



وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
جامعة عبدالحميد ابن باديس مستغانم
Université Abdelhamid Ibn Badis de Mostaganem
كلية العلوم و التكنولوجيا
Faculté des Sciences et de la Technologie



N° d'ordre : M...../GE/2018

MEMOIRE DE FIN D'ETUDE DE MASTER ACADEMIQUE

Filière : Électronique

Spécialité : Electronique des systèmes embarqués

Thème

**CONCEPTION ET REALISATION D'UNE MATRICE DE
COMMUTATION POUR TRAVAUX PRATIQUES
DISTANTS A BASE DE SWITCHS NUMERIQUES**

Présenté par :

- Yassine Ouzar
- Abdelhamid Mebrouka

Soutenu le 28 / 06 / 2018 devant le jury composé de :

Présidente : K.BERRADJA

Encadreur : A.BENACHENHOU

Examineur 1: M.REBHI

Examineur 2: W.BENSTAALI

Année Universitaire : 2017 / 2018



Remerciement

Ce périple de cinq années arrive à son terme. Nous l'avons parcouru avec d'autres personnes qui nous ont accompagnées sur tout ou partie du chemin. Les bonnes idées sont celles qui viennent en marchant.

En tout premier lieu, nous remercions DIEU le tout puissant de nous avoir éclairé le chemin du savoir afin de terminer ce travail.

Egalement, nous tenons à exprimer notre sincère gratitude envers tous ceux qui nous ont aidé ou ont participé au bon déroulement de ce projet.

Nous tenons à remercier tout particulièrement notre encadreur Mr. Benachenhou Abdelhalim directeur du laboratoire d'électromagnétisme et optique guidée pour sa générosité, sa compréhension et son aide inestimable.

Un grand merci également à Mr Abderahmen l'ingénieur du laboratoire LEOG, ainsi qu'à Mr BENATTIA Abderahmen et MOUSSA Mohammed pour leurs soutiens durant nos travaux au laboratoire.

Nous remercions les membres du jury de nous avoir accordé l'honneur d'examiner et de valoriser notre travail.

Enfin nos remerciements vont à nos chers parents, frères, sœurs, et amis pour leurs soutiens.

Yassine et Abdelhamid

Table des matières

Introduction générale	1
Chapitre1: Architecture matérielle	3
1. Introduction	4
2. Description de l'architecture matérielle du laboratoire distant	5
2.1.Description de l'interface EFHI	6
2.1.1. Contrôleur	7
2.1.2. Description de la matrice de commutation	8
2.2.Description de l'interface PEB	9
2.2.1. Potentiomètre numérique	10
2.2.2. Mémoire One Wire DS28E07	12
3. Conclusion	14
Chapitre2: Architecture logicielle	15
1. Introduction	16
2. Description de l'architecture logicielle du laboratoire distant	16
2.1.Coté client	17
2.1.1. HTML	18
2.1.2. CSS	19
2.1.3. JavaScript	21
2.2.Coté serveur	22
2.2.1. Node.js	23
2.3.Protocole SPI	24
2.3.1. Généralité sur le protocole de communication SPI	24
2.3.2. Principe de fonctionnement d'une liaison SPI	24
2.4.Protocole 1-Wire	26
2.4.1. Généralité sur le bus 1-Wire	26
2.4.2. Caractéristiques principales	26
2.4.3. Principe de fonctionnement du protocole 1-Wire	27

3. Conclusion	30
Chapitre3: Conception et réalisation	31
1. Introduction	32
2. Implémentation logicielle	33
2.1. Structure de l'interface web	33
2.2. Mise en place du serveur web	34
2.2.1. Installation de node.js	34
2.2.2. Le gestionnaire de paquets NPM	35
2.2.3. Etapes de création du script serveur	35
2.2.4. Pilotage des switchs numériques	36
2.2.5. Programmation du potentiomètre numérique MCP4231	37
2.2.6. Programmation de l'EEPROM DS28E07	38
3. Conception du typon	40
3.1. La matrice de commutation	40
3.2. La carte de TP1	41
3.3. La carte de TP2	42
4. Réalisation des circuits	43
5. Manipulation à distance des travaux pratiques	47
6. Conclusion	49
Conclusion générale	50
Références bibliographiques	52
Annexes	54

Table des figures

Chapitre 1

Figure I-1:	Architecture matérielle du laboratoire distant	6
Figure I-2:	Schéma et brochage du PcDuino V2	8
Figure I-3:	Schéma de l'interfaçage du PcDuino avec le switch MAX4678	9
Figure I-4:	Désignation des circuits MCP	11
Figure I-5:	Interfaçage du PcDuino avec le MCP4231	12
Figure I-6:	Interfaçage du PcDuino avec l'EEPROM DS28E07	13

Chapitre 2

Figure II-1:	L'architecture logicielle du laboratoire distant	17
Figure II-2:	Principe de fonctionnement d'une architecture client/serveur	17
Figure II-3:	Structure générale d'un fichier html	19
Figure II-4:	Schéma de principe simplifié d'une liaison SPI	25
Figure II-5:	Schéma d'un bus de communication 1-Wire	27
Figure II-6:	Procédure d'initialisation: réinitialisation et impulsion de présence	27
Figure II-7:	L'adresse physique d'un circuit 1-Wire	28
Figure II-8:	La lecture d'un bit dans une communication 1-Wire	28

Chapitre 3

Figure III-1:	L'interface de développement et manipulation	34
Figure III-2:	Exécution du serveur web	36
Figure III-3:	Schéma électrique de la matrice de commutation sur ISIS Proteus	40
Figure III-4:	Schéma du PCB de la matrice de commutation généré par ARES Proteus	41
Figure III-5:	Schéma électrique de la carte de TP 1 sur ISIS Proteus	41
Figure III-6:	Schéma du PCB de la carte de TP 1 généré par ARES Proteus	42
Figure III-7:	Schéma électrique de la carte de TP 2 sur ISIS Proteus	42
Figure III-8:	Schéma du PCB de la carte de TP 2 généré par ARES Proteus	43
Figure III-9:	Les étapes principales de la réalisation du circuit avec la méthode chimique	44
Figure III-10:	Etapes de réalisation du circuit par la machine CNC	45
Figure III-11:	Réalisation du circuit par la machine CNC	45
Figure III-12:	Circuit final de la matrice de commutation	46
Figure III-13:	Circuit final de la carte de TP	46
Figure III-14:	Schéma de la carte de TP à base d'AOP	48
Figure III-15:	Schéma de la carte de TP dipôle	48
Figure III-16:	Laboratoire distant en fonctionnement réel	49



Liste des tableaux

Chapitre 1

Tableau I-1:	Brochage du contrôleur avec l'interface EFHI	9
Tableau I-2:	Brochage de la nappe EFHI – PEB	10

Chapitre 3

Tableau III-1:	Les fonctions de la bibliothèque SpiDev	38
-----------------------	---	----



Glossaire

AJAX	Asynchronous JavaScript And XML	
CNC	Computer Numerical Control	Commande numérique par calculateur
CSS	Cascading Style Sheets	Les feuilles de style en cascade
EFHI	Ethernet Flexible Hardware Interface	
HTML	Hyper Text Markup Language	
http	Hyper Text Transfer Protocol	
IoT	Internet Of Things	Internet des objets
IT	Information Technology	Technologie de l'information
MISO	Master Input Slave Output	
Moodle	Modular Object-Oriented Dynamic Learning Environment	
MOSI	Master Output Slave Input	
PCB	Printed Circuit Board	
PEB	Practical Evaluation Board	Interface d'évaluation pratique
SCLK	Serial Clock	Horloge
SPI	Serial Peripheral Interface	
SS	Slave Select	Sélection d'esclave
TIC	Technologie de l'Information et de Communication	
USB	Universal Serial Bus	



Introduction générale

Au cours des dernières années, Les Technologies de l'Information et de la Communication (TIC), Internet en particulier ont envahi de nombreux domaines tels que le commerce, les administrations, l'enseignement...etc.

L'application des TIC au domaine de l'enseignement a conduit à la création d'un nouveau processus d'apprentissage appelée e-Learning. Par conséquent, il a fait émerger de nouveaux outils et de nouveaux repères, provoquant un changement dans les pratiques pédagogiques. La révolution dans le domaine de l'enseignement et de la formation à distance a donné naissance aux laboratoires à distance et télé-TP. L'apparition de ces derniers répond à un besoin reconnu d'activités pratiques dans les sciences techniques et d'ingénierie.

1. Contexte

Les récents travaux de recherche en télé-TP d'électronique tentent de donner un nouvel élan à cette discipline en proposant des solutions et des architectures génériques favorisant la flexibilité et la réutilisation. Par conséquent, il apparaît deux types de laboratoires : les laboratoires virtuels et les laboratoires réels à distance.

Dans le contexte de notre travail, nous nous limitons à la classe des laboratoires réels à distance.

2. Problématique

La plus part des laboratoires réels à distance actuels utilisent un logiciel propriétaire comme Labview et un matériel coûteux pour les mettre en œuvre. D'autre part, les travaux pratiques dans le domaine de l'électronique demandent beaucoup de manipulations de la part de l'étudiant. Cet aspect doit être appliqué dans l'architecture qui sera utilisée pour le pilotage à distance.



Il est donc nécessaire d'avoir une architecture flexible tenant compte de toutes les configurations de manipulation prévues durant une manipulation distante.

3. Objectifs

Notre travail a principalement deux objectifs:

- Architecture matérielle flexible et reconfigurable à faible coût.
- Architecture logicielle basée sur des outils open source et qui tient compte de la limitation des ressources disponibles sur l'unité de traitement utilisée.

4. Organisation du mémoire

Le mémoire est organisé en trois chapitres:

Le premier chapitre: Architecture matérielle

Ce chapitre décrit les deux interfaces EFHI et PEB qui constituent l'architecture matérielle du laboratoire distant.

Le deuxième chapitre: Architecture logicielle

Le deuxième chapitre présente l'architecture logicielle pour la mise à distance des travaux pratiques.

Le troisième chapitre: Conception et réalisation

Le dernier chapitre expose l'implémentation logicielle et la réalisation pratique des différents circuits.

Nous achèverons notre mémoire par une conclusion générale.



Chapitre 1: Architecture matérielle

1. Introduction

La révolution technologique de l'électronique embarquée et le mouvement open source du matériel ces dernières années est de plus en plus populaire dans le monde. Arduino est le chef de file dans ce mouvement puis l'émergence d'autres plateformes telles que PcDuino et Raspberry Pi...etc.

L'apparition d'une variété de plateformes matérielles open source permet une diffusion rapide des connaissances, stimule l'innovation et accélère la conversion de l'idée à la réalisation.

Le choix de la plateforme appropriée dépend essentiellement des exigences du projet à réaliser. Notre choix s'est porté sur le PcDuino V2, car il est disponible au laboratoire et il répond parfaitement au cahier des charges en termes de performance, de coût et de ressources nécessaires pour la mise en œuvre du laboratoire distant.

Dans ce chapitre nous allons présenter l'interface EFHI (Ethernet Flexible Hardware Interface) et l'interface PEB (Practical Evaluation Board) ainsi que les différents circuits et composants utilisés en décrivant leurs principes de fonctionnement.

2. Description de l'architecture matérielle du laboratoire distant

Les travaux pratiques dans le domaine de l'électronique demandent beaucoup de manipulations de la part de l'étudiant qui doit réaliser des montages, visualiser des signaux, mesurer des grandeurs électriques...etc.

L'architecture matérielle du laboratoire distant devrait tenir compte cet aspect ainsi que toutes les manipulations prévues. Elle devrait être également flexible pour pouvoir s'adapter à plusieurs matières de l'électronique : électronique analogique, électromagnétisme, traitement de signal et électronique de puissance...).

L'architecture matérielle que nous proposons se compose principalement d'une interface matérielle Ethernet flexible EFHI (Ethernet Flexible Hardware Interface) et d'une interface d'évaluation des travaux pratiques PEB (Practical Evaluation Board) prévue pour chaque manipulation. L'interface EFHI est bâtie sur un contrôleur qui commande une matrice de commutation, L'ensemble contrôle la configuration de la carte PEB. [1] La figure I.1 illustre l'architecture matérielle du laboratoire distant.

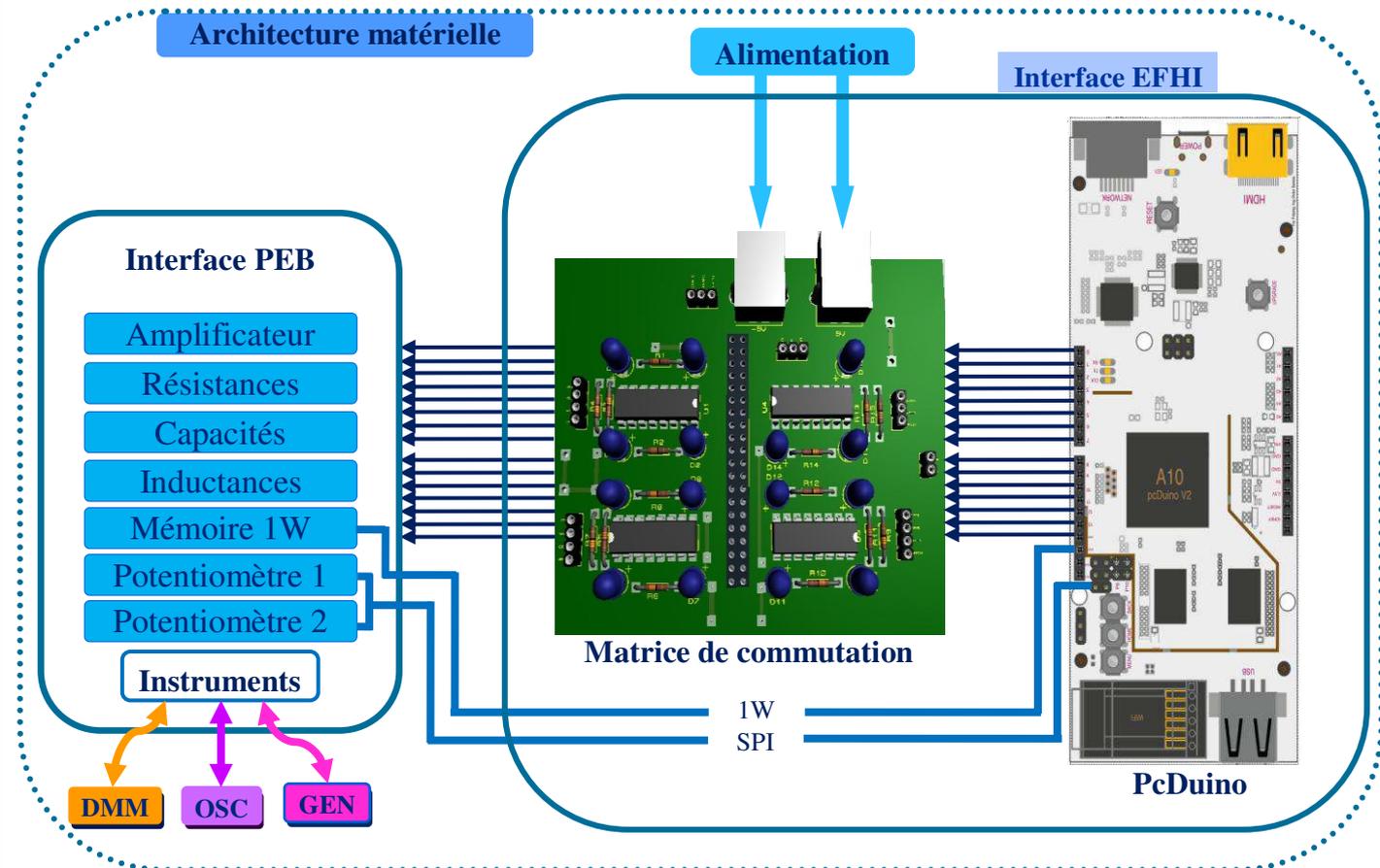


Figure I-1: Architecture matérielle du laboratoire distant.

2.1. Description de l'interface EFHI

L'interface EFHI est composée d'une matrice de commutation à base des switches numériques commandés par un contrôleur grâce à ses sorties numériques appelées GPIO ce qui permet de commander et de mettre en œuvre la configuration demandée par l'utilisateur distant en l'appliquant directement sur le PEB du TP concerné.

A chaque configuration on peut changer:

- Le montage à réaliser.
- La valeur des composants.
- Le signal à visualiser.

La carte EFHI est alimentée par 2 USB pour produire +5 Vcc et -5 Vcc

- ❖ Une sortie de +5 Vcc et GND pour alimenter le contrôleur.

- ❖ Une sortie de +5 Vcc et -5 Vcc avec une masse GND pour alimenter la PEB à travers la nappe.

2.1.1. Contrôleur

Le contrôleur utilisé doit répondre parfaitement aux exigences du projet en termes de coût et des spécifications matérielles et logicielles. Les principaux critères sur lequel nous nous sommes basés pour choisir le contrôleur approprié sont :

- Le nombre des GPIO disponible pour commander les switchs numériques.
- Les interfaces de communications (ETHERNET, UART, SPI, I2C...).
- la compatibilité avec les technologies du web et la possibilité d'accueillir un serveur web embarqué.
- Les ressources matérielles telles que le CPU, RAM et le stockage.

Il existe sur le marché une large gamme des cartes de type mini-ordinateur que nous pouvons utiliser dans notre projet, parmi lesquelles nous avons:

- pcDuino
- Red Pitaya
- Beagle Bone
- Raspberry Pi
- Intel Edison

Pour la mise en œuvre du laboratoire distant nous avons choisis la carte PcDuino V2 de LinkSprite, (les spécifications matérielles et logicielles de PcDuino sont mentionnés en annexe 1). Cette plateforme est très performante. Elle intègre les fonctionnalités d'un ordinateur équipée de la distribution gratuite de Linux "Lubuntu" et qui peut à la fois accueillir le serveur web indispensable dans l'architecture des laboratoires distants, et de piloter la matrice de commutation grâce à ses sorties numériques(GPIO). Ce mini-ordinateur permet également de contrôler les potentiomètres numériques via le protocole de communication SPI et l'EEPROM via le protocole OneWire. La figure I.2 donne son schéma de brochage.

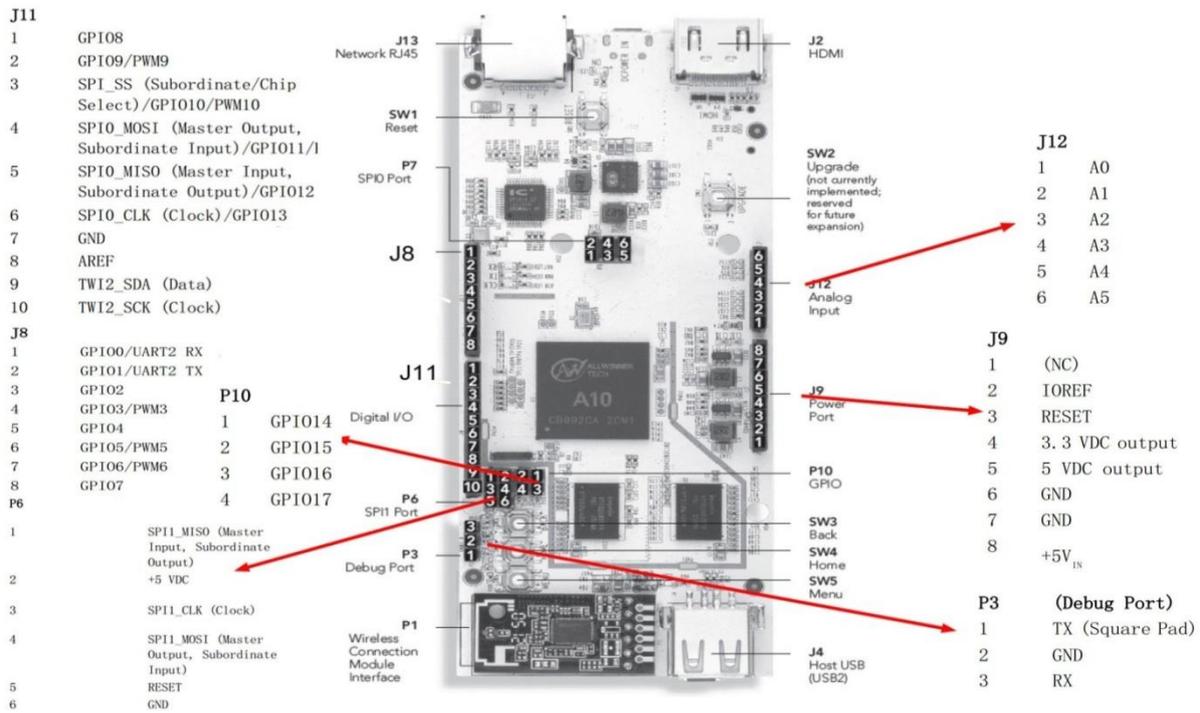


Figure I-2: Schéma et brochage du PcDuino V2. [2]

2.1.2. Description de la matrice de commutation

La matrice de commutation que nous avons réalisée tient compte de toutes les configurations de manipulation prévues, elle est constituée de 15 switchs numériques commandés par les sorties GPIO du PcDuino. Chaque TP a une configuration des états des switchs bien particulière qui est appliquée directement sur le PEB en ouvrant ou en fermant ces switchs.

Pour la conception de notre matrice de commutation, nous avons utilisé des switchs numériques de types MAX4678 à la place des relais mécaniques pour la commodité, la fiabilité et leur petite taille par rapport aux relais mécaniques.

Le MAX4678 est un switch analogique quadruple de type CMOS SPST en boîtier DIP 16 broches qui peut laisser passer les signaux numériques et analogiques. Il possède quatre switchs normalement ouverts (NO) chaque switch peut gérer des signaux analogiques et numériques rail à rail et le courant de fuite est de 0,1 nA à + 25 ° C. [3] La Figure I-3 représente l'interfaçage du PcDuino avec le switch MAX4678.

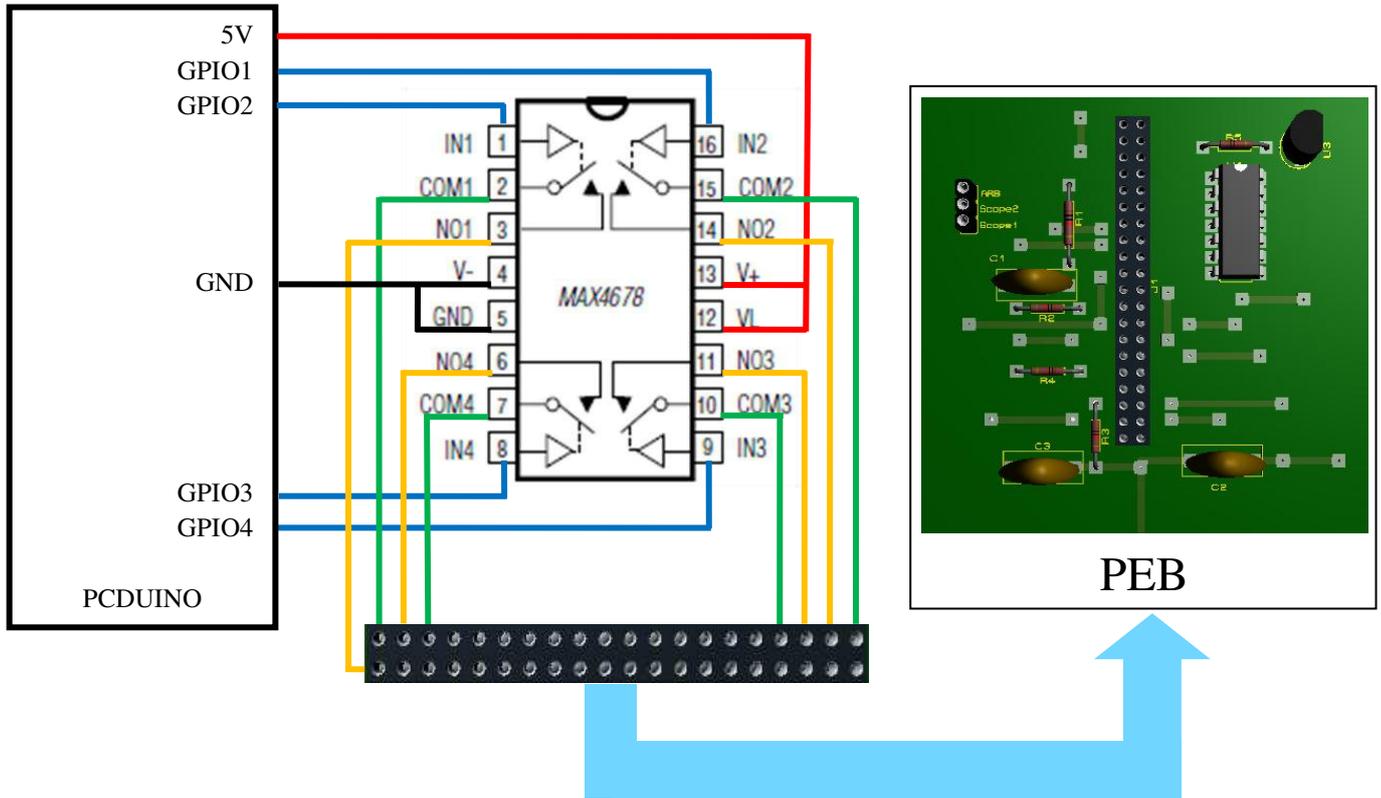


Figure I-3: Schéma de l'interfaçage du PcDuino avec le switch MAX4678.

Contrôleur-EFHI			
Pin	Fonction	Pin	Fonction
GPIO0	1W	GPIO9	Switch9
GPIO1	Switch1	GPIO10	Switch10
GPIO2	Switch2	GPIO11	Switch11
GPIO3	Switch3	GPIO12	Switch12
GPIO4	Switch4	GPIO13	Switch13
GPIO5	Switch5	GPIO14	Switch14
GPIO6	Switch6	GPIO15	Switch15
GPIO7	Switch7	GPIO16	CS1
GPIO8	Switch8	GPIO17	CS2

Tableau I-1: Brochage du contrôleur avec l'interface EFHI.

2.2. Description de l'interface PEB

L'interface PEB de notre projet représente les cartes des TP à effectuer, elle est reliée à la plateforme EFHI par une nappe standard de 40 lignes ce qui permet de changer la carte facilement.

Cette interface est un support pour les composants de TP (résistances, capacités, inductances, diode,...) qui sont reliés directement aux switches de la matrice de commutation. Cependant, les potentiomètres numériques et la mémoire OneWire sont reliés au contrôleur parce qu'ils sont contrôlés via un protocole de communication spécifique (SPI pour le potentiomètre numérique et 1-wire pour la mémoire). Le brochage des nappes est présenté dans le tableau suivant:

Nappe EFHI – PEB			
Pin	Fonction	Pin	Fonction
1	-5 V	21	GND
2	1W(GPIO0)	22	+5V
3	CS Pot1 (GPIO16)	23	MISO
4	CS Pot2 (GPIO17)	24	SCLK
5	S1 (GPIO1)	25	MOSI
6	D1	26	D15
7	S2 (GPIO2)	27	S15 (GPIO15)
8	D2	28	D14
9	S3 (GPIO3)	29	S14 (GPIO14)
10	D3	30	
11	S4 (GPIO4)	31	D13
12	D4	32	S13 (GPIO13)
13	S5 (GPIO5)	33	D12
14	D5	34	S12 (GPIO12)
15	S6 (GPIO6)	35	D11
16	D6	36	S11 (GPIO11)
17	S7 (GPIO7)	37	D10
18	D7	38	S10 (GPIO10)
19	S8 (GPIO8)	39	D9
20	D8	40	S9 (GPIO9)

Tableau I-2: Brochage de la nappe EFHI – PEB.

2.2.1. Potentiomètre numérique

Un potentiomètre numérique est un composant actif qui simule le comportement d'un potentiomètre analogique, mais à la différence de ce dernier, on ne fait pas varier sa résistance mécaniquement en le tournant (ou en le glissant). Il varie sa résistance par programmation en fonction d'une valeur numérique (souvent un octet) qu'il reçoit.

Le potentiomètre numérique que nous utiliserons dans notre projet est le MCP4231 de Microchip. Ce type de potentiomètre est un circuit intégré à 14 broches qui contient 2 potentiomètres sur le même boîtier qui sont séparés et indépendants les uns de l'autres. Son brochage est présenté en annexe 2.

La résistance du potentiomètre numérique est déterminée par une commande numérique envoyée à partir d'un contrôleur. En connectant ce circuit au PcDuino et grâce au programme que nous avons écrit, nous serons alors en mesure de modifier la sortie de la résistance aux bornes de wiper-lock du potentiomètre.

L'échelle de résistances est l'élément clé d'un potentiomètre numérique, Elle est sous forme d'une série de résistances de valeur égale connectées en série.

Dès la mise sous tension, certains potentiomètres numériques vont tout simplement affecter la position du wiper au milieu de l'échelle, ou à quelques autres points définis dans la fiche technique. Le circuit MCP4231 a une résolution de 7 bits et qui incorpore une RAM.

❖ Les circuits MCP sont désignés comme suit:

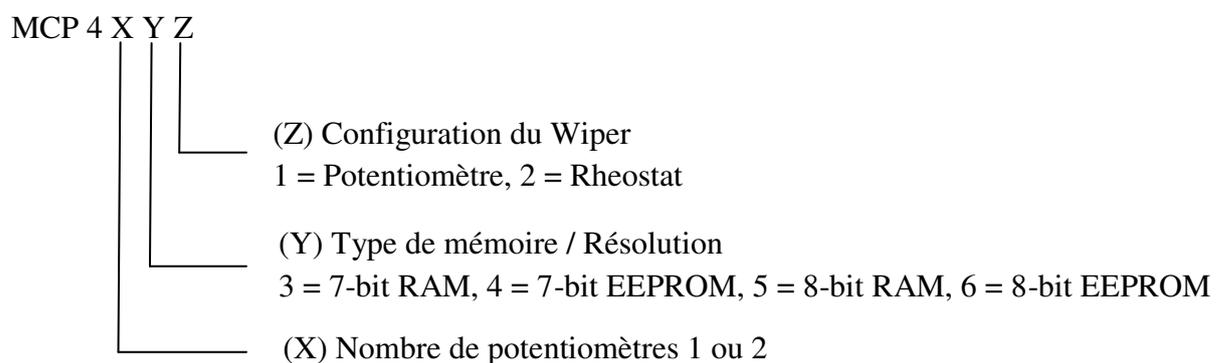


Figure I-4:Désignation des circuits MCP.

La valeur du potentiomètre est contrôlée par l'écriture dans son registre de commande à 7 bits via l'interface SPI, ce qui nous donne une résolution de 128 pas (0-127). Ces potentiomètres ne peuvent être réglés que par incréments de 1/127ème de leur résistance totale. [4]

La Figure I-5 représente l'interfaçage du PcDuino avec le MCP4231:

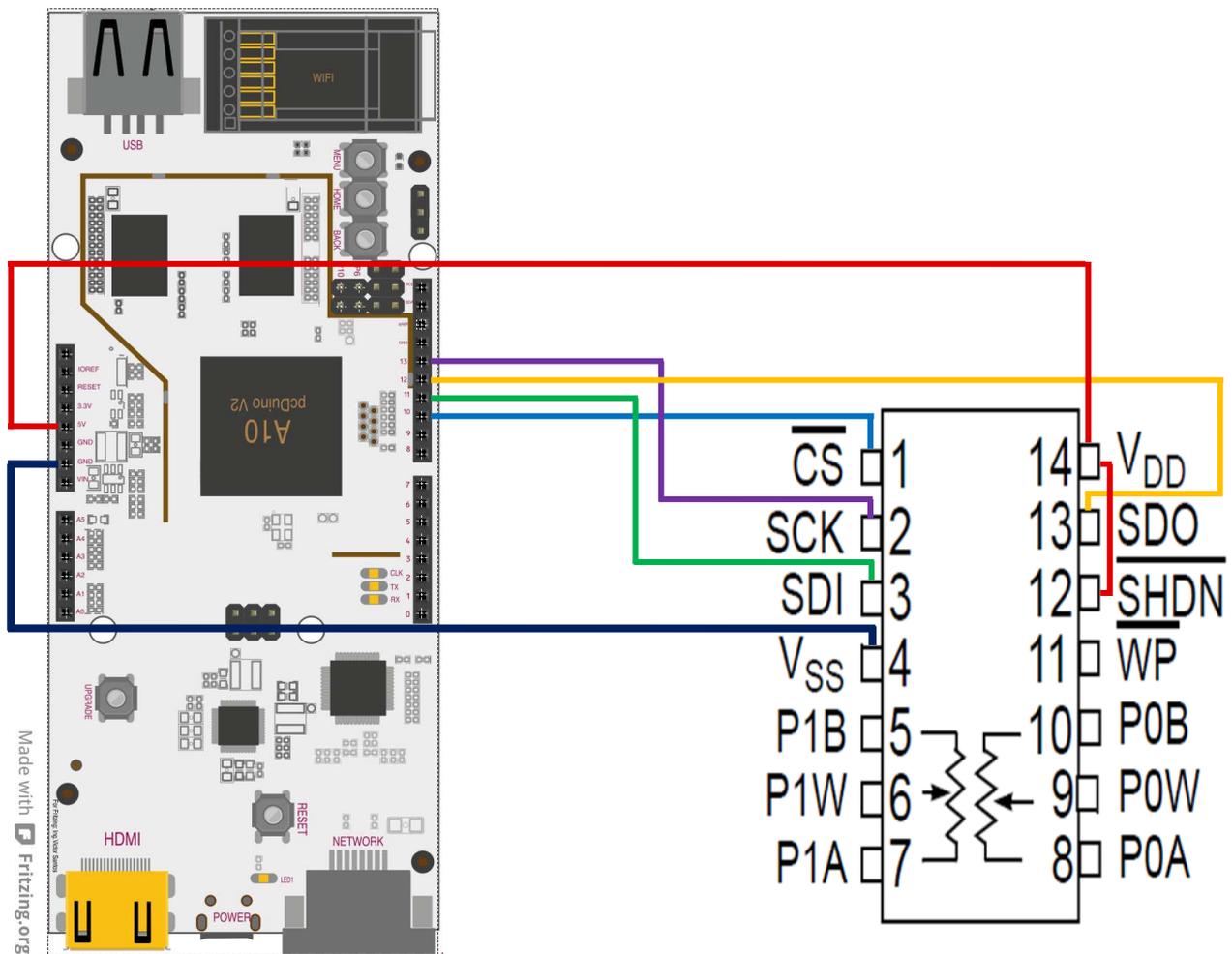


Figure I-5: Interfaçage du PcDuino avec le MCP4231.

2.2.2. Mémoire One Wire DS28E07

La mémoire DS28E07 utilisée dans notre projet est une EEPROM de 128 octets programmable par l'utilisateur, Elle est organisée en 4 pages de 32 octets chacune. Les données sont écrites sur mémoire scratchpad (bloc-notes) de 8 octets, vérifiées, puis copiées dans la mémoire EEPROM.

En tant que caractéristique spéciale, les quatre pages de mémoire utilisateur peuvent être individuellement protégées en écriture ou mises en mode d'émulation EPROM, où les bits ne peuvent être changés que d'un état 1 à un état 0. Chaque périphérique présente son propre numéro d'identification de ROM

(ROM ID) unique de 64 bits, qui est programmé en usine dans la puce. La mémoire DS28E07 communique avec le PcDuino via le protocole 1-Wire avec l'ID ROM agissant comme adresse de nœud dans le cas de réseau à plusieurs appareils. [5]

La Figure I-6 représente l'interfaçage du PcDuino avec la mémoire 1-wire:

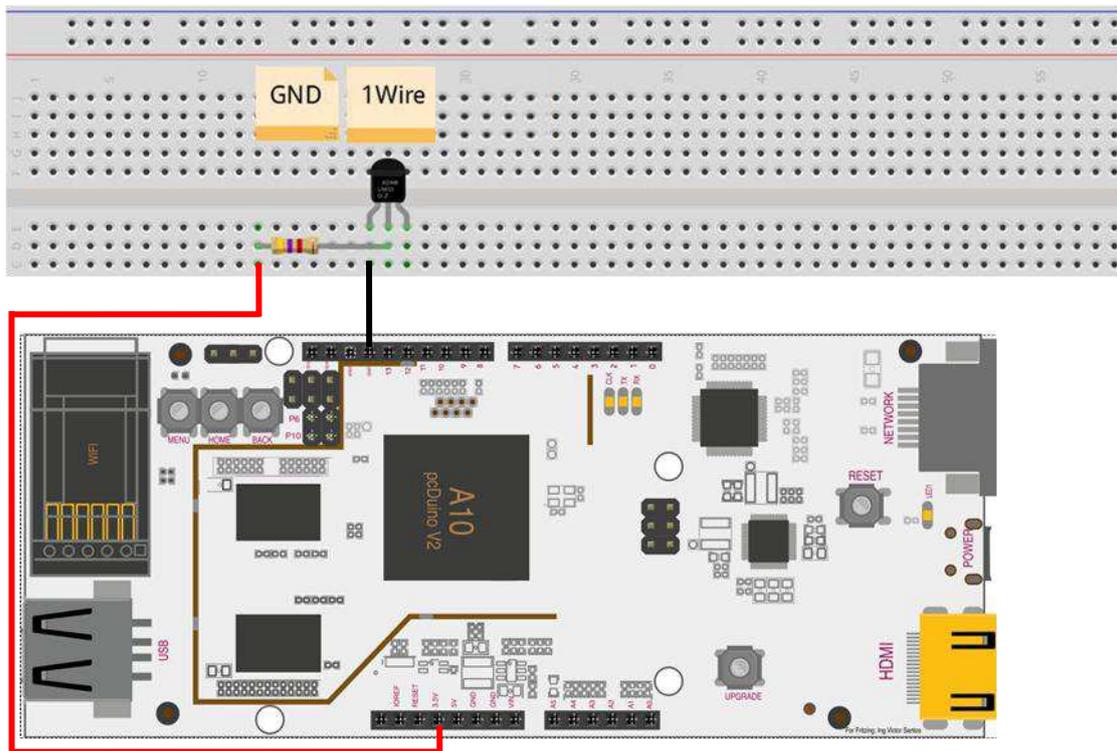


Figure I-6: Interfaçage du PcDuino avec l'EEPROM DS28E07.

3. Conclusion

Ce chapitre a été consacré à la description de l'architecture matérielle de notre laboratoire distant dédié aux travaux pratiques d'électronique.

Cette architecture présente plusieurs avantages:

- Flexibilité matérielle : le passage d'une expérience à une autre se fait par simple échange de carte de TP.
- L'utilisation de la matrice de commutation permet d'effectuer un nombre assez important de manipulations.
- Faible coût: grâce à l'utilisation de serveur sur ordinateur mono-carte de type PcDuino.



Chapitre 2: Architecture logicielle

1. Introduction

Le développement des technologies web et l'émergence des plateformes logicielles flexibles et open source comme node.js nous a incité à introduire ces dernières dans l'architecture du laboratoire distant pour répondre aux besoins de flexibilité tout en assurant le faible coût de développement et l'accessibilité à distance adaptée aux connexions à faible débit.

En effet, la plus part des laboratoires actuels utilisent un logiciel propriétaire comme Labview et un matériel coûteux (un serveur) pour les mettre en œuvre. Nous utilisons pour notre part uniquement des logiciels open source.

Le principal développement pour la partie logicielle concerne la mise en place du serveur web et le script de l'interface web pour la manipulation des TP.

Dans ce chapitre nous allons décrire l'architecture logicielle ainsi que les technologies web utilisées pour la mise à distance des travaux pratiques via Internet.

2. Description de l'architecture logicielle du laboratoire distant

L'architecture logicielle de notre projet est de type client-serveur. Le serveur assure en temps réel la réception des requêtes en provenance du client. Il les traite en commandant la carte de TP et les instruments de mesure et de visualisation du signal, puis renvoie les résultats au client. Le client c'est-à-dire l'apprenant effectue l'expérimentation via l'interface. Ce qui lui permet de sélectionner la manipulation à réaliser, les composants nécessaires et leurs valeurs, et visualiser les résultats sur la même page. La figure II-1 illustre les technologies web utilisées pour développer l'architecture logicielle.

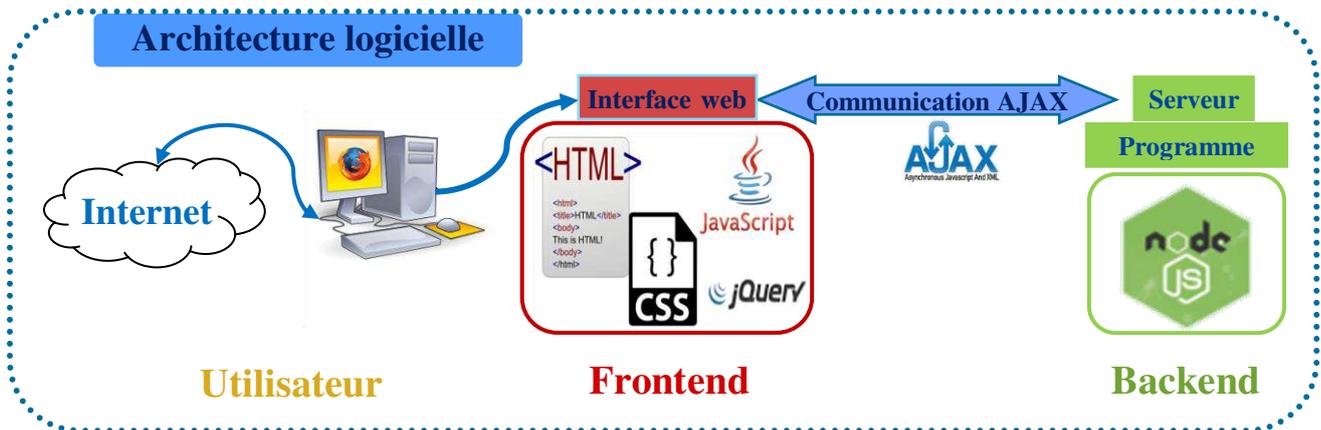


Figure II-1: L'architecture logicielle du laboratoire distant.

La communication entre les clients et le serveur est assurée par des appels AJAX sous formes des requêtes http (Voir figure II-2).

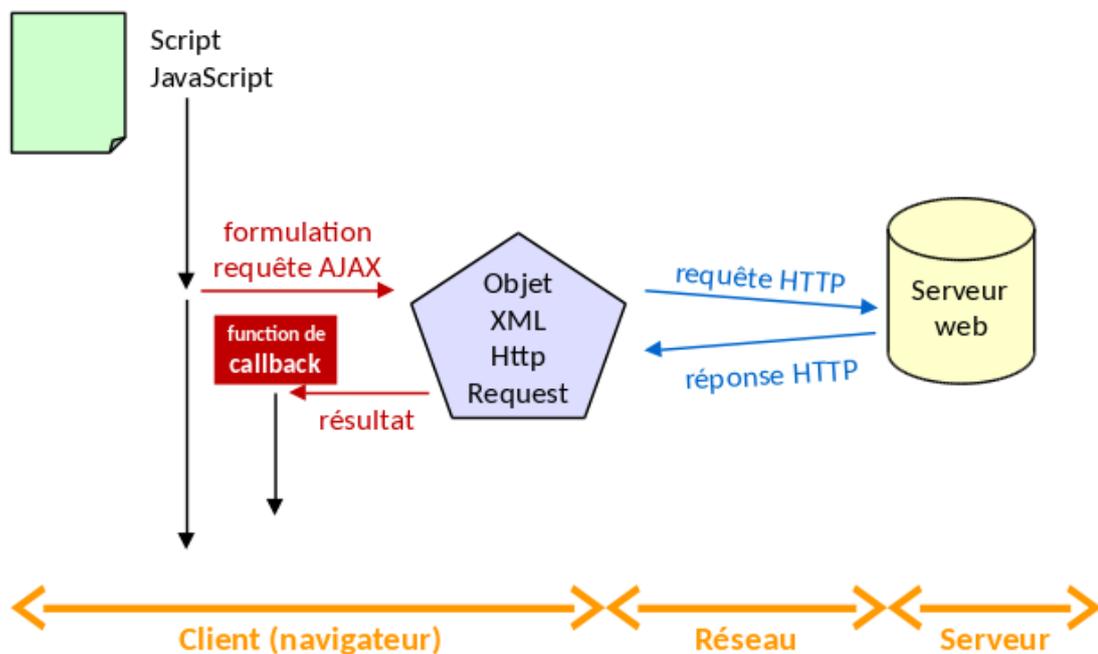


Figure II-2: Principe de fonctionnement d'une architecture client/serveur [6]

2.1.Coté client

La partie client (front-end) représente l'interface web qui permet aux utilisateurs distants d'accéder à la plateforme des TP. Elle est réalisée principalement en HTML, CSS et JavaScript.

Le HTML décrit le contenu. Le fichier CSS appelé feuille de style permet la mise en forme de l'affichage. Le JavaScript quant à lui permet d'ajouter de l'interaction côté client.

2.1.1. HTML

L'**Hypertext Markup Language**, généralement abrégé HTML, est le format de données conçu pour représenter les pages web. C'est un langage de balisage qui permet d'écrire de l'hypertexte, d'où son nom. HTML permet également de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des liens, et des éléments programmables tels que des applets. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web.

La figure II-3 représente la structure type d'un fichier html. Il est toujours composé d'une racine **html** suivie de deux fils **head** et **body**. Le head contient les méta-données du document tandis que le body contient le corps du fichier.

Le document contient en outre une ligne DOCTYPE indiquant la version du langage HTML utilisé (et donc la syntaxe à attendre). On peut par ailleurs ajouter des commentaires entre les balises spéciales `<!--commentaires-->`.

Le contenu est décrit par des balises qui peuvent contenir des attributs, qui vont se trouver entre les symboles "<" et ">" `<balise attribut1="texte1" attribut2="texte2" attribut3="texte3">`.

Un attribut est toujours une chaîne de caractères. Chaque balise HTML possède un certain nombre d'attributs valide possible. [7]

```
1  <!DOCTYPE html>
2  <html>
3    <head><!-- ceci est un commentaire -->
4      <metacharset="UTF-8">
5      <title> Titre du document </title>
6    </head>
7    <body>
8      Contenu du document
9    </body>
10 </html>
```

Figure II-3: Structure générale d'un fichier html.

2.1.2. CSS

CSS (Cascading Style Sheets: feuilles de style en cascade): est un langage informatique qui sert à décrire la présentation des documents HTML et XML. Il permet de définir la taille, la couleur ou l'alignement d'un texte, ainsi que la disposition visuelle et auditif, pour une variété de dispositifs. [8]

On peut écrire du code en langage CSS à trois endroits différents :

- dans un fichier.css en le liant au fichier HTML par la balise suivante:

```
<linkrel="stylesheet" type="text/css" href="style.css">
```

- dans l'en-tête <head> du fichier HTML.
- directement dans les balises du fichier HTML via un attribut style.

Pour chaque déclaration, la structure est toujours la même :

```
Sélecteur {  
  Propriété: valeur;  
}
```

Le sélecteur, c'est la balise (X) du fichier HTML (body ; h1 ; p, etc.), l'identifiant (id) ou la classe (class).

La propriété, c'est l'attribut qu'on veut appliquer (font ; background ; margin ; etc.).

La valeur qui précise les caractéristiques de la propriété.

Schématiquement, une feuille de style CSS ressemble à cela :

```
balise1  
{  
  propriete1: valeur1;  
}  
balise2  
{  
  propriete1: valeur1;  
  propriete2: valeur2;  
}  
balise3  
{  
  propriete1: valeur1;  
  propriete2: valeur2;  
}
```

2.1.3. JavaScript

JavaScript a été inventé en 1995 par Brendan Eich (membre du conseil d'administration de Mozilla). C'est un langage de programmation initialement introduit dans les navigateurs Web afin de rendre les pages HTML plus dynamiques dans leurs interactions avec l'utilisateur. Des optimisations en performances lui ont permis de se hisser comme un langage de programmation efficace aussi bien au niveau du client qu'au niveau du serveur. [9]

Il s'exécute coté client (navigateur) et s'insère dans le code HTML soit directement, soit par inclusion d'un script externe.

Exemple exécution d'une fonction en réaction à un clic sur un bouton.

L'ensemble des fonctions peuvent être regroupées dans un fichier externe et lié au fichier html par la balise :

```
<script type="text/javascript" src="js/script.js" async="async"></script>
```

Dans ce projet nous avons fait appel à la bibliothèque jQuery. C'est une bibliothèque JavaScript open-source créée en 2006 par John Resing qui fonctionne de façon unifiée sur tous les navigateurs. Cette bibliothèque facilite grandement le développement d'application web (ou de page web). C'est un simple fichier JavaScript que l'on inclut dans nos documents HTML comme n'importe quel fichier JavaScript. [10]

jQuery nous permet de :

- modifier très facilement l'aspect de nos pages (jouer sur les CSS).
- modifier le contenu de notre page (jouer sur le HTML).
- gérer les événements.

- créer des animations sur notre page.
- communiquer facilement avec un serveur.

La communication entre le client et le serveur à lieu par des requêtes AJAX.

AJAX veut dire "Asynchronous JavaScript and XML". Le code JavaScript d'une page peut effectuer des requêtes HTTP, sans changer de page. On travaille alors en mode asynchrone, le code JavaScript tourne en tâche de fond, et les requêtes HTTP faite en JavaScript ne sont pas bloquantes (on leur associe une fonction de callback). Cette technique fournit les avantages suivants :

- La page n'échange que les données qui ont besoin de changer, on a donc une bande passante et temps de transfert réduits.
- La page peut rester opérationnelle pendant l'attente d'une réponse à une requête.

L'objet essentiel qui permet de faire des requêtes HTTP en JavaScript est le **XMLHttpRequest**. On doit faire un certain nombre de choix:

- mode synchrone ou asynchrone.
- type de requête: POST, GET, PUT...
- format d'échange: XML ou JSON. [11]

2.2.Coté serveur

La partie serveur ou back-end de notre projet est fondée sur une plateforme de développement en JavaScript appelée node.js permettant de mettre en œuvre des applications réseaux asynchrones et une communication en temps réel. Ces applications mono-threadées peuvent prendre en charge une quantité très élevée de requêtes simultanément.

Le serveur reçoit les requêtes http envoyées par l'utilisateur distant, les traite et l'exécute en ouvrant ou en fermant les switchs numériques de la matrice de commutation.

2.2.1. Node.js

Node.js est une plateforme de développement open source orientée serveur qui a été créée par Ryan Dahl en 2009. Cette plateforme logicielle utilise la machine virtuelle V8 de chrome. C'est une technologie qui permet d'exécuter du JavaScript côté serveur.

Cette plateforme contient un ensemble de bibliothèques standard offrant des fonctionnalités qui étaient auparavant impossibles avec le JavaScript. Parmi lesquelles, nous avons la bibliothèque http qui nous intéresse pour la suite de notre projet.

Node.js est utilisé pour le développement des applications web coté serveur en utilisant du JavaScript. Il se distingue des autres plateformes grâce à une approche non bloquante permettant d'effectuer des entrées/sorties (I/O) de manière asynchrone. [12]

Le package Node.js inclut également l'outil **NPM** (Node Package Manager) qui permet le téléchargement et l'accès à de très nombreuses librairies.

Les grands principes qui sous-tendent Node.js sont:

- Exécution pilotée par les événements.
- Appels de fonctions asynchrones (utilisation de callback).
- Entrées/sorties non bloquantes.

Node.js est maintenant couramment utilisé dans le monde de l'IT (technologie de l'information) pour des middlewares, des web services temps-réel ou en full-stack (coté serveur et coté client) pour des applications web. On le voit aussi beaucoup dans le monde des objets connectés IOT. Il arrive également sur le PcDuino et Raspberry PI pour le pilotage à distance. [13]

2.3. Protocole SPI

Le potentiomètre numérique MCP4231 communique avec le PcDuino via le bus Serial Peripheral Interface, ou bus SPI.

2.3.1. Généralité sur le protocole de communication SPI

Une liaison SPI est un bus de données série synchrone baptisé par Motorola, qui opère en mode Full-duplex. Les circuits communiquent selon le mode maître-esclaves, où le maître s'occupe totalement de la communication. Plusieurs esclaves peuvent coexister sur un même bus, dans ce cas, la sélection du destinataire se fait par une ligne dédiée entre le maître et l'esclave appelée chip select.

Le bus SPI utilise 4 signaux logiques :

- SCLK: Serial Clock, Horloge (généralisé par le maître).
- MOSI: Master Output, Slave Input (généralisé par le maître).
- MISO: Master Input, Slave Output (généralisé par l'esclave).
- SS: Slave Select, Actif à l'état bas (généralisé par le maître).

2.3.2. Principe de fonctionnement d'une liaison SPI

Les périphériques SPI sont synchrones, ce qui signifie que les données sont transmises en synchronisation avec un signal d'horloge partagé (SCLK). Les

données peuvent être déplacées dans l'appareil esclave sur le front montant ou descendant du signal d'horloge.

Le PcDuino a un support SPI, contenant une bibliothèque intégrée et du matériel pour communiquer avec un potentiomètre contrôlable numériquement.

La figure II-4 correspond au schéma de principe simplifié d'une liaison de données SPI, avec ses principaux composants.

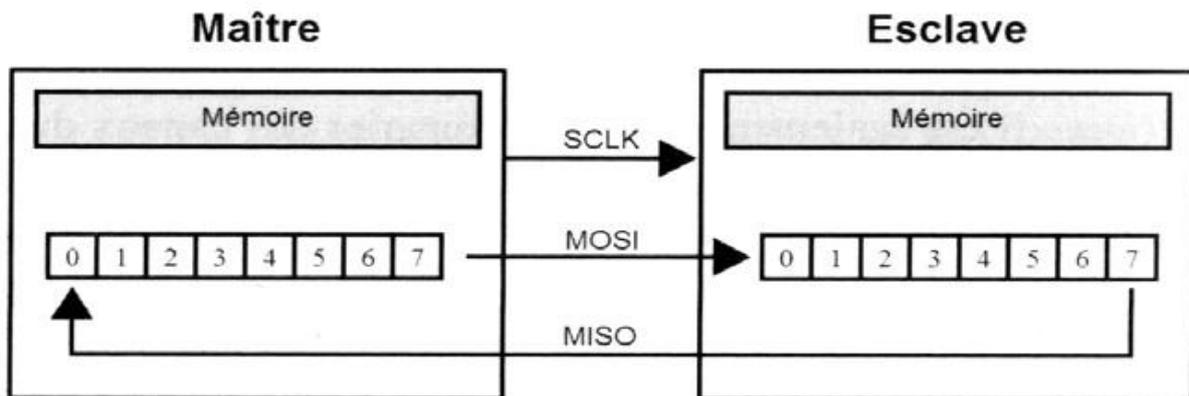


Figure II-4: Schéma de principe simplifié d'une liaison SPI. [14]

Deux registres à décalages sont généralement mis en œuvre. Ils peuvent être de type matériel ou logiciel, selon les dispositifs employés. Par exemple, le PcDuino implémente son registre à décalage de façon logiciel alors qu'il y a d'autres circuits qui sont équipés d'un registre à décalage matériel.

Quelque soit leur mise en œuvre, ces deux registres à décalage forment un buffer circulaire inter-puce, qui se trouve au cœur de la liaison SPI.

Les communications sont initiées par le maître, qui commence par sélectionner l'esclave. Au cours de chaque cycle d'horloge, le maître envoie un bit à l'esclave, qui le lit sur la ligne MOSI. En parallèle, l'esclave envoie un bit au maître, qui le lit sur la ligne MISO. Ces lectures et écritures simultanées entre le maître et l'esclave forment une communication Full-Duplex.

La fréquence de l'horloge dépend principalement des possibilités de réponse de l'esclave. [15]

2.4. Protocole 1-Wire

La mémoire EEPROM DS28E07 utilisée pour identifier les cartes de TP fonctionne avec un protocole de communication spécifique appelé 1-Wire (aussi connu sous le nom de bus Dallas ou OneWire).

2.4.1. Généralité sur le Bus 1-Wire

1-Wire est un bus conçu par Dallas Semiconductor qui permet de connecter et de faire dialoguer entre eux (en série, parallèle ou en étoile) des composants (des circuits) avec seulement un fil de données. Le fil unique du bus doit être tiré au +Vcc par une résistance de 4,7K Ω . Le second fil est le fil de masse.

Le protocole série 1-Wire est asynchrone et de nature similaire à I2C, il fonctionne suivant le principe maître / esclave. Il y a donc un seul maître, qui pourra dialoguer avec un ou plusieurs esclaves. Il est half-duplex et il a une vitesse maximale de 16 Kbps. La distance maximale entre les deux appareils est de 300 m. Ce protocole ne peut être utilisé entre deux appareils à la fois. [16]

2.4.2. Caractéristiques principales

Le principe global de ce bus repose sur une communication maître-esclaves. Le maître étant le contrôleur (PcDuino) 1wire et les esclaves sont l'ensemble des modules connectés à ce maître quelque soit la topologie de ce réseau.

Le maître va discuter avec un ou plusieurs esclaves sur un seul fil, en mode half-duplex, c'est-à-dire qu'un seul esclave parle à la fois sur la ligne, ce qui permet de discuter dans les deux directions.

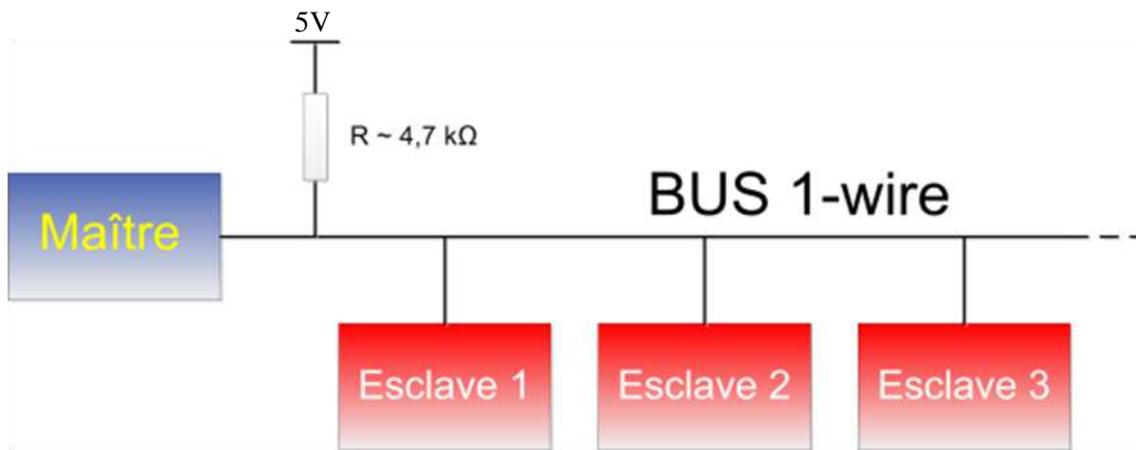


Figure II-5: Schéma d'un bus de communication 1-Wire. [17]

2.4.3. Principe de fonctionnement du protocole 1-Wire

La communication sur le bus 1-Wire est caractérisée par un ensemble de «pulse» ou changement d'état. L'état par défaut de la ligne data est +5V, ce qui permet d'alimenter les différents composants à partir de la ligne de données.

Avant toute communication, le maître met le bus à l'état bas (0V) pendant $480\mu\text{s}$ pour faire une réinitialisation (reset) des composants connectés. Après un délai de 15 à $60\mu\text{s}$, le ou les esclaves connectés, forcent le bus à l'état bas pendant 60 à $240\mu\text{s}$ pour signaler leur présence. Le maître est alors informé des esclaves connectés sur le bus.

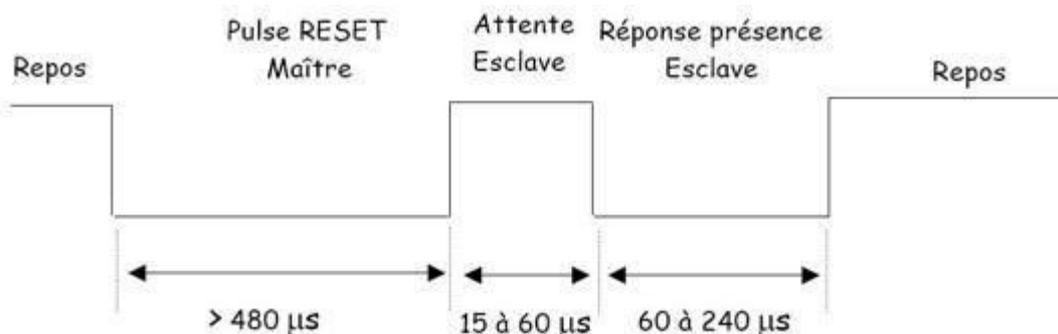


Figure II-6: Procédure d'initialisation: réinitialisation et impulsion de présence.

Chaque circuit possède une adresse physique unique, gravée dans la puce à la fabrication. Cette adresse est constituée de 64 bits soit 8 octets. Le premier octet détermine le type de famille auquel appartient le circuit. Les 6 octets suivants,

constituent le code propre du circuit. Le dernier octet est le CRC. C'est un octet de contrôle calculé à partir des 56 bits précédents.



Figure II-7: L'adresse physique d'un circuit 1-Wire.

Toute communication entre un maître et un ou plusieurs esclaves, débute par une initialisation, constituée par l'envoi du pulse de Reset par le maître. Le maître doit ensuite envoyer une commande de type ROM qui est propre au protocole 1-Wire, et que tous les circuits de ce type vont reconnaître. Cela va permettre entre autre de sélectionner un circuit parmi les différents esclaves qui ont répondu présents au pulse de Reset. Le dialogue et l'échange de données pourra ensuite commencer, entre le maître et l'esclave sélectionné.

Pour émettre un bit sur le bus, le maître force le bus à «0» pendant 1 à 15 μ s. L'esclave va lire le bus entre 15 et 45 μ s après le front descendant (valeur typique 30 μ s). Si on veut émettre un «1», il faut repasser le bus à «1» immédiatement, et ne plus rien faire jusqu'à $t = 60\mu$ s. Pour émettre un «0» il faut laisser le bus à «0» jusqu'à $t = 60\mu$ s, puis repasser le bus à «1».

La durée totale d'un bit est donc de 60 μ s, ce qui donne une vitesse de communication maximale de 16kbits/s.



Figure II-8: La lecture d'un bit dans une communication 1-Wire.

Après la réception d'une commande du maître, l'esclave peut renvoyer des données. Pour lire les données de l'esclave, le maître force le bus à « 0 » pendant au moins $1\mu\text{s}$. Si l'esclave veut émettre un «1», il laisse le bus libre, donc tiré à «1». Pour émettre un «0», l'esclave doit tirer le bus à «0» pendant $15\mu\text{s}$ au minimum. Le maître devra donc dans tous les cas lire le bus $15\mu\text{s}$ maximum après avoir tiré le bus à «0» pendant $1\mu\text{s}$. L'état du bus donnera alors le bit transmis par l'esclave. [18]

3. Conclusion

La partie logicielle représente la partie la plus importante qui constitue notre travail, ce chapitre a été consacré à la description de la conception de l'architecture logicielle de notre laboratoire distant dédié aux travaux pratiques d'électronique.

Cette architecture présente plusieurs avantages :

- ❖ Faible coût de développement grâce à l'utilisation de serveur sur la carte de type PcDuino et de logiciels open source.
- ❖ Implémentation facile.



Chapitre 3: Conception et réalisation

1. Introduction

La dernière étape de notre travail consiste à réaliser les circuits des différentes cartes (la matrice de commutation et PEB). Pour ce faire nous devons tout d'abord passer par la conception du typon puis nous passons à la réalisation.

D'autre part, la réalisation matérielle est suivie par l'implémentation des algorithmes pour pouvoir commander la matrice de commutation et effectuer le TP à distance.

Dans ce chapitre, nous présentons l'implémentation logicielle du laboratoire distant ainsi que la réalisation matérielle des différents circuits.

2. Implémentation logicielle

Après avoir conçu l'architecture logicielle de notre laboratoire, nous passons maintenant à l'étape de l'implémentation pour tester le bon fonctionnement du projet et les différents scripts que nous avons écrit.

2.1. Structure de l'interface web

L'interface web de notre projet donne l'accès aux utilisateurs distants pour réaliser leurs TP. Elle est développée en HTML/CSS/JavaScript avec communication AJAX. Pour chaque manipulation le navigateur envoie une requête http au serveur qui l'exécute et transmet le résultat sur la même page web de l'utilisateur. (Code source de l'interface web est donné en annexe 3).

Après avoir accédé à l'interface web des TP, l'étudiant sélectionne la manipulation qu'il veut réaliser, il configure les paramètres du TP, les valeurs des composants (résistance, capacité...). En arrière-plan, des requêtes http sont envoyées au serveur node.js installé sur le PcDuino en indiquant les paramètres du TP concerné (états des switches et la valeur du potentiomètre). Ces paramètres sont appliqués directement par le PcDuino qui commande la carte PEB à travers les switches de la matrice de commutation connectés à ses sorties GPIO et le protocole SPI pour contrôler les potentiomètres.

L'interface web développée permet ainsi aux étudiants de visualiser le résultat de la manipulation sur un oscilloscope virtuel, prendre des mesures de grandeurs électriques et répondre à des questions concernant le TP.

L'interface en dessous (figure III-1) est doté en plus des outils supplémentaire destiné aux développeurs tel que l'accès direct aux états des switches, control des potentiomètres et d'écrire ou lire à partir de l'eprom 1W.

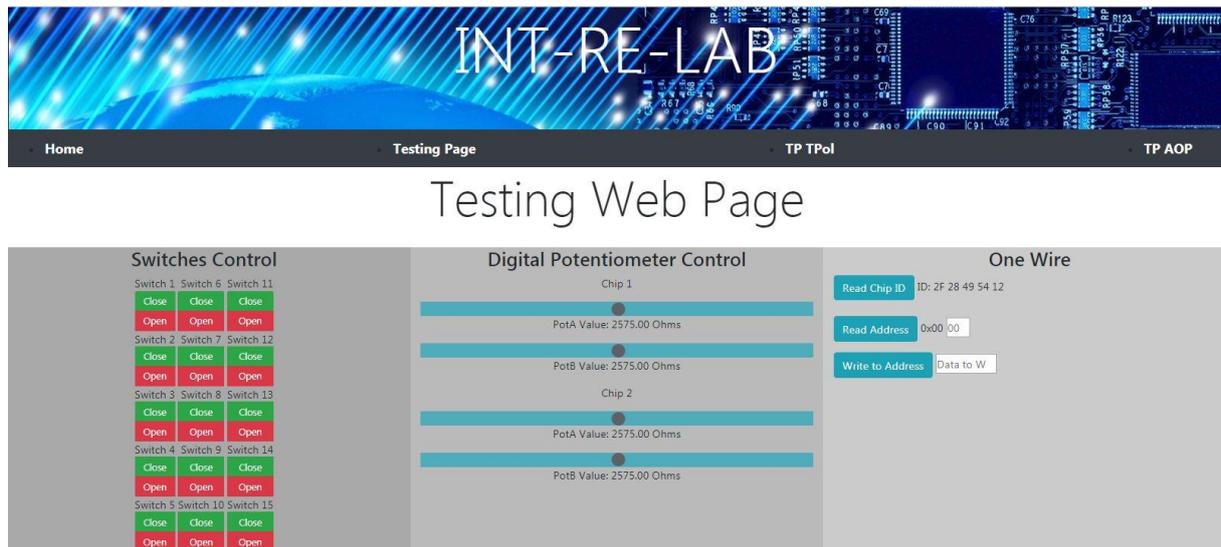


Figure III-1: L'interface de développement pour la manipulation des switches, les potentiomètres et 1-Wire.

2.2. Mise en place du serveur web

Le pilotage à distance de l'interface PEB à travers l'interface web est géré par un programme serveur écrit en JavaScript et basé sur node.js.

Le serveur web interprète les requêtes envoyées par l'utilisateur distant et les traduit en commandes. Ces dernières sont transformées en instructions sur les GPIO du PcDuino en ouvrant ou en fermant les switches numériques, ou en commandant le potentiomètre numérique via le protocole SPI. Le serveur nous fournit également la possibilité de lire ou écrire sur la mémoire DS28E07 via le protocole 1-Wire pour identifier les cartes de TP. [19]

2.2.1. Installation de node.js

Node.js peut être installé de trois façons :

- Utiliser un gestionnaire de paquets (brew, apt-get, ...).
- Compiler le code source.
- Télécharger les binaires en se rendant sur le site officiel <https://nodejs.org/>

2.2.2. Le gestionnaire de paquets npm

Node.js dispose d'une communauté assez importante qui partage ses modules au travers du gestionnaire de paquet npm. Il permet de télécharger rapidement un module ainsi que ses différentes dépendances.

Il y a des milliers de modules déjà développés, il suffit de se rendre sur le site <https://www.npmjs.com> et rechercher un paquet.

L'installation d'un module se fait par l'instruction: `npm install<nom du paquet>`

Node.js propose une infinité de bibliothèques, parmi lesquelles que nous avons utilisé dans notre projet:

- **fs** : manipuler le système de fichiers.
- **http**: créer et interroger des ressources via le protocole HTTP.
- **path** : gestion des chemins sur un système de fichier.
- **querystring** : échapper, analyser les arguments d'une requête.
- **child_process** : appeler un exécutable système. [20]

2.2.3. Etapes de création du script serveur

- Création d'un fichier serveur.js en utilisant un éditeur de texte comme sublime text ou notepad ++, code source du serveur est montré en annexe 4.
- Inclure le module http avec la fonction require.
- `var http = require("http");`
- Création du serveur HTTP avec la méthode `createServer`, en lui fournissant l'URL de la page HTML qui permet de:
 - commander les switchs numériques connectés aux GPIO du PcDuino.

- modifier la valeur du potentiomètre via le protocole SPI.
 - lire ou écrire dans la mémoire DS28E07 via le protocole 1-Wire.
- Dès la réception d'une requête, le serveur la traite puis il renvoie une réponse.
 - Mettre le serveur en écoute sur le port 8080.
 - Indiquer dans la console que le serveur était lancé.
 - Exécution: node serveur.js
 - Ouvrir le navigateur, visiter <http://127.0.0.1:8080/>

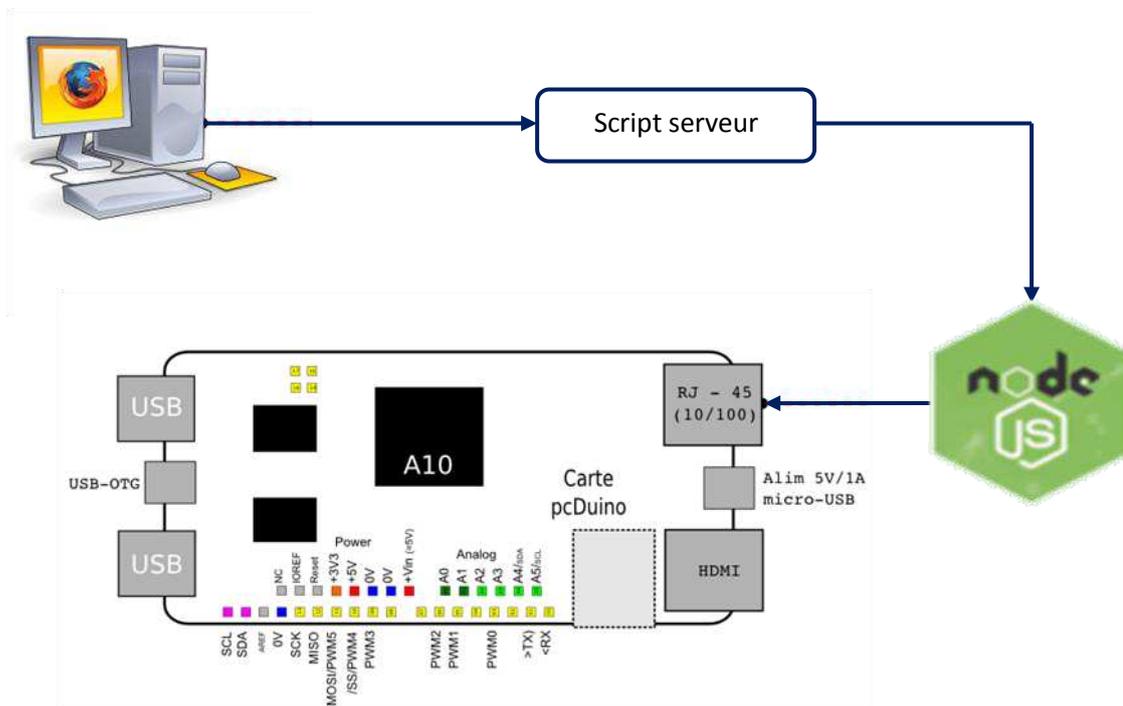


Figure III-2: Exécution du serveur web.

2.2.4. Pilotage des switches numériques

Pour commander les switches numériques de la matrice de commutations qui sont connectés aux sorties numériques du PcDuino, nous avons développé un script crée en JavaScript et exécuté par node.js qui permet de changer l'état des GPIO de ce mini-ordinateur pour configurer le circuit PEB.

Le serveur interprète les requêtes, il extrait les switches concernés (id) et leurs états (valeur) puis il applique le résultat sur les GPIO du PcDuino en utilisant l'instruction suivante:

❖ `exec('echo '+valeur +' >/sys/devices/virtual/misc/gpio/pin/gpio'+ id);`

2.2.5. Programmation du potentiomètre numérique MCP4231

Le MCP4231 est un potentiomètre numérique programmable contrôlé via l'interface SPI.

❖ Pour commander ce type de potentiomètre numérique par le PcDuino, nous avons utilisé le langage de programmation Python qui dispose une bibliothèque prête à utiliser appelée SPIDEV.

- Pour contrôler le MCP4231, nous devons tout d'abord installer l'environnement de développement intégré pour le langage Python et la bibliothèque SpiDev.
- Créer un script puis importer les modules nécessaires (annexe 5):
 - ✚ SpiDev: permet d'utiliser l'interface SPI.
 - ✚ Time: gérer le temps.
 - ✚ Sys: permet un accès direct aux arguments de la ligne de commande.
- Après avoir importé les modules nécessaires, nous créons un objet SPI puis nous l'activons et nous configurons les autres paramètres comme CS et nombre de bits et la vitesse à chaque transfert de données.
- Sélectionner le mode transmission SPI: le potentiomètre que nous avons utilisé fonctionne en mode SPI 00 ou 11.
 - ❖ `spi.mode = 0b00`
 - ❖ `spi.mode = 0b11`
- Pour modifier les valeurs de la résistance nous utilisons la fonction `xfer2([@pot,valeur])`.

Les principales fonctions utilisées de la bibliothèque SpiDev sont présentées par le tableau ci-dessous:

<code>spi = spidev.SpiDev()</code>	Création d'un objet spi
<code>spi.open(bus, device)</code>	Activer SPI (0 ou 1) Device est l'état du CS (0 ou 1)
<code>spi.max_speed_hz = 100000</code>	Vitesse de transfert en Hz
<code>spi.cshigh = False</code>	Indiquer si CS est actif à l'état haut (généralement CS est actif à l'état bas)
<code>spi.bits_per_word = 8</code>	Nombre de bits par transfert (généralement 8)
<code>spi.mode = 0b00</code>	Mode 0 de transmission SPI
<code>xfer2([@pot, valeur])</code>	Echange des données entre le PcDuino et le potentiomètre en fournissant son adresse et la valeur de résistance
<code>spi.close()</code>	Désactiver l'interface spi

Tableau III-1: Les fonctions de la bibliothèque SpiDev.

Le contrôle du MCP4231 qui contient deux potentiomètres dans le même boîtier se fait par la commande suivante:

 Python script_pot.py valeur1 valeur 2

2.2.6. Programmation de l'EEPROM DS28E07:

La mémoire 1-Wire de type DS28E07 utilisée dans notre projet pour stocker l'identificateur des cartes de TP communique avec le PcDuino via le protocole 1-Wire.

Pour commander ce type des mémoires soit en lecture ou en écriture, nous avons utilisé une bibliothèque spécifique appelée OneWire.h écrite en langage C.

Pour lire des données depuis la mémoire, on doit :

- Spécifier l'adresse et le format des données à lire.
- Créer un script puis importer les modules nécessaires :

- Spécifier le GPIO auquel est connecté l'EEPROM.
- Vérifier en permanence la présence d'un circuit 1-wire.
- Fournir deux arguments représentant l'adresse et le format des données à lire (8 bits par exemple).
- Utiliser la commande `ds.write(0xF0);` pour lire depuis la mémoire.
- Spécifier l'adresse de l'octet poids faible et l'octet poids fort à lire.

Pour écrire des données dans la mémoire, on doit :

- Spécifier l'adresse et le format des données à lire.
- Créer un script puis importer les modules nécessaires :
- Spécifier le GPIO auquel est connecté l'EEPROM.
- Vérifier en permanence la présence d'un circuit 1-wire.
- Fournir deux arguments représentant l'adresse et la donnée à écrire dans la mémoire.
- Pour écrire une donnée dans la mémoire nous devons passer par la mémoire scratchpad en utilisant la commande `ds.write(0x0F);`
- Ecrire l'adresse puis les données dans la mémoire scratchpad.
- Lire à nouveau la mémoire scratchpad par la commande `ds.write(0xAA);` qui doit contenir l'adresse, le registre E/S et les données.
- Copier les données à partir de la mémoire scratchpad vers l'adresse où on veut stocker les données par la commande `ds.write(0x55);` ensuite le registre E/S par la commande `ds.write(data[2]);`

Le programme permettant de commander la mémoire 1-Wire est montré en annexe 6.

3. Conception du typon

Pour préparer le typon de chaque circuit nous avons recours à la conception assistée par ordinateur appliquée aux circuits imprimés en utilisant le logiciel Proteus 8.6 produit par la société LABCENTER ELECTRONICS.

Initialement les circuits sont réalisés sur ISIS Proteus pour dessiner le schéma électrique, dans cette étape nous devons prendre en considération le type, l'emplacement et les packages correspondant aux composants utilisés. Puis nous avons utilisé ARES PROTEUS pour simuler les connexions de tous les composants implantés sur nos cartes. Les résultats sont illustrés par les figures suivantes: [21]

3.1. La matrice de commutation

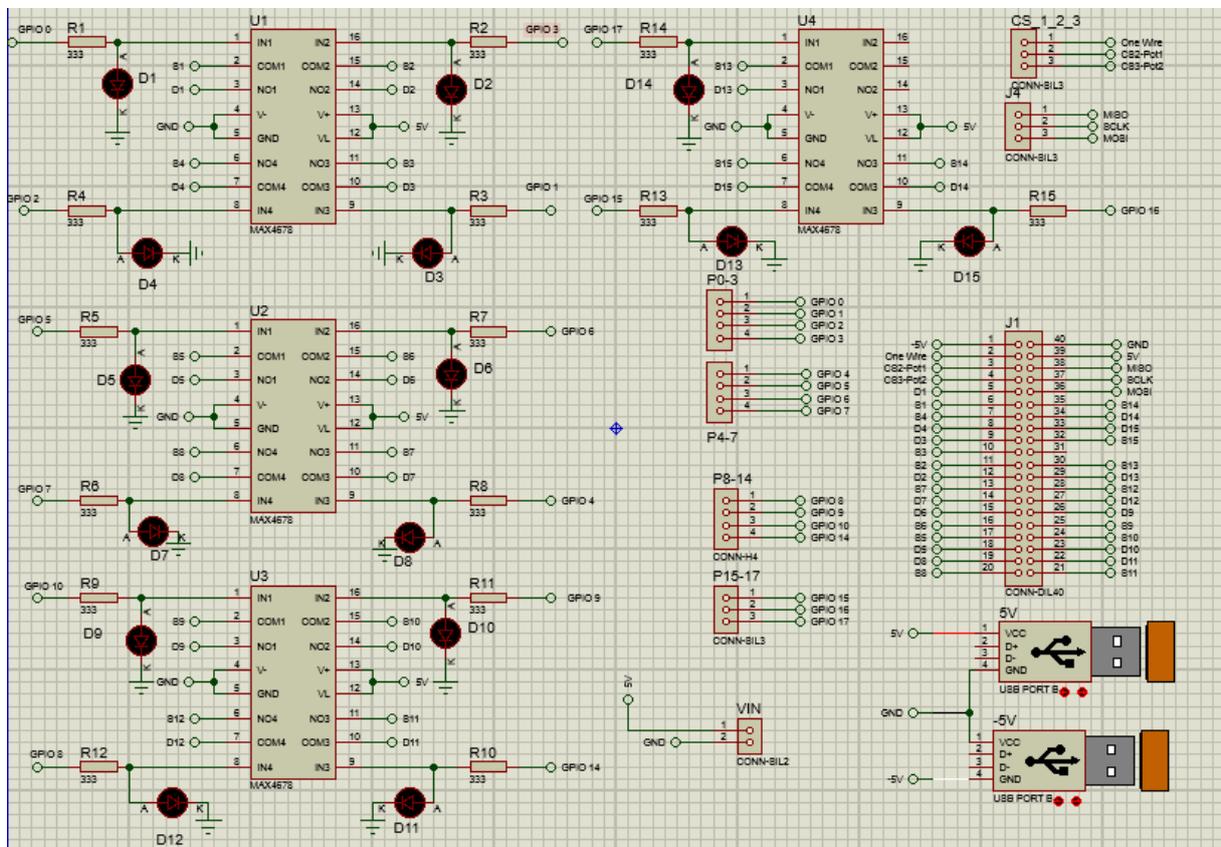


Figure III-3: Schéma électrique de la matrice de commutation sur ISIS Proteus.

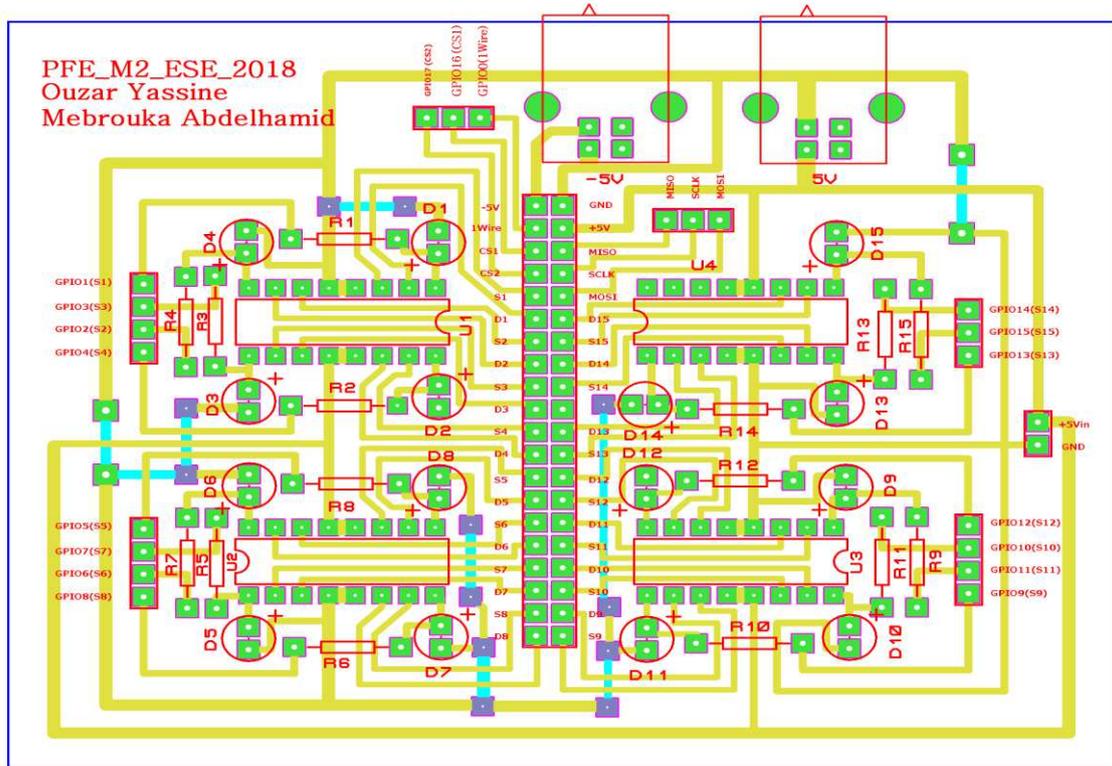


Figure III-4: Schéma du PCB de la matrice de commutation généré par ARES Proteus.

3.2. Carte de TP 1

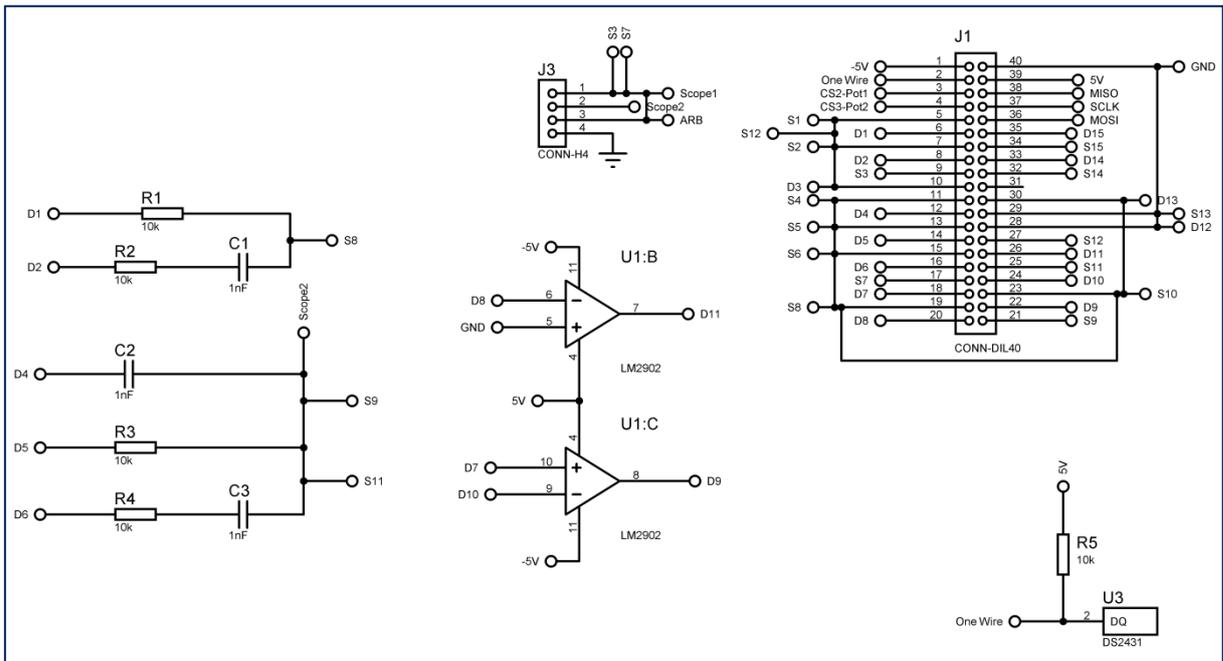


Figure III-5: Schéma électrique de la carte de TP 1 sur ISIS Proteus.

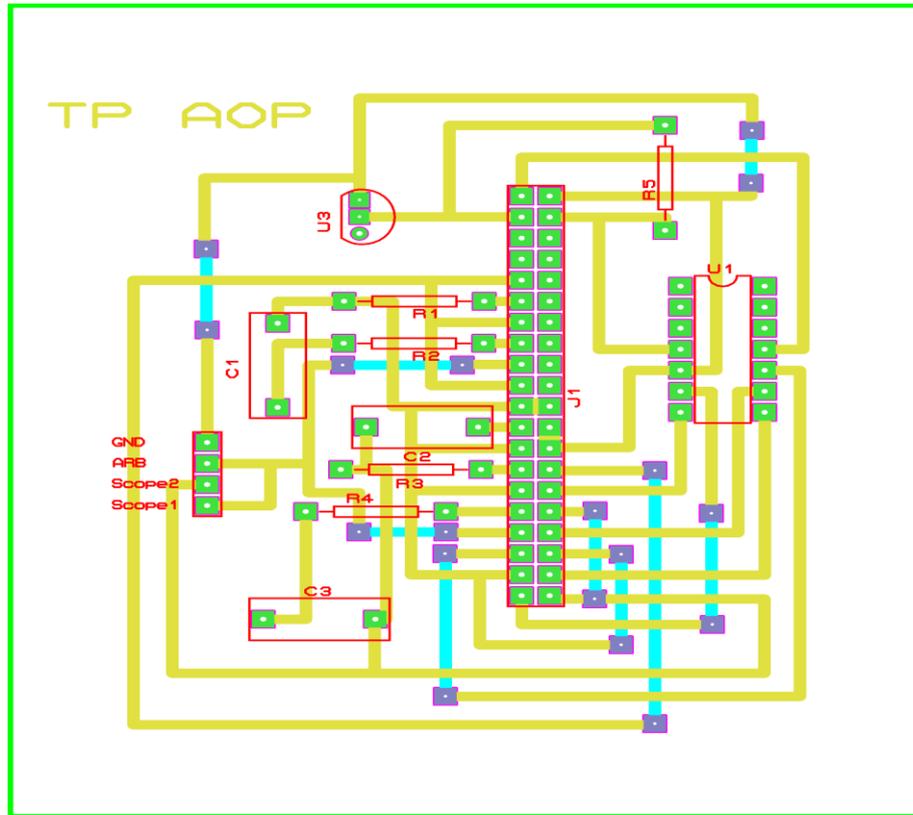


Figure III-6: Schéma du PCB de la carte de TP 1 généré par ARES Proteus.

3.3. Carte de TP 2

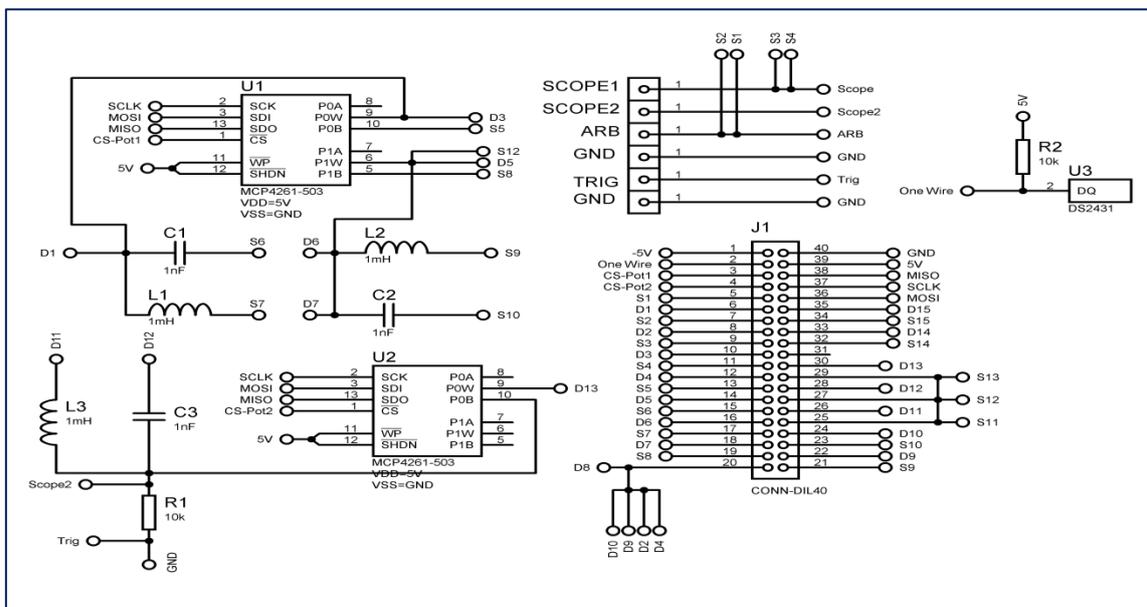


Figure III-7: Schéma électrique de la carte de TP 2 sur ISIS Proteus.

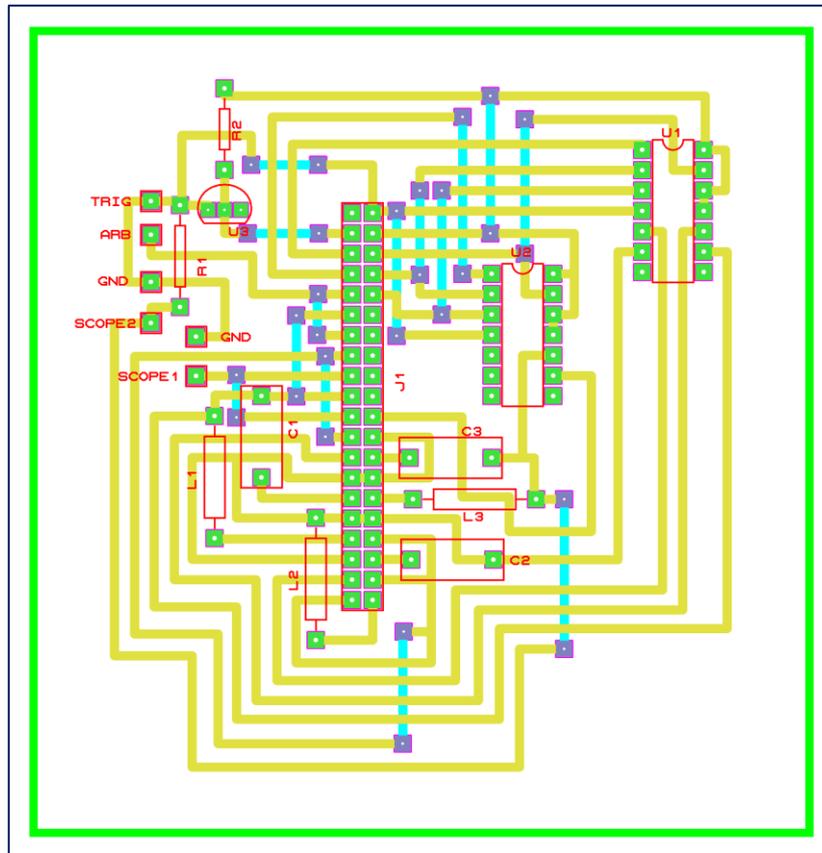


Figure III-8: Schéma du PCB de la carte de TP 2 généré par ARES Proteus.

4. Réalisation des Circuits

Pour la réalisation final des circuits nous avons utilisé deux méthodes, la première est la méthode chimique que nous avons l'habitude de l'utiliser, Les procédures de réalisation du PCB avec cette méthode sont présentées ci-dessous:

1. Impression du typon sur un transparent d'acétate.
2. Faire la superposition du typon sur la plaque photosensible et tracer le trait de coupe, puis la découper à l'aide de la scie à métaux selon les dimensions du CI.
3. Retirer le film protecteur de la plaque époxy et scotcher le typon sur la plaque.
4. Insolation de la résine photosensible à l'aide du rayonnement ultraviolet (UV) pour bruler la résine photosensible à l'UV (Temps d'insolation 3mn).
5. Plonger la plaque dans le révélateur (NaOH) et remuer pendant 30s, ça permet d'éliminer la couche de résine des zones exposées à l'UV.

6. Rincer la plaque à l'eau sans frotter à fin d'éliminer le révélateur restant sur la plaque avant la gravure.
7. Placez la plaque dans la graveuse pendant 20 min. cela permet d'éliminer le film de cuivre exposé à l'UV sauf sur les pistes tracées sur le typon, puis rincer pour éliminer le perchlorure de fer restant. [22]



Insolation



Révélation



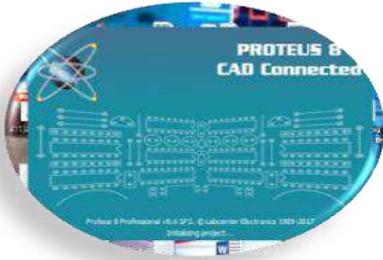
Gravure

Figure III-9: Les étapes principales de la réalisation du circuit avec la méthode chimique.

La deuxième méthode consiste à utiliser la machine CNC à forêt qui est disponible au laboratoire. Pour réaliser le circuit par la CNC il faut exporter un fichier sous format Gerber à partir du schéma PCB réalisé sur Proteus. Ce fichier est interprété par le logiciel (IsoCam) de la machine CNC à forêt.

Après avoir importé le fichier Gerber qui correspond au schéma du circuit à réaliser dans IsoCam, nous utilisons "Create milling data" pour générer le fichier PLT compréhensible par la machine. Puis nous passons à un autre logiciel qui s'appelle RoutePro3000, ce dernier permet d'exécuter le fichier PLT et lancer la gravure du circuit imprimé.

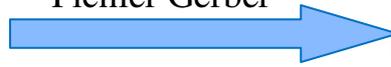
Schéma PCB sur Proteus



Logiciel IsoCam



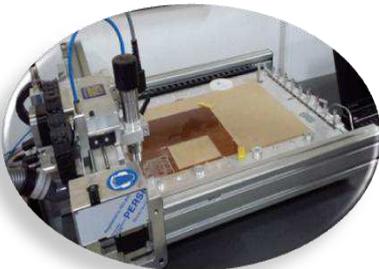
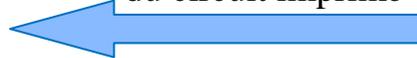
Fichier Gerber



Fichier PLT



Exécution de la gravure
du circuit imprimé



Machine CNC



Logiciel RoutePro3000

Figure III-10: Etapes de réalisation du circuit par la machine CNC.

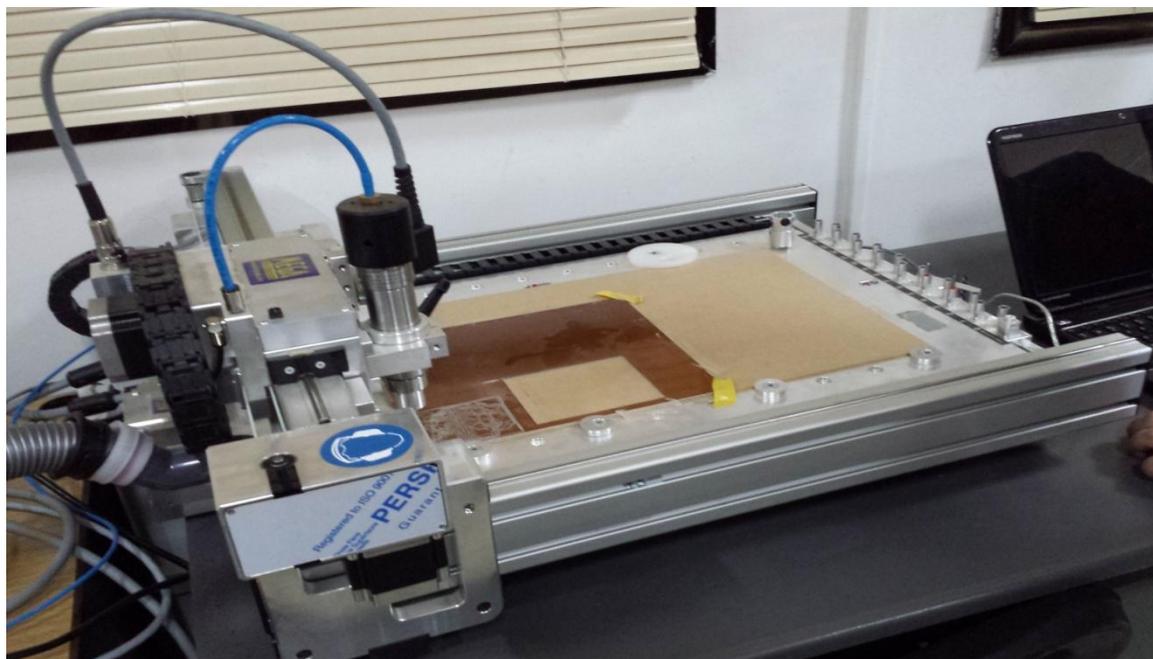


Figure III-11: Réalisation du circuit par la machine CNC.

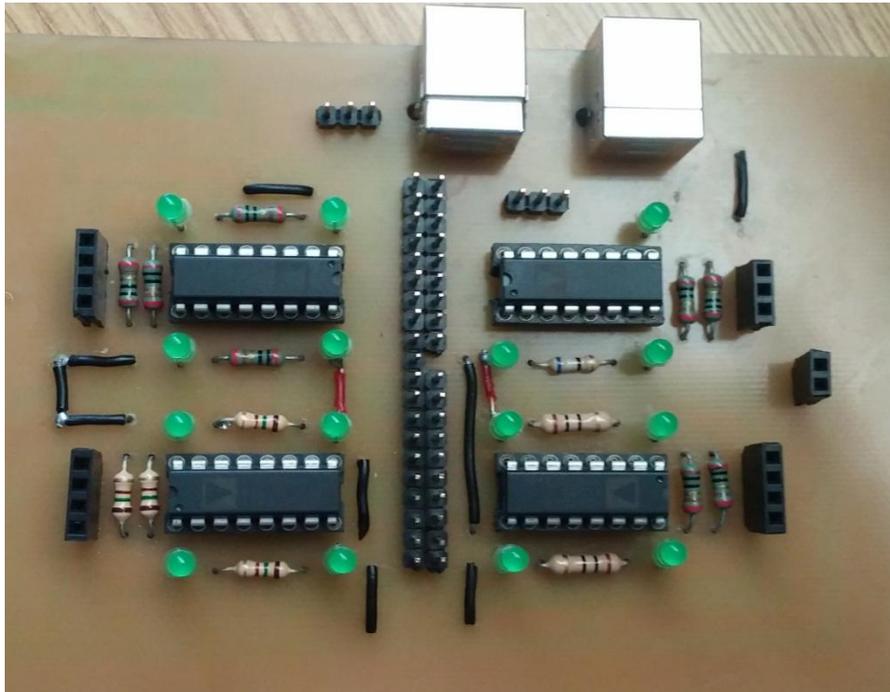


Figure III-12: Circuit final de la matrice de commutation.

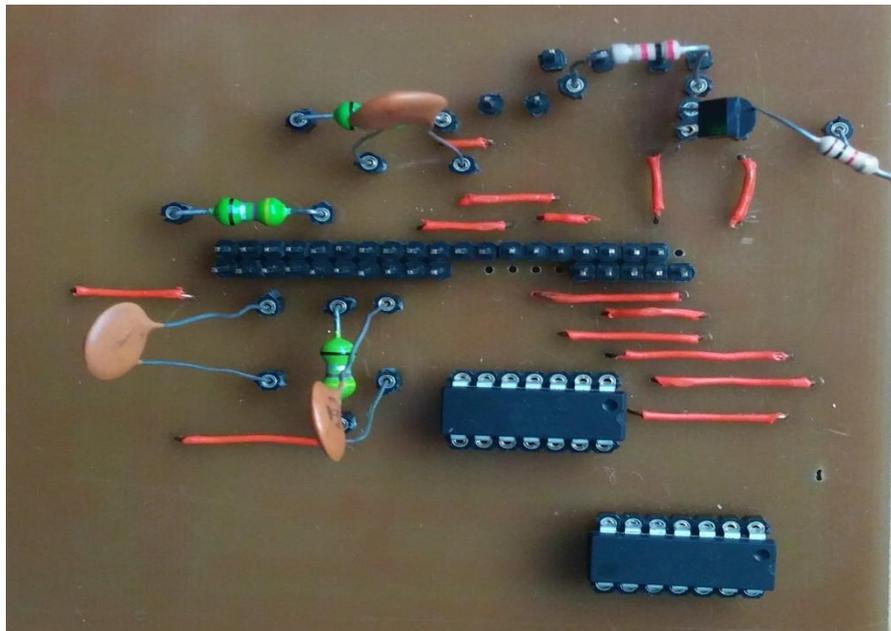


Figure III-13: Circuit final de la carte de TP.

5. Manipulation à distance des travaux pratiques

La réalisation des travaux pratiques à distance se fait à travers l'interface web développée en html/css/javascript qui se connecte d'une manière transparente au serveur node.js implémenté sur PcDuino via des requêtes AJAX en spécifiant son adresse IP. Les travaux pratiques représentés par l'interface web sont conçus spécialement pour des cartes PEB bien particulières.

L'utilisateur sélectionne la manipulation à réaliser parmi les TP disponibles. En arrière-plan, des requêtes http sont envoyées au serveur qui va les interpréter et les traduire en commandes. Ces derniers sont transformés en instructions sur les sorties numériques du PcDuino (GPIO).

Chaque TP dispose une configuration des états des switches bien particulière, ainsi que des valeurs de potentiomètres permettant de réaliser la manipulation désirée. Ces paramètres sont appliqués directement sur les GPIO du PcDuino. Ils sont mis à l'état haut ou à l'état bas pour piloter les switches numériques de la matrice de commutation et quant à la résistance du potentiomètre peut être modifiée via l'interface SPI.

Pour tester le bon fonctionnement de notre laboratoire, Nous avons réalisé deux cartes de TP différentes. Chaque carte possède un identificateur spécifique stocké dans l'EEPROM DS28E07.

La première carte réalisée est à base d'amplificateur opérationnel de type LM2902N, et l'autre un dipôle sous formes de T composée par seulement des résistances, capacités et inductances.

Avec la carte de TP à base d'AOP représentée par la figure III-14, on peut réaliser jusqu'aux 42 montages différents inverseur et non inverseur. En changeant la valeur des potentiomètres alors on arrive à effectuer un nombre très

important que l'on ne peut pas atteindre pendant les séances des travaux pratiques classiques.

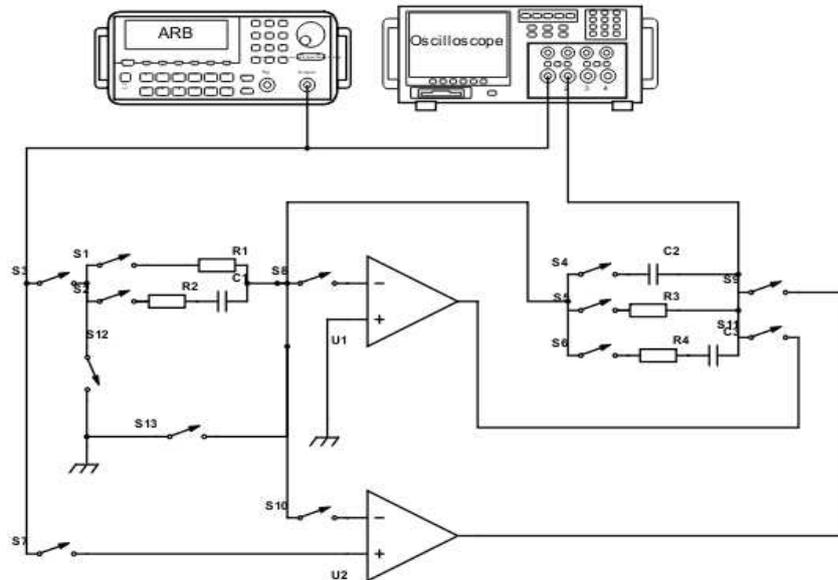


Figure III-14: Schéma de la carte de TP à base d'AOP.

La deuxième carte de TP dipôle a pour but d'étudier les caractéristiques d'un circuit dipôle comme le montre la figure III-15. Les manipulations prévues en utilisant cette carte de TP arrivent jusqu'aux 343 combinaisons différentes ce qui représente un nombre assez important. On peut également visualiser les signaux, effectuer des mesures et calculer des éléments d'impédance (Z) en utilisant un oscilloscope et un générateur de signaux. [23]

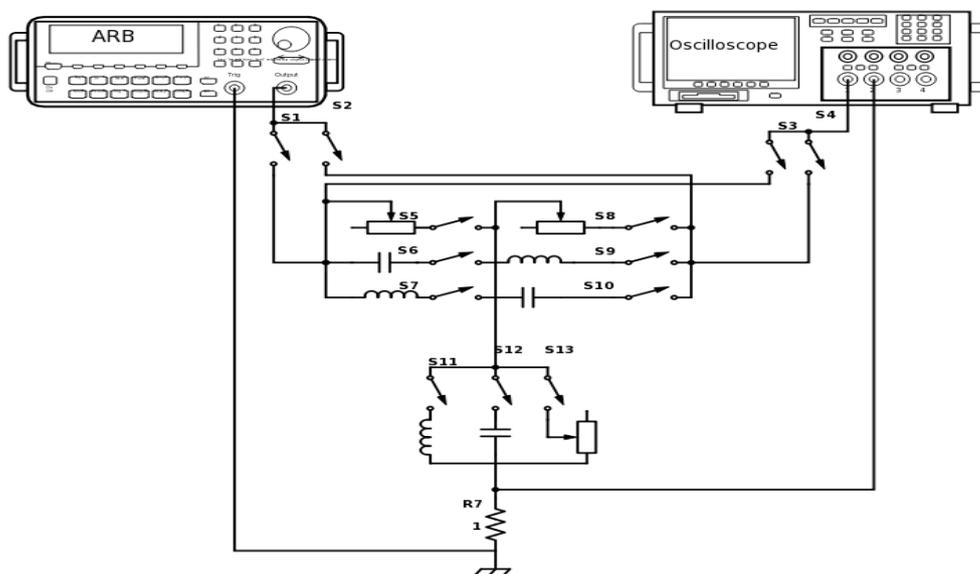


Figure III-15: Schéma de la carte de TP dipôle.

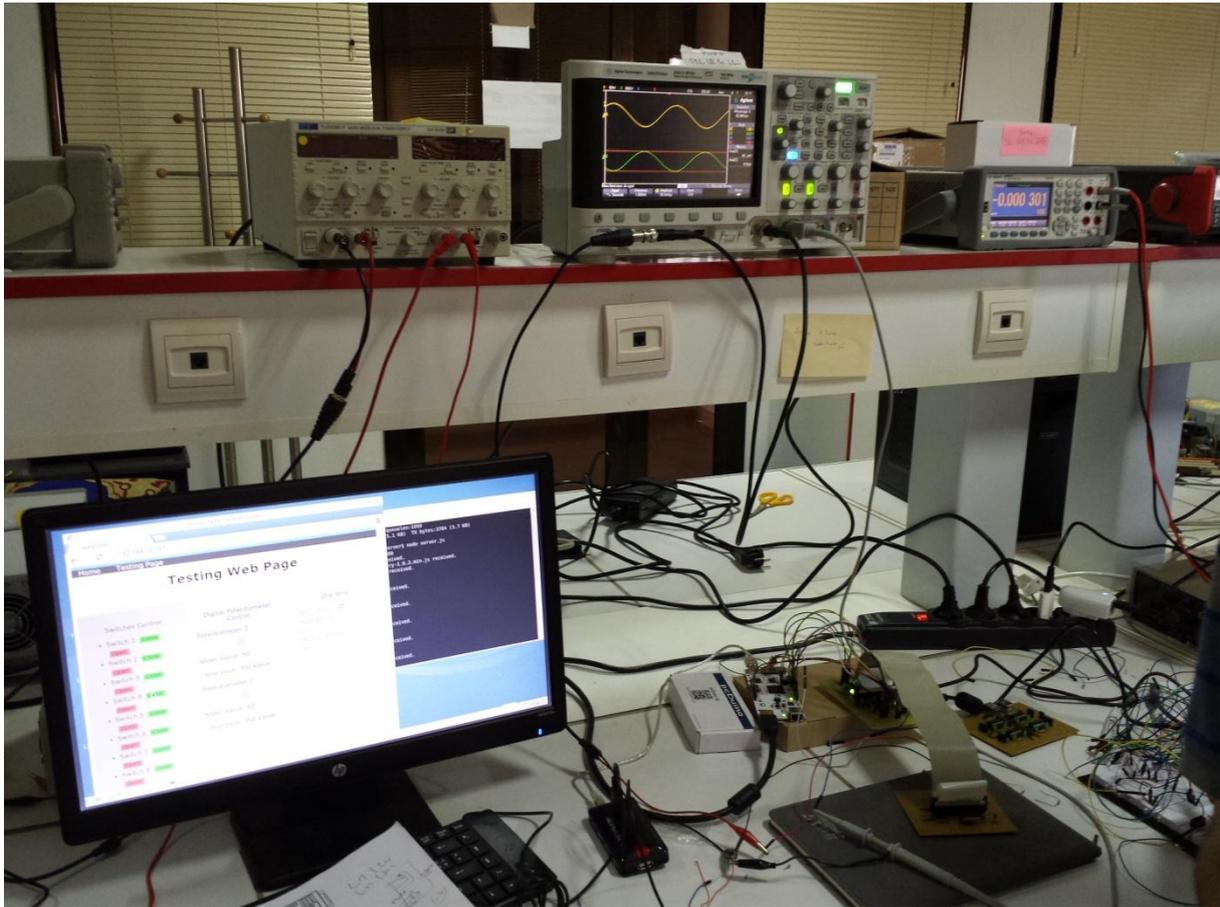


Figure III-16: Laboratoire distant en fonctionnement réel.

6. Conclusion

Ce chapitre a été consacré à l'implémentation des différents algorithmes de l'architecture logicielle ainsi que la réalisation des circuits de la matrice de commutation et l'interface PEB pour tester le bon fonctionnement du laboratoire distant.



Conclusion générale

Conclusion générale

Le travail réalisé est pour nous une très bonne expérience personnelle et pédagogique afin de suivre le rythme de la recherche scientifique dans le domaine d'électronique et notamment le développement des laboratoires à distance. En effet, nous avons pu réaliser un projet d'actualité qui représente une solution pour les problèmes rencontrés dans les séances de travaux pratiques.

Grace à cette plateforme du laboratoire distant on peut effectuer les TP d'électronique à distance sans la nécessité de se déplacer au laboratoire, il suffit d'utiliser un ordinateur connecté à Internet. D'autre part nous proposons une architecture flexible et réutilisable permettant d'effectuer un nombre très important de manipulations, manipuler des équipements coûteux et visualiser les résultats à travers une interface web.

L'avantage de notre projet qu'il est basé sur des outils open source et efficace avec un coût de développement très faible par rapport aux travaux des autres laboratoires tout en offrant une implémentation très facile.

L'architecture innovante de ce laboratoire distant peut être exploitée pour effectuer des TP de plusieurs modules de l'électronique (électronique analogique, électromagnétisme, traitement de signal et électronique de puissance...). Le pilotage à distance de ces TP se fait par un simple échange de la carte PEB.

En guise de perspectives, un complément de ce travail peut être réalisé en développant une plateforme rassemblant l'ensemble des appareils que peut contenir une manipulation de TP dans un seul boîtier en utilisant le Red Pitaya qui est destiné à remplacer de nombreux instruments de mesure et de contrôle de laboratoire coûteux. Il intègre un oscilloscope et un GBF à la fois. En développant une application web approprié qui gère le laboratoire distant nous serons en mesure d'abandonner les équipements coûteux du laboratoire.

Références bibliographiques

- [1] S. Farah, Etude et conception d'un système embarqué pilotable à distance pour la mesure temporelle dynamique in situ d'amplificateurs de très fortes puissances adaptatifs. Thèse de doctorat de l'Université de Mostaganem, Algérie, 2017.
- [2] PcDuino, Page consultée le 03/2018 à partir de:
<http://www.linksprite.com/linksprite-pcduino2/>
- [3] 2Ω, Quad, SPST, CMOS Analog Switches, Document téléchargé le 03/2018 à partir de:
<https://www.maximintegrated.com/en/datasheet/index.mvp/id/2274>
- [4] MCP4231, Document téléchargé le 03/2018 à partir de:
<https://www.microchip.com/datasheet/MCP4231>
- [5] DS28E07, Page consultée le 04/2018 à partir de:
<https://www.maximintegrated.com/en/products/digital/memory-products/DS28E07.html>
- [6] Communication AJAX, Page consultée le 04/2018 à partir de:
<http://wdi.supelec.fr/appliouaibe/Cours/AJAX>
- [7] HTML, Page consultée le 04/2018 à partir de:
http://www.standard-du-web.com/hypertext_markup_language.php
- [8] CSS, Page consultée le 04/2018 à partir de:
<https://www.w3.org/Style/CSS/Overview.fr.html>
- [9] JavaScript, Page consultée le 04/2018 à partir de:
<https://fr.wikipedia.org/wiki/JavaScript>
- [10] Introduction à jquery, Page consultée le 04/2018 à partir de:
<https://openclassrooms.com/courses/introduction-a-jquery-4/javascript-jquery-c-est-quoi-la-difference>
- [11] Communication AJAX, Page consultée le 04/2018 à partir de:
<http://wdi.supelec.fr/appliouaibe/Cours/AJAX>

- 
- [12] Développer avec node.js, Page consultée le 05/2018 à partir de:
<https://www.bocasay.com/pourquoi-developper-avec-node-js/>
- [13] <https://www.supinfo.com/articles/single/946-npm-package-manager-nodejs>
- [14] Bus SPI, Document téléchargé le 04/2018 à partir de:
<http://robert.cireddu.free.fr/SIN/Bus%20SPI.pdf>
- [15] SPI, Page consultée le 05/2018 à partir de:
https://fr.wikipedia.org/wiki/Serial_Peripheral_Interface
- [16] 1-Wire, Page consultée le 05/2018 à partir de:
<https://fr.wikipedia.org/wiki/1-Wire>
- [17] Mesure de température 1-wire DS18B20 avec le Raspberry Pi, Page consultée le 04/2018 à partir de: <https://www.framboise314.fr/mesure-de-temperature-1-wire-ds18b20-avec-le-raspberry-pi/>
- [18] Principe du protocole 1-wire, Page consultée le 04/2018 à partir de:
<https://blog.domadoo.fr/guides/principe-du-protocole-1-wire/>
- [19] S. Farah, A. Benachenhou, G. Neveux, D. Barataud, G. Andrieu, T. Fredon, "Flexible and real-time remote laboratory architecture based on Node.js server," 2015 3rd Experiment International Conference (exp.at'15), Ponta Delgada, 2015, pp. 155-156.
- [20] Proteus (ISIS et ARES), Page consultée le 05/2018 à partir de:
<http://www.elektronique.fr/logiciels/proteus.php>
- [21] Modules & NPM, Page consultée le 04/2018 à partir de:
<https://www.grafikart.fr/formations/nodejs/npm>
- [22] Gravure de circuit imprimé: Premier essai concluant, Page consultée le 05/2018 à partir de:
https://acolab.fr/2014/04/pcb_premiers-essai-concluant/
- [23] A.Adda Benattia, A.Benachenhou, M.Moussa, "Development of an Automatic Assessment in Remote Experimentation over Remote Laboratory", 2011 15th International Conference on Remote Engineering and Virtual Instrumentation, At Duesseldorf, Germany.

Annexes

Annexe1: Présentation du PcDuino

Le PcDuino est un mini-ordinateur avec une performance élevée et une plateforme compatible avec des fonctionnalités qui peuvent fonctionner sur des systèmes d'exploitation tels que Ubuntu et Android. Il suffit de raccorder cette carte à une alimentation 5 Vcc, un clavier, une souris et un écran pour être opérationnel.

L'écran de sorties PcDuino à HDMI et son matériel est compatible avec l'Arduino. Le PcDuino prend en charge les langages de programmation tels que C, C ++, Java, Python, Arduino et bien d'autres. Nous pouvons même utiliser l'USB-OTG pour connecter à distance. Il suffit de le connecter au réseau avec le module WiFi ou la prise Ethernet pour pouvoir enregistrer des données, exécuter un serveur Web ou contrôler des périphériques à distance.

Description du PcDuino V2:

La carte pcDuino V2 que nous avons utilisé dans notre projet est basée sur un processeur ARM Cortex A8 cadencé à 1 GHz. Elle a 1 Go de RAM, 2 Go de Flash, 2 ports USB hôte, 1 port USB OTG, vidéo HDMI, Ethernet et slot Micro-SD. Elle est livrée avec Ubuntu 12.04 préinstallé et son bureau graphique est étonnamment réactif pour une petite machine. Elle est en fait une des single board computers (SBC) les plus puissantes sur le marché.

Cette carte est conçue pour faciliter le développement de projets pour la communauté open source. Cette plateforme peut exécuter un système d'exploitation et dispose d'une chaîne d'outils facile à utiliser. Elle est également compatible avec tous les systèmes de la célèbre Arduino tels que les shields ou encore avec des projets open source. Une API a été développée et permet aux utilisateurs du PcDuino d'utiliser le langage de programmation Arduino.

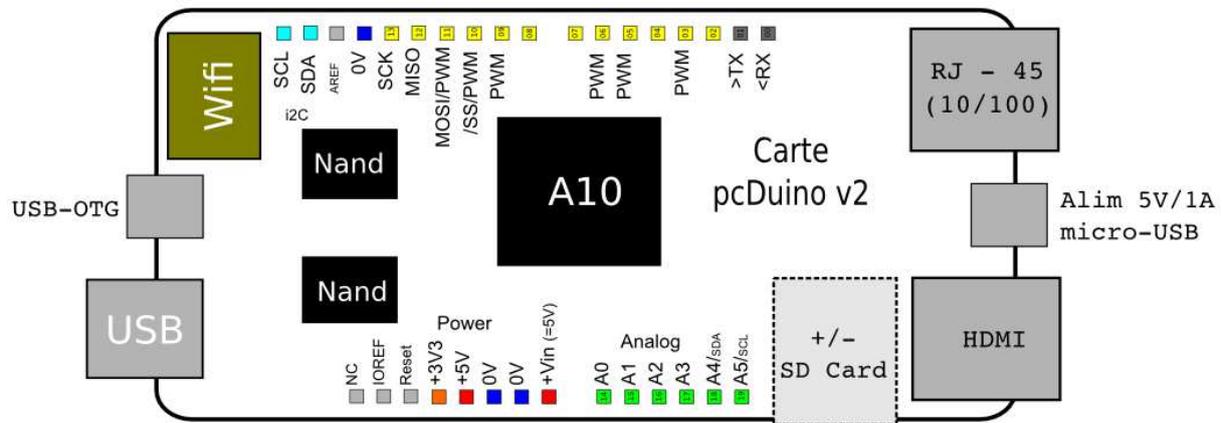


Schéma du PcDuino V2

Spécifications matérielles :

Alimentation	5 Vcc / 2 A (via micro-USB)
GPIO	18 E/S en 3.3 V
Entrées analogiques	2 x 6 bits en 0-2V et 4 x 12 bits en 0-3.3V
Sorties analogiques	6 x PWM
CPU	AllWinner A10 SoC 1 GHz ARM Cortex A8
GPU	OpenGL ES2.0, OpenVG 1.1 Mali 400 Core
DRAM	1 GB
Mémoire flash	2 GB
Support pour microSD	32 GB (non incluse)
Sortie vidéo	HDMI
Bus	UART, SPI et I2C
Connexion ethernet	10/100 Mbps
Interface réseau	RJ45 et WiFi intégré
Dimensions	128x52mm

Spécifications logicielles:

OS	Linux 3.0 + Ubuntu 12.10 et Android ICS 4.0
API	Il se compose d'API pour accéder aux interfaces suivantes: UART, ADC, PWM, GPIO, I2C, SPI
Support de la langue de programmation	C/C ++ avec chaîne d'outils GNU Java avec SDK Android standard

Connectique nécessaire:

- 1 câble HDMI.
- 1 câble USB (micro-USB).
- 1 câble ethernet.

Initiation à la mise en œuvre matérielle du PcDuino:

Accès aux broches GPIO:

Le PcDuino est livré avec 23 broches auxquelles on peut accéder, entre autres, des E / S à usage général GPIO. L'accès aux broches GPIO n'est pas difficile, il suffit d'ouvrir simplement le fichier associé à la broche qu'on veut accéder et lire ou écrire ce fichier. Les fichiers sont situés dans les emplacements suivants:

`/sys/devices/virtual/misc/gpio/mode/`

`/sys/devices/virtual/misc/gpio/pin/`

Certaines broches peuvent faire un double (ou triple) fonctions. L'écriture de certaines chaînes dans le fichier "mode" active les différents modes (série, SPI, entrée, sortie, etc...).

- ❖ '0' → configurer la broche en entrée.
- ❖ '1' → configurer la broche en sortie.
- ❖ '8' → configurer la broche en entrée, avec une résistance pull-up.

Chaque broche a son propre fichier, il y a 23 fichiers et seulement 18 d'entre eux sont effectivement mappés aux broches disponibles.

Entrée et sortie analogiques:

Contrairement au Raspberry Pi, le PcDuino possède des fonctionnalités d'entrée et de sortie analogiques.

L'entrée analogique sur le PcDuino se fait via six broches dédiées étiquetées A0-A5 sur les en-têtes Arduino-ish. A0 et A1 sont des entrées à six bits, renvoyant une valeur de 0 à 63 sur une plage de 0 à 2 V, A2-A5 sont des entrées 12 bits fonctionnant sur toute la plage de 3,3 V.



La sortie analogique sur le PcDuino est réalisée via PWM. Il y a six broches compatibles PWM dans les en-têtes Arduino-ish: 3, 5, 6, 9, 10 et 11. Ceux-ci, correspondent aux broches PWM des cartes Arduino. Les broches 5 et 6 sont de véritables PWM à 520 Hz à 8 bits, les autres sont limités à une plage de 0-20 à 5Hz. La tension de sortie pour toutes les broches est de 3,3V.

Communications série

Le PcDuino est livré avec plusieurs ports série intégrés. Le processeur A10 sur ce mini-ordinateur a huit ports série. Seuls deux de ces ports sont mappés vers un périphérique Linux: (/ dev/ttyS0 ; /dev/ttyS1).

Paramètres du mode Pin:

Comme mentionné sur la page GPIO, les registres "mode" pour chaque broche contrôlent la fonctionnalité de cette broche. Pour les E/S série, nous devons écrire '3' dans les fichiers de mode pour les broches 0 et 1.

Communication SPI:

Le PcDuino a des headers exposant deux bus SPI différents. Cependant, un seul bus et un seul périphérique sur ce bus sont pris en charge. Il est possible d'utiliser d'autres broches GPIO pour ajouter des lignes de sélection d'esclaves avec une petite configuration.

Le périphérique SPI actuellement disponible SPI0 est accessible via quatre broches compatibles Arduino situés le long du bord inférieur de la carte. Ces broches répartissent les lignes MOSI, MISO, SCK et CS du processeur, ce qui permet une prise en charge complète des communications SPI par le périphérique.

Annexe2: Brochage du potentiomètre numérique MCP4231

Potentiomètre numérique MCP4231		
Pin	Nom de broche	Description
1	CS	Chip Select est la broche de sélection d'esclave pour l'interface SPI. Elle est active à l'état bas. 0V signifie que le CI est sélectionné et 5V signifie qu'il n'est pas sélectionné
2	SCLK	La ligne d'horloge SPI.
3	SDI	La broche d'entrée des données série, également connu sous le nom MOSI
4	V _{SS}	La masse
5	P1B	La borne du potentiomètre 1
6	P1W	La borne WiperLock du potentiomètre 1.
7	P1A	La borne du potentiomètre 1.
8	PA0	La borne du potentiomètre 0.
9	PW0	La borne WiperLock du potentiomètre 0.
10	PB0	La borne du potentiomètre 0.
11	WP	Write Protect (Il n'est pas connecté (NC))
12	SHDN	La broche d'arrêt. Active à l'état bas. Lorsqu'il est maintenu à l'état bas, le matériel déconnecte le WiperLock du réseau de résistances interne. Puisque nous voulons toujours que le potentiomètre soit actif, nous gardons cette broche connectée à 5V.
13	SDO	La broche de sortie des données série. Cette broche est également connue sous le nom de MISO.
14	V _{DD}	La source de tension

Annexe 3: Code source de l'interface web

Index.htm:

```
<html>
<head>
  <title>Home Page</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="Styles/style.css">
  <link rel="stylesheet" href="Styles/bootstrap.min.css">
  <script type="text/javascript" src="Scripts/jquery.min.js"></script>
  <script type="text/javascript" src="Scripts/popper.min.js"></script>
  <script type="text/javascript"
src="Scripts/bootstrap.min.js"></script>
</head>
<body>
  <div class="page-header" ><h1 class="display-1 text-white" align="center"
>INT-RE-LAB</h1></div>
  <nav class="navbar navbar-expand-sm bg-dark navbar-dark justify-content-
center">
    <div class="container-fluid font-weight-bold">
      <li class="nav-item" ><a href="/index.htm" id="Home"
class="navbar-brand" >Home</a></li>
      <li class="nav-item" ><a href="/test.htm" id="test"
class="navbar-brand" >Testing Page</a></li>
      <li class="nav-item" ><a href="/tpot.htm" id="tpot"
class="navbar-brand" >TP TPoL</a></li>
      <li class="nav-item" ><a href="/aop.htm" id="aop"
class="navbar-brand" >TP AOP</a></li>
    </div>
  </nav>
  <div class="container-fluid bg-1 text-center">
    <h1 class="display-1">PFE Master ESE</h1>
  </div>

  <div class="container-fluid bg-3 text-center" style="background-color:
#0f74a5e3;"><br/>
    <h3>Cette interface été développée pour tester le fonctionnement et
pour la mise en oeuvre de Laboratoire à distance. </h3> <br/>
    <div class="row text-center">
      <div class="col-sm-4">
        <a href="#testcolaps" data-toggle="collapse">
          
        </a>
        <div id="testcolaps" class="collapse">
```

```

        <p>Cette page permet de commander manuellement les
switches, les potentiomètres numériques et de communiquer avec la mémoire 1-
Wire. </p>
    </div>
    <a href="/test.htm" class="text-white" > <h3>Testing Page</h3>
</a>
</div>
<div class="col-sm-4">
    <a href="#tpolcolaps" data-toggle="collapse">
        
    </a>
    <div id="tpolcolaps" class="collapse">
        <p>Exemple d'un tp Di-Pol RLC.</p>
    </div>
    <a href="/tpot.htm" class="text-white" > <h3>Tp TPol</h3> </a>
</div>
<div class="col-sm-4">
    <a href="#taopcolaps" data-toggle="collapse">
        
    </a>
    <div id="taopcolaps" class="collapse">
        <p>Exemple d'un tp Amplificateur Opérationnel.</p>
    </div>
    <a href="/tpot.htm" class="text-white" > <h3>Tp AOP</h3> </a>
</div>
</div>
</div>
</body>
</html>

```

test.htm:

```

<html>
  <head>
    <title>Testing Web Page</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="Styles/style.css">
    <link rel="stylesheet" href="/Styles/bootstrap.min.css">
    <script type="text/javascript" src="Scripts/jquery.min.js"></script>
    <script type="text/javascript" src="Scripts/popper.min.js"></script>
    <script type="text/javascript"
src="Scripts/bootstrap.min.js"></script>

    <script type="text/javascript">

    var ip = "http://192.168.1.7:8080";

```

```

$(function(){
    $("#[id='switch']").click(function(e){
        e.preventDefault();
        console.log('button clicked');
        var data = {};
        data.pins = this.name;
        if (this.value == "Close") { data.val = "1"; }
        else if (this.value == "Open") { data.val = "0"; }
        $.ajax({
            type: 'GET',
            data: data,
            contentType: 'application/json',
            url: ip+'/setPin.js',
        });
    });
    $("#[type='range']").click(function(e){
        e.preventDefault();
        console.log('Slider changed');
        var data = {};
        data.cs = this.name;
        data.val = this.value;
        data.pot = this.id;
        $.ajax({
            type: 'GET',
            data: data,
            contentType: 'application/json',
            url: ip+'/pd-spi-arg.py',
        });
    });
    $("#[id='eprom']").click(function(e){
        e.preventDefault();
        console.log('ROM button clicked');
        var data = {};
        data.cmd = this.name;
        if (this.name == "readadr") {
            data.addr=document.getElementById("addr").value; }
        if (this.name == "writeadr") {
            data.addr=document.getElementById("addr").value;
            data.val=document.getElementById("val").value;
        }
        $.ajax({
            type: 'GET',
            data: data,
            contentType: 'application/json',
            url: ip+'/eprom.c',
            success: function(result){
                switch(data.cmd){
                    case 'readid':
                        var chipid = document.getElementById("romid");

```

```

        chipid.innerHTML = result;
        break;
    case 'readadr':
        var chipid =
document.getElementById("memdata");
        chipid.innerHTML = result;
        break;
    case 'writeadr':
        var chipid = document.getElementById("wfeed");
        chipid.innerHTML = result;
        break;
    }
    });
});
</script>
</head>
<body>
    <div class="page-header" ><h1 class="display-1 text-white"
align="center" >INT-RE-LAB</h1></div>
    <nav class="navbar navbar-expand-sm bg-dark navbar-dark justify-
content-center">
        <div class="container-fluid font-weight-bold">
            <li class="nav-item" ><a href="/index.htm" id="Home"
class="navbar-brand" >Home</a></li>
            <li class="nav-item" ><a href="/test.htm" id="test"
class="navbar-brand" >Testing Page</a></li>
            <li class="nav-item" ><a href="/tpot.htm" id="tpot"
class="navbar-brand" >TP TPol</a></li>
            <li class="nav-item" ><a href="/aop.htm" id="aop"
class="navbar-brand" >TP AOP</a></li>
        </div>
    </nav>
    <h1 align="center" class="display-3" >Testing Web Page</h1><br>

    <div class="row">
        <div class="col" align="center" style="background-color:#aaa;" >
            <h3 align="center" >Switches Control </h3>
            <div >
                <div class="btn-group-vertical btn-group-sm">
                    Switch 1 <input type="button" name="1" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="1"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
                    Switch 2 <input type="button" name="2" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="2"
value="Open" class="btn btn-danger btn-sm" id="switch"/>

```

```

        Switch 3 <input type="button" name="3" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="3"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 4 <input type="button" name="4" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="4"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 5 <input type="button" name="5" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="5"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
    </div>

    <div class="btn-group-vertical btn-group-sm">
        Switch 6 <input type="button" name="6" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="6"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 7 <input type="button" name="7" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="7"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 8 <input type="button" name="8" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="8"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 9 <input type="button" name="9" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="9"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 10 <input type="button" name="10" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="10"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
    </div>

    <div class="btn-group-vertical btn-group-sm">
        Switch 11 <input type="button" name="11" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="11"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 12 <input type="button" name="12" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="12"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 13 <input type="button" name="13" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="13"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 14 <input type="button" name="14" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="14"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
        Switch 15 <input type="button" name="15" value="Close"
class="btn btn-success btn-sm" id="switch"/> <input type="button" name="15"
value="Open" class="btn btn-danger btn-sm" id="switch"/>
    </div>

</div>
</div>

```

```

<div class="col" style="background-color:#bbb;">
  <h3 align="center">Digital Potentiometer Control</h3>
  <div align="center" class="slidecontainer">
    <p>Chip 1</p>
    <input type="range" name="CS1" min="0" max="127"
value="64" step="1" class="slider" id="pot0">
    <p> PotA Value: <span id="slider1val"></span> Ohms
</p>

    <input type="range" name="CS1" min="0" max="127"
value="64" step="1" class="slider" id="pot1">
    <p> PotB Value: <span id="slider2val"></span> Ohms
</p>
  </div>
  <div align="center" class="slidecontainer">
    <p>Chip 2 </p>
    <input type="range" name="CS2" min="0" max="127"
value="64" step="1" class="slider" id="pot0">
    <p> PotA Value: <span id="slider3val"></span> Ohms
</p>

    <input type="range" name="CS2" min="0" max="127"
value="64" step="1" class="slider" id="pot1">
    <p> PotB Value: <span id="slider4val"></span> Ohms
</p>
  </div>
  <script>
    var pot1slider = document.getElementById("pot0");
    var slider1_val =
document.getElementById("slider1val");
    slider1_val.innerHTML = ((pot1slider.value*5000)/128 +
75).toFixed(2);

    var pot2slider = document.getElementById("pot1");
    var slider2_val =
document.getElementById("slider2val");
    slider2_val.innerHTML = ((pot2slider.value*5000)/128 +
75).toFixed(2);

    var pot3slider = document.getElementsByName("CS2")[0];
    var slider3_val =
document.getElementById("slider3val");
    slider3_val.innerHTML = ((pot3slider.value*5000)/128 +
75).toFixed(2);

    var pot4slider = document.getElementsByName("CS2")[1];
    var slider4_val =
document.getElementById("slider4val");
    slider4_val.innerHTML = ((pot4slider.value*5000)/128 +
75).toFixed(2);
  </script>

```

```

        pot1slider.oninput = function() {
            slider1_val.innerHTML = ((this.value*5000)/128 +
75).toFixed(2);
        }
        pot2slider.oninput = function() {
            slider2_val.innerHTML = ((this.value*5000)/128 +
75).toFixed(2);
        }
        pot3slider.oninput = function() {
            slider3_val.innerHTML = ((this.value*5000)/128 +
75).toFixed(2);
        }
        pot4slider.oninput = function() {
            slider4_val.innerHTML = ((this.value*5000)/128 +
75).toFixed(2);
        }
    }
</script>
</div>
<div class="col" align="left" style="background-color:#ccc;">
    <h3 align="center">One Wire</h3>
    <input type="button" name="readid" value="Read Chip ID"
class="btn btn-info" id="eprom"/> ID: <span id="romid"></span></br></br>
    <form>
        <input type="submit" name="readadr" value="Read Address"
class="btn btn-info" id="eprom"/> 0x00 <input type="text" name="Address"
id="addr" size="1" maxlength="2" pattern="[A-Fa-f0-9]{2}" title="Enter Two
HEX Digits" placeholder="00"><span id="memdata"></span><br>
    </form>
    <form>
        <input type="submit" name="writeadr" value="Write to
Address" class="btn btn-info" id="eprom"/> <input type="text" name="Data"
id="val" size="8" maxlength="8" placeholder="Data to W"><br><span
id="wfeed"></span><br>
    </form>
    </div>
</div>
</body>
</html>

```

Annexe 4: Code source du serveur

Serveur.js :

```

var http = require('http');
var fs = require('fs');
var url = require('url');
var querystring = require('querystring');

```

```

var child_process = require('child_process');

http.createServer( function (request, response) { // Création du serveur
    var pathname = url.parse(request.url).pathname; // Récupération du chemin
    a partir de la requete HTTP
    if(pathname=='/'){ pathname='/test.htm' }
    if (pathname=="ser_pcdduino_2018.js"){ //Requete AJAX pour Modifiée
    les switches
        var params = querystring.parse(url.parse(request.url).query);
        console.log(params);
        console.log(typeof params);
        if ('v1', 'v2', 'v3', 'v4', 'v5', 'v6', 'v7', 'v8', 'v9',
            'v10', 'v11', 'v12', 'v13', 'v14', 'v15' in params) {
            for (i = 1; i < 15; i++) { child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/mode/gpio'+i); } // pin en mode out '1'
            child_process.exec('echo '+params['v1'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio1');
            child_process.exec('echo '+params['v2'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio2');
            child_process.exec('echo '+params['v3'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio3');
            child_process.exec('echo '+params['v4'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio4');
            child_process.exec('echo '+params['v5'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio5');
            child_process.exec('echo '+params['v6'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio6');
            child_process.exec('echo '+params['v7'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio7');
            child_process.exec('echo '+params['v8'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio8');
            child_process.exec('echo '+params['v9'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio9');
            child_process.exec('echo '+params['v10'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio10');
            child_process.exec('echo '+params['v11'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio11');
            child_process.exec('echo '+params['v12'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio12');
            child_process.exec('echo '+params['v13'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio13');
            child_process.exec('echo '+params['v14'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio14');
            child_process.exec('echo '+params['v15'] +' >
/sys/devices/virtual/misc/gpio/pin/gpio15');
        }
        else { console.log('not executed') }
        response.writeHead(200, {'Content-Type': 'text/html'});
    response.end();
}

```

```

}

if(pathname=="setPin.js"){ //Requete AJAX pour Modifié un switch
  console.log('Switch command');
  var params = querystring.parse(url.parse(request.url).query);
  console.log(params);
  var workerProcess = child_process.exec('node Scripts/setPin.js '+
params.pins + ' ' + params.val ,function (error, stdout, stderr) {
    if (error) {
      console.log(error.stack);
      console.log('Error code: '+error.code);
      console.log('Signal received: '+error.signal);
    }
  });
  response.writeHead(200, {'Content-Type': 'text/html'});
response.end();
}
if(pathname=="pd-spi-arg.py"){ //Requete AJAX pour commander
potentiomètre
  console.log('Set Digital pot');
  var params = querystring.parse(url.parse(request.url).query);
  console.log(params);
  var cs1=16;      var cs2=17;
  var gpio10 =
fs.readFileSync('/sys/devices/virtual/misc/gpio/pin/gpio10');
  var gpio11 =
fs.readFileSync('/sys/devices/virtual/misc/gpio/pin/gpio11');
  var gpio12 =
fs.readFileSync('/sys/devices/virtual/misc/gpio/pin/gpio12');
  var gpio13 =
fs.readFileSync('/sys/devices/virtual/misc/gpio/pin/gpio13');
  child_process.exec('echo 2 >
/sys/devices/virtual/misc/gpio/mode/gpio10'); // pin en mode SPI '2'
  child_process.exec('echo 2 >
/sys/devices/virtual/misc/gpio/mode/gpio11');
  child_process.exec('echo 2 >
/sys/devices/virtual/misc/gpio/mode/gpio12');
  child_process.exec('echo 2 >
/sys/devices/virtual/misc/gpio/mode/gpio13');

  child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/mode/gpio'+cs1);
  child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/mode/gpio'+cs2);

  if(params.cs=='CS1'){
    child_process.exec('echo 0 >
/sys/devices/virtual/misc/gpio/pin/gpio'+cs1);

```

```

        child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/pin/gpio'+cs2);
    }

    if(params.cs=='CS2'){
        child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/pin/gpio'+cs1);
        child_process.exec('echo 0 >
/sys/devices/virtual/misc/gpio/pin/gpio'+cs2);
    }

    var workerProcess = child_process.exec('python Scripts/pd-spi-arg.py '
+ ' ' +params.val+ ' ' + params.pot ,function (error, stdout, stderr) {
        console.log(stdout)
        if (error) {
            console.log(error.stack);
            console.log('Error code: '+error.code);
            console.log('Signal received: '+error.signal);
        }
        else {
            child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/pin/gpio'+cs1);
            child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/pin/gpio'+cs2);

            child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/mode/gpio10');
            child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/mode/gpio11');
            child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/mode/gpio12');
            child_process.exec('echo 1 >
/sys/devices/virtual/misc/gpio/mode/gpio13');

            child_process.exec('echo '+gpio10 + ' >
/sys/devices/virtual/misc/gpio/pin/gpio10');
            child_process.exec('echo '+gpio11 + ' >
/sys/devices/virtual/misc/gpio/pin/gpio11');
            child_process.exec('echo '+gpio12 + ' >
/sys/devices/virtual/misc/gpio/pin/gpio12');
            child_process.exec('echo '+gpio13 + ' >
/sys/devices/virtual/misc/gpio/pin/gpio13');
        }
    });
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.end();
}
if(pathname=="/eprom.c"){ //Requete AJAX pour communiquer avec l'eprom
    console.log('Eprom');
}

```

```

var params = querystring.parse(url.parse(request.url).query);
var content="";
console.log(params);
switch(params.cmd){
    case 'readid':
        var fpath='Scripts\\DS28.exe'; break;
    case 'readadr':
        var fpath='Scripts\\DS28.exe ' + params.addr + ' 8'; break;
    case 'writeadr':
        var fpath='Scripts\\DS28.exe ' + params.addr + ' ' +
params.val; break;
    }
    var workerProcess = child_process.exec(fpath,function (error, stdout,
stderr) {
        if (error) {
            console.log(error.stack);
            console.log('Error code: '+error.code);
            console.log('Signal received: '+error.signal);
        }
        else {
            response.writeHead(200, {'Content-Type': 'text/html'});
            response.write(stdout);
            response.end();
        }
    });
}
// Lecture du fichier a partir du stockage serveur
if(pathname=='/test.htm' || pathname=='/index.htm' ||
pathname=='/Styles/style.css' || pathname=='/Scripts/jquery.min.js' ||
    pathname=='/tpot.htm' || pathname=='/Scripts/bootstrap.min.js' ||
pathname=='/Styles/bootstrap.min.css' || pathname=='/Scripts/popper.min.js' ||
    pathname=='/aop.htm' ){
    fs.readFile(pathname.substr(1), function (err, data) {
        if (err) {
            console.log(err);
            // HTTP Status: 404 : NOT FOUND
            // Content Type: text/plain
            response.writeHead(404, {'Content-Type': 'text/html'});
        }
        else {
            //Fichier trouver HTTP Status: 200 : OK
            response.writeHead(200, {'Content-Type': 'text/html'});
            //Renvoyer le fichier demander
            response.write(data.toString());
        }
        // Send the response body
        response.end();
    });
}
}

```

```

else if (pathname=='/Img/aop.png' || pathname=='/Img/tpot.png' ||
pathname=='/Img/world_circuit.jpg' || pathname=='/Img/OSC.png' ||
        pathname=='/Img/Test.jpg' || pathname=='/Img/Tpol.jpg'){
    fs.readFile(pathname.substr(1), function (err, data) {
        if (err) {
            console.log(err);
            // HTTP Status: 404 : NOT FOUND
            response.writeHead(404, {'Content-Type': 'text/html'});
        }
        else {
            response.writeHead(200, {'Content-Type': 'image/gif'});
            response.end(data, 'binary');
        }
    });
}

}).listen(8080);
console.log("Serveur à l'écoute sur le port 8080");

```

Annexe 5: Code source du potentiomètre numérique

Pd-spi-arg.py :

```

import spidev
import sys

spi = spidev.SpiDev() # creation d'objet spi
spi.open(0,0)
spi.max_speed_hz = 100000
spi.cshigh = False # CS activier a l'etat bas
spi.bits_per_word = 8
spi.mode = 0b00

print "argv 1 val :"+sys.argv[1]
print "argv 2 pot :"+sys.argv[2]

if sys.argv[2]=="pot0":
    pot=0
if sys.argv[2]=="pot1":
    pot=16
try:
    n=sys.argv[1]
    spi.xfer2([int(pot), int(n)]) #transferer la valeur n vers le
potoniometre pot
finally:
    spi.close()

```

Annexe 6: Code source de l'EEPROM 1W

Write.c :

```
#include <core.h>
#include <stdio.h>
#include <OneWire.h>
#include <stdlib.h>
#include <string>

OneWire ds(0); // OneWire bus on digital pin 0

byte i;          boolean present;          uint8_t
adr,len;
byte data[14];  // container for the data from device

void setup() {

    if (argc < 3) { // Il faut 2 arguments
        fprintf(stderr, "%s", "Invalid arguments\n");
        exit (1);
    }
    adr = (uint8_t)strtol(argv[1], NULL, 16); // adr pour l'ecriture

    if (adr < 0 || adr > 127 ) { // Tester si l'adresse est valide
        fprintf(stderr, "%s", "Address out of user's memory range\n");
        exit (1);
    }

    std::string arg2(argv[2]); // Données a ecrire
    len=arg2.length(); // la longueur des données

    if (len != 8) {
        fprintf(stderr, "%s", "Les données doivent etre formater en 8 Octets\n");
        exit (1);
    }

    present = ds.reset();
    if (present == true) {
        ds.write(0xCC); // skip rom
        ds.write(0x0F); // Commande pour ecrire dans la Scratchpad
        ds.write(adr); // LSB first
        ds.write(0x00);
        for ( i = 0; i < len ; i++) { ds.write( uint8_t(argv[2][i])); } // Ecriture
des données
        data[12]= ds.read(); // lecture du CRC
        data[13]= ds.read();
        present = ds.reset(); // OneWire bus reset
        if (present == true) {
            ds.write(0xCC, 1); //skip rom
            ds.write(0xAA, 1); // Commande pour lire Scratchpad
        }
    }
}
```

```

ds.read_bytes(data, len+3);
}

present = ds.reset();           // OneWire bus reset
if (present == true) {         // check if the device is present
    ds.write(0xCC);             // Skip rom
    ds.write(0x55);             // Commande pour copier Copy Scratchpad
    ds.write(adr);              // Adress LSB first
    ds.write(0x00);
    ds.write(data[2]);          // E/S register
    delay(10); // delay
    data[14]=ds.read();        // Lire Le feedback
}
}
if(data[14]='AA') {printf("passed");}
else printf("Failed");
exit(0);
}
void loop() {}

```

Read.c:

```

#include <core.h>
#include <stdio.h>
#include <OneWire.h>
#include <stdlib.h>

OneWire ds(0);

byte i;
boolean present;
byte data[8];
uint8_t adr,len;

void setup() {
if (argc < 3) {
    fprintf(stderr, "%s", "Invalid arguments\n");
    exit (1);
}
    adr = (uint8_t)strtol(argv[1], NULL, 16);

    if (adr < 0 || adr > 127 ) {
        fprintf(stderr, "%s", "Address out of user's memory range\n");
        exit (1);
    }

    len = (uint8_t)strtol(argv[2], NULL, 10);

    if (len > 8 ) {

```



```
fprintf(stderr, "%s", "Can't read more than 8 bytes at once\n");
exit (1);
}

present = ds.reset();
if (present == true) {
ds.write(0xCC);          //skip rom
ds.write(0xF0);         // Commande pour Read Memory

ds.write(adr); // Adress LSB first
ds.write(0x00);
ds.read_bytes(data,8);  // Lecture a partire du memoire
for ( i = 0; i < len; i++) { printf("%c",data[i]); } //Printout the data
printf("\n");
}
else {
printf("Reading failed or nothing connected");
}
exit(0);
}
void loop() {}
```