République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A. MIRA-BEJAIA

Faculté de la technologie

Département génie électrique

MEMOIRE DE FIN D'ETUDE

En vue de l'obtention du diplôme

Master en Automatique

Spécialité : Automatique et systèmes

Thème

Distributeur de boissons à base d'Arduino

Présenté par :

Mr Bourenane Lyes

Mr Latbi Boussaad

Encadré par :

Mr Hadji Slimane

Membres du jury :

Mr Sadji Mustapha

Mme Mezzah Samia

Année universitaire :2018/2019

Remerciements

Avant tout nous tenons nos remerciements à notre Dieu de nous avoir donné la force et le courage de mener à terme ce modeste travail.

A la suite nous remercions très sincèrement Mr S. HADJI notre promoteur, pour ses conseils pertinents, ses orientations judicieuses, sa patience et aussi pour ses suggestions qui nous ont grandement facilité le travail.

Nos remerciements s'adressent aussi à toute l'équipe de DENZER Technologies, en particulier Monsieur LADDI Samir.

Nous tenons à exprimer notre gratitude aux membres de jury qui ont bien voulu examiner notre travail.

Nous tenons également à remercier tous les enseignants et tous ceux à qui nous devons notre formation.

Nous remercions vivement toutes les personnes qui ont contribué de près ou de loin, à la réalisation de ce modeste travail.

Dédicaces

Je dédie ce modeste travail
à :
Mon père et ma Mère pour
leurs soutiens et leurs
sacrifices, qui ont sacrifié
leur vie pour ma réussite et
m'ont éclairé le chemin par
leurs conseils judicieux. Que
Dieu leurs procure bonne
santé et longue vie.
Mon frère
Mes grands parents
Mes oncles et tantes
Mes cousins et cousines
Ma famille et mes proches
Mes amis et collèques.

Lyes

Dédicaces

Je dédie ce modeste travail à :
Mes deux parents, c'est grâce à eux que j'ai pu arriver là où j'en suis actuellement.
Ma réussite est le fruit de leurs soutiens et sacrifices durant toute leur vie, je ne pourrai jamais les remercier assez.
Mes frères et sœurs pour leurs précieux conseils.
Mes cousins et cousines.
Mes ami(e)s, qui sont ma deuxième famille autant qu'ils sont.
Mes collègues, avec qui j'ai pu partager cette expérience durant tout ce parcours.

Boussaad

SOMMAIRE

Table de matière

Liste des figures

Liste des tableaux

INTRODUCTION GENERALE

Introduction générale

1. Généralités

L'électronique embarquée est de plus en plus utilisée dans notre quotidien, elle est utilisée sur de nombreux objets comme les téléphones, les agendas électroniques et les voitures. L'objectif de l'utilisation des systèmes embarqués est de donner aux objets usuels la possibilité de réagir à l'environnement.

Les distributeurs automatiques de boissons sont maintenant bien connus du grand public. Ils sont en effet présents dans de nombreux lieux publics comme par exemple dans des grandes surfaces, aéroports, gares, parkings, écoles et entreprises etc. Ces machines délivrent automatiquement, après payement, plusieurs variétés de produits.

La question posée est comment réaliser un distributeur de boissons à base de l'électronique embarquée ?

2. Les étapes du projet

Dans ce projet, trois objectifs ont été visés :

- Le premier est de regrouper suffisamment d'informations sur l'Arduino, sa constitution et les modules associés.
- Le deuxième est de faire une étude approfondie sur le fonctionnement, le branchement et la programmation des modules à utiliser.
- Le troisième est de réalisé le produit (distributeur de boissons) et le tester.

3. Présentation du mémoire

Pour structurer notre travail, on a consacré :

- Le premier chapitre à la présentation de l'entreprise DENZER Technologies ses missions, ses valeurs et ainsi que ses services.

- Le deuxième chapitre aux généralités sur l'Arduino, là où on mettra la lumière sur le hardware et le software de l'Arduino, et on parlera sur les différents composants qui constituent la carte ainsi que leurs rôles, puis on parlera de l'IDE, méthode de programmation, compilation, …etc. Et à la fin on va citer quelques modules de l'Arduino d'une manière globale.
- Le troisième chapitre à l'étude théorique (c'est le chapitre le plus important), on commencera par la présentation du distributeur de boissons, son principe de fonctionnement puis on va étudier chaque module utilisé, on donnera une définition du module, son branchement et un programme simple pour le tester, ensuite on illustra le schéma de branchement reliant tous les modules utilisés et finir par le programme principal.
- Le quatrième chapitre à la réalisation et aux tests pratiques.
- Enfin, on terminera avec une conclusion générale sur le travail réalisé.

CHAPITRE I : PRESENTATION DE L'ENTREPRISE DENZER TECHNOLOGIES

Chapitre I. Présentation de l'entreprise DENZER Technologies

I.1. Introduction

DENZER Technologies est une jeune entreprise innovante fondé par Dr LADDI Samir en 2016, spécialisé dans le domaine du développement électronique et informatique.

L'entreprise possède deux unités, une unité de recherche et développement situé à la cité Aouchiche, Route de Boukhiama, Bejaia, et un siège social à Akbou (Boulevard Aissat Idir, Akbou, Bejaia).

I.2. Missions et valeurs

Les missions de DENZER Technologies s'inscrivent dans le cadre du développement de l'économie nationale hors hydrocarbures.

La mission fondamentale de l'entreprise consiste à développer de nouvelles techniques informatiques et électroniques exploitées dans les différents secteurs d'activité (Industrie, agriculture et services).

Pour pallier à l'importation, l'entreprise s'investie sur plusieurs pistes de production, elle accueillit d'avantage toute nouvelle idée de production substitutive aux produits importés.

Les valeurs de DENZER Technologies sont :

- Favoriser l'intérêt général sur l'intérêt individuel.
- Favoriser la sous-traitance algérienne sur l'étrangère.
- Innover sans cesse.
- Valoriser les compétences algériennes/estudiantines.
- Encourager l'esprit de collaboration pluridisciplinaire.
- Dynamiser la recherche scientifique.
- Embrasser les nouvelles idées innovantes et rentables.[1]

I.3. Les services

L'entreprise réalise et produit des systèmes électroniques complets visant à mieux commander des machines, des appareils et des objets domestiques ou professionnels, dans les usines industrielles, fermes agricoles et établissements administratifs; à titre d'exemple : des armoires électriques pour la commande des ascenseurs, un système de gestion de file d'attente, des bracelets électroniques pour le suivi des malades Alzheimer, système de réservation à distance et de gestion de parkings, machine à commande numérique (CNC), imprimante 3D pour le prototypage rapide, drones (non commercialisable en Algérie).

Plusieurs systèmes sont en cours de développement, il s'agit de dongles pour la transmission automatique des SMS, un système de sécurité et d'alarme, un système de géolocalisation des engins et des véhicules, des chargeurs de batteries de véhicules ainsi que celles des PC portables et des téléphones mobiles.

I.3.1. Réalisation et maintenance industrielle

DENZER Technologies présente un volet de maintenance, elle s'est engagée, dès sa fondation dans la maintenance industrielle électronique et électrotechnique.

Elle effectue avant tout la maintenance de ses produits et la mise en service du matériel acheté par ses clients, elle intervient dans le cas de pannes au niveau des chaines de production industrielle, et réalise des armoires électriques comme l'armoire « DENZER AscE4 » de commande des ascenseurs qui est réalisé par l'équipe d'ingénieurs électroniciens et électrotechniciens.

Figure I.1. L'armoire « DENZER AscE4 »

I.3.2. Installation des ascenseurs

Elle assure l'installation des différents modèles d'ascenseurs, quelle que soit la technologie utilisée, et un service d'entretien périodique pour le bon fonctionnement et la survie de l'ascenseur. Quant aux immeubles dotés de vieux ascenseurs (en marche ou en panne), elle propose ses services de rénovation, de mise aux normes et de modernisation.

I.3.3. Solutions technologiques

DENZER Technologies engage ses équipes dans la recherche des solutions les plus optimales qui répondront aux besoins des clients. Et travail pour développer des outils pas couteux, qui permettent aux clients d'optimiser leurs méthodes de travail, la rentabilité et l'efficacité.

En agriculture, DENZER propose l'automatisation des outils et machines de travail, et leurs commandes à distance, tout en mettant en place des systèmes de control et d'alerte, pour réduire les besoins de main d'œuvre, réduire le temps d'arrêt et augmenter le rendement.

En industrie, elle propose aussi l'automatisation des outils et des machines qui sont habituellement pilotés par l'humain. Et mettre en place des mécanismes de commande à distance, de control et d'alerte, et aide à la conception des logiciels simples pour la commande, le control et la gestion des usines.

I.3.4. Prototypage rapide et impression 3D

Le prototypage rapide permet de créer physiquement un objet 3D à partir d'une modélisation informatique. Cette technique permet l'obtention, en quelques heures, des modèles en résine, en plastique ou en toute autre matière.

Pour imprimer un objet en 3D, il faudra un fichier au format STL, l'équipe d'ingénieurs de DENZER peuvent concevoir des dessins des idées et des besoins des clients pour pouvoir l'imprimer grâce aux imprimantes 3D avec une extrême haute résolution.

Figure I.2. Imprimante 3D

I.3.5. Développement de logiciels

DENZER développe des logiciels répondant aux besoins croissants des entreprises dans l'automatisation des tâches de gestion, de management et de collaboration.

Elle assure le développement des logiciels :

- Dans plusieurs langages de programmation (Delphi, C++, VB, Java…),
- Avec plusieurs systèmes de bases de données (MS SQL Server, MySQL, Access, SQLite).

Figure I.3. Développement de logiciels

I.3.6. Conception des applications mobile

DENZER développe des applications mobiles qui permettent d'ouvrir un large et très performant canal de communication avec les clients, réalise aussi des applications performantes et faciles à manipuler, pour tous les secteurs d'activités et pour tout usage.

Fourni aussi des fonctionnalités permettant de commander des machines et appareils à distance et en temps réel.

I.4. Données sur l'entreprise

- Raison sociale : EURL DENZER TECHNOLOGIES.
- Forme juridique : Entreprise Unipersonnelle à Responsabilité Limitée (EURL).
- Adresse du siège social : Boulevard Aissat Idir, Akbou, Wilaya de Béjaia.
- Adresse de l'Unité Recherche et Développement : Cité Aouchiche, Route de Boukhiama, Béjaia.
- Fondateur : Dr LADDI Samir.
- Gérant : Dr LADDI Samir.
- Capital social : 1.000.000,00 DA soit un million de dinar algérien.[2]

CHAPITRE II : GENERALITES SUR L'ARDUINO

Chapitre II. Généralités sur l'Arduino

II.1. Introduction

Aujourd'hui, l'électronique câblée est de plus en plus remplacée par de l'électronique programmée. On parle aussi de système embarqué, son but est de simplifier les schémas électroniques et par conséquent réduire l'utilisation des composants électroniques, réduisant ainsi le coût de fabrication d'un produit. Il en résulte des systèmes plus complexes et performants pour un espace réduit. [3]

Un microcontrôleur est un petit ordinateur confiné dans un unique circuit intégré (une puce). Il constitue un excellent moyen pour programmer et pour contrôler des équipements électroniques. Il existe une grande variété de cartes à microcontrôleur, certaines des plus utilisées sont la platine Wiring, le PIC et bien sûr Arduino, dans notre travail on s'intéresse à l'Arduino pour sa simplicité, ses performances et sa disponibilité.

II.2. Définition d'Arduino

Arduino est un circuit imprimé, il englobe des composants le plus important est le microcontrôleur qui est le cerveau de la carte, programmé pour tester et produire des signaux électriques de manière à effectuer plusieurs tâches.

L'Arduino est utilisé dans beaucoup d'applications comme l'électrotechnique industrielle, mais aussi dans des domaines différents comme le pilotage d'un robot, commande des moteurs, faire des jeux de lumières, communiquer avec l'ordinateur et commander des appareils mobiles. Chaque module d'Arduino possède un régulateur de tension et un oscillateur. Pour programmer cette carte, on utilise l'logiciel IDE Arduino.

II.3. Les différentes cartes Arduino

Il n'existe pas de modèle unique de carte Arduino ; on trouve plusieurs variations, chacune étant conçue pour convenir à différents usages. Choisir la bonne carte n'est pas toujours aisé, car le nombre des cartes s'accroît sans cesse.

Toutefois, une carte peut être considérée comme la pierre angulaire de toute aventure dans le monde Arduino, il existe des différentes cartes d'Arduino on site :

II.3.1. Arduino Uno R3

La carte Arduino Uno est basée sur un ATMega328 cadencé à 16 MHz. C'est la plus récente et la plus économique carte à microcontrôleur d'Arduino. Des connecteurs situés sur les bords extérieurs du circuit imprimé permettent d'enficher une série de modules complémentaires, la carte Uno est le choix de prédilection d'un débutant, peu chère et facile à utiliser c'est celle que l'on conseille le plus souvent à ceux qui souhaitent se lancer dans l'aventure Arduino. [4]

Figure II.1 Arduino Uno R3

II.3.2. Arduino Leonardo

Leonardo est l'une des toutes dernières cartes de la gamme Arduino officielle. Elle adopte la même empreinte que Uno, mais le microcontrôleur utilisé est différent, ce qui lui permet de reconnaître un clavier ou une souris d'ordinateur.

Figure II.2. Arduino Leonardo

II.3.3. Arduino Méga 2560

Comme son nom le suggère, Méga 2560 est une carte plus grande que l'Uno. Elle est destinée à ceux qui en veulent : plus d'entrées, plus de sorties, et plus de puissance de calcul. Le Méga dispose de 54 broches numériques et de 16 broches analogiques, alors que l'Uno n'aligne que 13 broches numériques et 6 broches analogiques.

Figure II.3. Arduino Mega 2560

II.3.4. Arduino Nano 3.0

L'Arduino Nano est un condensé d'Arduino Uno qui ne mesure que 1,85 cm sur 4,3 cm. Ces dimensions sont parfaites pour réduire celles d'un projet. La Nano a toute la puissance de l'Arduino Uno, puisqu'il utilise le même microcontrôleur ATmega328, mais ne fait qu'une fraction de sa taille. Il tient à merveille sur une platine d'essai, ce qui le rend idéal pour le prototypage.

Figure II.4 Arduino Nano 3.0

II.3.5. Arduino Mini R5

Contrairement à ce que son nom suggère, l'Arduino Mini est plus petit que la Nano. Cette carte utilise aussi le microcontrôleur ATmega328, mais elle est plus concentrée, les connecteurs externes et le connecteur Mini-USB du Nano disparaissant. Elle est parfaitement indiquée si l'espace est pour vous un enjeu, mais il faut la manipuler avec soin lorsqu'on la connecte, car une connexion incorrecte peut facilement la détruire.

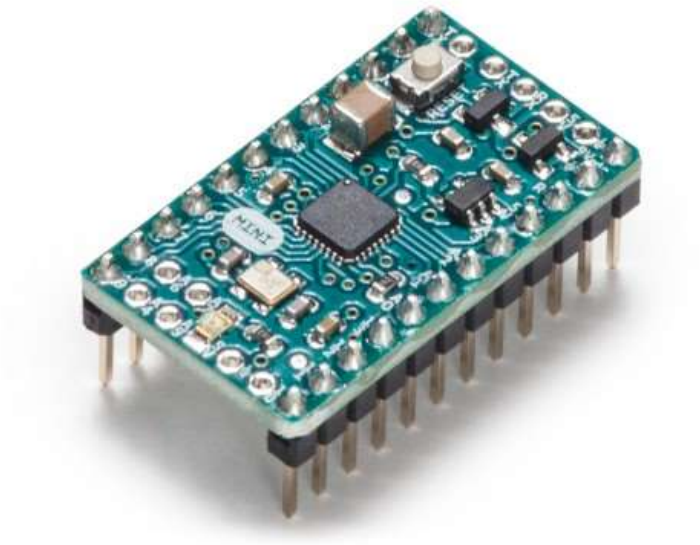


Figure II.5 Arduino Mini R5

Dans notre projet on s'intéresse à la carte Arduino Méga 2560.

II.4. Pourquoi Arduino Méga 2560 ?

Il existe plusieurs modèles de cartes, le choix dépend de plusieurs critères de sélection dont le développeur doit tenir compte :

- Type du microcontrôleur ;
- Nombre d'entrées/sorties ;
- Conversion analogique numérique et numérique analogique ;
- Les logiciels de programmation (assembleur, c, micro…) ;
- Les évolutions prévisibles du composant, son prix, les sources.

La Méga est une carte composée de bien plus d'entrées/sorties (16 analogiques et 54 digitales pouvant fournir jusqu'à 20mA). Mais ses 70 I/O (au total) ne sont pas ses seuls avantages, la Méga 2560 dispose de 256Kb de mémoire flash. De plus cette carte dispose de ports série matériel (UART). Mais pour accueillir ses 70 I/O il faut de l'espace donc la carte mesure 101mm*53mm.[8]

La Méga dispose des mêmes connecteurs USB et DC 2.5mm que la UNO, et les Shields (extensions venant s'ajouter au-dessus de votre carte) compatibles avec la UNO sont compatibles avec la Méga.

II.5. La constitution de la carte Arduino Méga

Le système Arduino est composé de deux parties principales : matériel et logiciel.

II.5.1. Partie matérielle

La carte comporte, outre le microcontrôleur, un certain nombre de petits composants, comme les représente la figure II.6 suivante :

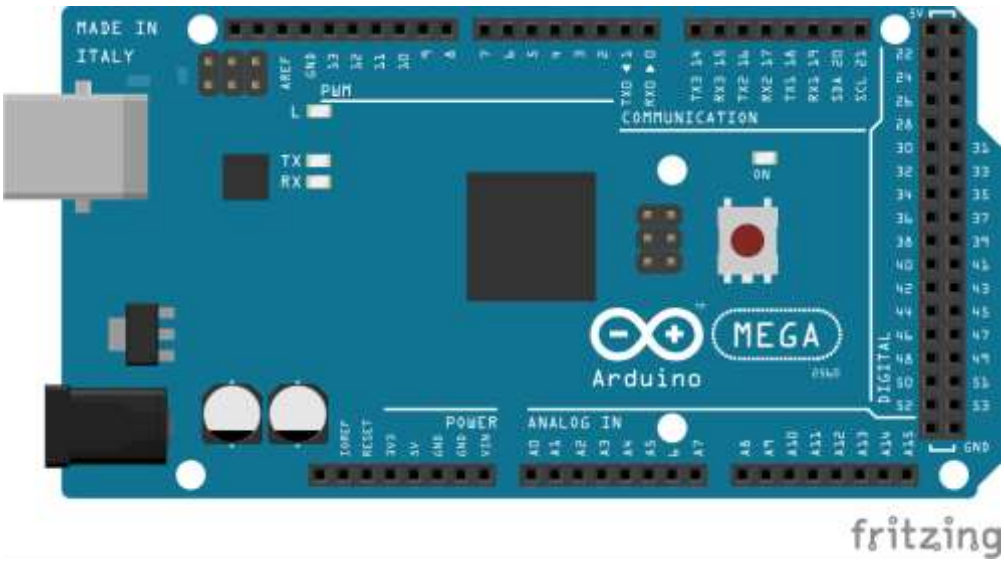


Figure II.6 Constitution de la carte Arduino Méga

Elle est dotée :

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

II.5.1.1. Le Microcontrôleur ATMega2560

Le Atmel ATMEGA2560-16AU est un microcontrôleur 8 bits CMOS basse puissance basée sur l'architecture RISC améliorée par AVR. En exécutant des instructions puissantes en un seul cycle d'horloge, le ATMEGA2560-16AU atteint des débits approchant les 1MIPS par MHz permettant aux concepteurs de système d'optimiser la consommation d'énergie par rapport à la vitesse de traitement., il serait tel que montré en Figure II.7:

Figure II.7. Microcontrôleur ATmega2560

Globalement, l'architecture interne de ce circuit programmable se compose essentiellement sur :

- La mémoire Flash : C'est celle qui contiendra le programme à exécuter. Cette mémoire est effaçable et réinscriptible mémoire programme de 256 KB.

- RAM : c'est la mémoire dite "vive", elle va contenir les variables du programme. Elle est dite "volatile" car elle s'efface si on coupe l'alimentation du microcontrôleur.

- EEPROM : C'est le disque dur du microcontrôleur. On y enregistre des informations qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme.

- Le registre : c'est un type de mémoire utilisé par le processeur.

- La mémoire cache : c'est une mémoire qui fait la liaison entre les registres et la RAM.

II.5.1.2. Les entrées / sorties numériques

Chacune des 54 broches numériques (numérotée de 0 à 53) sur la Mega2560 peut être utilisé comme une entrée ou une sortie, en utilisant pinMode() , digitalWrite() et digitalRead() fonctions. Ils fonctionnent à 5 volts. Chaque broche peut fournir ou recevoir 20 mA en état de fonctionnement. Un maximum de 40mA est la valeur qui ne doit pas être dépassée pour éviter des dommages permanents au microcontrôleur.

II.5.1.3. Les entrées / sorties analogiques

La carte Mega2560 dispose de 16 entrées analogiques, chacune pouvant fournir une mesure d'une résolution de 10 bits (c-à-d sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead() du langage Arduino. Par défaut, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023). Les broches analogiques peuvent être utilisées en tant que broches numériques.

Figure II.8. Les entrées / sorties analogiques

II.5.1.4. Autres broches

Reset : Mettre cette broche au niveau BAS entraîne la réinitialisation (le redémarrage) du microcontrôleur. Typiquement, cette broche est utilisée pour ajouter un bouton de réinitialisation sur le circuit qui bloque celui présent sur la carte.

AREF : Tension de référence pour les entrées analogiques (si différent du 5V). Utilisée avec l'instruction analogReference().

II.5.1.5. Les tensions de références

La carte Arduino Méga 2560 peut être alimentée soit via la connexion USB (qui fournit 5V jusqu'à 500mA) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte.

Figure II.9. Les tensions de références

Les broches d'alimentation sont les suivantes :

GND est la référence de la carte Arduino par rapport à laquelle toutes les différences de tension sont mesurées. Si la carte est reliée à l'ordinateur par un câble USB, cette tension est celle de la terre.

Les ports **5V** et **3V3** donnent accès aux tensions de 5 V et de 3.3 V. Ces tensions sont normalement régulées et précises. Une exception : quand la carte est branchée sur un port USB sans alimentation externe, le port 5 V ne provient plus de la carte Arduino mais directement du câble USB, la tension de référence 5 V n'est alors plus aussi bien régulée.

VIN est la tension de l'alimentation externe, quand il y en a une.[5]

II.5.1.6. Le port USB

Le port USB permet à la fois l'alimentation de la carte Arduino et la communication série entre la carte et l'ordinateur. Une fois connectée, la carte Arduino apparaît dans le gestionnaire de matériel de votre ordinateur, connecté à un port série (COM1, COM4, …).

Figure II.10. Port USB

II.5.2. Partie logicielle

Une carte qui se base sur un microcontrôleur doit être dotée d'une interface de programmation comme est le cas de notre carte. L'environnement de programmation open-source pour Arduino peut être téléchargé gratuitement (pour Mac OS X, Windows, et Linux) par le lien suivant : https://www.arduino.cc/en/main/software.

Le logiciel Arduino a pour fonctions principales :

- De pouvoir écrire et compiler des programmes pour la carte Arduino
- De se connecter avec la carte Arduino pour y transférer les programmes
- De communiquer avec la carte Arduino

Cet espace de développement intégré (IDE) dédié au langage Arduino et à la programmation des cartes Arduino, comporte :

Une **BARRE DE MENUS** : comme pour tout logiciel une interface graphique (GUI),

Une **BARRE DE BOUTONS** : qui donne un accès direct aux fonctions essentielles du logiciel et fait toute sa simplicité d'utilisation,

Un **EDITEUR** : pour écrire le code de vos programmes, avec onglets de navigation,

Une **ZONE DE MESSAGES** : qui affiche indique l'état des actions en cours,

Une **CONSOLE TEXTE** : qui affiche les messages concernant le résultat de la compilation du programme.

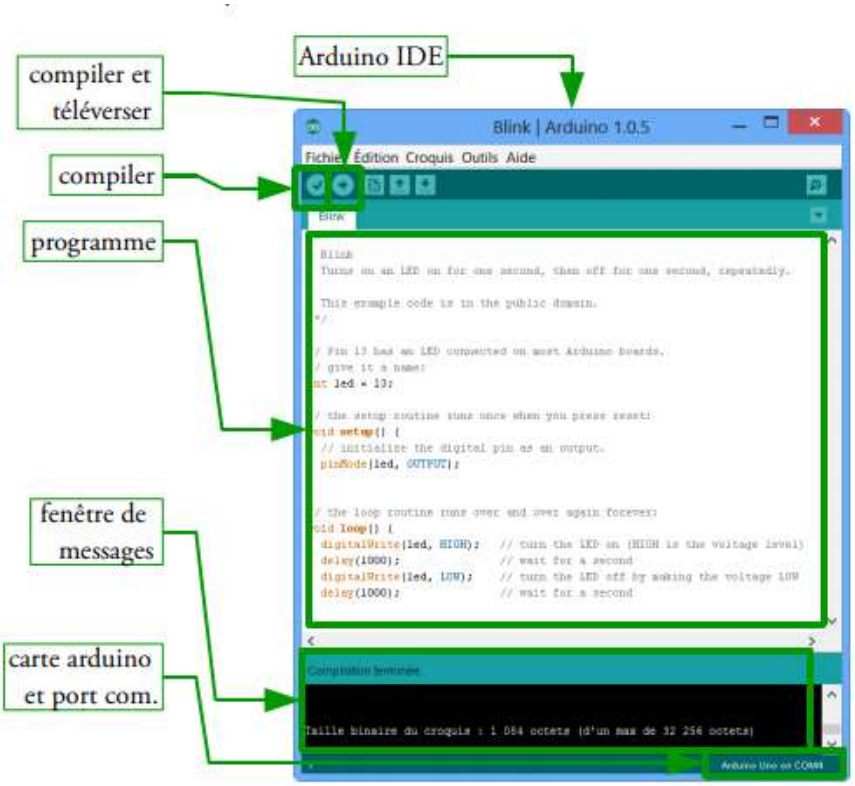


Figure II.11. Interface IDE Arduino

Le logiciel Arduino intègre également :

Un **TERMINAL SERIE** (fenêtre séparée) qui permet d'afficher des messages textes reçus de la carte Arduino et d'envoyer des caractères vers la carte Arduino. Cette fonctionnalité permet une mise au point facilitée des programmes, permettant d'afficher sur l'ordinateur l'état de variables, de résultats de calculs ou de conversions analogique-numérique : un élément essentiel pour améliorer, tester et corriger ses programmes.

Le code écrit avec le logiciel Arduino est appelé un programme (ou une séquence - sketch en anglais) :

Ces programmes sont écrits dans **l'éditeur de texte**. Celui-ci a les fonctionnalités usuelles de copier/coller et de rechercher/remplacer le texte.

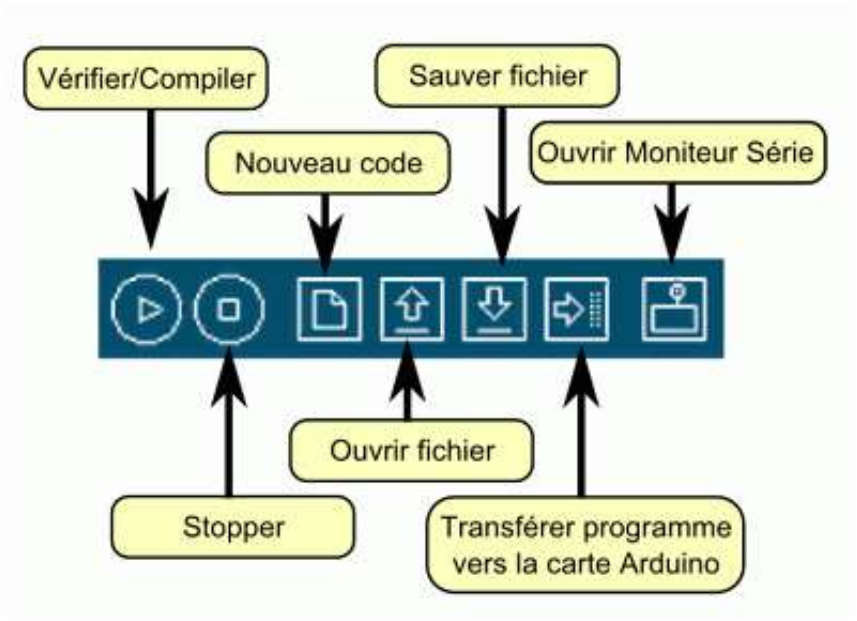
La **zone de messages** donne l'état de l'opération en cours lors des sauvegardes, des exportations et affiche également les erreurs.

La **console texte** affiche les messages produits par le logiciel Arduino incluant des messages d'erreur détaillés et autres informations utiles.

La **barre de boutons** vous permet de vérifier la syntaxe et de transférer les programmes, créer, ouvrir et sauver votre code, et ouvrir le moniteur série.

La barre des menus vous permet d'accéder à toutes les fonctionnalités du logiciel Arduino.

II.5.2.1. Description de la barre des boutons



[6]

Figure II.12. Description de la barre des boutons

II.5.2.2. Transfert des programmes vers la carte Arduino

- **Saisir un programme et vérifier le code.**

On suppose ici qu'un programme correctement écrit se trouve dans la fenêtre éditeur. Pour votre première programmation de la carte, aller dans le menu **File>Examples>Digital>Blink** : un programme s'ouvre avec du code dans la fenêtre éditeur.

On appuye alors sur le bouton **Verify** de la barre d'outils pour lancer la vérification du code.

Si tout va bien, aucun message d'erreur ne doit apparaître dans la console et la zone de message doit afficher **Done Compiling** attestant que la vérification s'est bien déroulée.

- **Sélectionner le bon port série (et la bonne carte Arduino...).**

Avant de transférer votre programme vers la carte Arduino, il faut vérifier que la bonne carte Arduino est bien sélectionné depuis le menu **Tools>Board** (Outils>Carte). Les cartes sont décrites ci-dessous. Votre carte doit évidemment être connectée à l'ordinateur via un câble USB.

On sélectionne le bon port série depuis le menu **Tools > Serial Port** (Outils > Port Série) :

Sur un Mac, le port série ressemble probablement à quelque chose comme /dev/tty.usbserial-1B1 (pour une carte USB), ou /dev/tty.USA19QW1b1P1.1 (pour une carte série connectée avec un adaptateur USB-vers-Série).

Sous Windows, c'est probablement COM1 ou COM2 (pour une carte série) ou COM4, COM5, COM7 ou supérieur (pour une carte USB) - pour trouver le bon, on cherche dans la rubrique des ports série USB dans la section des ports du panneau de configuration ou du gestionnaire de périphériques.

Sous Linux, çà devrait être /dev/ttyUSB0, /dev/ttyUSB1 ou équivalent.

- **Clic sur le bouton upload (transfert du programme)**

Une fois que vous avez sélectionné le bon port série et la bonne carte Arduino, on clique sur le bouton **UPLOAD** (Transfert vers la carte) dans la barre d'outils, ou bien on sélectionne le menu **File>Upload to I/O board** (Fichier > Transférer vers la carte). Avec les versions récentes (Duemilanove notamment), la carte Arduino va alors automatiquement se réinitialiser et démarrer le transfert. Avec les versions précédentes qui ne sont pas équipées de l'auto-réinitialisation, on appuye sur le bouton "reset" de la carte juste avant de démarrer le transfert.

Une fois le transfert terminé, le logiciel Arduino doit afficher un message indiquant que le transfert est bien réalisé, ou montrer des messages d'erreurs.

En transférant un programme, avec le bootloader Arduino, un petit programme (code binaire) a été chargé dans le microcontrôleur sur la carte Arduino. Cette technique permet de transférer le programme sans aucun matériel externe. Une fois le transfert terminé, le bootloader est actif une petite seconde, une fois que la carte est réinitialisée à la fin du transfert, puis le dernier programme programmé dans la carte s'exécute.

II.6. Les modules de la carte Arduino

La carte Arduino est généralement associée aux accessoires qui simplifient les réalisations.

II.6.1. Servomoteur

Le servomoteur est un moteur capable de maintenir une opposition à un effort statique et dont la position est vérifiée en continu et corrigée en fonction de la mesure, il est capable d'attendre des positions prédéterminées dans les instructions qui lui on était donné, puis de les maintenir. Le servomoteur a l'avantage d'être asservi en position angulaire, cela signifie que l'axe de sortie du servomoteur respectera la consigne d'instruction que vous envoyée a l'entrée.

Figure II.13. Servomoteur

II.6.2. Lecteur carte SD

Le module de carte Arduino Micro SD est un appareil basé sur la communication SPI. Peut être utilisé pour fournir une sorte de stockage externe aux projets basés sur un microcontrôleur et un microprocesseur, afin de stocker différents types de données, des images ou vidéos. Les cartes SD sont généralement des périphériques de niveau logique 3.3v, mais à l'aide du module de carte Micro SD, les signaux sont convertis en 5v via un convertisseur de niveau logique implémenté sur le module de carte SD.

Figure II.14. Module lecteur carte SD

II.6.3. Afficheur 7 segments

Les afficheurs 7 segments sont des afficheurs numériques composés de segments généralement appelé A B C D E F et G que l'on va pouvoir allumer ou non afin d'afficher différentes choses, sont particulièrement présent sur les calculatrices et les montres à affichage numérique.

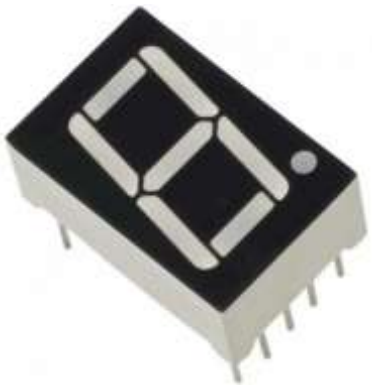


Figure II-15 Afficheur 7 segments

II.6.4. Capteur de température LM35

Le capteur de température LM35 est un capteur analogique de température, il est extrêmement populaire en électronique, car précis, peu couteux, très simple d'utilisation et d'une fiabilité à toute épreuve. Le capteur de température LM35 est capable de mesurer des températures allant de -55°C à +150°C dans sa version la plus précise et avec le montage adéquat, de quoi mesurer n'importe quelle température.

La sortie analogique du capteur est proportionnelle à la température. Il suffit de mesurer la tension en sortie du capteur pour en déduire la température. Chaque degré Celsius correspond à une tension de +10mV.[7]

Figure II.15. Capteur de température LM35

II.6.5. Module RTC

Le module DS1307 de Maxim Integrated est une horloge temps réel (aussi appelé "RTC", "Real Time Clock"). C'est une horloge numérique autonome qui donne l'heure quand on la lui demande. Ce genre d'horloge est très utile dans des projets de mesure de grandeurs physiques.

Ce module RTC est capable de gérer l'heure (heures, minutes, secondes) et la date (jours, mois, année) tout en s'occupant des mois de 30 ou 31 jours, des années bissextiles, etc. Le calendrier intégré dans le module DS1307 est valable de l'an 2000 à l'an 2100, ce qui devrait être suffisant pour la plupart des projets.

Figure II.16. Module RTC

II.7. Conclusion

Dans ce chapitre, on a donné un aperçu général sur la carte Arduino puis on a illustré les différents types de des cartes, donnant ainsi les raisons pour lesquelles on a choisi la carte Arduino-méga. Ensuite, nous avons expliqué les deux parties essentielles de l'Arduino, (la partie matérielle et la partie de programmation), à la fin on a présenté quelques modules qui peuvent être associés à la carte.

Le prochain chapitre sera consacré à l'étude théorique de notre système, principe de fonctionnement, algorithmes et programmation.

CHAPITRE III :
ETUDE THEORIQUE

Chapitre III. Etude théorique

III.1. Introduction

Dans ce chapitre, on présentera de manière sommaire une vue d'ensemble du dispositif expérimental « réalisation d'un distributeur de boissons à base d'Arduino ».

Après avoir donné dans le chapitre précédent une description théorique sur le module Arduino et son environnement de développement, on va procéder à l'application expérimentale, pour cette raison, plusieurs blocs ont été nécessaires afin de réaliser une telle combinaison.

Nous allons commencer par le cahier de charge associé à notre projet, ensuite on teste les différents modules utilisés et à la fin on combine tout.

III.2. Description du distributeur

Le distributeur est une machine qui permet d'obtenir des biens sans l'intervention de l'homme grâce à l'automatique, notre distributeur est divisé en deux parties :

La première partie qui est subdivisée en 4 étages, chaque étage contient un panier subdivisé en 9 couloirs, chaque couloir est inséré une bobine attachée sur son bout antérieur à un moteur pas-à-pas. Et c'est à l'intérieur de cette bobine qu'on insère nos produits, devant chaque couloir est collée une étiquette portant le prix et le code du produit.

La deuxième partie est séparée du reste pour servir au mécanisme électronique de commande du distributeur, là où on trouve un clavier attaché au mur extérieur, un afficheur LCD et un lecteur de badge RFID.

Au-dessous de la première partie il est réservé un compartiment dans lequel les produits tombent afin qu'ils soient récupérés par le client via une petite fenêtre.

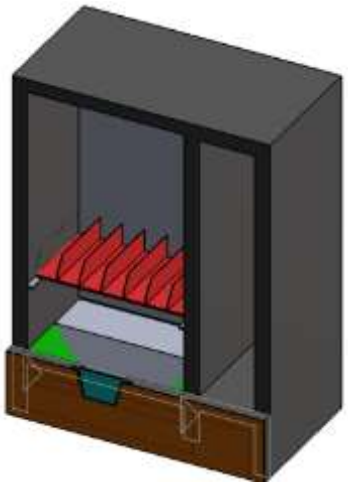


Figure III.1. Modèle du distributeur de boissons

III.3. Principe de fonctionnement

Au début, l'afficheur affiche « Bienvenue insérer votre carte SVP ! », une fois que le client insère sa carte dans le coin qui porte le lecteur RFID, l'Arduino fait une recherche si le code de la carte est sauvegardé dans la base de données, elle sera identifiée .Une fois la carte est identifiée ,l'Arduino affiche via l'afficheur LCD « Bienvenue » suivi du nom du client et affiche le solde du client ,et affiche aussi « Veuillez choisir un produit » , pour choisir un produit, il faut taper le code associé au produit via le module clavier. Une fois que le code sera inséré, l'Arduino effectue une recherche dans la base de données afin de tirer le prix du produit, puis il vérifie si le prix du produit est inférieur à la variable « solde », si oui, il soustrait le prix du produit du solde du client puis faire fonctionner le moteur qui fera tourner la bobine pour libérer le produit, et il affiche « Bonne consommation » .Et si le prix du produit et supérieure au solde ,l'afficheur affiche «Votre solde est insuffisant recharger votre carte», et si le code inséré est incorrect l'afficheur affiche « code incorrect taper le code à nouveau ».

III.4. Test des modules

Il est branché à la carte Arduino méga les modules suivants :

- Un afficheur LCD 16*2 pour l'affichage des directives et des différentes informations.
- Un clavier 4*3 pour l'insertion des codes des produits.
- Un lecteur de badge RFID compatible Arduino qui lit les cartes insérées et transmet le code à l'Arduino.
- Cinqs moteurs pas-a-pas piloté par des drivers 8825.

III.4.1. Afficheur LCD

LCD est l'abréviation du terme anglais "**L**iquid **C**rystal **D**isplay" qui signifie en français « Écran à cristaux liquides ». D'où afficheur LCD.[8]

L'afficheur LCD est en particulier une interface visuelle entre un système (projet) et l'homme (utilisateur). Son rôle est de transmettre les informations utiles d'un système à un utilisateur. Il affichera donc des données susceptibles d'être exploiter par l'utilisateur d'un système.

Le branchement est représenté dans la figure suivante :

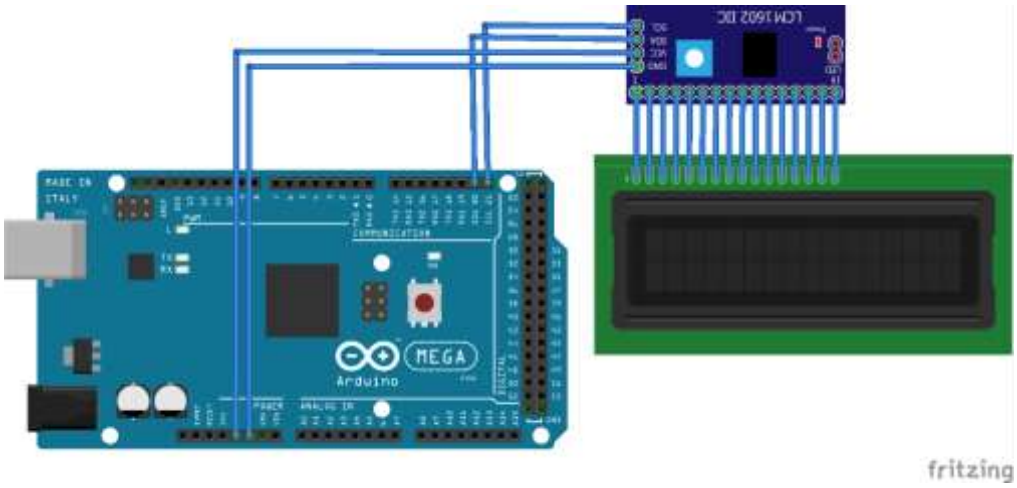


Figure III.2 Branchement de l'afficheur LCD

On connecte d'abord les 16 broches de l'afficheur LCD aux 16 broches du module LCM 1602 IIC qui permet d'économiser les broches sur la carte Arduino.

Puis on suit le branchement suivant :

GND → GND

VCC → 5V

SDA → SDA

SCL → SCL

Pour pouvoir utiliser l'afficheur LCD, la bibliothèque LiquidCrystal_I2C doit être ajouter, ouvrez l'IDE Arduino et aller dans Croquis → Inclure une bibliothèque →LiquidCrystal.

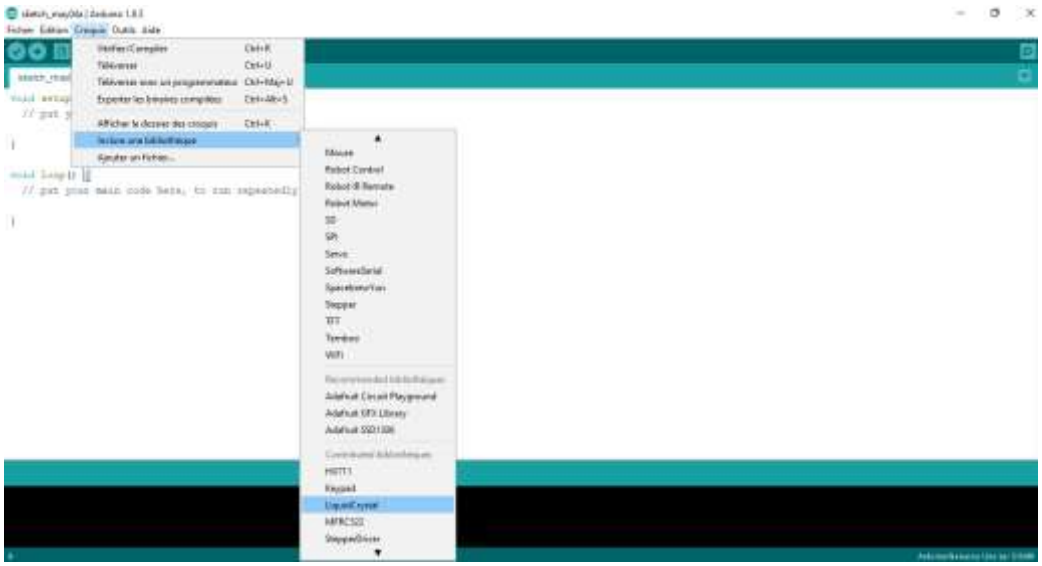


Figure III.3 l'inclusion de la librairie LiquidCrystal

Après avoir ajouté la bibliothèque « LiquidCrystal », on téléverse l'exemple du code représenté dans la figure « Figure III.3» qui initialise l'afficheur et affiche un texte.

Dans le programme on commence par l'inclusion de la librairie de l'afficheur LCD alphanumérique puis on déclare les constantes des broches utilisées dans le programme, ensuite on déclare un objet LCD alphanumérique, au niveau de la fonction setup () on commence par l'initialisation de l'utilisation de l'afficheur LCD alpha-numérique et on le test en affichant le message suivant « Master 2 Automatique ».

```
//Définition des bibliothéques
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
//Definitoin des variables
#define I2C_ADDR          0x27        //Define I2C Address where the PCF8574A is
#define BACKLIGHT_PIN      3
#define En_pin             2
#define Rw_pin             1
#define Rs_pin             0
#define D4_pin             4
#define D5_pin             5
#define D6_pin             6
#define D7_pin             7
//Initialisation de LCD
LiquidCrystal_I2C      lcd(I2C_ADDR, En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

void setup()
 {
    lcd.begin (20,4);
    lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
    lcd.setBacklight(HIGH);
    lcd.setCursor(0,0);
    lcd.print("Master 2");
    lcd.setCursor(0,1);
    lcd.print("Automatique");
void loop(){ }
```

Figure III.4 Le code associé à l'afficheur LCD

III.4.2. Clavier 4*3

Un clavier est l'un des périphériques d'entrée les plus couramment utilisés dans les applications à microcontrôleurs. Sur un clavier standard câblé en tant que matrice de commutateurs X-Y, les commutateurs normalement ouverts connectent une ligne à une colonne lorsqu'ils sont enfoncés. Si un clavier est câblé en 4 colonnes sur 4 lignes on dit que c'est un claver 4*4, un clavier comporte 12 touches, s'il est câblé en 3 colonnes sur 4 lignes, comme illustré dans la figure « Figure III.5 ».

Pour que l'Arduino détermine quel bouton est enfoncé, il doit d'abord tirer sur chacun des boutons, trois colonnes (broches 1 à 3), l'une basse ou l'autre haute, l'une après l'autre, puis interrogez les états des quatre lignes (broches 4 à 7).[9]

La connexion 1 correspond à la colonne 3 (C3)
La connexion 2 correspond à la colonne 2 (C2)
La connexion 3 correspond à la colonne 1 (C1)
La connexion 4 correspond à la ligne 4 (L4)
La connexion 5 correspond à la ligne 3 (L3)
La connexion 6 correspond à la ligne 2 (L2)
La connexion 7 correspond à la ligne 1 (L1)

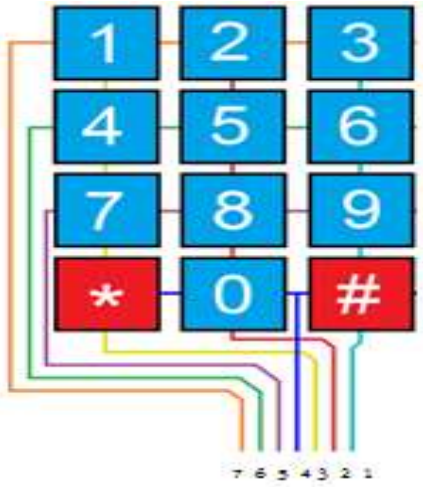


Figure III.5. Les connexions du clavier matriciel 4*3

On raccorde les connexions L1, L2, L3, L4, C1, C2 et C3 aux pins 8 jusqu'à 2 respectivement de l'Arduino.

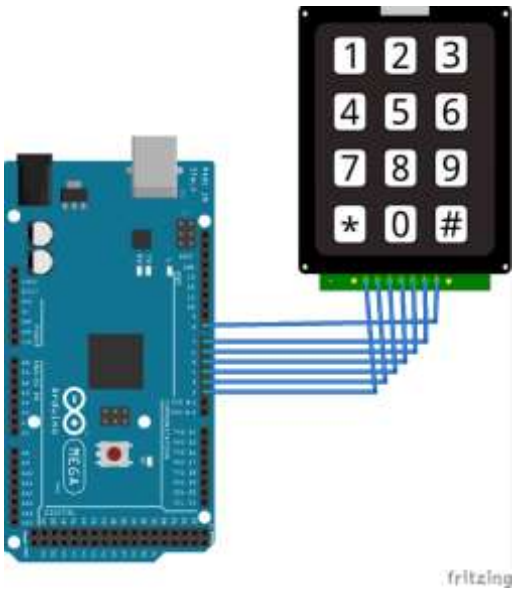


Figure III.6 . Branchement d'un clavier 4*3

Après avoir ajouté la bibliothèque Keypad, on charge le programme suivant, et on ouvre le moniteur série, lorsque on presse une touche, cette dernière est affichée dans le moniteur série.

```
#include <Keypad.h>

int led_pin=13;
const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns
//define the cymbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};
byte rowPins[ROWS] = {8, 7, 6, 5}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {4, 3, 2}; //connect to the column pinouts of the keypad

//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){
  pinMode(led_pin,OUTPUT);
  Serial.begin(9600);
}


void loop(){
  char customKey = customKeypad.getKey();

  if (customKey){
    digitalWrite(led_pin,HIGH);
    Serial.println(customKey);
    delay(100);
  }
   else digitalWrite(led_pin,LOW);
}
```

Figure III.7. Code associé au clavier

Voilà les résultats obtenus.

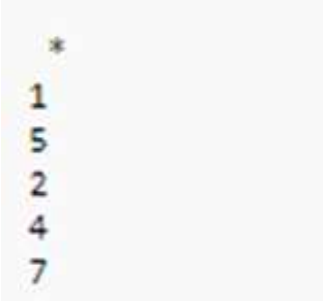


Figure III.8. Résultats obtenus sur le moniteur série

III.4.3. Lecteur RFID

RFID signifie identification par radiofréquence. La RFID utilise des champs électromagnétiques pour transférer des données sur de courtes distances. Le système d'identification par radiofréquence comprend deux composants principaux : une étiquette attaché à un objet à identifier et également appelé lecteur.

Un lecteur se compose d'un module radiofréquence et d'une antenne générant un champ électromagnétique haute fréquence. Par contre, l'étiquette est généralement un périphérique passif, ce qui signifie qu'elle ne contient pas de batterie. Il contient une puce qui stocke et traite les informations, ainsi qu'une antenne pour recevoir et transmettre un signal.[10]

Figure III.9. Module RFID

Le module RC522 a au total de 8 broches qui l'interfèrent avec le monde extérieur. Les connexions sont les suivantes :

VCC : Fournit l'alimentation au module, cela peut être n'importe où de 2,5 à 3,3 volts. Vous pouvez le connecter à la sortie 3.3V de votre Arduino.

RST : Est une entrée pour la réinitialisation et la mise hors tension.

GND : Est la broche de terre et doit être connecté à la broche GND de l'Arduino.

IRG : Est une broche d'interruption pouvant alerter l'Arduino lorsque l'étiquette RFID se trouve à proximité.

MISO : (Master In Slave Out) agit en tant qu'horloge série.

MOSI : (Master Out Slave In) est l'entrée d'Arduino du module RC522.

SCK : Accepte les impulsions d'horloge fournies par le bus de l'Arduino.

SS/SDA : La broche agit comme entrée du signal.

Le branchement des broches MOSI, MISO, SCK et CS change d'une carte a une autre,

Dans la figure suivante on illustre le branchement sur les différentes cartes.

	MOSI	MISO	SCK	CS
Arduino Uno	11	12	13	10
Arduino Nano	11	12	13	10
Arduino Mega	51	50	52	53

Tableau III.1. Configuration des pins sur les cartes Uno,Nano et méga

Dans notre exemple on a suivi le branchement suivant :

MISO → pin 50
MOSI → pin 51
SCK → pin 52
CS → pin 53
3.3V → 3.3V
GND → GND
RST → pin 10

Comme le représente la figure suivante :

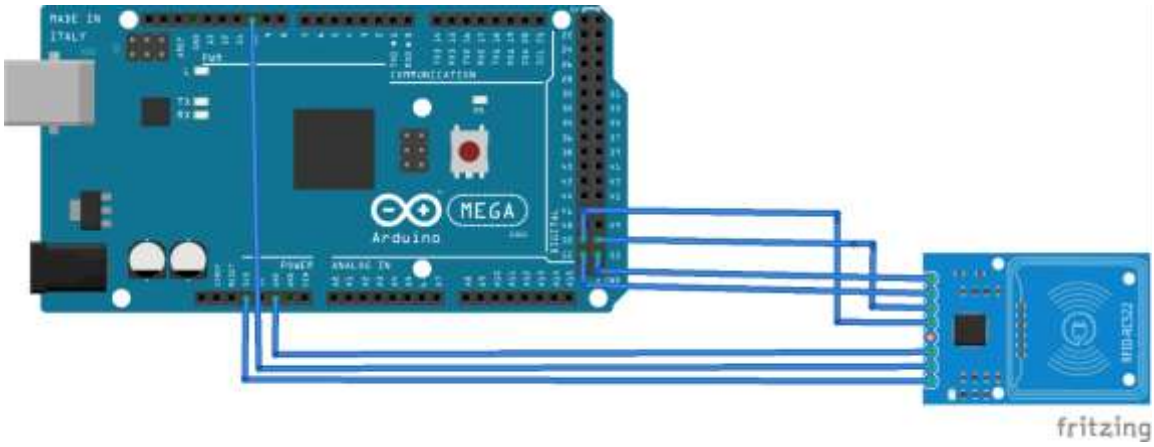


Figure III.10. Branchement du module lecteur RFID

Après avoir ajouté les deux bibliothèques SPI et **MFRC522,** on téléverse le programme suivant et on ouvre le moniteur série qui affichera l'ID du badge.

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 53
#define RST_PIN 10
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.

void setup()
{
  Serial.begin(9600);   // Initiate a serial communication
  SPI.begin();      // Initiate  SPI bus
  pinMode(10, OUTPUT);
  digitalWrite(10, HIGH);
  mfrc522.PCD_Init();   // Initiate MFRC522
  Serial.println("Approximate your card to the reader...");
```

```
}
void loop()
{
// Look for new cards
  while ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  // Select one of the cards
  while ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  String content= "";

  for (byte i = 0; i < mfrc522.uid.size; i++)              // save the RFID ID in a string variable
  {
     content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : ""));
     content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  content.toUpperCase();
  Serial.println(content);
}
```

Figure III-11 Code associé au lecteur RFID

On aura les résultats suivants :

Figure III.11. L'affichage de l'ID du badge

III.4.4. Moteur pas-à-pas

Un moteur pas à pas est constitué de deux parties principales, un rotor et un stator. Le rotor est la partie du moteur qui tourne et fournit du travail. Le stator est la partie fixe du moteur qui abrite le rotor. Dans un moteur pas à pas, le rotor est un aimant permanent. Le stator est constitué de plusieurs bobines qui agissent comme des électroaimants lorsqu'un courant électrique les traverse. La bobine électromagnétique obligera le rotor à s'aligné sur elle lorsqu'il sera chargé.

Figure III.12. Stepper moteur

Les moteurs pas à pas présentent de nombreux avantages. Ils sont peu coûteux et faciles à utiliser. Lorsqu'il n'y a pas de courant envoyé au moteur, les steppers tiennent leur position. Les moteurs pas à pas peuvent également tourner sans limites et changer de direction en fonction de la polarité fournie.

Afin d'utiliser un moteur pas à pas il est nécessaire d'utiliser un "driver de moteur pas à pas". Ces drivers permettent de transmettre la puissance électrique au moteur afin de le faire tourner.

Généralement les drivers de moteurs pas à pas permettent de commander les moteurs en fractions de pas complet. En mode demi pas, le moteur devra alors faire 400 demis pas pour faire un tour complet soit 0.9° par demi pas. Certains drivers permettent de faire du 16ème de pas et même du 128ème de pas, dans notre projet on a utilisé les drivers 8825.

Le drv8825 sont des petits modules qui permettent de contrôler la rotation d'un moteur en fonction des instructions reçues de la carte de pilotage. Ils sont connectés à cette dernière, soit directement, soit par l'intermédiaire d'une interface.

La puce comporte plusieurs fonctions de sécurité intégrées telles que la protection contre les court-circuits et la protection contre la surchauffe.

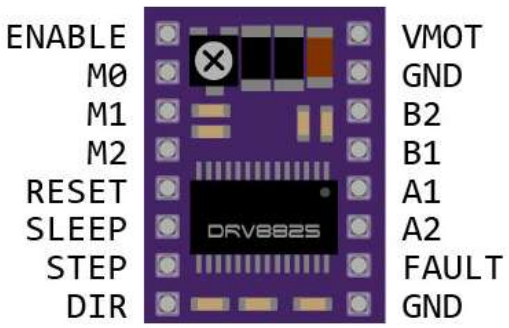


Figure III.13. Le driver drv8825

Le dirver 8825 se compose de 16 broches sont :

Enable : Logique inversée, permet d'activer ou désactivé le moteur.

M0, M1, M2 : Ces broches permettent de choisir une résolution de micro-stepping parmi les 6 disponibles (pas complet, demi pas, 1/4 pas, 1/8 pas, 1/16 pas et 1/32 pas).

Reset : Logique inversée. Permet de faire une réinitialisation du module.

Sleep : Logique inversée. Généralement connecté sur la broche "Reset" du module.

Step : Envoye un signal d'horloge pour avancer le moteur d'un pas.

DIR : Permet d'indiquer la direction de rotation du moteur.

VMot : Tension d'alimentation pour les moteurs pas à pas. Tension entre 8.2 et 45v.

GND : Masse pour l'alimentation moteur.

A1 A2 : Première bobine du moteur pas à pas bipolaire.

B1 B2 : Deuxième bobine du moteur pas à pas bipolaire.

Le schéma de câblage ci-dessus vous montre comment connecter le pilote DRV8825 à un moteur pas à pas et à l'Arduino.

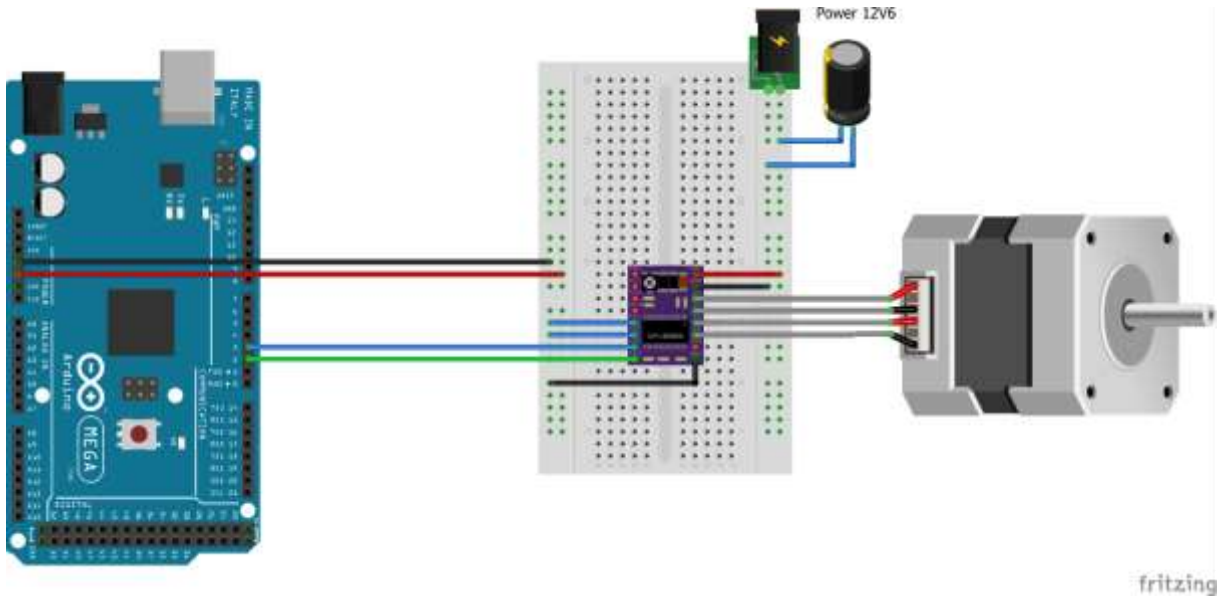


Figure III.14. Schéma de câblage du pilote DRV8825 avec moteur pas à pas et Arduino.

L'alimentation du moteur est connectée à GND et à VMOT (en haut à droite).

Les deux bobines du moteur pas à pas sont connectées à A1, A2 et B1, B2.

La broche GND (en bas à droite) est connectée à la broche de terre de l'Arduino.

Les broches STP (step) et DIR (direction) sont connectées aux broches numériques 3 et 2 respectivement.

Vous devez connecter RST (réinitialisation) et SLP (veille) à 5V, sinon le pilote ne s'allume pas.

La broche EN (activée) peut être laissée déconnectée, elle est tirée vers le bas par défaut. (Lorsque cette broche est élevée, le pilote est désactivé).

Pour protéger le pilote, on a connecté un condensateur entre VMOT et GND. Pololu (drv8825) suggère un condensateur de 47 µF ou plus (on a utilisé un condensateur de 100 µF).

Le code suivant peut-être utiliser pour faire fonctionner un ou plusieurs moteurs pas à pas en continu à une vitesse constante. (Aucune accélération ou décélération n'est utilisée).

```
#include <AccelStepper.h>
//Define stepper motor connections
#define dirPin 2
#define stepPin 3
//Create stepper object
AccelStepper stepper(1, stepPin, dirPin); //motor interface type must be set to 1 when using a driver.
void setup()
{
  stepper.setMaxSpeed(1000); //maximum steps per second
}
void loop()
{
  stepper.setSpeed(400); //steps per second
  stepper.runSpeed(); //step the motor with constant speed as set by setSpeed()
}
```

Figure III.15. Le code associé pour faire fonctionner un moteur pas à pas à une vitesse constante

La première étape consiste à définir les broches de pas et de direction, comme avant. Ensuite, on a créé un objet stepper avec le type d'interface moteur et les broches de pas et de direction appropriée. Comme nous utilisons un pilote, nous avons défini le type d'interface moteur sur 1.

Dans ce cas, on a appelé le moteur pas à pas « stepper », mais on peut également utiliser d'autres noms. AccelStepper liftmotor (1, stepPin, dirPin). On peut créer plusieurs objets de moteur pas à pas avec des noms, des broches de pas et de direction différente. Cela permet de contrôler facilement 2 moteurs pas à pas ou plus en même temps.

Dans la section setup () du code, nous définissons la vitesse maximale du moteur pas à pas en pas / seconde.

Dans la section void loop (), nous définissons d'abord la vitesse à laquelle le moteur doit tourner, puis nous laissons le moteur tourner.[11]

III.5. Schéma de principe

Il est branché sur la carte Arduino un clavier 4*3, un afficheur LCD, un lecteur RFID et 5 moteurs pas à pas soit directement ou avec l'intermédiaire des drivers (drv8825 pour le moteur) ou le module (LCM 1602 IIC pour l'afficheur).

Dans le tableau suivant on représentera le branchement des différentes broches des modules sur les pins de l'Arduino.

	Broche du module	Pin sur l'Arduino
Clavier	C3	2
	C2	3
	C1	4
	L4	5
	L3	6
	L2	7
	L1	8
L'afficheur	GND	GND
	VCC	5V
	SDA	SDA
	SCL	SCL
Lecteur RFID	MISO	50
	MOSI	51
	SCK	52
	CS	53
	3.3	3.3V
	GND	GND
	RST	9
Moteur 1	DIR	32
	STEP	33
Moteur 2	DIR	34
	STEP	35
Moteur 3	DIR	36
	STEP	37
Moteur 4	DIR	38
	STEP	39
Moteur 5	DIR	40
	STEP	41

Tableau III.2 Branchement des différents modules sur les pins de la carte Arduino

Dans la figure suivante on va représenter le schéma de câblage des différents modules utilisé sur la carte Arduino.

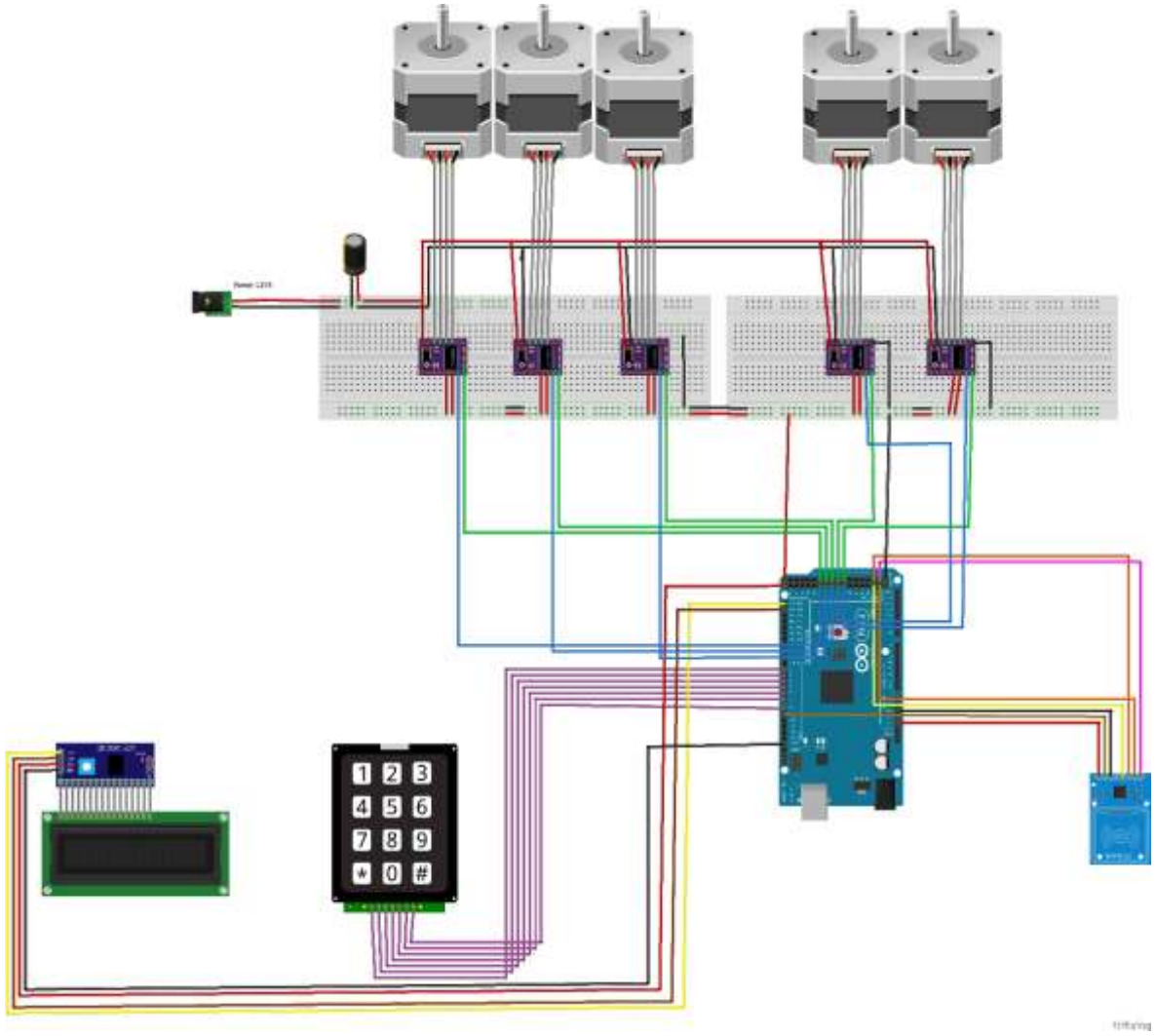


Figure III.16. Schéma de câblage global

III.6. Algorithme

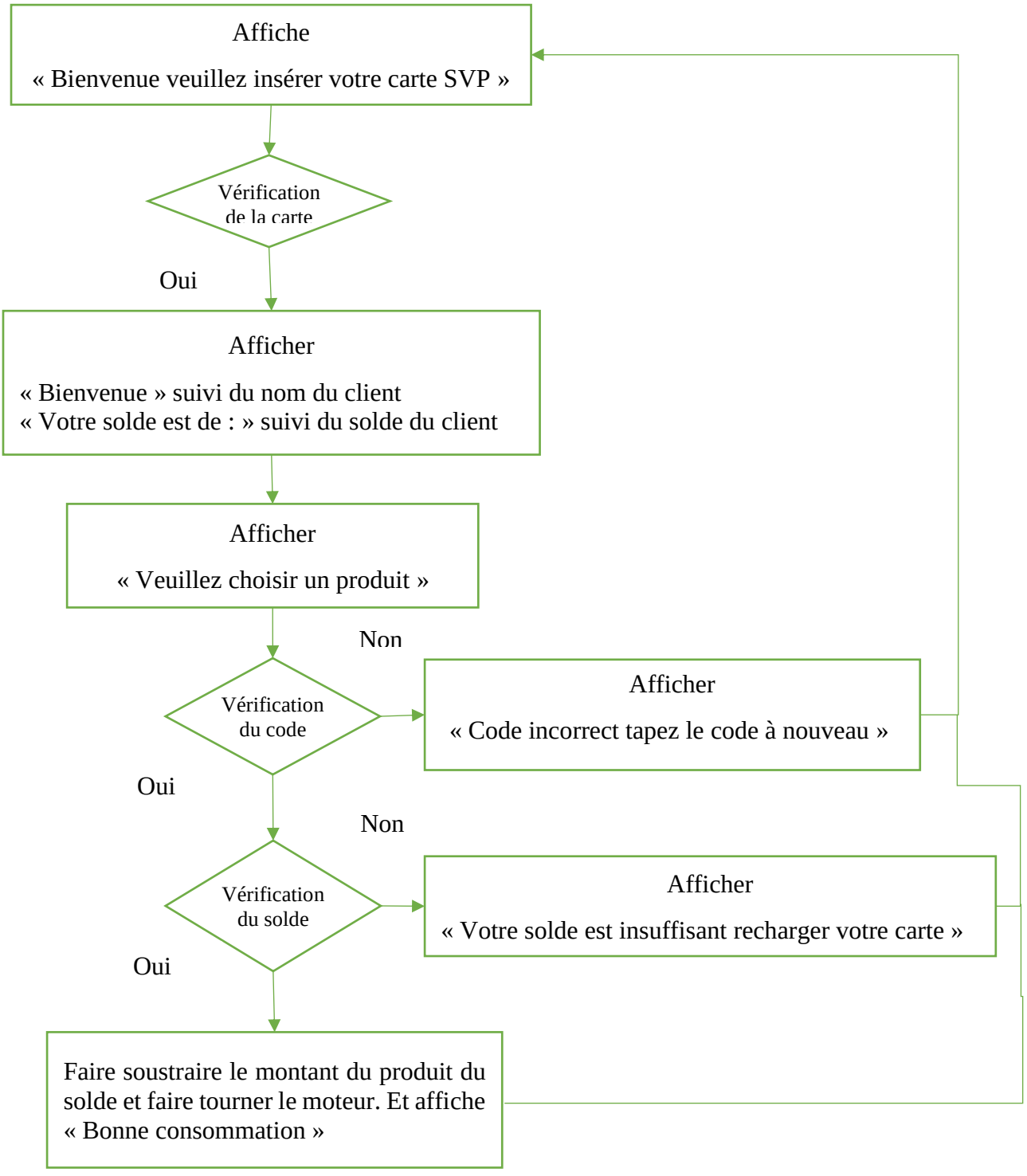


Figure III.17 L'organigramme du programme

III.7. Le programme

Après avoir terminé le câblage on passera à la programmation, le programme principal est présenté dans la section sous dessous.

Ce programme comprend :

- Une entête déclaratif.
- Une partie « configuration » qui ne sera exécutée qu'une fois (fonction setup ( )).
- Une partie constituée d'une boucle sans fin que le programme répètera à l'infini (fonction loop ( )) : c'est le cœur du programme.

Au niveau de la partie déclarative, on inclut les librairies des fonctionnalités utilisées, librairie pour l'afficheur LCD alphanumérique, pour le clavier matriciel, le lecteur RFID et pour les moteurs pas à pas.

```
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <MFRC522.h>
#include <SPI.h>
#include <AccelStepper.h>
```

Ensuite on déclare les variables et les constantes utilisés ; la constante « montant » qui représente le prix des produits, la constante « code » qui représente les codes associés aux produits, la constante « NombreDeClient » qui représente le nombre de client, « nom » le nom du client, « solde » c'est le crédit du client.

```
const int montant[5]={20,25,40,35,30};
const int NombreDeClient=2;
int solde[NombreDeClient]={1000,800};
String nom[NombreDeClient]={"Boussaad","Lyes"};
String mot[NombreDeClient]={"1686D212","01305C79"};
int client;
const int code[5]={111,222,333,444,555};
int sum,a1,a2,a3,a4;
int codeRead = 0;
char m1,m2,m3,m4;
String uidString;
```

Dans cette partie du programme, on déclare les constantes des broches utilisées, les constantes et variables globales utiles et les objets pour chaque module utilisé.

```
//RFID
#define OLED_RESET 4
#define SS_PIN 53
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

//Moteurs
#define dirPin_1 32
#define stepPin_1 33
#define dirPin_2 34
#define stepPin_2 35
#define dirPin_3 36
#define stepPin_3 37
#define dirPin_4 38
#define stepPin_4 39
#define dirPin_5 40
#define stepPin_5 41
#define stepsPerRevolution 200


//Afficheur
#define I2C_ADDR 0x27
#define BACKLIGHT_PIN 3
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
LiquidCrystal_I2C lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

//Clavier
const byte numRows= 4;
const byte numCols= 3;
char keymap[numRows][numCols]=
{
    {'#','0','*'},
    {'9','8','7'},
    {'6','5','4'},
    {'3','2','1'}
};
byte rowPins[numRows] = {2, 3, 4, 5};
byte colPins[numCols]= {6, 7, 8};
Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols);
```

Dans la section setup () du code, toutes les broches de commande du moteur sont déclarées comme OUTPUT, puis on initialise l'afficheur LCD alpha-numérique et le bus du lecteur RFID.

```
void setup(){
  pinMode(stepPin_1, OUTPUT);
  pinMode(dirPin_1, OUTPUT);
  pinMode(stepPin_2, OUTPUT);
  pinMode(dirPin_2, OUTPUT);
  pinMode(stepPin_3, OUTPUT);
  pinMode(dirPin_3, OUTPUT);
  pinMode(stepPin_4, OUTPUT);
  pinMode(dirPin_4, OUTPUT);
  pinMode(stepPin_5, OUTPUT);
  pinMode(dirPin_5, OUTPUT);
  Serial.begin(9600);
  SPI.begin();
  mfrc522.PCD_Init();
   delay(2000);
    lcd.clear();
    lcd.begin (16,2);
    lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
    lcd.setBacklight(HIGH);}
```

Au début l'afficheur affiche « Bienvenue veuillez insérer votre carte svp ! ».

```
void loop()
{
    lcd.setCursor(0,0);
    lcd.print("***Bienvenue***");
    delay(2000);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Veuillez inserer");
    lcd.setCursor(0,1);
    lcd.print("votre carte SVP!");
    delay(1000);
    lcd.clear();
```

Une fois la carte est insérée, cette partie du programme consiste a identifié la présence de la carte la lire et sauvegarde l'adresse ID de la carte dans une variable appelée « content », puis il compare si l'adresse existe déjà, si oui la carte sera identifiée.

```
  while ( ! mfrc522.PICC_IsNewCardPresent()) { return;}
  while ( ! mfrc522.PICC_ReadCardSerial()) { return;}
String content= "";
for (byte i = 0; i < mfrc522.uid.size; i++)
{
   content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : ""));
   content.concat(String(mfrc522.uid.uidByte[i], HEX));
}
content.toUpperCase();
content;
boolean test = false;
for (int NombreDeClient=0; NombreDeClient<2; NombreDeClient++) {
  if (mot[NombreDeClient] == content) {
    test = true;
    client = NombreDeClient;
  }
}
int j;
```

Une fois la carte a été identifiée, on affiche avec l'afficheur « Bienvenue » suivi du nom du client identifié, « votre solde est de : » le solde du client, « veuillez choisir le produit ».

```
if (test){
   lcd.clear();
    lcd.setCursor(0,0);
   lcd.print("Bienvenue");
    lcd.setCursor(0,1);
   lcd.print(nom[client]);
   delay(2000);
   lcd.clear();
   lcd.setCursor(0,0);
   lcd.print("Votre solde est:");
   lcd.setCursor(0,1);
   lcd.print(solde[client]);
   lcd.setCursor(4,3);
   lcd.print("DA");
   delay(5000);
   lcd.clear();
```

```
   lcd.setCursor(0, 0);
   lcd.print("Veuillez Choisir");
   lcd.setCursor(0, 1);
   lcd.print("un produit");
```

Le programme suivant permet de saisir le code du produit et l'affiché.

```
char keypressed = myKeypad.getKey();
  keypressed = myKeypad.waitForKey();
  lcd.clear();
  if (keypressed){
  m1 = keypressed;
  lcd.setCursor(0,0);
  lcd.print(keypressed);}
  keypressed = myKeypad.waitForKey();
  if (keypressed)}
  m2 = keypressed;
  lcd.setCursor(1,0);
  lcd.print(keypressed);}
  keypressed = myKeypad.waitForKey();
  if (keypressed){
  m3 = keypressed;
  lcd.setCursor(2,0);
  lcd.print(keypressed);}
  a1=(m1-48)*100;
  a2=(m2-48)*10;
  a3=(m3-48)*1;
  sum=a1+a2+a3;
  lcd.clear();
```

Dans ce qui suit, Arduino fait une comparaison entre le code saisi et les codes existant, s'il existe déjà il compare une autre fois si le montant du produit et inférieur au solde et fait soustraire le montant du produit dans le solde du client et faire tourner le moteur de 400 pas, et affiche « en cours …… » et « Bonne consommation ». Si le solde est inférieur au montant l'afficheur affiche « votre solde est insuffisant recharger votre carte », et c'est la même syntaxe pour les 5 moteurs.

Si le code n'existe pas, il affiche « code incorrect taper le code à nouveau ».

III.8. Conclusion

Au cours de ce chapitre nous avons essayé de faire une étude théorique sur le travail réaliser.

On a commencé par la description du distributeur de boissons suivis du principe de fonctionnement, ensuite, l'étude et les tests des différents modules utilisés (afficheur LCD, clavier, le lecteur RFID et les moteurs pas à pas) là où on a expliqué les broches de chaque module et leurs branchements sur la carte Arduino et donner un programme simple pour chaque module.

Puis on est passé au schéma principal du branchement et présenter l'algorithme qui explique le fonctionnement du distributeur, et finalement on a donné le programme principal.

Le chapitre suivant sera consacré à la réalisation et aux tests de notre distributeur.

CHAPITRE IV : REALISATION PRATIQUE

Chapitre IV. Réalisation pratique

IV.1. Introduction

Dans ce chapitre, on présentera le dispositif expérimental « réalisation d'un distributeur de boissons ».

Après avoir donné dans les chapitres précédents une description théorique sur la carte Arduino, l'afficheur LCD, clavier, lecteur RFID et les moteurs pas à pas, le branchement de chacun, le schéma de câblage global et le programme.

Dans le chapitre qui se suit, on va procéder au montage, l'assemblage des composants et aux tests pratiques.

IV.2. Les principaux modules du distributeur

Le distributeur est composé de plusieurs composants électriques et mécaniques, de deux parties complémentaires partie de commande (affichage, saisi …) et une partie opérative.

Il se compose principalement :

IV.2.1. Une carte Arduino Méga 2560

Comme on l'a définie précédemment, l'Arduino Méga 2560 est une carte à microcontrôleur basée sur l'ATmega2560. Il possède 70 broches d'entrée / sortie numériques en tout, c'est l'élément de commande.

Caractéristiques:

- Microcontroller : ATmega2560
- Operating Voltage : 5V
- Input Voltage (recommended) : 7-12V
- Input Voltage (limits) : 6-20V
- Digital I/O Pins : 54
- Analog Input Pins : 16
- DC Current per I/O Pin : 40 mA
- DC Current for 3.3V Pin : 50 mA
- Flash Memory : 256 KB

- EEPROM : 4 KB
- Clock Speed : 16 MHz [12]

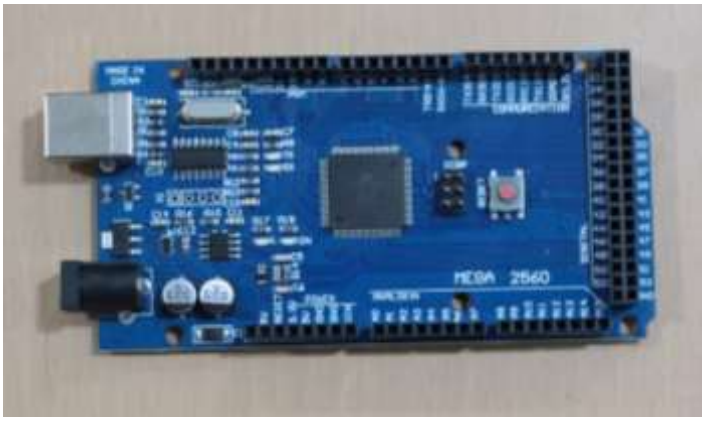


Figure IV.1 Carte Arduino méga 2560

IV.2.2. **Une alimentation 12V 1A**

C'est un bloc alimentation secteur disposant d'une connectique en Jack 2,1mm. Ce bloc alimentation permettra d'alimenter facilement la carte Arduino via le connecteur Jack.

Caractéristiques :

- Connecteur de sortie : Jack 2,1mm x 5,5 mm x 12mm
- Courant de sortie : 1A
- Tension de sortie : 12Vcc

Figure IV.2 Alimentation 12V 1A

IV.2.3. **Une alimentation 12V 30A**

L'alimentation est un élément clé de toute construction, cette alimentation est utilisée pour alimenter les moteurs pas à pas.

Caractéristiques :

- Tension d'entrée : 115V / 230V AC
- Tension d'entrée : 12V DC
- Output Current : 0 ~ 29.2A
- Protection : Protection de surcharge ; Protection de survoltage
- Le poids : 660 g
- Dimension : 215 * 114 * 50mm

Figure IV.3 Alimentation 12V 30A

IV.2.4. Cinq moteurs pas à pas Nema 17

Un moteur pas à pas est un moteur à haute durée de vie, qui se commande comme son nom l'indique, pas par pas, avec précision, ces moteurs ont 200 pas par tour soit 1.8° par pas.

Ce moteur pas à pas est un moteur à quatre fils : Rouge, Bleu, Vert, Noir. Il y a 4 demi-bobines, câblées en parallèles 2 à 2 pour former au final 2 bobines : 2 fils par bobines, les fils "rouge" et "bleu" étant les extrémités d'une bobine, les fils "noir" et "vert" les extrémités de l'autre bobine. Utiliser pour faire tourner les bobines.

Caractéristiques :

- Model Number : SM42HT47-1684A
- Type : Hybrid
- Phase :2
- Tension: 2.8V
- Current / Phase :1.68A

- Couple :5kg.cm
- Angle par pas (dégrées) :1.8 [13]

Figure IV.4 Moteur pas à pas Nema17

IV.2.5. Cinq drivers drv8825

Le Drv8825 est un contrôleur de moteur pas à pas, cette carte est utilisée pour servir d'interface entre le microcontrôleur et le moteur pas à pas.

Le Drv8825 est capable de fournir jusqu'à 2,5 A et peut être contrôlé avec une simple interface étape / direction, le dispositif dispose d'une protection contre la surcharge et les courts circuits.

Caractéristiques :

- Tension de la partie commande (logic) : 3-5.25V
- Tension de la partie opérative (vmot) : 12-24V
- Courant maximal : 2.5A
- Dimensions : 20.4x15.6mm [14]

Figure IV.5 Driver 8825

IV.2.6. Un clavier 4*3

Ce clavier a 12 boutons, disposés dans une grille téléphonique 3x4, les clés sont connectées à une matrice, donc on a besoin que de 7 broches du microcontrôleur (3 colonnes et 4 lignes) pour numériser à travers le clavier. Utiliser pour saisir le code des produits.

Figure IV.6 Clavier 4*3

IV.2.7. Un afficheur LCD 16*2

Les afficheurs LCD sont devenus indispensables dans les systèmes techniques qui nécessitent l'affichage des paramètres de fonctionnement. Ces Afficheurs permettent d'afficher des lettres, des chiffres et quelques caractères spéciaux. Les caractères sont prédéfinis. Utiliser pour afficher des messages aux clients.

Figure IV.7 Afficheur LCD 16*2

IV.2.8. Un lecteur RFID RC522

RFID signifie identification par radiofréquence et peut être utilisé pour de nombreuses applications nécessitant un mécanisme d'identification, tels que les cartes de crédit, Le module RFID-RC522 est un lecteur RFID capable de lire des étiquettes RFID à courte portée. Pour lire une étiquette RFID, le lecteur et l'étiquette doivent avoir la même fréquence. Le module RFID-RC522 ne lit que les étiquettes haute fréquence à 13,56 MHz. Utiliser pour identifier les cartes des clients.

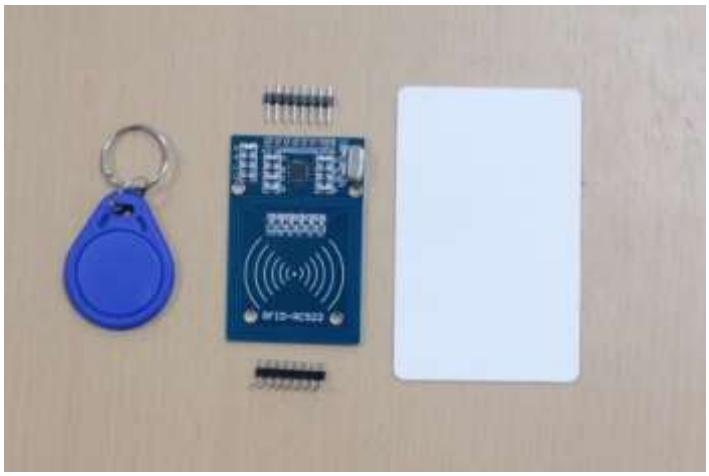


Figure IV.8 Lecteur RFID RC522

IV.2.9. Breadboard

Une platine d'expérimentation ou platine de prototypage en anglais breadboard, est un dispositif qui permet de réaliser le prototype d'un circuit électronique et de le tester L'avantage de ce système est d'être totalement réutilisable, car il ne nécessite pas de soudure, on peut de plus câbler sur une platine d'expérimentation une grande variété de composants afin de réaliser des circuits électroniques.

On l'utilise pour fixer les drivers 8825, l'alimenter et pour faire leurs câblages.

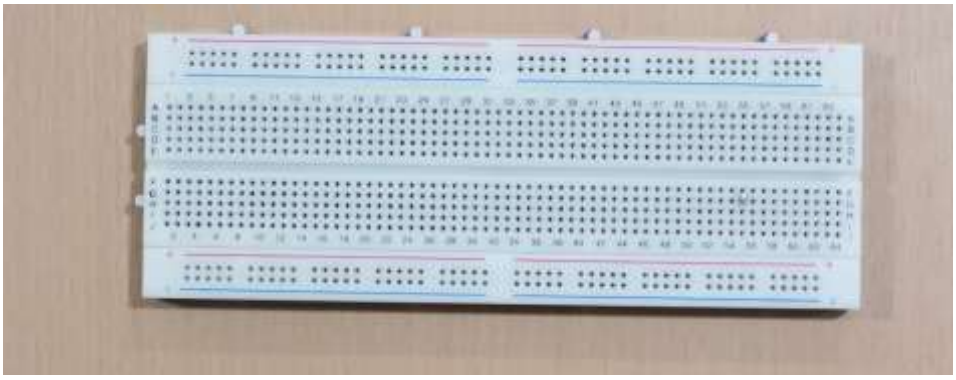


Figure IV.9 Breadboard

IV.3. Description et étapes de réalisations

L'ensemble des dispositifs Arduino exige une alimentation de 5V, pour notre travail on a utilisé une alimentation de 12V 1A pour alimenter l'Arduino. Par contre on a utilisé une alimentation 12V 30A pour alimenter les moteurs.

Initialement, on teste le fonctionnement de la carte Arduino, en connectant cette dernière avec le port USB de PC. Si la LED power s'allume la carte est bonne. Ensuite, on clique sur le bouton Reset de la carte, pour supprimer tout ancien programme et de la réinitialiser.

Maintenant, on connecte les broches des composants suivant la Figure III.16 du chapitre précédent, puis on compile le programme principal présenté aussi dans le Chapitre III sur Arduino IDE, et on téléverse le programme vers la carte pour l'exécution via le câble USB.

IV.4. Tests et résultats

Après avoir terminé le montage des différents modules dans l'armoire, on passe aux tests finals du produit.

Apres l'alimentation du distributeur, l'afficheur LCD affichera le message suivant : « Bienvenue insérer votre badge SVP ! »

Figure IV.10. Affichage du message « Bienvenue insérer votre carte SVP ! »

On insère un badge connu par l'Arduino, puis l'afficheur affiche « Bienvenue » suivi du nom du client identifié et affiche aussi son solde, puis affiche une autre fois le message suivant : « Veuillez choisir un produit ».

Figure IV.11 Affichage du message « bienvenue » et « Veuillez choisir un produit »

Figure IV.12 Affichage du message « Veuillez choisir un produit »

Pour choisir le produit, on doit saisir le code associé au produit via le clavier si le code est correct, le moteur associé au produit choisi, tourne et fait tomber le produit pour que le client puisse le récupérer et affiche « Bonne consommation ».

Figure IV.13 Affichage du message « bonne consommation »

Par contre si le code est incorrect l'afficheur affiche « code incorrect taper le code à nouveau ».

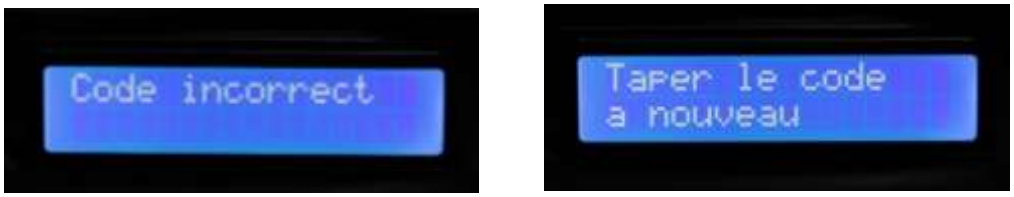


Figure IV.14 Affichage du message « Code incorrect taper le code à nouveau »

Et si le solde du client est inférieur au prix du montant l'afficheur affiche « Votre solde est insuffisant recharger votre solde ».

Figure IV.15 Affichage du message « Votre solde est insuffisant recharger votre carte »

IV.5. Conclusion

Dans ce chapitre on a réussi à réaliser un distributeur de boissons à l'aide de la carte Arduino Méga.

Au début du chapitre on a commencé par la présentation des différents éléments constituant le distributeur, en donnant une brève définition, les caractéristiques. Puis la description et les étapes de réalisations en passant par le câblage et la programmation.

Et enfin la mise en marche du distributeur, aux essais et aux tests des différents cas.

CONCLUSION GENERALE

Conclusion générale

Notre projet comporte un travail théorique accompagné d'une réalisation, son objectif consiste sur l'étude et la réalisation d'un distributeur de boissons. Nous avons utilisé des alimentations, un afficheur LCD, un clavier, un lecteur RFID et des stepper moteurs.

L'ensemble des modules utilisés sont commandés par la carte Arduino programmable. Pour notre cas ; on a utilisé la carte Arduino Méga 2560 dont ses caractéristiques particulières nous ont facilité les taches surtout en ce qui concerne sa programmation.

Notre projet a été fait en deux parties : La première partie est l'étude théorique là où on a testé tous les modules utilisés et fait une étude approfondie sur le branchement des modules et leurs programmations, un schéma global de branchement et le programme principal. La deuxième partie est la réalisation du produit, cette partie est consacrée au montage des modules et des composants et à la mise en marche.

Et on a terminé notre projet par le test et la simulation selon le principe de fonctionnement demandé.

Suite au manque du matériels sur le marché et plus exactement les moteurs pas à pas nema17, on a utilisé que cinq moteurs. Donc le distributeur a toujours besoin d'amélioration et développement.

Pendant notre travail, on a pu développer et approfondir nos connaissances théoriques acquises durant notre formation et d'acquérir une bonne expérience de la réalisation pratique et la programmation. Une expérience professionnelle, et l'esprit de travailler en groupe après la période de stage passé au sein de l'entreprise DENZER Technologies.

Liste des acronymes et abréviations

3D	Trois dimensions
IDE	Integrated Development Environment
USB	Universal Serial Bus
UART	Universal Asynchronous Reiciving Transmiting
I/O	Input/Output
DC	Direct Current
ICSP	In Circuit Serial Programming
CMOS	Complimentary Métal Oxyde Semiconductor
RISC	Reduced Instruction Set Computer
MIPS	Million d'Instruction Par Seconde
EEPROM	Electrically Erasible Programmable Read-Only Memory
RESET	Réinitialiser
AREF	Alimentation de Reference
GND	Ground
LCD	Liquid Crystal Display
RTC	Real Time Clock
RFID	Radio Frequency Identification
SS	Slave Select
SCL	Serial Clock Line
MISO	Master In Slave Out
MOSI	Master Out Slave In
SCK	Serial Clock
CS	Serial Clock Line
RST	Réinitialiser
SDA	Serial Data
DIR	Data Terminal Ready
DRV	Driver
PWM	Pulse Width Modulation (en français modulation par large impulsions)
RAM	Random Access Memory

Références bibliographiques

[3] LAOUAR BACHIR AYACHI AMOR Hamza. Etude et Réalisation d'une Commande Domotique par ARDUINO Via Infrarouge. Master. Electrotechnique Industrielle. UNIVERSITE KASDI MERBAH OUARGLA.2017.

[4] NUSSEY John. « Electronique pour les nuls ». (2 e édition).2017.

[7] https://www.carnetdumaker.net/articles/mesurer-une-temperature-avec-un-capteur-lm35-et-une-carte-arduino-genuino/ .Consulté en mai 2019.

[6] http://www.mon-club-elec.fr/pmwiki_reference_arduino/pmwiki.php?n=Main.DebuterPresentationLogiciel. Consulté en mars 2019.

[2] TAZAMOUCHT Yanis MOUZAOUI Melissa. Réalisation et automatisation d'une machine à commande numérique. Master. AUTOMATISMES INDUSTRIELS. Université A.MIRA-BEJAIA.2018.

[1] http://www.denzertech.com/. Consulté en mai 2019.

[5] https://www.robot-maker.com/ouvrages/tuto-arduino/choisir-carte-arduino-adaptee/. Consulté en mars 2019.

[11] https://www.makerguides.com/drv8825-stepper-motor-driver-arduino-tutorial .Consulté en mai 2019.

[8] https://plaisirarduino.fr/afficheur-lcd-comment-lexploiter/ .Consulté en mai 2019.

[10] https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/ .Consulté en mai 2019.

[9] https://www.parallax.com/sites/default/files/downloads/27899-4x4-Matrix--Keypad-v1.2.pdf. Consulté en mai 2019.

[12] Arduino mega datasheet. http://www.mantech.co.za/datasheet .Consulté en mai 2019.

[13] SM42HT47-1684A datasheet.

[14] Drv8825 datasheet.

ANNEXES

1.8° Nema 17, Size 42mm High Torque Hybrid Stepping Motor

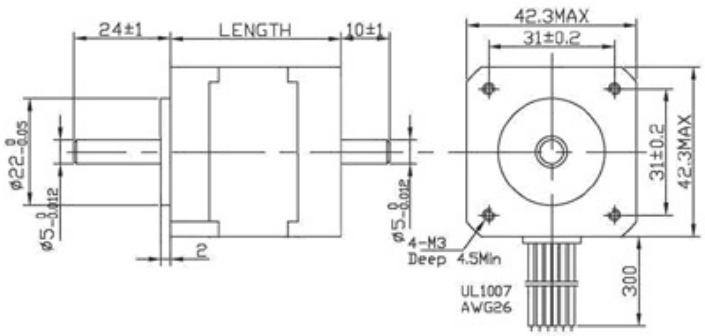
Technique Parameter

Item	Specification
Step Angle Accuracy	±5%（full step,no load）
Resistance Accuracy	±10%
Inductance Accuracy	±20%
Temperature Rise	80℃Max.(rated current,2 phase on)
Ambient Temperature	-10℃-+50℃
Insulation Resistance	100MΩMin.500VDC
Dielectric Strength	500VAC for one minute
Shaft Radial Play	0.06Max.(450 g-load)
Shaft Axial Play	0.08Max.(450 g-load)

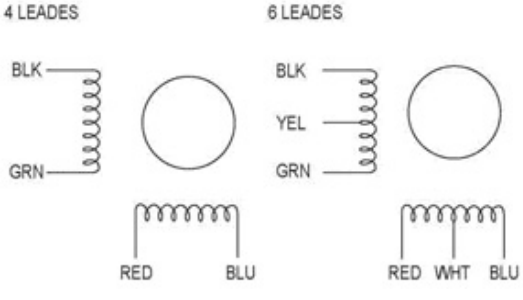
Technique Specification

Model No.		Rated Voltage	Current /Phase	Resistance /Phase	Inductance /Phase	Holding Torque		#Of Leads	Rotor Inertia	Weight	Length
Single Shaft	Double Shaft	V	A	Ω	mH	Oz-in	g-cm		g-cm2	Kg	mm
SM42HT33-0956A	SM42HT33-0956B	4	0.95	4.2	2.5	22	1580	6	35	0.22	33
SM42HT33-0406A	SM42HT33-0406B	9.6	0.4	24	15						
SM42HT33-0316A	SM42HT33-0316B	12	0.31	38.5	21						
SM42HT33-1334A	SM42HT33-1334B	2.8	1.33	2.1	2.5	30	2200	4			
SM42HT38-1206A	SM42HT38-1206B	4	1.2	3.3	3.2	36	2590	6	54	0.28	38
SM42HT38-0806A	SM42HT38-0806B	6	0.8	7.5	6.7						
SM42HT38-0406A	SM42HT38-0406B	12	0.4	30	30						
SM42HT38-1684A	SM42HT38-1684B	2.8	1.68	1.65	3.2	50	3600	4			
SM42HT47-1206A	SM42HT47-1206B	4	1.2	3.3	2.8	44	3170	6	68	0.35	47
SM42HT47-0806A	SM42HT47-0806B	6	0.8	7.5	6.3						
SM42HT47-0406A	SM42HT47-0406B	12	0.4	30	25						
SM42HT47-1684A	SM42HT47-1684B	2.8	1.68	1.65	2.8	62	4400	4			

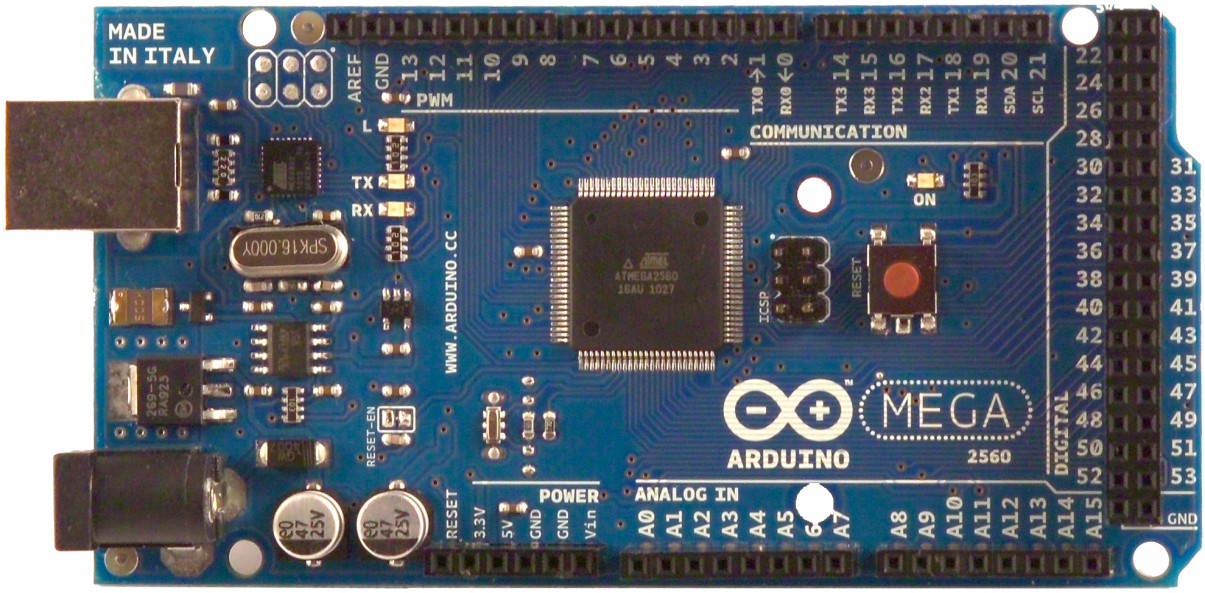
Dimensions



Wiring Diagram



Arduino MEGA 2560



Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

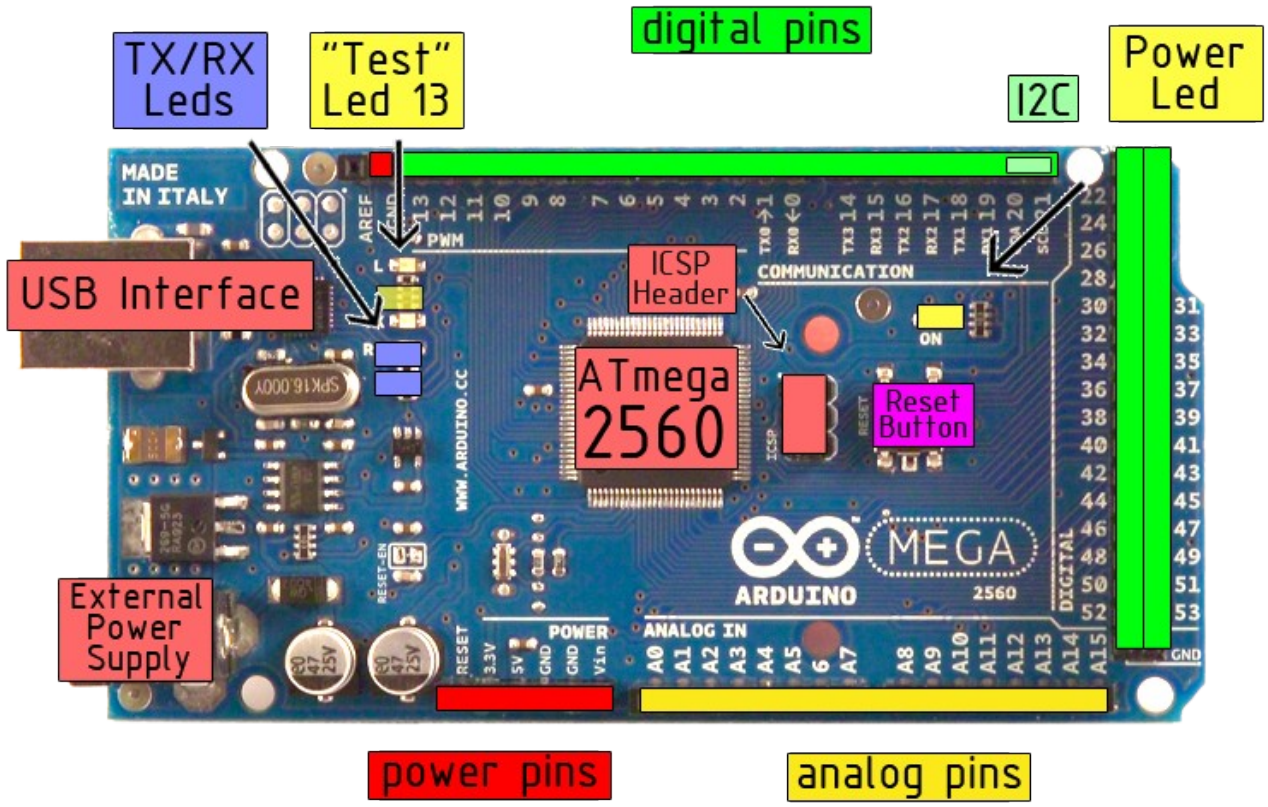
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

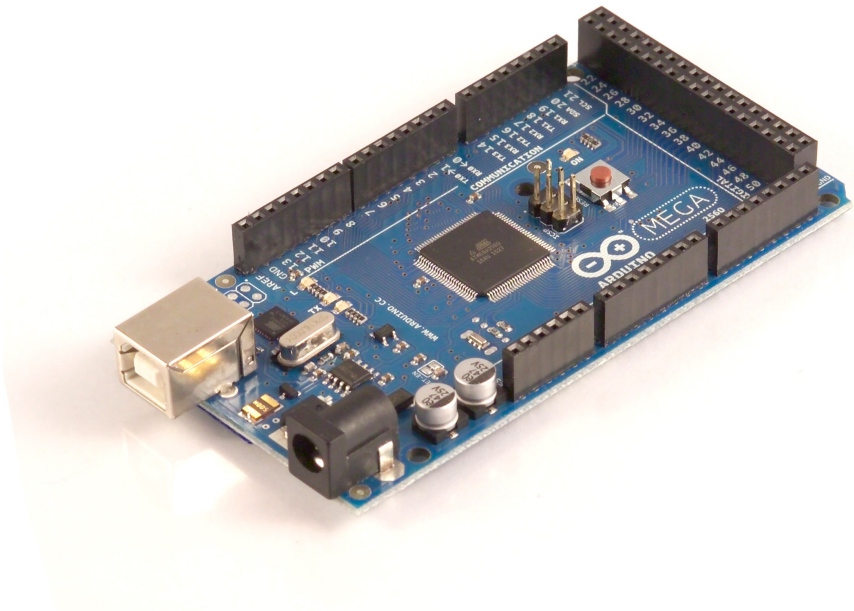
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.



Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

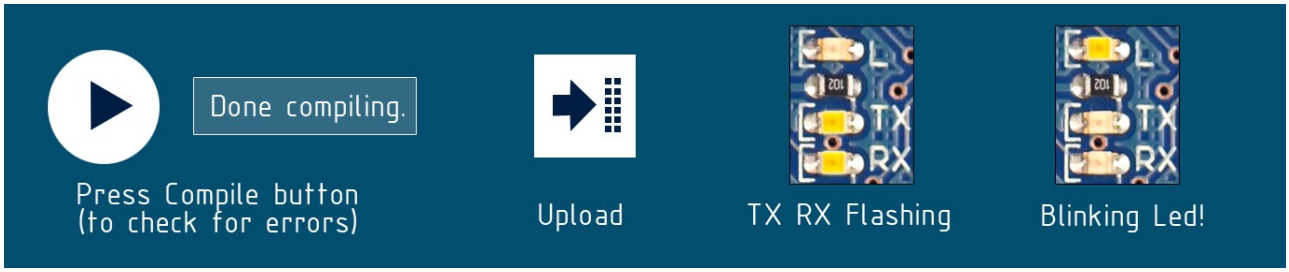
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

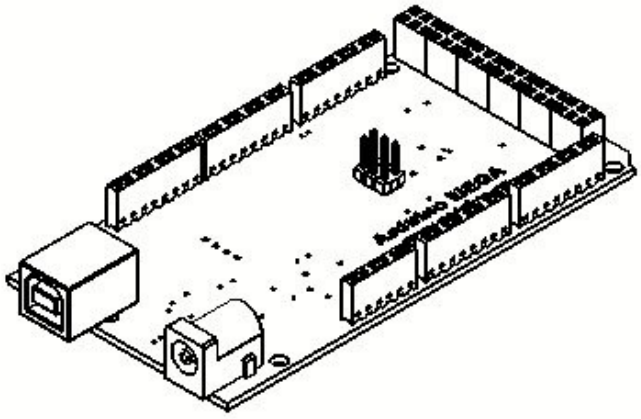
Once you have your skecth you'll see something very close to the screenshot on the right.

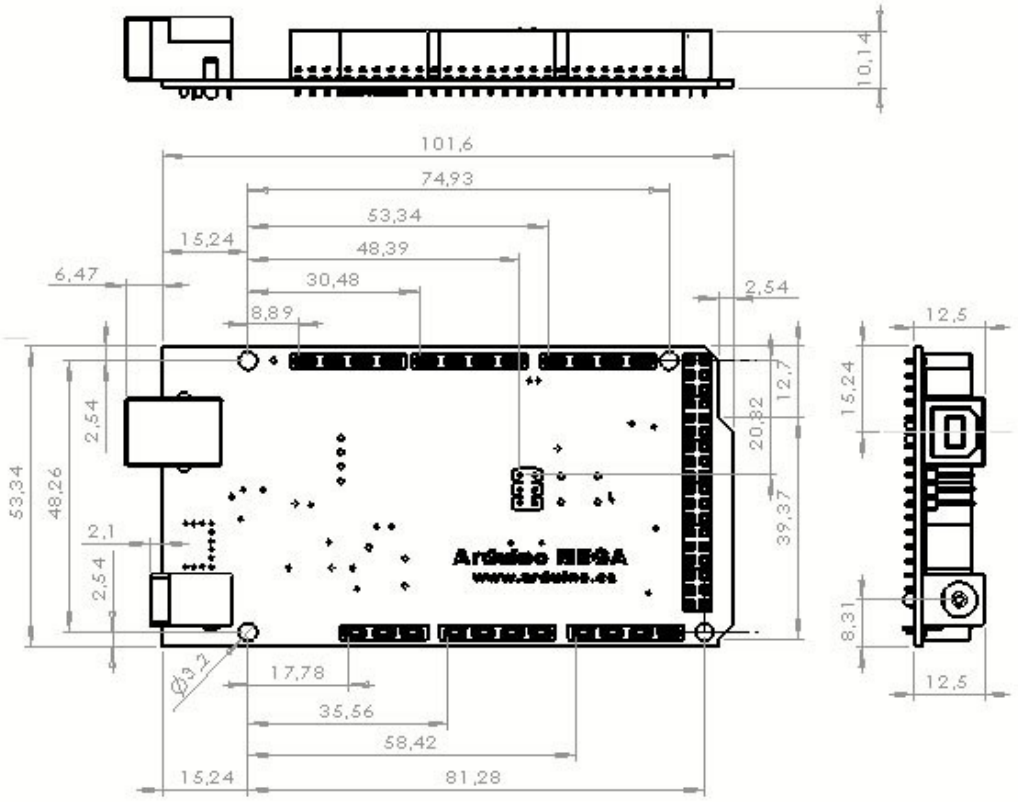
In **Tools>Board** select MEGA

Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.



Dimensioned Drawing





Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

Enviromental Policies

The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

الملخص

الغرض من هذا المشروع هو إنشاء آلة لبيع المشروبات بالاردوينو مع نظام الدفع ببطاقات الائتمان الممغنطة.

في الجزء الأول من عملنا، بدأنا بدراسة عامة حول وحدة الاردوينو بالإضافة إلى البرمجة مع واجهة البرمجة الخاصة به، الجزء الثاني مكرس للدراسة النظرية للوحدات النمطية المختلفة المستخدمة، مع تحديد مبدأ العملية، مخطط الأسلاك وكذلك البرنامج، الجزء الأخير مكرس لتحقيق العملي والاختبارات النهائية.

موزعنا هو منتج يحتاج إلى تحسين في المستقبل، حيث يضيف العديد من محركات، ويضيف أيضًا وحدات مثل قارئ بطاقة لتخزين المعلومات من أجل التعديل السهل.

أتاح لنا تطوير هذا العمل في إطار مشروع نهاية الدراسة تعميق معرفتنا النظرية المكتسبة خلال التدريب لدينا واكتساب خبرة جيدة على مستوى الإدراك العملي.

Abstract

The aim of this project is to create an Arduino-based drinks vending machine with a magnetic credit card payment system.

In the first part of our work, we started with a general study on the Arduino module as well as programming with IDE programming interface, the second part is devoted to the theoretical study of the different modules used, define the principle of operation, wiring diagram as well as the program, the last part is dedicated to the practical realization and the final tests.

Our distributor is a product that needs improvement in the future, adding several stepper motors, also add modules such as an SD card reader to store information for easy modification.

The development of this work within the framework of the end-of-study project, allowed us to deepen our theoretical knowledge acquired during our training and to acquire a good experience at the level of the practical realization.

Résumé

Ce projet a pour but de réaliser un distributeur de boissons à base d'une carte Arduino avec un système de payement avec carte de crédit magnétique.

Dans la première partie de notre travail, nous avons commencé par une étude générale sur le module Arduino ainsi que la programmation avec l'interface de programmation IDE, la deuxième partie est consacré à l'étude théorique des différents modules utilisés, définir le principe de fonctionnement, schéma de câblage ainsi que le programme, la dernière partie est dédié à la réalisation pratique et aux tests finals.

Notre distributeur est un produit qui a besoin d'amélioration dans le futur, en ajoutant plusieurs moteurs pas à pas, ajouter aussi des modules tels qu'un lecteur de carte SD pour stocker les informations afin de les modifier facilement.

L'élaboration de ce travail dans le cadre du projet de fin d'étude, nous a permis d'approfondir nos connaissances théoriques acquises durant notre formation et d'acquérir une bonne expérience au niveau de la réalisation pratique.