commencera à partir de 59°C) et la valeur PWM transmise sera affichée (à partir de 0). Ces valeurs sont visualisées dans la **Figure II.19**.

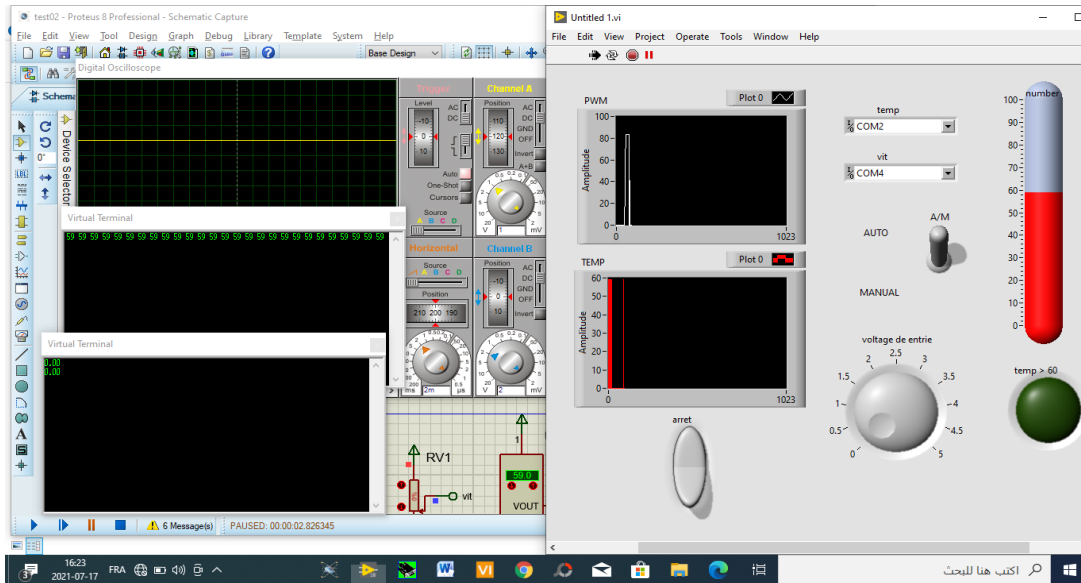


Figure II.19 : Démarrage de la simulation

Pendant la période de test, la valeur de la tension est modifiée (cela se fait en faisant tourner la roue représentant la résistance variable) provoquant la modification de la valeur PWM envoyée au Proteus et la vitesse de rotation du moteur (la valeur du cycle de service change). La température est constante et n'a pas été modifiée. Les modifications de la valeur PWM et de la température sont affichées dans les graphes de l'interface de la **Figure II.20**.

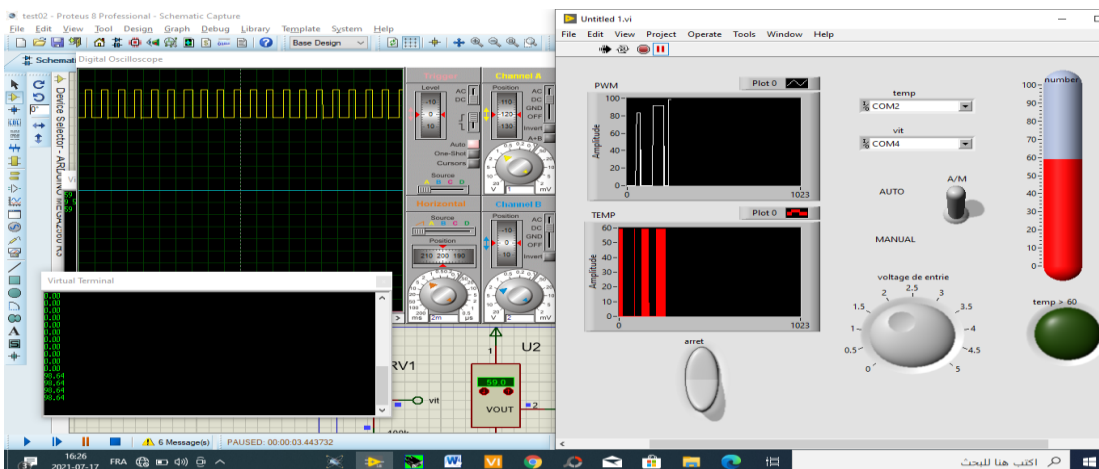


Figure II 20 : Changement de la vitesse de moteur

Dès que la température dépasse les limites de 60 degrés Celsius, une alerte est émise en allumant une lampe dans l'interface de commande, puis le ventilateur de refroidissement fonctionne automatiquement et la simulation se poursuit normalement. La **Figure II.21** donne les différentes mesures et indications.

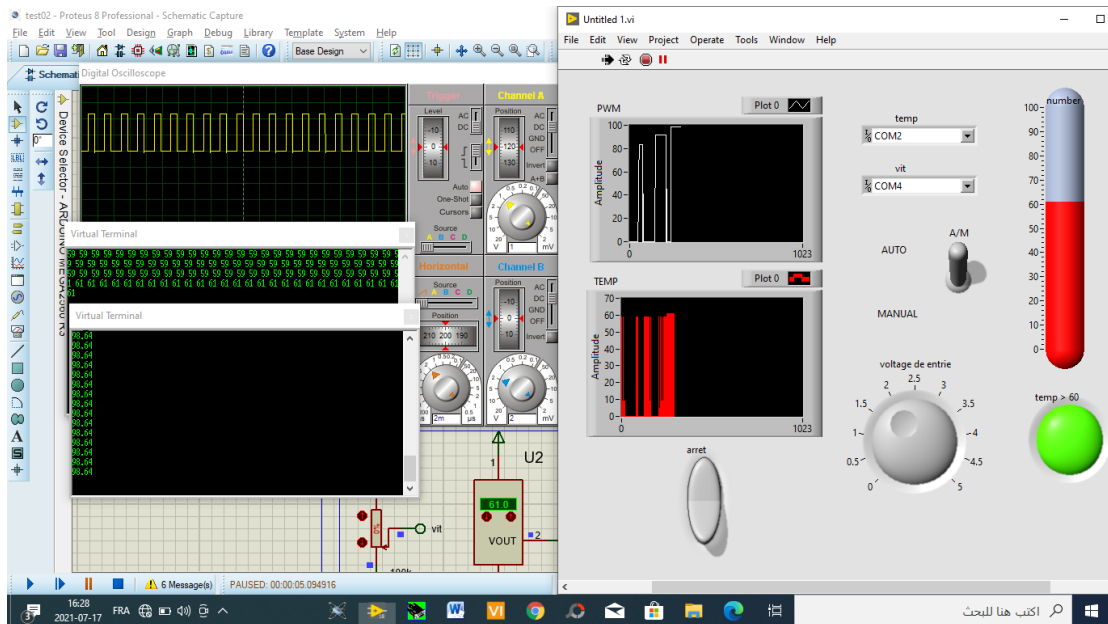


Figure II.21 : les différentes mesures dans le cas de dépassement de température

Lors du passage du mode "auto" au mode "manuel", Proteus arrête de recevoir les informations PWM et commence à contrôler manuellement via la résistance variable. Les changements de température sont affichés comme d'habitude dans un graphique et un thermomètre, tandis que l'affichage de la valeur PWM est arrêté dans le graphique dans l'interface de contrôle.

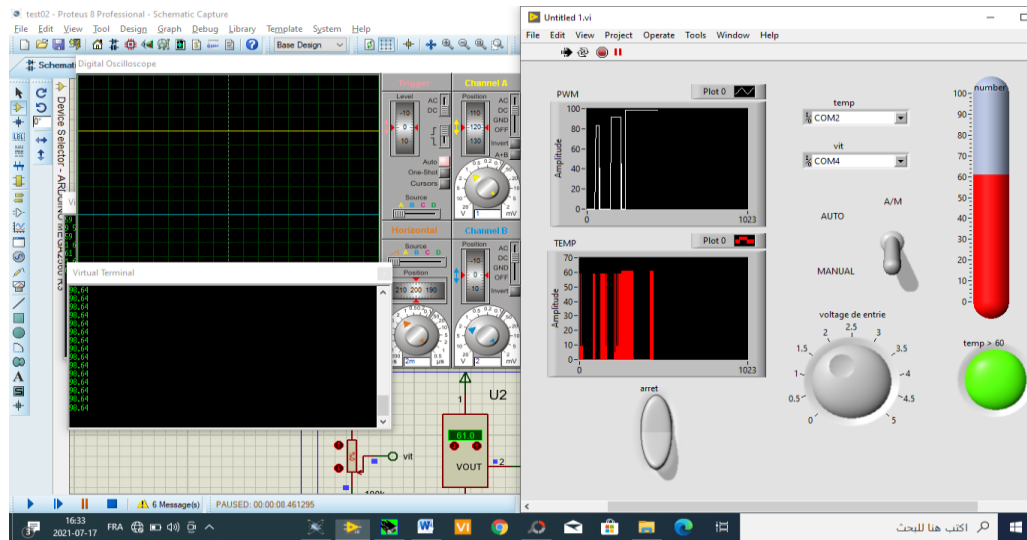


Figure II.22 : Changement méthode de contrôle de la vitesse de moteur

La valeur de la tension dans la résistance variable est modifiée, ce qui entraîne la modification de la valeur PWM entrant dans l'Arduino et la modification de la vitesse de rotation du moteur. Tout cela se fait sans aucune intervention de l'interface de contrôle.

On observe les modifications et les mesures de la température et de la PWM dans les commandes et les indicateurs de l'interface et dans l'interface de Proteus illustrés à

La Figure II.23

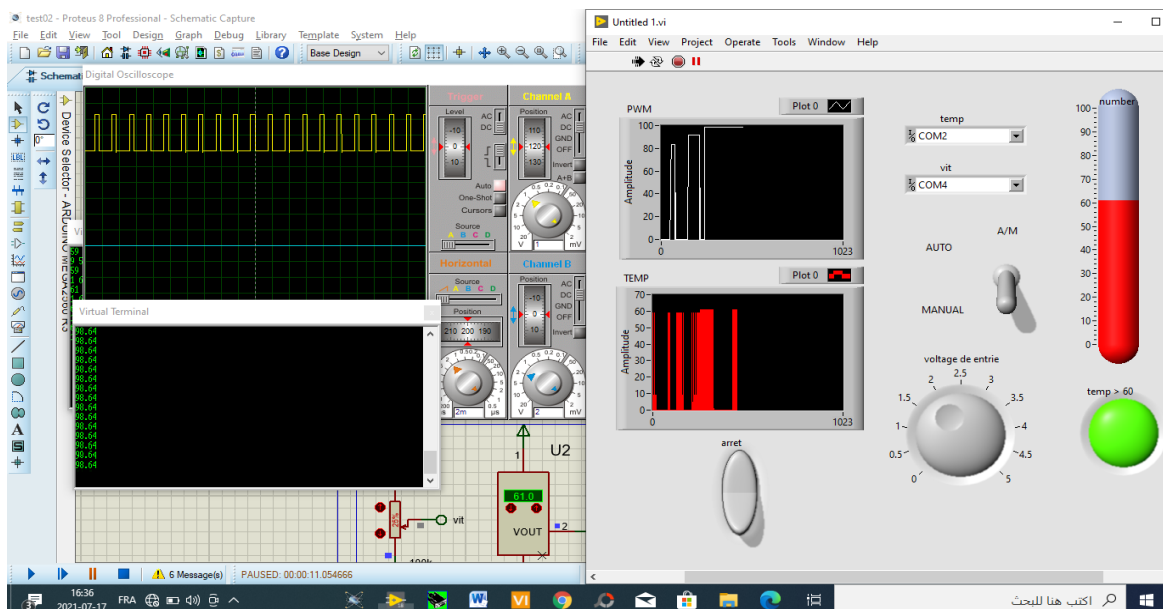


Figure II.23 : Le changement de la vitesse de moteur avec le proteus

Conclusion :

Grace aux avantages de la programmation graphique, on a arrivé à modéliser une interface de commande de moteur à courant continu. La simulation de cette interface dans différents états offre la possibilité de contrôler virtuellement le moteur d'une part, et de prendre quelque mesures et réglages des paramètres d'autre part.

Conclusion Générale

Dans le cadre de ce projet, on s'est intéressé à la réalisation et la simulation, d'une carte électronique de commande d'un moteur à courant continu avec l'Arduino, d'une part et d'une interface de contrôle de la vitesse de ce moteur, d'autre part.

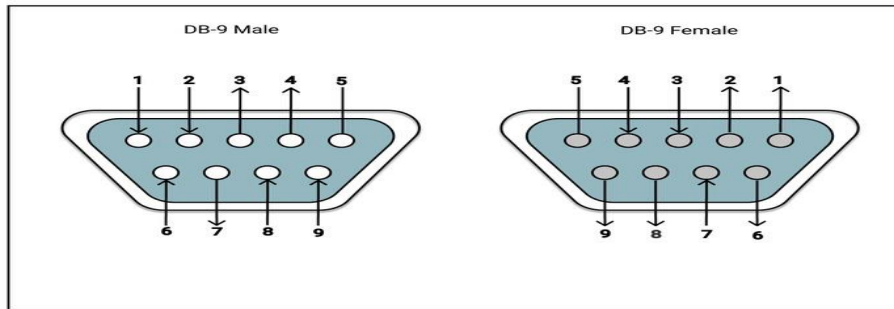
Pour cela, on a étudié et utilisé un outil de simulation et de programmation virtuel qui est destiné à la modélisation matérielle. Et on est arrivé à programmer les plusieurs parties fonctionnelles de l'interface de commande.

Ce projet était une occasion pour acquérir des connaissances sur un outil de programmation électronique et sur la conception électronique de la carte de contrôle de moteur DC.

Ce travail présente pour nous une première expérience dans le domaine d'électronique des moteurs, aussi il a été très bénéfique en matière d'acquis de langage graphique.

Annexe A : Brochage et configuration d'un port RS232 DB9

Ces connecteurs peuvent avoir des fils conducteurs RS232 femelles ou des fils conducteurs DB9 ou DB25 mâles illustrés par la figure ci-dessous.



Chaque broche d'un connecteur série possède sa propre fonction come il est indiqué dans ce tableau :

DB-9 Male				DB-9 Female			
Pin	Signal Direction	Signal Name	Signal Function	Pin	Signal Direction	Signal Name	Signal Function
1	→	CD	Carrier Detected	1	←	CD	Carrier Detected
2	←	RxD	Receive Data	2	→	TxD	Transmit Data
3	→	TxD	Transmit Data	3	←	RxD	Receive Data
4	→	DTR	Data Terminal Ready	4	→	DTR	Data Terminal Ready
5	—	GND	Ground	5	—	GND	Ground
6	←	DSR	Data Set Ready	6	←	DSR	Data Set Ready
7	←	RTS	Request To Send	7	→	CTS	Clear To Send
8	→	CTS	Clear To Send	8	←	RTS	Request To Send
9	→	RI	Ring Indicator	9	←	RI	Ring Indicator

→ Transmitted from DTE Device ← Receive by DTE Device	→ Transmitted from DCE Device ← Receive by DCE Device
--	--

Annexe B : Liste des signaux utilisés par un port COM RS232

- **Masse de protection** : Ce signal est connecté à la masse du châssis du connecteur métallique.
 - **Masse commune** : Tension nulle de référence pour tous les signaux de contrôle.
 - **TxD (Broche de transmission)** : Pour transmettre les données d'un ETTD vers un ETCD.
 - **RxD (Broche de réception)** : Envoie les données d'un ETCD vers un ETTD.
 - **DTR (Données prêtes)** : L'ETTD est prêt à accepter la requête.
 - **DCD (Détection d'un signal sur la ligne)** : L'ETCD accepte une transmission depuis un ETTD distant.
 - **DSR (Envoyez les données)** : L'ETCD est prêt à envoyer et à recevoir des informations.
 - **RI (Indicateur de sonnerie)** : Détecte la sonnerie entrante sur la ligne téléphonique.
 - **RTS (Demande de transmission)** : L'ETTD demande à l'ETCD d'envoyer les données.
 - **RTR (Réception des données)** : L'ETTD est prêt à recevoir les données provenant de l'ETCD.
- CTS (Prêt pour transmission)** : L'ETCD est prêt à accepter les données provenant de l'ETTD.

Bibliographie :

- [1] : <https://domolex.fr/>
- [2] : <https://www.cours-gratuit.com/cours-arduino/tutoriel-carte-arduino-mega-2560-pdf>
- [3] : Iabbaden Zinedine et Lahlou Farid, 'Réalisation d'un module de distribution d'énergie à base d'une carte Arduino méga 2560', mémoire de licence de l'université Mouloud M'Ammeri de Tizi ousou, 2016-2017
- [4] : Hezzat Nassima, 'Variateur de vitesse d'un moteur à courant continu 12V par NE555', mémoire de licence, Université Mohamed Bachir Ibrahim, 2018-2019
- [5] : <https://tutoduino.fr/blog-PWM>
- [6] : <https://www.arduino.cc/en/Tutorial/foundations/PWM>
- [7] : <https://electrotoile.eu/c-est-quoi-la-PWM-avec-arduino.php>
- [8] : <https://www.locoduino.org/spip.php?article213>
- [9] : https://www.ni.com/pdf/manuals/373427b_0114.pdf, 'LabVIEW™ Initiation à LabView National Instruments.
- [10] : https://d1wqtxts1xzle7.cloudfront.net/59382228/Control_and_Simulation_in_LabVIEW
- 20
- [11] : Projets Expérimentaux de Physique, (Exercices d'initiation à LabVIEW), Licence et magistère de physique fondamentale, Université Paris-Sud, 2018-2019