REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE DES SCIENCES APPLIQUEES
–T L E M C E N–

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

École Supérieure En Sciences Appliquées de Tlemcen

Département de la formation du Second Cycle

Filière: Génie industriel

Polycopie des Travaux Pratiques

Électronique Embarquée – Arduino

Élabore Par

Dr MEGNAFI Hicham

Préface

Ce polycopié de travaux pratiques dédié au module d'Électronique Embarquée basé sur la carte Arduino, spécialement conçu pour les étudiants de 3ème année du cycle ingénieur en spécialité Génie Industriel. Ce document a pour objectif de vous familiariser avec les principes fondamentaux de l'électronique embarquée en utilisant la populaire carte Arduino comme plateforme de développement.

Ce module vous offrira l'opportunité d'explorer les aspects essentiels de l'électronique embarquée, du développement de projets simples à des applications plus complexes. Vous serez amenés à acquérir des compétences pratiques en programmation, en conception de circuits électroniques et en intégration de systèmes, tout en mettant l'accent sur les applications industrielles.

Le polycopié comprendra plusieurs travaux pratiques, chacun étant accompagné d'un guide détaillé pour vous aider pas à pas dans la réalisation des projets. Vous serez amenés à manipuler des capteurs, des actionneurs, à programmer en langage C++ sur l'IDE Arduino, et à intégrer ces éléments pour créer des systèmes embarqués fonctionnels.

En tant qu'étudiants en génie industriel, l'électronique embarquée est une compétence clé qui vous permettra de concevoir, de contrôler et d'optimiser des processus industriels, des systèmes automatisés et des équipements intelligents.

Avant de commencer ce module, voici les prérequis nécessaires pour profiter pleinement de cette expérience :

- Connaissances en électronique de base : Vous devriez avoir une compréhension solide des concepts de base de l'électronique, y compris les composants électroniques, les circuits, et les lois de l'électricité.
- Programmation : Une connaissance préalable des bases de la programmation en langage C ou C++ sera utile pour tirer le meilleur parti de ce module.

Nous sommes convaincus que ce polycopié de travaux pratiques enrichissant vous permettra d'acquérir des compétences pratiques essentielles en électronique embarquée et de renforcer vos capacités d'ingénieur en génie industriel. Nous espérons que vous apprécierez cette expérience d'apprentissage stimulante et que vous développerez un intérêt croissant pour les technologies embarquées et leurs applications dans l'industrie.

Sommaire

Liste des Figures

Liste des Tables

Introduction générale

I. Introduction à l'Électronique Embarquée

Les systèmes embarqués jouent un rôle de plus en plus essentiel dans notre vie quotidienne, tant dans les appareils électroniques grand public que dans les domaines industriels et scientifiques. Ces systèmes, basés sur l'électronique embarquée, offrent une intégration étroite entre le matériel et le logiciel, leur permettant de réaliser des tâches spécifiques avec efficacité et précision.

I.1 Principes de base de l'électronique embarquée

Les principes de base de l'électronique embarquée sont basés sur les aspects suivants :

- **Intégration** : L'électronique embarquée vise à intégrer les différents composants électroniques nécessaires au bon fonctionnement du système dans un espace restreint, réduisant ainsi la taille et le coût du système.
- **Fonctions spécifiques** : L'électronique embarquée est conçue pour effectuer des fonctions spécifiques en réponse aux données des capteurs ou aux entrées utilisateur. Elle est optimisée pour ces fonctions particulières.
- **Fiabilité** : Étant souvent utilisée dans des applications critiques, l'électronique embarquée doit être fiable et robuste pour éviter les défaillances.
- **Faible consommation d'énergie** : L'électronique embarquée est conçue pour consommer le moins d'énergie possible, car elle est souvent alimentée par des sources d'énergie limitées.

La figure ci-dessous (Figure 1) représente le principe de base des systèmes embarqués :

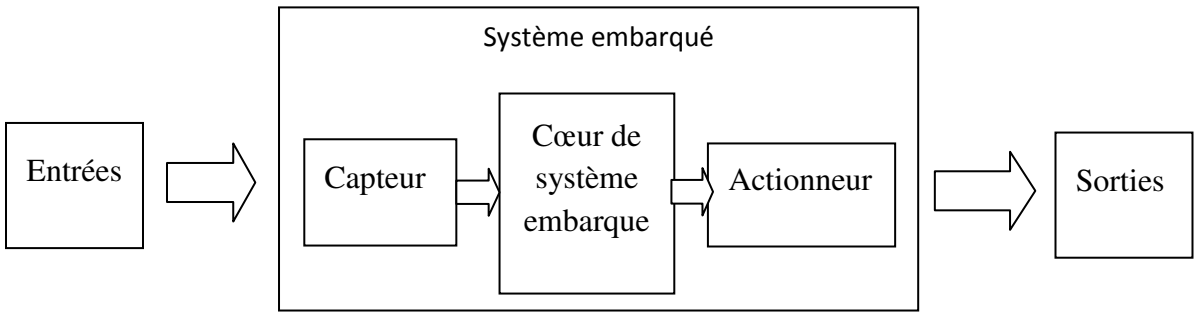


Figure 1. Principe de base des systèmes embarqués

Description des éléments :

Entrées : Ce sont les signaux ou les données provenant de différents capteurs ou périphériques externes. Les capteurs collectent des informations de l'environnement ou des entrées utilisateur.

Capteur : Le capteur est un dispositif qui détecte les changements dans l'environnement physique et les transforme en signaux électriques ou numériques compréhensibles par le système embarqué. Les données des capteurs sont envoyées au cœur du système pour traitement.

Cœur de système : C'est le bloc central de l'électronique embarquée, généralement un microcontrôleur ou un processeur dédié. Le cœur du système embarqué traite les données des capteurs, exécute les programmes embarqués et prend des décisions en fonction des instructions programmées.

Actionneur : L'actionneur est un dispositif qui transforme les signaux du cœur du système en actions physiques. Ils peuvent être des moteurs, des relais, des afficheurs, des LED, etc. Les actionneurs permettent au système de produire des résultats en réponse aux données des capteurs et aux décisions du cœur du système.

Sorties : Ce sont les résultats des actions effectuées par le système embarqué en utilisant les actionneurs. Les sorties peuvent être des actions physiques, des affichages, des signaux électriques, etc., en fonction de la nature du système embarqué et de ses objectifs.

I.2 Applications de l'électronique embarquée

L'électronique embarquée trouve des applications dans un large éventail de domaines. Dans les systèmes grand public, on le retrouve dans les smart phones, les tablettes, les appareils électroménagers, les wearables (objets connectés portables), les téléviseurs intelligents, les drones, etc. Dans le domaine industriel, les systèmes embarqués sont utilisés dans l'automatisation industrielle, le contrôle de la robotique, les systèmes de surveillance, la logistique et la gestion des processus. On les retrouve également dans les domaines de l'automobile (systèmes d'info divertissement, contrôle moteur, assistance à la conduite), de l'aérospatiale (systèmes avioniques), de la santé (dispositifs médicaux), de l'énergie (gestion de l'énergie) et bien d'autres secteurs.

I.2.1 Avantages et applications de l'électronique embarquée avec Arduino

L'utilisation d'Arduino dans les systèmes embarqués présente plusieurs avantages. En tant que plateforme de prototypage rapide, Arduino permet aux ingénieurs, aux étudiants de matérialiser rapidement leurs idées en réalisant des prototypes fonctionnels. Cela facilite l'expérimentation, le développement itératif et l'apprentissage. Les applications de l'électronique embarquée avec Arduino sont nombreuses, allant des projets ludiques et éducatifs aux applications plus sérieuses, comme la surveillance environnementale, les objets connectés, les systèmes de contrôle automatisés et les solutions d'automatisation industrielle.

I.3 Bases de l'électronique

I.3.1 Composants électroniques essentiels pour les projets Arduino

A. Résistances, condensateurs, diodes

a) Résistances

Une résistance est un composant électronique passif conçu pour résister au passage du courant électrique. Elle est mesurée en ohms (Ω) et est utilisée dans les circuits pour limiter le courant, diviser la tension, et créer des diviseurs de tension. Pour mesurer la valeur d'une résistance, on utilise un multimètre en mode "résistance" (Ω). On place les pointes de test du multimètre de chaque côté de la résistance et la valeur s'affiche sur l'écran. Ou bien on utilise le code des couleurs des résistances comme la montre la figure 2.

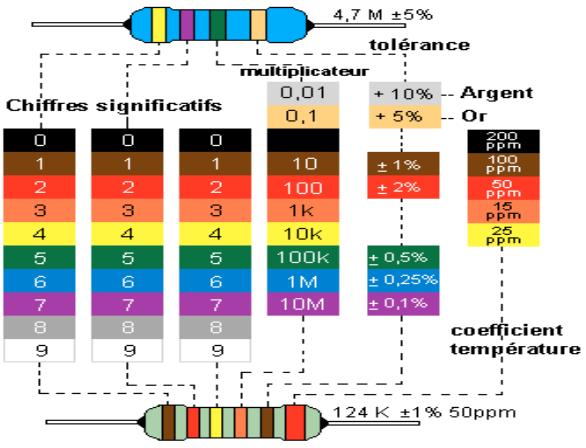


Figure 2. Code des couleurs des résistances

b) Condensateurs

Un condensateur (figure 3) est un composant électronique qui stocke de l'énergie électrique sous forme de charge électrique. Il est mesuré en farads (F) ou en sous-multiples tels que microfarads (µF) et picofarads (pF). Les condensateurs sont utilisés pour le filtrage des signaux, la stabilisation des tensions et le stockage d'énergie. Pour mesurer la valeur d'un condensateur, on utilise un multimètre en mode "capacité" (F). On place les pointes de test du multimètre sur les broches du condensateur, et la valeur s'affiche sur l'écran.

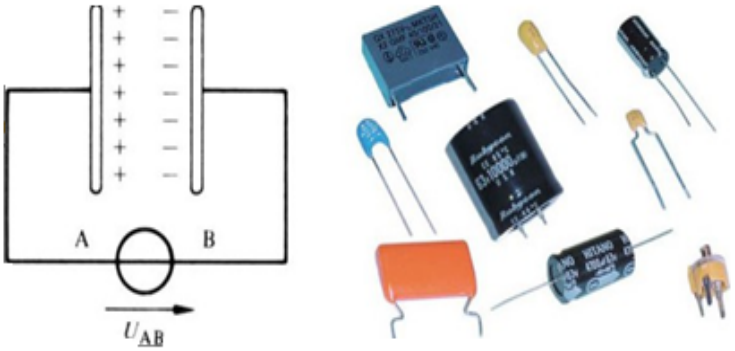


Figure 3. Condensateurs

c) Diodes

Une diode (figure 4) est un composant électronique non linéaire qui permet au courant de circuler dans une direction spécifique. Elle est utilisée pour redresser le courant alternatif en courant continu, protéger les circuits contre les surtensions et moduler des signaux. Pour tester une diode, on utilise également un multimètre en mode "diode" (généralement représenté par un symbole de diode sur le sélecteur). En plaçant les pointes de test du multimètre aux bornes de la diode dans les deux sens, on peut déterminer si elle est en bon état de fonctionnement.

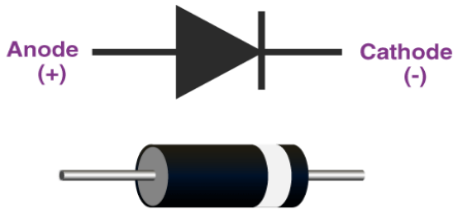


Figure 4. Diode

B. Transistors, capteurs, actionneurs

a) Transistors

Un transistor (figure 5) est un composant électronique actif utilisé pour amplifier et commuter les signaux électriques. Il existe différents types de transistors, tels que les transistors bipolaires (NPN, PNP) et les transistors à effet de champ (MOSFET). Les transistors sont utilisés dans les projets Arduino pour contrôler des moteurs, des LEDs, et pour réaliser des amplificateurs audio. Leur caractéristique principale est leur capacité à amplifier un petit signal de commande pour contrôler un courant plus important dans un circuit.

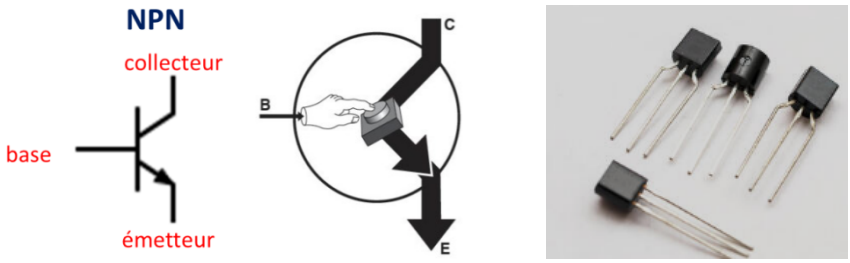


Figure 5. Transistors (symbole, fonctionnement et photos real)

b) Capteurs

Un capteur est un dispositif qui détecte des grandeurs physiques ou des variations de l'environnement et les convertit en signaux électriques. Les capteurs peuvent être de différents types, tels que des capteurs de lumière, de température, d'humidité, de distance, etc. Ils sont essentiels pour permettre aux projets Arduino de percevoir et de réagir à leur environnement. La figure suivante (figure 6) montre quelques exemples de capteurs utilise avec la carte Arduino.

Figure 6. Exemple des capteurs utilise avec la carte Arduino

c) Actionneurs

Un actionneur est un composant qui convertit un signal électrique en action physique. Les actionneurs sont utilisés dans les projets Arduino pour créer des mouvements, des vibrations, des sons, etc. Les exemples d'actionneurs courants incluent les moteurs DC, les servomoteurs, les haut-parleurs, les relais, etc. La figure suivante (figure 7) montre quelques exemples d'actionneur utilise avec la carte Arduino.

Figure 7. Exemple des actionneurs utilise avec la carte Arduino

C. Circuits électroniques de base

a) Montages en série et en parallèle

Dans un montage en série, les composants sont connectés les uns à la suite des autres, formant ainsi un chemin unique pour le courant. Le courant est le même dans chaque composant, mais la tension totale est répartie entre eux. La résistance totale dans un montage en série est la somme des résistances individuelles. Tan disque dans un montage en parallèle, les composants sont connectés côte à côte, partageant ainsi la même tension, mais ayant des chemins de courant distincts. La tension est la même pour chaque composant, mais le courant total est la somme des courants individuels.

b) Utilisation des breadboards et des fils de connexion

Le breadboard ou la plaque d'essai est un outil essentiel pour les projets Arduino. Il permet de créer des circuits sans soudure, ce qui facilite grandement le prototypage. Grâce au breadboard, les apprenants peuvent expérimenter rapidement avec différents composants électroniques et les connecter à Arduino. Les fils de connexion sont utilisés pour relier les composants entre eux sur le breadboard, ce qui permet de créer des circuits temporaires sans

endommager les composants. La figure suivante (figure 8) montre la plaque d'essai avec leur état de fonctionnement.

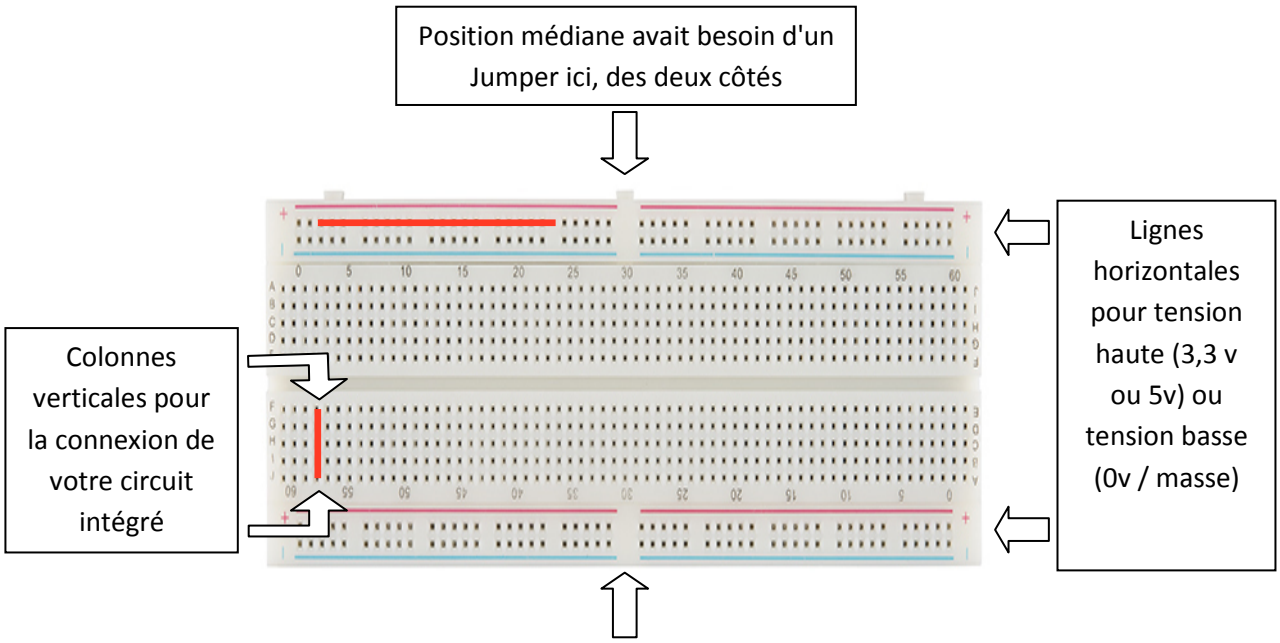


Figure 8. Principe de breadboard

II. Présentation d'Arduino et de son environnement de développement

Arduino est une plateforme de prototypage rapide largement utilisée pour les systèmes embarqués. Elle se compose de cartes de développement équipées de microcontrôleurs et d'une interface de programmation simple à utiliser. Les cartes Arduino sont conçues pour faciliter le développement rapide de prototypes électroniques en offrant une variété de broches d'entrée/sortie (E/S) permettant de connecter des capteurs, des actionneurs et d'autres composants électroniques. Grâce à son IDE (Environnement de développement intégré), les utilisateurs peuvent écrire et téléverser facilement le code dans la carte Arduino pour contrôler le système embarqué.

II.1 Composants de base d'Arduino Mega 2560

II.1.1 Spécifications techniques de la carte Arduino Mega

Les spécifications techniques de la carte Arduino Mega 2560 offrent une vue d'ensemble détaillée de ses capacités avancées. De la mémoire généreuse aux multiples entrées/sorties, ces spécifications fournissent la base pour des projets électroniques variés et complexes. Ces spécifications sont decrits dans le tableau suivant (tableau 1) :

Tableau 1. Spécifications techniques de la carte Arduino Mega 2560

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	**9V**
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 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
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

II.1.2 Microcontrôleur

L'ATmega2560 est un microcontrôleur, utilisé dans la carte Arduino Mega 2560. Doté d'une architecture RISC 8 bits, il offre des performances fiables et une grande flexibilité pour les projets électroniques. Avec une fréquence d'horloge allant jusqu'à 16 MHz, il dispose de 256 Ko de mémoire Flash pour le stockage du programme et 8 Ko de mémoire SRAM pour les données en cours d'exécution.

Ce microcontrôleur propose 86 broches d'E/S, dont 54 numériques et 16 analogiques. Il prend en charge divers protocoles de communication tels que l'I2C, le SPI et l'USART, facilitant l'interaction avec des capteurs et des périphériques externes. Il est également équipé

de fonctions avancées, comme le générateur PWM, le compteur/timer et le convertisseur analogique-numérique (CAN).

II.1.3 Alimentation

La carte Arduino Mega offre plusieurs options d'alimentation, incluant la connexion USB ou une alimentation externe, avec une sélection automatique de la source. L'alimentation externe peut être fournie par une alimentation à courant continu ou une batterie. Pour cela, on peut utiliser un adaptateur avec une fiche positive de 2,1 mm branchée dans la prise d'alimentation de la carte, ou connecter les fils d'une batterie aux broches Gnd et Vin du connecteur POWER.

La carte peut fonctionner sur une plage d'alimentation de 7 à 12 volts. Les principales broches d'alimentation de la carte Arduino Mega sont décrites dans le tableau les suivantes (tableau 2):

Tableau 2. Principales broches d'alimentation de la carte Arduino Mega

Type	Description
VIN	Entrée de tension de la carte via une source d'alimentation externe.
5V	Alimentation régulée pour le microcontrôleur et les autres composants, provenant de VIN, de l'USB ou d'une source 5V régulée.
3.3V	Alimentation de 3,3V générée par la puce FTDI embarquée, avec une consommation maximale de 50 mA.
GND	Masse ou référence à 0V de la carte.

II.1.4 Broches d'E/S (Entrées/Sorties)

Il y a 54 entrées/sorties numériques, identifiées de 0 à 53, sur lesquelles le programme peut agir pour les configurer en mode entrée ou sortie. De plus, leur fonctionnement peut changer de manière dynamique pendant l'exécution du programme. Ces entrées/sorties sont compatibles avec les signaux logiques TTL, avec une plage de tension comprise entre 0 et 5V. Chaque entrée/sortie peut fournir ou délivrer un courant maximal de 40 mA. Cependant, il est essentiel de noter que l'ensemble des sorties combinées ne doit en aucun cas dépasser 200

mA. La figure suivante (figure 9) montre les types emplacements de chaque port E/S de la carte Arduino.

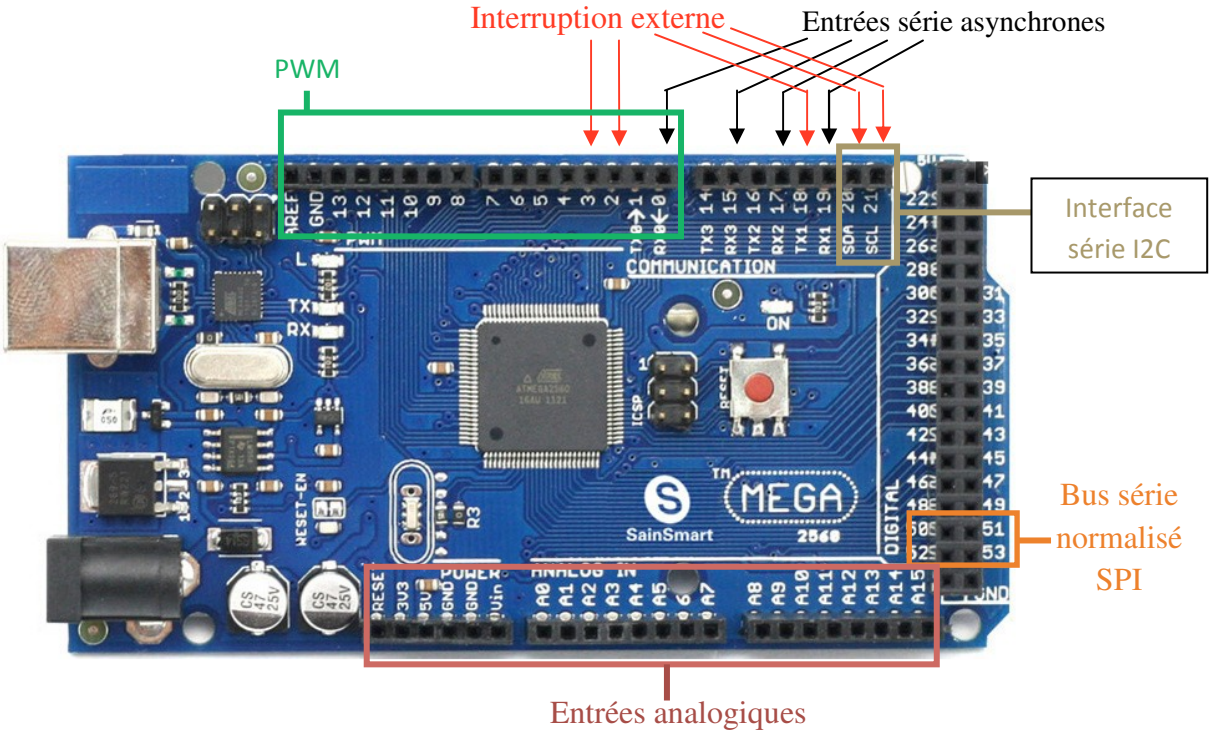


Figure 9. Présentation des Broches d'E/S de la carte arduino mega 2560

Description des broches d'entres / sorties :

A. Entrées série asynchrones

Il y a 4 entrées séries asynchrones (Rx0, Rx1, Rx2, Rx3) et 4 sorties séries asynchrones (Tx0, Tx1, Tx2, Tx3). Ces entrées et sorties sont partagées avec les lignes 0, 19, 17 et 15 pour la partie Rxn, et les lignes 1, 18, 16 et 14 pour la partie Txn.

B. Interruption externe

Il y a 6 entrées d'interruption externes qui sont partagées avec les lignes 2, 3, 18, 20 et 21. Lorsque des interruptions sont en cours d'exécution, les données sur les ports de communication ne sont pas prises en compte. Ainsi, les variables utilisées dans les interruptions doivent être déclarées comme "volatile" pour garantir leur mise à jour correcte. De plus, il est important de noter que la fonction "delay" ne fonctionne pas au sein des routines d'interruption. Par conséquent, son utilisation à cet endroit doit être évitée. De la

même manière, la fonction "millis()" n'a aucun effet au sein de ces routines d'interruption, et son utilisation ne sera pas pertinente pour mesurer le temps.

C. Les sorties PWM

Il y a 14 sorties PWM (Modulation de Largeur d'Impulsion ou MLI) disponibles sur les lignes numériques de 0 à 13. La MLI est une technique permettant d'obtenir des effets semblables à des signaux analogiques à partir de broches numériques. Elle consiste à créer une onde carrée où le signal alterne entre un niveau HAUT (5V) et BAS (0V) en utilisant un contrôle numérique.

D. Bus série normalisé SPI

Il y a 4 entrées/sorties pour le bus série SPI (Serial Peripheral Interface) normalisé, avec les broches suivantes : SS (Slave Select) sur la broche 53, MOSI (Master Out Slave In) sur la broche 51, MISO (Master In Slave Out) sur la broche 50, et SCK (Serial Clock) sur la broche 52.

Le bus SPI est un protocole de communication série synchrone utilisé par les microcontrôleurs pour permettre des échanges rapides avec un ou plusieurs périphériques sur de courtes distances. Il peut également être employé pour les communications entre deux microcontrôleurs.

Pour utiliser la librairie SPI dans un programme, il suffit d'ajouter au début du code la ligne suivante : #include <SPI.h>

E. Entrée / sortie pour interface série I2C

Il y a 2 entrées/sorties pour le bus de données I2C réparties comme suit : SDA (Serial Data) sur la broche 20, et SCL (Serial Clock) sur la broche 21.

Le bus I2C est un protocole de communication qui a été développé lors de la "guerre des standards" dans le monde de l'électronique. Il a été conçu par Philips pour les applications de domotique et d'électronique domestique, permettant de connecter facilement un microprocesseur à différents circuits, notamment ceux d'une télévision moderne.

Pour utiliser la librairie Wire dans un programme, il suffit d'ajouter au début du code la ligne suivante : #include <Wire.h>.

F. Entrées analogiques

La carte Arduino est équipée de 16 entrées, identifiées de A0 à A15, capables de recevoir des tensions analogiques comprises entre 0 et 5 Volts. Cependant, il est crucial de ne jamais dépasser cette limite de 5 Volts, car cela pourrait entraîner des dommages au microcontrôleur.

Pour mesurer des tensions supérieures à 5 Volts, il est nécessaire de munir l'entrée d'un pont diviseur. Les cartes Arduino incluent un convertisseur analogique-numérique (A/N) pour effectuer des mesures de tensions analogiques. Ce convertisseur possède une résolution de 10 bits, ce qui signifie qu'il renvoie des valeurs de mesure sous forme d'entiers compris entre 0 et 1023.

II.2 Environnement de développement Arduino IDE

IDE (Integrated Development Environment) est spécialement conçu pour faciliter la programmation des cartes Arduino, ouvrant ainsi la voie à des projets innovants et passionnants. Arduino IDE offre une plate-forme conviviale qui permet à tous d'explorer et de donner vie à leurs idées électroniques. Découvrons comment cet environnement de développement stimulant nous permet d'exploiter tout le potentiel de ces microcontrôleurs. La fenêtre principale de l'IDE est présente comme suite (figure 10):

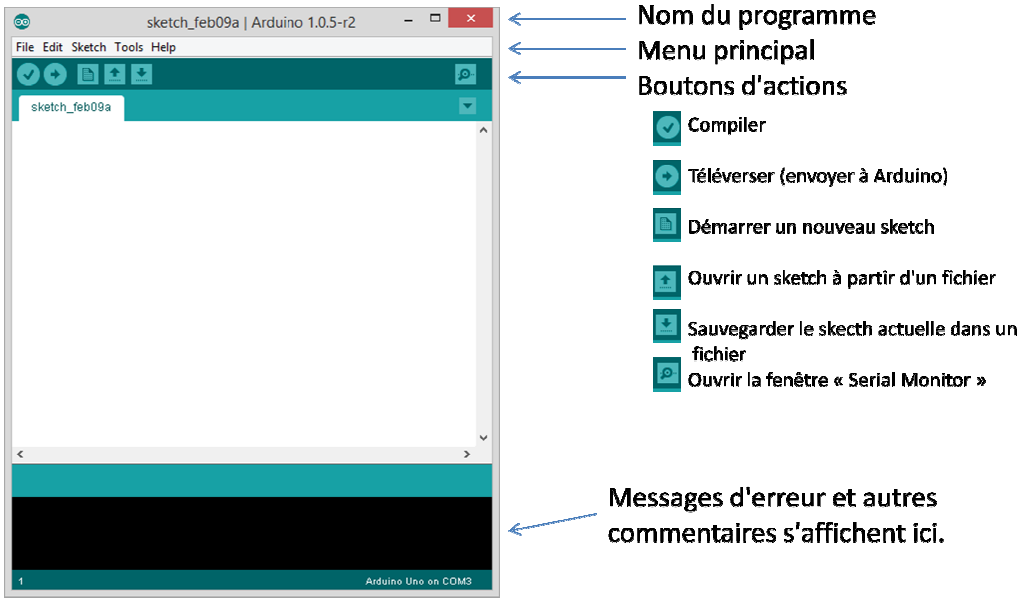


Figure 10. Fenetre principale de l'IDE d'Arduino

II.2.1 Langage de programmation Arduino (C/C++)

L'Arduino utilise le langage de programmation C/C++ pour écrire des programmes qui contrôlent ses entrées et sorties numériques et analogiques. Bien qu'Arduino utilise C/C++, la plupart des fonctions et bibliothèques spécifiques à Arduino simplifient considérablement la programmation pour les débutants et permettent d'accéder facilement aux fonctionnalités matérielles.

A. Les fonctions setup() et loop()

- **setup**() : La fonction "void **setup**()" est appelée une seule fois au démarrage du programme. Elle est utilisée pour initialiser les paramètres, configurer les broches d'entrée/sortie, et effectuer toute autre opération qui doit être effectuée une seule fois au début du programme. Voir figure figure 11.
- **loop**() : La fonction "void **loop**()" est exécutée en boucle en continu après le "setup()". Elle est utilisée pour les tâches qui doivent être exécutées en continu, comme la surveillance des capteurs, la gestion des entrées utilisateur ou le contrôle des sorties. Voir figure figure 11.

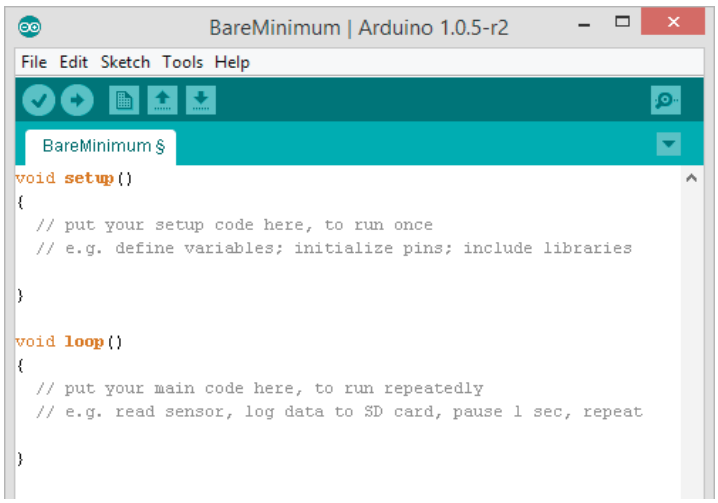


Figure 11. Présentation des fonctions de base dans IDE d'arduino

B. Syntaxe de base du langage C/C++ utilisé par Arduino

La syntaxe de base du langage C/C++ utilisée par Arduino est similaire à celle du langage C/C++ standard. Voici quelques points importants de la syntaxe utilisée dans les programmes Arduino :

- **Point-virgule** : Comme en C/C++, chaque instruction doit se terminer par un point-virgule.
- **Commentaires** : Les commentaires sur une seule ligne sont ajoutés en utilisant "//". Les commentaires multilignes sont inclus entre "/" et "/".
- **Variables** : Les variables sont déclarées en spécifiant le type de données suivi du nom de la variable. Les types de données courants sont int (entier), float (nombre à virgule flottante), char (caractère), et bool (booléen). Les variables globales sont déclarées en dehors des fonctions setup() et loop() et peuvent être utilisées dans tout le programme.
- **Structures de contrôle** : Les structures de contrôle comme "if", "else", "while", "for", "switch" sont utilisées de la même manière qu'en C/C++ standard. Elles permettent de contrôler l'exécution du code en fonction de certaines conditions.
- **Opérateurs** : Les opérateurs arithmétiques (+, -, *, /, %), les opérateurs de comparaison (==, !=, <, >, <=, >=) et les opérateurs logiques (&&, ||, !) sont utilisés pour effectuer des opérations mathématiques et logiques.
- Fonctions prédéfinies : L'Arduino IDE inclut des fonctions prédéfinies qui simplifient la programmation. Voici quelques exemples de fonctions couramment utilisées :
  - **pinMode**(pin, mode) : Configure une broche comme entrée (INPUT) ou sortie (OUTPUT).
  - **digitalWrite**(pin, value) : Écrit une valeur HIGH ou LOW (1 ou 0) sur une broche de sortie.
  - **digitalRead**(pin) : Lit la valeur (HIGH ou LOW) d'une broche d'entrée.
  - **analogRead**(pin) : Lit la valeur analogique (0 à 1023) d'une broche analogique.
  - **analogWrite**(pin, value) : Écrit une valeur de PWM (0 à 255) sur une broche PWM (Pulse-Width Modulation).

Ces fonctions prédéfinies permettent de manipuler facilement les broches numériques et analogiques, de contrôler les périphériques connectés à l'Arduino et de réaliser une variété de projets électroniques.

C. Utilisation des bibliothèques Arduino pour faciliter la programmation

Arduino dispose d'une vaste bibliothèque standard qui simplifie la programmation et permet d'accéder facilement aux fonctionnalités matérielles de la carte. Les bibliothèques fournissent des fonctions préécrites qui permettent d'interagir avec les périphériques tels que les LED, les capteurs, les afficheurs LCD, etc.

Pour utiliser une bibliothèque, vous devez inclure son nom en haut de votre code avec une instruction "#include <nom_de_la_bibliothèque.h>". Une fois inclus, vous pouvez utiliser les fonctions de la bibliothèque dans votre code.

Par exemple, pour contrôler une LED, vous pouvez utiliser la bibliothèque "Arduino.h" qui est incluse par défaut. La fonction "digitalWrite()" est une fonction de cette bibliothèque qui permet d'allumer ou d'éteindre une broche numérique pour contrôler la LED.

III. Applications avancées et extensions possibles

L'utilisation de la carte Arduino ne se limite pas aux projets basiques. En effet, cette plateforme offre de nombreuses applications avancées et des extensions passionnantes. Parmi celles-ci, la communication sans fil avec des modules RF, tels que le nRF24L01, permet de créer des réseaux d'objets interconnectés. L'intégration de capteurs de gaz offre la possibilité de détecter des fuites potentielles et de garantir la sécurité dans divers environnements. De plus, en combinant Arduino avec des modules Wi-Fi tels que l'ESP8266, il est possible de créer des systèmes IoT intelligents, permettant la collecte de données en temps réel et le contrôle à distance des appareils. Ces possibilités d'extensions ouvrent de nouvelles perspectives pour des projets électroniques innovants et interactifs.

III.1 Communication sans fil avec des modules RF (par exemple, nRF24L01)

Les modules RF (Radio-Fréquence) tels que le nRF24L01 permettent d'établir une communication sans fil entre différents nœuds d'un réseau. Ces modules sont très populaires

pour les projets Arduino car ils offrent une solution simple et économique pour créer des réseaux sans fil à courte portée. Ils peuvent être utilisés pour transmettre des données, des commandes ou des informations entre différents appareils Arduino. Par exemple, vous pourriez utiliser ces modules pour créer un système de contrôle à distance pour un robot, un réseau de capteurs pour la surveillance environnementale ou un système de communication pour des appareils portables.

III.2 Utilisation d'un capteur de gaz pour détecter des fuites potentielles

L'intégration d'un capteur de gaz dans un projet Arduino permet de détecter et de surveiller la présence de gaz nocifs ou potentiellement dangereux dans l'environnement. Les capteurs de gaz peuvent détecter des gaz tels que le monoxyde de carbone (CO), le dioxyde de carbone (CO2), les gaz inflammables, les gaz toxiques, etc. Ces capteurs peuvent être utilisés pour la sécurité domestique, la surveillance industrielle, la détection de fuites de gaz dans les systèmes de chauffage ou la mesure de la qualité de l'air. Lorsque le capteur détecte la présence d'un gaz dangereux, il peut déclencher une alarme, envoyer des notifications ou prendre des mesures pour protéger les occupants ou l'environnement.

III.3 Intégration de l'Internet des objets (IoT) avec un module Wi-Fi (comme ESP8266)

L'Internet des objets (IoT) est un domaine en plein essor qui consiste à connecter des objets et des appareils du quotidien à Internet pour permettre la collecte et l'échange de données. L'utilisation d'un module Wi-Fi tel que l'ESP8266 avec Arduino permet de transformer vos projets en dispositifs IoT. Vous pouvez envoyer des données en temps réel vers des serveurs en ligne, contrôler à distance vos appareils depuis une application mobile ou recevoir des notifications lorsque certains événements se produisent. Par exemple, vous pourriez créer un système de surveillance à domicile où les capteurs détectent les mouvements, la température et l'humidité, puis envoient les données à un serveur en ligne pour une visualisation à distance via une application mobile. L'ajout de la connectivité Wi-Fi ouvre un large éventail de possibilités pour créer des projets IoT innovants et interactifs.

IV. Dépannage et bonnes pratiques

Le dépannage et les bonnes pratiques de l'électronique embarquée sont essentiels pour assurer le bon fonctionnement et la fiabilité des projets. En identifiant et résolvant rapidement les problèmes de circuits et de code, le dépannage permet de garantir des résultats optimaux. Les bonnes pratiques de câblage, l'utilisation appropriée des composants et la gestion efficace de l'alimentation sont des éléments clés pour développer des systèmes électroniques embarqués robustes et performants. Dans ce qui suite, nous prenons on détails la technique et les différents points à suivre pour un bon dépannage des circuits.

IV.1 Techniques de dépannage des circuits et du code

IV.1.1 Vérification du câblage

Lorsque vous rencontrez un problème avec votre circuit Arduino, commencez par vérifier soigneusement le câblage. Assurez-vous que toutes les connexions sont correctement réalisées, qu'il n'y a pas de fils cassés ou de connexions lâches. Un simple mauvais contact peut parfois être à l'origine du dysfonctionnement.

IV.1.2 Utilisation des LED de diagnostic

Les cartes Arduino sont souvent équipées de LED de diagnostic intégrées, telles que la LED ON et la LED L (ou RX/TX sur certaines versions). Ces LED peuvent être utilisées pour indiquer l'état de la carte et pour identifier les problèmes potentiels. Par exemple, si la LED ON est allumée, cela signifie que la carte est correctement alimentée, tandis que si la LED L clignote, cela indique qu'il y a une communication série.

IV.1.3 Affichage des messages de débogage

Utilisez la fonction Serial.print() ou Serial.println() dans le code pour afficher des messages de débogage sur le moniteur série de l'IDE Arduino. Cela vous permettra de suivre l'exécution du programme, d'afficher des valeurs de variables et d'identifier les erreurs éventuelles dans le code.

IV.1.4 Utilisation des points de contrôle

Insérez des points de contrôle (par exemple, des LEDs ou des afficheurs) à des endroits clés du circuit ou du code pour vérifier que certaines parties du programme fonctionnent correctement. Cela vous aidera à isoler les problèmes et à cibler les parties du circuit qui nécessitent une attention particulière.

IV.1.5 Test des composants individuels

Si un composant ne fonctionne pas comme prévu, vérifiez-le individuellement en utilisant un multimètre ou un autre outil de test approprié. Cela vous permettra de déterminer si le composant est défectueux ou s'il y a un problème avec son intégration dans le circuit.

IV.2 Bonnes pratiques de câblage et d'utilisation des composants

IV.2.1 Utilisation de couleurs pour les fils

Utilisez des fils de couleurs différentes pour faciliter le suivi et la compréhension des connexions. Par exemple, utilisez du rouge pour l'alimentation, du noir pour la masse, et des couleurs différentes pour les signaux. Cela rendra le câblage plus clair et vous évitera de vous mélanger dans les connexions.

IV.2.2 Éviter les croisements de fils

Évitez que les fils se croisent sur le breadboard ou dans le circuit imprimé, car cela pourrait entraîner des courts-circuits ou des connexions incorrectes. Organisez les fils de manière ordonnée et logique pour minimiser les risques d'erreurs de câblage.

IV.2.3 Fixation des composants

Assurez-vous que les composants sont correctement insérés dans les trous du breadboard ou soudés sur le circuit imprimé, et qu'ils sont solidement fixés pour éviter les connexions lâches.

IV.2.4 Utilisation de résistances de limitation

Lorsque vous connectez des LEDs, des capteurs ou d'autres composants sensibles, utilisez des résistances de limitation appropriées pour éviter de les surcharger et les endommager.

IV.2.5 Gestion des polarités

Faites attention à la polarité des composants, tels que les diodes, les condensateurs électrolytiques et les LEDs. Assurez-vous de les connecter correctement pour éviter de les endommager.

IV.3 Gestion de l'alimentation et des ressources

IV.3.1 Économie d'énergie

Utilisez des techniques d'économie d'énergie pour prolonger la durée de vie de la batterie et réduire la consommation d'énergie. Par exemple, désactivez les périphériques inutilisés ou utilisez des modes de veille lorsque possible.

IV.3.2 Utilisation de sources d'alimentation adéquates

Assurez-vous d'utiliser des sources d'alimentation appropriées pour votre projet. Vérifiez la tension, le courant et la puissance requis par les composants et assurez-vous que l'alimentation choisie peut les fournir en toute sécurité.

IV.3.3 Protégez les broches d'E/S

Utilisez des résistances de limitation et des diodes de protection pour éviter de surcharger les broches d'entrée/sortie et les endommager.

IV.3.4 Gestion de la mémoire

Si votre programme utilise beaucoup de mémoire, assurez-vous d'optimiser le code pour économiser de l'espace et éviter les problèmes de dépassement de mémoire.

IV.3.5 Utilisation des régulateurs de tension

Si votre projet nécessite une tension régulée, utilisez des régulateurs de tension appropriés pour obtenir une tension stable et fiable pour vos composants.

En suivant ces bonnes pratiques et en utilisant les techniques de dépannage appropriées, vous pourrez résoudre efficacement les problèmes éventuels dans vos projets d'électronique embarquée avec Arduino. Vous pourrez ainsi développer des projets plus fiables, durables et efficaces tout en évitant les erreurs courantes liées au câblage, à l'alimentation et au code.

Travaux Pratiques

- **Travaux Pratique N°1** : Introduction à la Programmation Arduino : Clignotant LED et Contrôle de la Luminosité
- **Travaux Pratique N°2** : Contrôle d'un Système d'Éclairage avec des Boutons-Poussoirs et des Relais
- **Travaux Pratique N°3** : Système de Suivi et de Traçabilité des Produits avec Capteurs RFID
- **Travaux Pratique N°4** : Système de Surveillance de la Chaîne Logistique avec Capteurs Ultrason
- **Travaux Pratique N°5** : Contrôle Automatisé de la Température et de l'Humidité Actionné par un Ventilateur/Moteur à Courant Continu

Travaux Pratiques N° 1:

Introduction à la Programmation Arduino : Clignotant LED et Contrôle de la Luminosité

I.1 Objectif

- Introduire les bases de la programmation Arduino.
- Enseigner le contrôle des sorties numériques pour clignoter une LED.
- Comprendre le contrôler de la luminosité de la LED.
- Pratiquer la connexion matérielle et développer l'aisance avec l'environnement Arduino IDE.

I.2 Introduction

L'IDE Arduino (Environnement de Développement Intégré) est un logiciel gratuit et open-source conçu pour programmer et développer des projets avec les cartes Arduino. Il offre une interface (figure 12) conviviale permettant aux utilisateurs de créer, modifier, vérifier et téléverser facilement du code sur les cartes Arduino. Grâce à son intégration étroite avec les bibliothèques Arduino, il facilite également l'utilisation de nombreux capteurs et modules, rendant le processus de développement plus rapide et plus efficace.

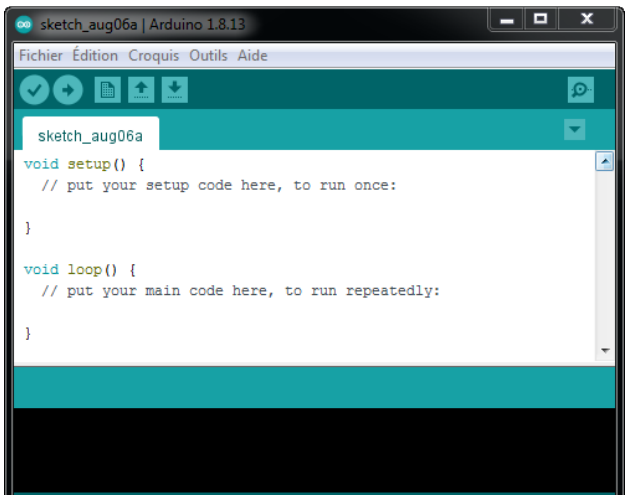


Figure 12. Interface principale de l'IDE Arduino

Installation et configuration de l'environnement de développement Arduino IDE :

1) Téléchargement de l'Arduino IDE : Rendez-vous sur le site officiel d'Arduino (https://www.arduino.cc/) et téléchargez la dernière version de l'Arduino IDE adaptée à votre système d'exploitation (Windows, macOS, Linux).
2) Installation de l'Arduino IDE : Une fois le fichier d'installation téléchargé, lancez-le et suivez les instructions d'installation pour installer l'Arduino IDE sur votre ordinateur.
3) Connexion de l'Arduino : Si vous possédez déjà un Arduino Mega ou tout autre modèle, connectez-le à votre ordinateur à l'aide d'un câble USB. Assurez-vous que l'Arduino est correctement détecté et reconnu par votre système d'exploitation. (Dans la plupart des cas, les pilotes nécessaires seront installés automatiquement.)
4) Lancement de l'Arduino IDE : Après l'installation, lancez l'Arduino IDE. Vous devriez voir une interface graphique conviviale avec une barre de menus et une zone de saisie du code.
5) Configuration du type de carte : Allez dans le menu "Outils" (Tools) > "Carte" (Board) et sélectionnez "Arduino Mega" ou le modèle de carte que vous utilisez dans la liste des cartes disponibles.
6) Sélection du port série : Toujours dans le menu "Outils" (Tools), allez dans "Port" et choisissez le port série qui correspond à votre Arduino. Si vous n'êtes pas sûr du port, connectez votre Arduino, allez dans le menu "Port" et notez le port qui apparaît. C'est généralement marqué comme "Arduino" suivi d'un numéro de port.
7) Vérification du fonctionnement de l'IDE : Pour vérifier que tout fonctionne correctement, sélectionnez un exemple de programme Arduino pré-installé en allant dans le menu "Fichier" (File) > "Exemples" (Examples) > "01. Basics" > "Blink". Cela ouvrira l'exemple de programme "Blink" qui fait clignoter une LED intégrée sur l'Arduino. Compilez (vérifiez) le programme en cliquant sur le bouton de coche (✓) dans la barre d'outils. Si tout est correct, vous devriez voir "La compilation s'est bien déroulée" en bas de l'IDE.
8) Téléversement du programme sur l'Arduino : Assurez-vous que l'Arduino est correctement connecté et que le bon port est sélectionné (étape 6). Ensuite, cliquez sur le bouton "Téléverser" (Upload) représenté par une flèche vers la droite dans la

barre d'outils. Cela transférera le programme sur l'Arduino. Vous devriez voir "Téléversement terminé" en bas de l'IDE.

I.3 Travail à réaliser

I.3.1 Manipulation 1 : Allumer une LED

Le but de cette première manipulation est d'allumer une LED de couleur rouge connectée à la broche 7 de la carte Arduino.

A. Matériel requis

- Une carte Arduino (par exemple, Arduino Mega)
- Une LED
- Une résistance de 220 ohms (pour protéger la LED)
- Deux fils de connexion (jumper wires)

B. Étapes de la manipulation

Pour allumer une LED en utilisant une carte Arduino, suivez les étapes ci-dessous :

Étape 1 : Configuration matérielle

Réaliser le circuit suivant (voir figure 13):

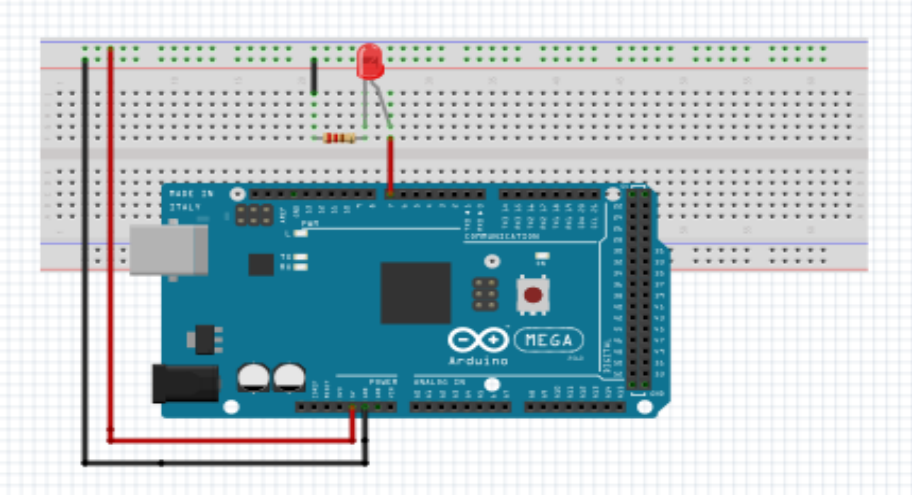


Figure 13. Circuit électronique « Commande la LED avec l'Arduino Mega »

1) Connectez votre breadbord aux broches 5V et Ground (notée GND) de la carte.
2) Placez la LED rouge sur la breadbord.

3) Attachez la cathode (patte courte de la LED) au Ground, via une résistance 220 Ohms.
4) Connectez l'anode de la LED rouge à la broche 7 d'arduino.

Étape 2 : Programmation

Ouvrez l'environnement de développement Arduino IDE sur votre ordinateur et créez un nouveau projet. Ecrire le programme suivant :

```
int PIN_LED_Rouge= 7 ;

void setup()

{   pinMode(PIN_LED_Rouge, OUTPUT);

}

void loop()

{  digitalWrite(PIN_LED_Rouge, HIGH);

   delay(500);

}
```

Étape 3 : Téléversement (Upload) du code sur la carte Ar**duino**

1) Branchez votre carte Arduino à votre ordinateur via un câble USB.
2) Sélectionnez le type de carte et le port correct dans le menu Outils de l'IDE Arduino.
3) Vérifiez votre code en cliquant sur l'icône de vérification (checkmark).
4) Si la vérification réussit, cliquez sur l'icône de téléversement (flèche vers la droite) pour charger le code sur la carte Arduino.
5) La LED devrait maintenant s'allumer dès que le code est téléversé sur la carte Arduino.

N'oubliez pas que la LED restera allumée jusqu'à ce que vous téléversez un nouveau code ou éteignez manuellement la LED dans le programme. Pour éteindre la LED, vous pouvez utiliser digitalWrite(7, LOW); dans le programme.

Refaire la 2ème étape avec le programme suivant (pour voir l'affichage dans le serial) :

```
int PIN_LED_Rouge= 7 ;
Int count =0;
void setup()
{   pinMode(PIN_LED_Rouge, OUTPUT);
   Serial.begin(9600); // communications à 9600 baud (char /second)
}
void loop()
{      digitalWrite(PIN_LED_Rouge, HIGH);
           delay(500);
      Serial.print("Count is ");
           Serial.println(count);
           count++;  }
```

I.3.2 Manipulation 2 : Clignotement d'une LED

Dans cette manipulation, nous allons modifier notre programme pour que la LED clignote. La LED sera allumée pendant une seconde, puis éteinte pendant une autre seconde. Nous utiliserons le même circuit que dans la première application. Dans ce programme, nous ferons appel à la fonction delay() pour introduire un délai en millisecondes.

```
int PIN_LED_Rouge= 7 ;
Int count =0;
void setup()
{
 pinMode(PIN_LED_Rouge, OUTPUT);
 Serial.begin(9600); // communications à 9600 baud (char /second)
}
void loop()
{
           digitalWrite(PIN_LED_Rouge, HIGH);
           delay(1000);
      digitalWrite(PIN_LED_Rouge, LOW);
           delay(1000);
      Serial.print("Count is ");
           Serial.println(count);
           count++;
}
```

I.3.3 Manipulation 3 : Variation de l'intensité de la LED

Dans cette manipulation, nous allons apprendre à contrôler l'intensité lumineuse d'une LED en utilisant la modulation de largeur d'impulsion (PWM - Pulse Width Modulation) sur une carte Arduino. La PWM nous permet de simuler une sortie analogique en ajustant le rapport cyclique (rapport entre le temps d'allumage et le temps total) de la LED.

```
#define PIN_LED_Rouge1  10
// Définition d'une constante.
int Intensite_LED = 0;   // Intensité de la LED
int Quantite_de_variation= 1;
// Quantité de variation de l'intensité

void setup()  { // Initialise le PIN digital pinA comme OUTPUT
            pinMode(PIN_LED_Rouge1 , OUTPUT);
}
void loop()  { // Etablit l'intensité de la LED connectée à la PIN pinA
analogWrite(PIN_LED_Rouge1 , Intensite_LED);
// Change l'intensité de la LED
Intensite_LED = Intensite_LED + Quantite_de_variation;
// Si on atteint une extreme, change le sens de variation d'intensité
if (Intensite_LED == 0 || Intensite_LED == 255) {
  Quantite_de_variation = -Quantite_de_variation;
  // Si l'intensité est à son minimum, éteint la LED et attend un petit moment
  if (Intensite_LED == 0) {
    digitalWrite(PIN_LED_Rouge1 , LOW);
    delay(500);
    }
  }
delay(10);
}
```

Vous devriez maintenant voir la LED varier en intensité lumineuse, passant progressivement de l'obscurité à la pleine luminosité, puis revenant à l'obscurité. Cela démontre comment utiliser la modulation de largeur d'impulsion pour contrôler la luminosité de la LED. Vous pouvez modifier les délais et les valeurs de luminosité pour expérimenter différents effets visuels.

I.3.4 Manipulation 4 : Clignotement de plusieurs LED

L'objectif de cette manipulation est de faire clignoter de manière alternée 3 LEDs de couleur rouge, connectées aux broches 7 à 9 de la carte Arduino. Les LEDs s'allumeront pendant une demi-seconde et s'éteindront pendant la demi-seconde suivante.

Réaliser le circuit suivant (figure 14):

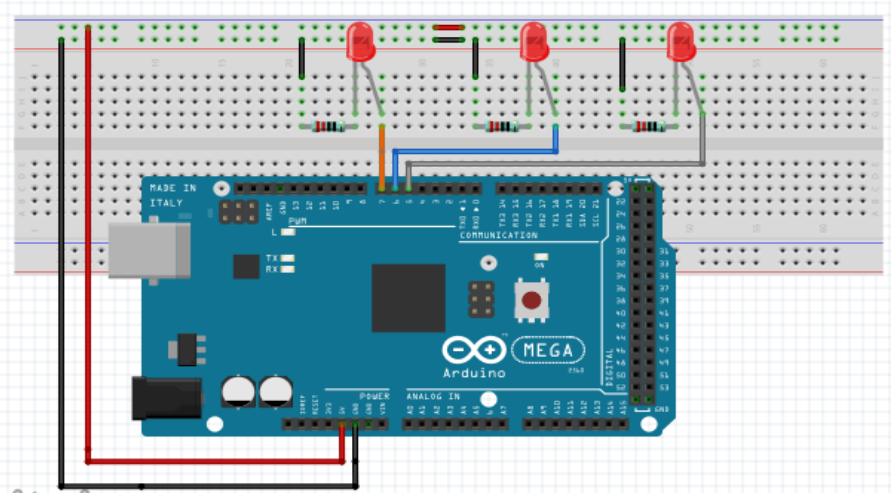


Figure 14. Circuit électronique « commandement des trois LEDs par Arduino »

Pour réaliser le clignotement de plusieurs LEDs connectées à une carte Arduino, vous pouvez utiliser un programme comme celui-ci :

```
int PIN_LED_Rouge1= 7 ;
int PIN_LED_Rouge2= 8 ;
int PIN_LED_Rouge3= 9 ;
Int count =0;
void setup()
{
 pinMode(PIN_LED_Rouge1, OUTPUT);
 pinMode(PIN_LED_Rouge2, OUTPUT);
 pinMode(PIN_LED_Rouge3, OUTPUT);
 Serial.begin(9600); // communications à 9600 baud (char /second)
}
void loop()
{
          digitalWrite(PIN_LED_Rouge1, HIGH);
       digitalWrite(PIN_LED_Rouge2, HIGH);
       digitalWrite(PIN_LED_Rouge3, HIGH);
          delay(500);
       digitalWrite(PIN_LED_Rouge1, LOW);
       digitalWrite(PIN_LED_Rouge2, LOW);
       digitalWrite(PIN_LED_Rouge3, LOW);
          delay(500);
     Serial.print("Count is ");
          Serial.println(count);
          count++;
}
```

I.3.5 Manipulation 5 : Un seu1 feu tricolore

Le but de cette manipulation est de simuler la commande d'un seul feu tricolore à l'aide de la carte Arduino. Le fonctionnement du feu tricolore est détaillé comme suit :

1) Allumer le feu vert pendant 5 secondes.
2) Allumer le feu orange pendant 4 secondes.
3) Allumer le feu rouge pendant 1 secondes.
4) Reprendre le cycle du début.

I.3.6 Manipulation 6 : Deux feux tricolores synchronisés

Dans cette manipulation, vous allez créer une simulation réaliste de feux de croisement composés de deux feux tricolores synchronisés. Ces deux feux vont permettre de réguler la circulation d'un carrefour à deux voies, comprenant une route principale et une route secondaire.

Figure 15. Deux feux tricolores synchronisés.

Le fonctionnement des deux feux tricolores est détaillé comme suit :

1) Allumer le feu vert pendant 30 secondes sur la route principale et le rouge sur la route secondaire.
2) Allumer le feu orange pendant 05 secondes sur la route principale et toujours le rouge sur la route secondaire.
3) Allumer le feu rouge pendant 25 secondes sur la route principale et le feu vert sur la route secondaire.
4) Allumer le feu orange pendant 05 secondes sur la route secondaire et toujours le rouge sur la route principale.
5) Reprendre le cycle du début

Travaux Pratiques N° 2:

Contrôle d'un Système d'Éclairage avec des Boutons-Poussoirs et des Relais

II.1 Objectif

- Maîtriser l'utilisation des boutons-poussoirs en tant qu'entrées numériques sur une carte Arduino.
- Comprendre le fonctionnement et l'interfaçage des relais pour contrôler des charges électriques plus importantes.
- Mettre en œuvre un système de commande d'éclairage, permettant d'allumer et d'éteindre un éclairage avec les boutons-poussoirs.

II.2 Introduction

II.2.1 Bouton-poussoir

Les boutons-poussoirs sont de simples interrupteurs momentanés utilisés pour détecter des pressions manuelles. Dans de nombreuses applications électroniques, ils jouent un rôle essentiel en permettant aux utilisateurs d'interagir avec des appareils.

Vous trouverez ci-dessous les deux schémas (figure 16 et figure 17) de raccordement de bouton-poussoir avec Arduino en pull-up et pull-down.

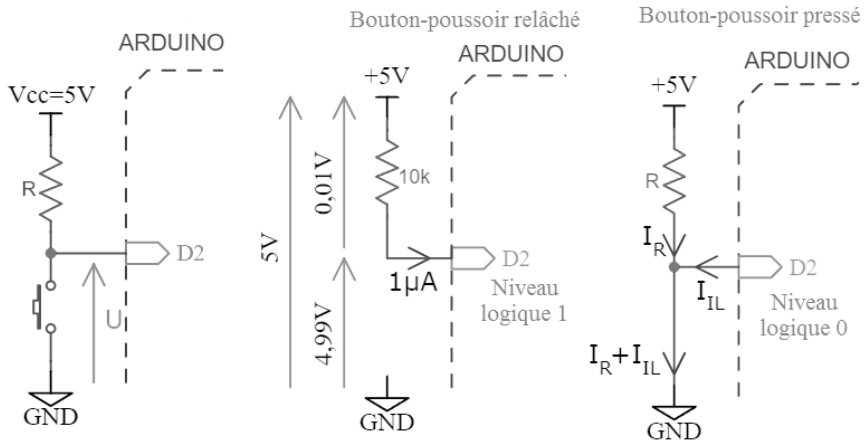


Figure 16. Bouton-poussoir avec une résistance de tirage vers le haut (pull-up).

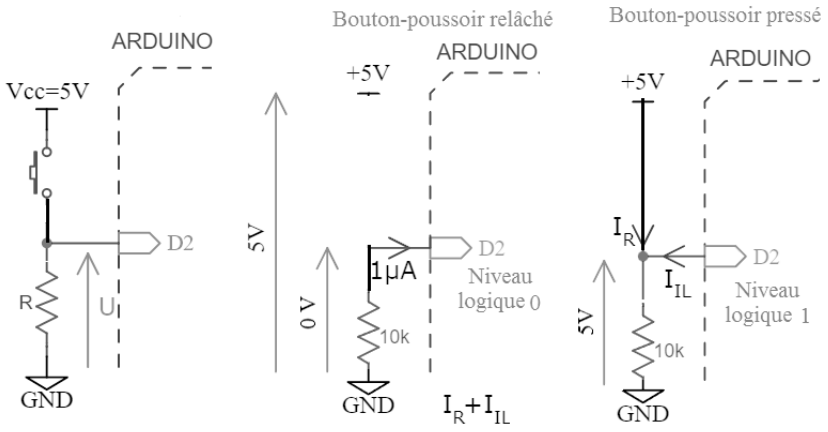


Figure 17. Bouton-poussoir avec une résistance de tirage vers le bas (pull-down).

II.2.2 Relai

Les relais sont des composants électromécaniques largement utilisés dans les systèmes électroniques et électriques pour contrôler des charges plus importantes à partir de signaux de commande faibles. Ils agissent comme des interrupteurs électriques qui peuvent ouvrir ou fermer un circuit en réponse à un signal électrique. Les relais offrent une solution fiable pour isoler des circuits de faible puissance des circuits de puissance élevée, protégeant ainsi les composants sensibles et les microcontrôleurs.

La figure suivante (figure 18) illustre la manière dont les relais fonctionnent, leur symbole électronique, ainsi que la représentation d'un relais en shield.

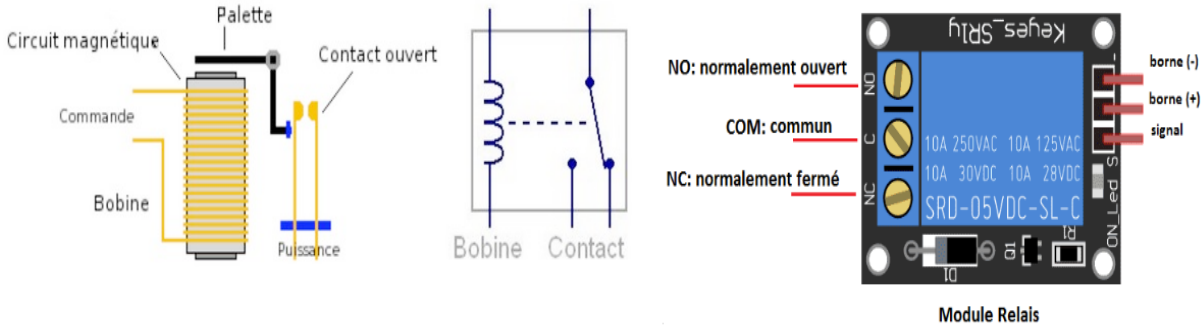


Figure 18. Fonctionnement de relai, le symbole électronique de relai et leur bronchement.

A) Schéma de Raccordement de relai

Le schéma de raccordement ci-dessous (figure 19) illustre comment connecter un relais à une carte Arduino Mega.et le circuit de puissance.

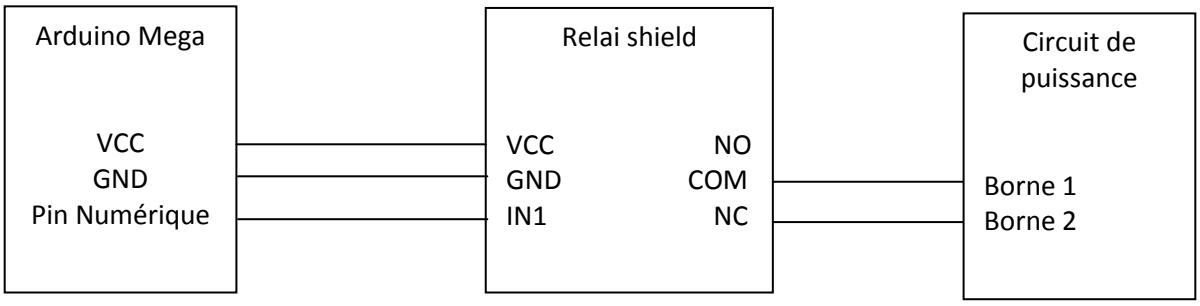


Figure 19. Schéma de raccordement de relai avec Arduino Mega

II.3 Travail à réaliser

II.3.1 Manipulation 1 : Commandement d'une LED par bouton poussoir

Dans cette manipulation, vous apprendrez à contrôler une LED à l'aide d'un bouton-poussoir. Nous configurerons le bouton-poussoir en tant qu'entrée numérique sur Arduino pour détecter les pressions et allumer/éteindre la LED en conséquence. Vous découvrirez également comment gérer les rebonds électroniques pour un contrôle fiable.

A. Matériel requis

- Une carte Arduino (par exemple, Arduino Mega)
- Une LED
- Un bouton-poussoir
- Une résistance de 220 ohms (pour protéger la LED)
- Deux résistances de 10 kohms (pour le pull-up ou le pull-down)
- Deux fils de connexion (jumper wires)

B. Étapes de la manipulation

Pour commande la LED par un bouton-poussoir en utilisant une carte Arduino, suivez les étapes ci-dessous :

Étape 1 : Configuration matérielle

Réaliser le circuit suivant (voir figure 20):

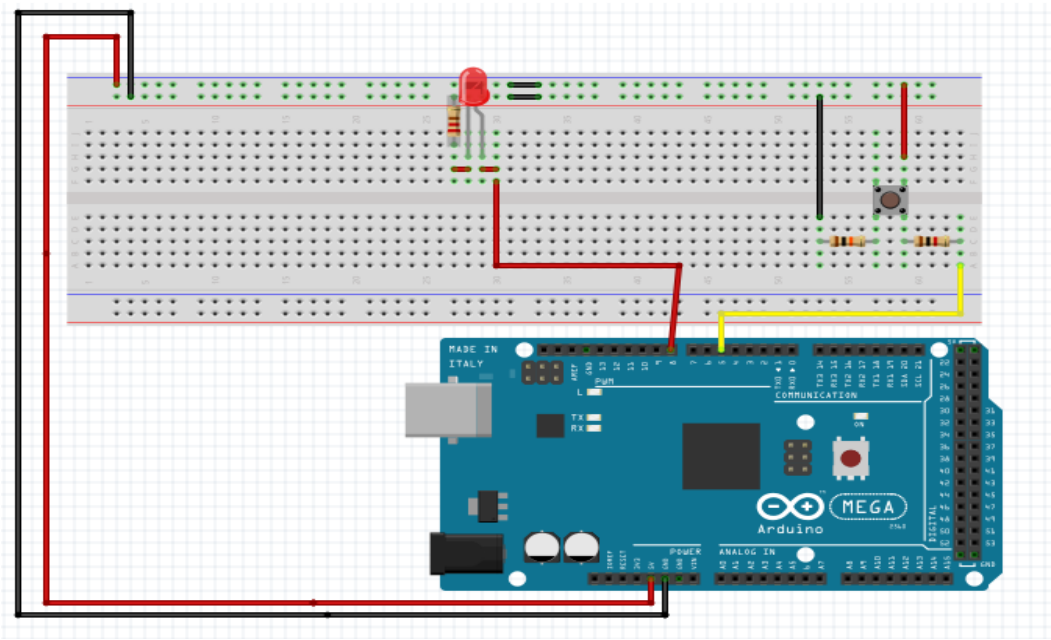


Figure 20. Circuit électronique «Commandement une LED par bouton poussoir»

- Connectez une LED (anode) avec une résistance de limitation de courant (220 ohms) à une broche numérique de l'Arduino (par exemple, D8).
- Connectez l'autre patte de la LED à la masse (GND) de l'Arduino.
- Connectez une patte du bouton poussoir à une autre broche numérique de l'Arduino (par exemple, D5).
- Connectez l'autre patte du bouton poussoir à la masse (GND) de l'Arduino.

Étape 2 : Programmation de l'Arduino

- Utilisez la fonction pinMode() pour configurer la broche D8 en mode sortie pour la LED et D5 en mode entrée pour le bouton poussoir.
- Dans la boucle loop() :
- Utilisez digitalRead() pour lire l'état de la broche D5 (le bouton poussoir).
- Si le bouton est enfoncé (état LOW à cause de la résistance de pull-down), utilisez digitalWrite() pour allumer ou éteindre la LED en fonction de son état actuel.
- Utilisez une courte pause (delay()) pour éviter des lectures erratiques dues aux rebonds du bouton.

I.3.2 Manipulation 2 : Affichage de l'état du bouton poussoir dans le serial

Le but de cette manipulation est de lire l'état d'un bouton poussoir connecté à une broche de l'Arduino et d'afficher son état (appuyé ou relâché) dans le moniteur série (serial monitor) de l'IDE Arduino.

Le code permettant de lire l'état du bouton poussoir connecté à une broche de l'Arduino est le suivant :

```
const int buttonPin = 2; // Broche numérique 2 utilisée pour lire l'état du bouton poussoir

void setup() {
  pinMode(buttonPin, INPUT_PULLUP); // Définir la broche du bouton poussoir en entrée avec une résistance de pull-up interne activée
  Serial.begin(9600); // Initialiser la communication série avec une vitesse de 9600 bps
}

void loop() {
  int buttonState = digitalRead(buttonPin); // Lire l'état du bouton poussoir (HIGH ou LOW)

  if (buttonState == LOW) {
    Serial.println("Bouton appuye"); // Afficher "Bouton appuyé" dans le moniteur série lorsque le bouton est appuyé
  } else {
    Serial.println("Bouton relache"); // Afficher "Bouton relâché" dans le moniteur série lorsque le bouton est relâché
  }

  delay(100); // Délai pour éviter les rebonds du bouton poussoir
}
```

Lorsqu'un bouton poussoir est pressé ou relâché, les contacts mécaniques peuvent rebondir plusieurs fois avant de se stabiliser dans un état. Cela peut provoquer des lectures erronées et des actions indésirables dans votre code.

Pour éviter ce problème, vous pouvez mettre en place un mécanisme anti-rebonds dans votre code. Voici une solution simple à ce problème en utilisant la fonction **millis**() pour gérer le temps entre les lectures du bouton :

```
const int buttonPin = 2; // Broche numérique 2 utilisée pour lire l'état du bouton poussoir
const unsigned long debounceDelay = 50; // Délai en millisecondes pour gérer les rebonds

int buttonState = HIGH; // État actuel du bouton (HIGH ou LOW)
```

```
int lastButtonState = HIGH; // État précédent du bouton (HIGH ou LOW)
unsigned long lastDebounceTime = 0; // Dernier temps où le bouton a été lu

void setup() {
  pinMode(buttonPin, INPUT_PULLUP); // Définir la broche du bouton poussoir en entrée avec une résistance de pull-up interne activée
  Serial.begin(9600); }
void loop() {
  // Lire l'état du bouton poussoir avec gestion des rebonds
  int reading = digitalRead(buttonPin);

  // Vérifier si l'état du bouton a changé et attendre le délai anti-rebonds
  if (reading != lastButtonState) { lastDebounceTime = millis();}

  // Si le délai anti-rebonds s'est écoulé sans rebond, mettre à jour l'état du bouton
  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (reading != buttonState) {  buttonState = reading;
      // Afficher l'état du bouton dans le moniteur série
      if (buttonState == LOW) {  Serial.println("Bouton appuye");}
                              else { Serial.println("Bouton relache");}
                                   }
                                                                  }
  lastButtonState = reading;
}
```

II.3.3 Manipulation 3 : Commandes des LEDs par 2 boutons

L'objectif de cette manipulation est de contrôler qutre LEDs à l'aide de deux boutons poussoirs connectés à l'Arduino. Lorsqu'un bouton est appuyé, la LED correspondante s'allume, et lorsqu'il est relâché, la LED s'éteint.

A. Matériel requis

- Arduino (ex. : Arduino Uno, Arduino Mega)
- Quatre LEDs (Light Emitting Diodes)
- Quatre résistances de limitation de courant (220 ohms recommandées)
- Deux boutons poussoirs
- Résistances de pull-up ou pull-down (10k ohms recommandées)
- Breadboard (plaque d'essai)
- Câbles de connexion jumper (mâle-mâle)

B. Étapes de la manipulation

Etape 1 : Préparation du matériel

- Connectez les quatre LEDs, chaque LED avec sa propre résistance de limitation de courant, à des broches numériques de l'Arduino (par exemple, D9, D10, D11, D8).
- Connectez les cathodes (pattes plus courtes) des LEDs à la masse (GND) de l'Arduino.
- Connectez les anodes (pattes plus longues) des LEDs aux résistances de limitation de courant, puis aux broches numériques respectives de l'Arduino.
- Connectez deux boutons poussoirs à deux broches numériques distinctes de l'Arduino (par exemple, D5 et D6).
- Placez des résistances de pull-up ou pull-down entre les broches des boutons et la tension d'alimentation (5V).

Le schéma du circuit pour la manipulation proposée sera illustré comme suit (figure 21) :

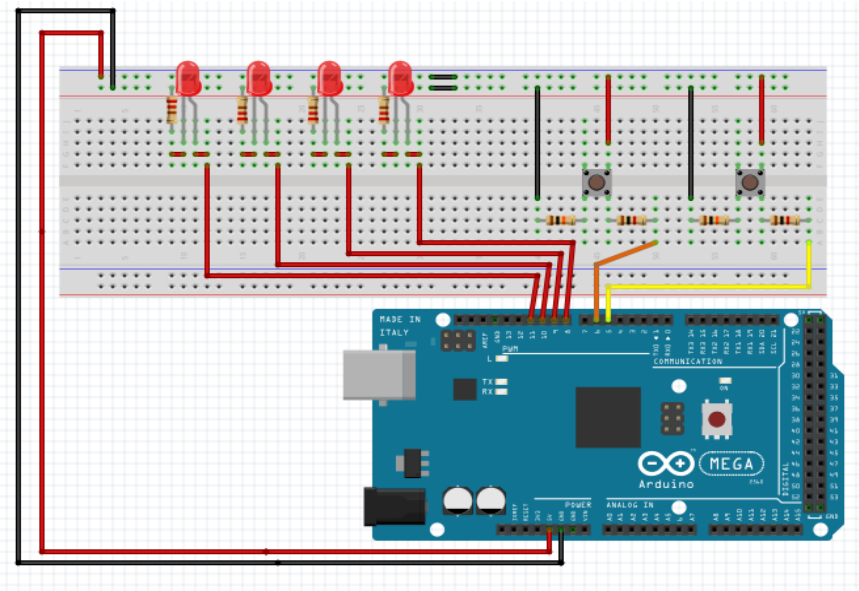


Figure 21. Commandes des LEDs par 2 boutons

Etape 2 : Programmation de l'Arduino

Pour contrôler les LEDs qui sont connectées à une carte Arduino à l'aide de boutons poussoirs, vous avez la possibilité d'utiliser un programme similaire à celui-ci :

```
int PIN_LED_Rouge1 = 8;
int PIN_LED_Rouge2 = 9;
int PIN_LED_Rouge3 = 10;
int PIN_LED_Rouge4 = 11;
int bouton_poussoir1 = 5;
int bouton_poussoir2 = 6;
void setup() {
  pinMode(PIN_LED_Rouge1, OUTPUT);pinMode(PIN_LED_Rouge2, OUTPUT);
pinMode(PIN_LED_Rouge3, OUTPUT);pinMode(PIN_LED_Rouge4, OUTPUT);
pinMode(bouton_poussoir1, INPUT);pinMode(bouton_poussoir2, INPUT);
}
void loop() {
   if(digitalRead(bouton_poussoir1)==HIGH)
  {digitalWrite(PIN_LED_Rouge1,HIGH);
  digitalWrite(PIN_LED_Rouge2,HIGH);
  }else{
  digitalWrite(PIN_LED_Rouge1,LOW);
  digitalWrite(PIN_LED_Rouge2,LOW);
  }
    if(digitalRead(bouton_poussoir2)==HIGH)
  {digitalWrite(PIN_LED_Rouge3,HIGH);
  digitalWrite(PIN_LED_Rouge4,HIGH);
  }else{
  digitalWrite(PIN_LED_Rouge3,LOW);
  digitalWrite(PIN_LED_Rouge4,LOW);
  }
}
```

Etape 3 : Expérimentation :

- Téléversez le code sur l'Arduino.
- Appuyez sur les boutons pour contrôler les LEDs correspondantes.

I.3.4 Manipulation 3 : Contrôle d'un Système d'Éclairage avec des Boutons-Poussoirs et des Relais

Dans cette manipulation, vous allez créer un système d'éclairage contrôlé par des boutons-poussoirs et des relais à l'aide d'une carte Arduino. Le but est d'allumer ou d'éteindre des lampes en utilisant les boutons, en faisant passer le courant à travers des relais.

A. Matériel requis

- Arduino (ex. : Arduino Uno, Arduino Mega) ;
- Breadboard (plaque d'essai) ;
- Câbles de connexion jumper (mâle-mâle) ;
- Boutons poussoirs (2) ;
- Relais (2) ;
- Lampes (2) ou LEDs avec résistances de limitation de courant ;
- Résistances de pull-up ou pull-down (10k ohms recommandées).

B. Étapes du projet

Etape 1 : Préparation du matériel

- Connectez les boutons poussoirs aux broches numériques de l'Arduino (par exemple, D2 et D3).
- Connectez les relais à des broches numériques de l'Arduino (par exemple, D4 et D5) pour le contrôle.
- Reliez les lampes (ou LEDs avec résistances) aux contacts des relais.

Etape 2 : Programmation de l'Arduino

- Utilisez pinMode() pour configurer les broches des boutons en mode entrée et les broches des relais en mode sortie.
- Dans la boucle loop() :
- Utilisez digitalRead() pour détecter l'état des boutons.
- Si un bouton est pressé, utilisez digitalWrite() pour activer le relais correspondant, ce qui allumera la lampe.
- Si le bouton est relâché, utilisez à nouveau digitalWrite() pour désactiver le relais, éteignant ainsi la lampe.

Etape 3 : Expérimentation

- Téléversez le code sur l'Arduino.
- Appuyez sur les boutons pour allumer et éteindre les lampes contrôlées par les relais.

Travaux Pratiques N° 3:

Système de Suivi et de Traçabilité des Produits avec Capteurs RFID

III.1 Objectif

- Intégrer les capteurs RFID avec la carte Arduino Mega pour la lecture précise des données des produits.
- Afficher les informations de suivi (identifiants uniques, date et heure) sur l'écran LCD I2C pour une visualisation en temps réel.
- Transmettre les données de suivi via une connexion série pour une gestion efficace des informations.
- Développer des compétences en électronique embarquée, en programmation Arduino et en gestion de la traçabilité pour des applications industrielles et logistiques avancées.

III.2 Introduction

Dans cette partie, nous fournirons un aperçu des RFID et des LCD-I2C, en détaillant le schéma de raccordement avec Arduino Mega. De plus, nous expliquerons la bibliothèque utilisée pour chacun des composants et offrirons un exemple de code pour faciliter leur intégration dans des projets de suivi et de traçabilité des produits.

III.2.1 Lecteur RFID

Le module RFID-RC522 est un dispositif électronique largement utilisé pour la lecture et l'écriture des cartes RFID (Radio-Frequency Identification). Il s'agit d'un moyen pratique et efficace d'intégrer la technologie RFID dans des projets d'électronique embarquée, tels que le système de suivi et de traçabilité des produits. Ce module permet d'interagir avec des cartes RFID sans contact, ce qui en fait une solution idéale pour des applications telles que le contrôle d'accès, l'identification d'objets et bien d'autres.

A. Schéma de Raccordement de RFID avec Arduino Mega

Le module RFID-RC522 peut être facilement interfacé avec la carte Arduino Mega grâce à quelques connexions simples (figure22). Voici le schéma de raccordement pour connecter le module RFID-RC522 à l'Arduino Mega :

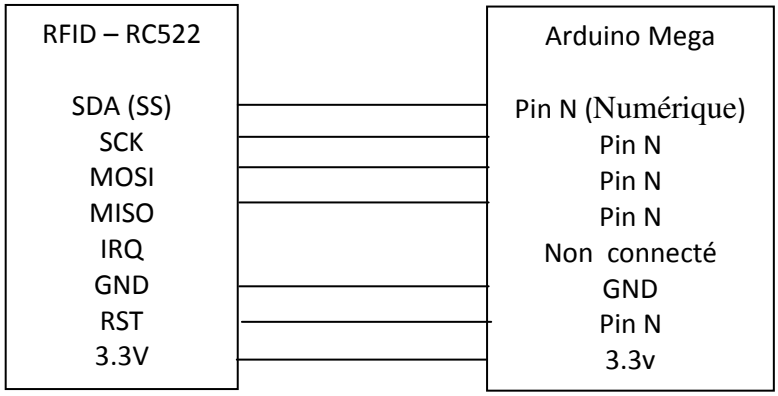


Figure 22. Schéma de raccordement de l'RFID avec Arduino Mega

A. Téléchargement de la Bibliothèque RFID

Pour utiliser le module RFID-RC522 avec l'Arduino Mega, vous devez installer la bibliothèque RFID-RC522. Voici comment la télécharger et l'installer :

- Ouvrez l'IDE Arduino sur votre ordinateur.
- Allez dans le menu "Croquis" (Sketch) > "Inclure une Bibliothèque" (Include Library) > "Gérer les Bibliothèques" (Manage Libraries).
- Dans la fenêtre "Gérer les Bibliothèques", recherchez "MFRC522" (le nom de la bibliothèque pour le module RFID-RC522).
- Cliquez sur le bouton "Installer" (Install) pour installer la bibliothèque.

Une fois la bibliothèque installée, vous pouvez maintenant utiliser le module RFID-RC522 dans vos projets Arduino Mega.

B. Exemple de Code pour Utiliser le RFID dans Arduino

Voici un exemple de code simple pour lire l'identifiant unique d'une carte RFID à l'aide du module RFID-RC522 et l'afficher via la connexion série (liaison UART) dans l'IDE Arduino :

```
#include <SPI.h> // SPI
#include <MFRC522.h> // RFID
#define SS_PIN 10
#define RST_PIN 9
MFRC522 rfid(SS_PIN, RST_PIN);  // Déclaration
byte nuidPICC[4];                          // Tableau contentent l'ID

void setup()
{
 SPI.begin();           // Init SPI bus
 rfid.PCD_Init();     // Init MFRC522
}

void loop()
{
  // Initialisé la boucle si aucun badge n'est présent
  if ( !rfid.PICC_IsNewCardPresent())
    return;
  // Vérifier la présence d'un nouveau badge
  if ( !rfid.PICC_ReadCardSerial())
    return;
    …
}
```

Assurez-vous que la bibliothèque RFID-RC522 est correctement installée avant de téléverser ce code sur votre Arduino Mega. Une fois téléversé, ouvrez le moniteur série (Serial Monitor) dans l'IDE Arduino pour voir les identifiants uniques des cartes RFID détectées par le module RFID-RC522.Haut du formulaire

III.1.2 Afficheur LCD –I2C

Voici le schéma (figure 23) de raccordement pour connecter un afficheur LCD avec interface I2C à l'Arduino Mega :

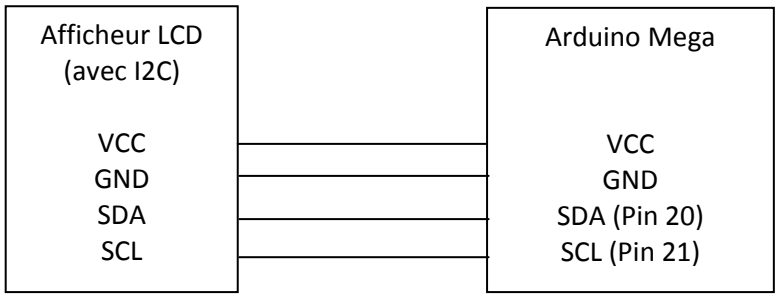


Figure 23. Schéma de Raccordement de l'LCD –I2C avec Arduino Mega

A. Téléchargement de la Bibliothèque LCD-I2C

Pour utiliser l'afficheur LCD avec interface I2C dans vos projets Arduino Mega, vous devrez télécharger et installer la bibliothèque LiquidCrystal_I2C. Voici comment procéder :

- Ouvrez l'IDE Arduino sur votre ordinateur.
- Allez dans le menu "Croquis" (Sketch) > "Inclure une Bibliothèque" (Include Library) > "Gérer les Bibliothèques" (Manage Libraries).
- Dans la fenêtre "Gérer les Bibliothèques", recherchez "LiquidCrystal_I2C" (le nom de la bibliothèque pour l'afficheur LCD avec interface I2C).
- Cliquez sur le bouton "Installer" (Install) pour installer la bibliothèque.
- Après avoir installé la bibliothèque LiquidCrystal_I2C, vous pourrez utiliser l'afficheur LCD avec interface I2C dans vos projets Arduino Mega.

B. Exemple de Code pour Utiliser l'Afficheur LCD - I2C

Voici un exemple de code simple pour afficher "Hello, World!" sur l'afficheur LCD avec interface I2C :

```
#include <LiquidCrystal_I2C_AvrI2C.h>
LiquidCrystal_I2C_AvrI2C lcd(0x27, 16, 2);
void setup()
{
  lcd.begin();
  lcd.backlight();
  lcd.setCursor(0, 0);
}

void loop()
{
  lcd.clear();
  delay(1000);
  lcd.print("Hello, world!");
  lcd.setCursor(0, 1);
  lcd.print("mutlu aysu");
  delay(1000);
}
```

Assurez-vous que la bibliothèque LiquidCrystal_I2C est correctement installée avant de téléverser ce code sur votre Arduino Mega. Une fois le code téléversé, vous verrez le message "Hello, World!" s'afficher sur l'afficheur LCD avec interface I2C.

III.2 Travail à réaliser

Le travail demande est de mettre en œuvre un système de suivi et de traçabilité des produits en utilisant des capteurs RFID (Radio-Frequency Identification) avec une carte Arduino Mega. Le système devra détecter les mouvements d'objets équipés de balises RFID dans un environnement simulé de production ou de logistique, et afficher les données de suivi en temps réel sur un écran ou les transmettre à un ordinateur via une connexion série.

III.2.1 Matériel requis

Pour réaliser le projet de système de suivi et de traçabilité des produits basé sur la carte Arduino Mega, vous aurez besoin des éléments suivants :

- **Carte Arduino Mega :** La carte Arduino Mega est la principale plateforme matérielle qui exécutera le programme et interagira avec les capteurs et l'écran LCD.
- **Module RFID (RFID-RC522 ou équivalent) :** Le module RFID est essentiel pour lire les étiquettes RFID des produits. Choisissez un module RFID compatible avec la carte Arduino Mega et assurez-vous qu'il est correctement câblé.
- **Écran LCD 16x2 avec interface I2C :** L'écran LCD permettra d'afficher les informations des produits lus à partir des étiquettes RFID. Optez pour un écran LCD 16x2 avec une interface I2C pour réduire le nombre de broches utilisées sur la carte Arduino Mega.
- **Câbles de connexion :** Vous aurez besoin d'une variété de câbles pour connecter les composants entre eux. Des fils de connexion mâle-mâle et mâle-femelle seront nécessaires pour les connexions entre la carte Arduino, le module RFID, l'écran LCD et les broches.
- **Breadboard (plaque d'essai) :** Une plaque d'essai peut être utilisée pour effectuer les connexions temporaires et faciliter le prototypage du circuit.
- **Ordinateur avec logiciel de surveillance série :** Un ordinateur avec l'IDE Arduino installé et un logiciel de surveillance série, tel que l'Arduino Serial Monitor, sera nécessaire pour afficher les données de suivi transmises par l'Arduino.

III.2.2 Étapes du projet

Les étapes à suivre pour la réalisation d'un système de suivi et de traçabilité des produits basé sur la technologie RFID ainsi que leur schéma électronique (figure 24) sont décrites comme suit:

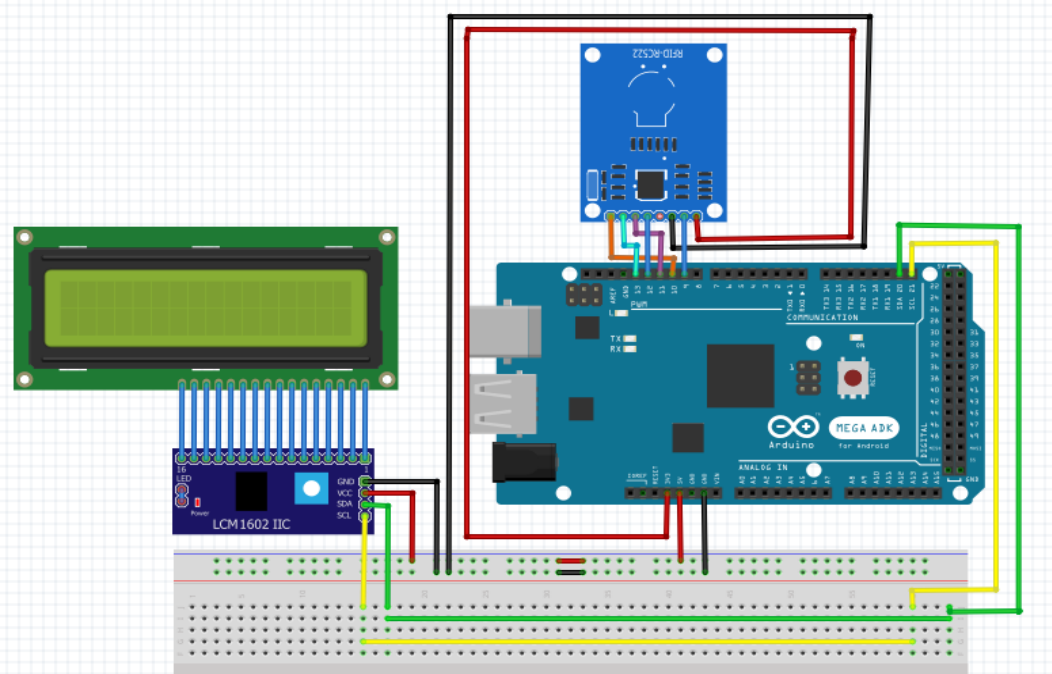


Figure 24. Schéma électronique « Système de Suivi et de Traçabilité des Produits avec Capteurs RFID »

Étape 1 : Préparation du Matériel

Avant de démarrer le projet, assurez-vous d'avoir tous les composants matériels nécessaires, tels que la carte Arduino Mega, le module RFID, l'écran LCD, les câbles de connexion et la plaque d'essai. Vérifiez également que les bibliothèques requises pour le module RFID et l'écran LCD sont installées dans l'IDE Arduino.

Étape 2 : Configuration du Logiciel

Démarrez l'IDE Arduino sur votre ordinateur et créez un nouveau projet. Importez les bibliothèques nécessaires pour le module RFID et l'écran LCD dans votre projet. Assurez-vous que la configuration du projet est compatible avec la carte Arduino Mega.

Étape 3 : Initialisation du Système

Dans la fonction **setup**() de votre code Arduino, initialisez le module RFID et l'écran LCD en configurant les broches d'entrée/sortie appropriées. Cette étape permet de mettre en place les éléments nécessaires pour le bon fonctionnement du système.

Étape 4 : Lecture des Données RFID

Utilisez la bibliothèque RFID pour détecter et lire les données des étiquettes RFID. Lorsqu'une étiquette RFID est détectée, capturez l'identifiant unique du produit ainsi que la date et l'heure de la lecture. Enregistrez ces informations pour pouvoir les afficher et les transmettre ultérieurement.

Étape 5 : Affichage sur l'Écran LCD

Programmez l'affichage des informations du produit (ID unique, date et heure) sur l'écran LCD. Assurez-vous que l'affichage est mis à jour chaque fois qu'une nouvelle lecture RFID est effectuée. Cela permettra de visualiser les données de suivi en temps réel.

Étape 6 : Transmission des Données en Série

Établissez une connexion série (liaison UART) entre la carte Arduino Mega et l'ordinateur. Transmettez les données de suivi (ID unique du produit, date et heure) à l'ordinateur via cette connexion série. Utilisez le logiciel de surveillance série (Arduino Serial Monitor) pour vérifier que les données sont correctement transmises.

Étape 7 : Test et Vérification

Testez le système en plaçant différentes étiquettes RFID à proximité du module RFID. Vérifiez que les informations du produit sont affichées correctement sur l'écran LCD et transmises avec précision à l'ordinateur via la connexion série. Assurez-vous également que le système fonctionne de manière fiable et stable.

Étape 8 : Documentation et Améliorations

Une fois le projet fonctionnel, documentez soigneusement le code, les connexions matérielles et les résultats obtenus. Si vous le souhaitez, explorez des améliorations potentielles telles que le stockage des données dans une carte SD, l'intégration avec une base de données ou l'ajout de fonctionnalités avancées de traçabilité.

Travaux Pratiques N° 4:

Système de Surveillance de la Chaîne Logistique avec Capteurs Ultrason

IV.1 Objectif

- Se familiariser avec l'utilisation d'un capteur ultrason pour détecter la distance entre le capteur et les objets simulée.
- Acquérir des compétences pratiques en intégration et en programmation d'un afficheur LCD avec interface I2C pour afficher les données de distance en temps réel.
- Comprendre le concept de surveillance automatisée dans le contexte de la logistique et développer des compétences en électronique embarquée en concevant un système fonctionnel de surveillance.

IV.2 Introduction

Le détecteur de distance ultrason, également connu sous le nom de capteur ultrason, est largement utilisé pour mesurer la distance entre le capteur et un objet en se basant sur le temps qu'un signal ultrasonique met pour voyager jusqu'à l'objet et revenir. Voici comment l'utiliser avec Arduino Mega.

A. Schéma de raccordement de capteur Ultrason avec Arduino Mega

Le schéma (figure 25) de raccordement avec Arduino Mega montre comment connecter correctement le capteur ultrason aux broches de l'Arduino Mega.

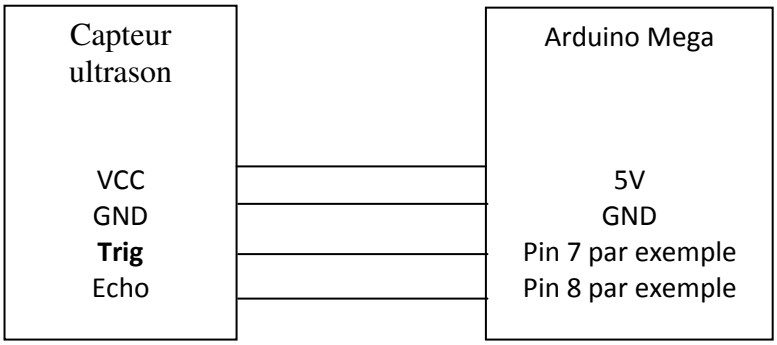


Figure 25. Schéma de raccordement de capteur Ultrason avec Arduino Mega

B. Téléchargement de la Bibliothèque de capteur ultrason

Pour utiliser le capteur ultrason avec l'Arduino Mega, vous devez installer la bibliothèque NewPing. Voici comment la télécharger et l'installer :

- Ouvrez l'IDE Arduino sur votre ordinateur.
- Allez dans le menu Croquis > Inclure une bibliothèque > Gérer les bibliothèques.
- Recherchez "NewPing" dans la barre de recherche.
- Cliquez sur "Installer" pour installer la bibliothèque NewPing, qui facilite l'utilisation des capteurs ultrason avec Arduino.

C. Exemple de code pour Utiliser le capteur ultrason dans Arduino

Voici un exemple de code simple qui utilise le capteur ultrason pour mesurer la distance et afficher les résultats.

```
#include <NewPing.h>              // Inclure la bibliothèque NewPing

#define TRIGGER_PIN 7             // Broche de déclenchement du capteur ultrason
#define ECHO_PIN 8                // Broche d'écho du capteur ultrason
#define MAX_DISTANCE 200          // Distance maximale à mesurer en centimètres
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
                                  // Créer une instance du capteur ultrason
void setup() {
 Serial.begin(9600); // Initialiser la communication série
}

void loop() {
 delay(50); // Attendre un court instant
 unsigned int distance = sonar.ping_cm();          // Mesurer la distance en centimètres
 if (distance == 0) {
  Serial.println("Pas de détection");              // Aucun objet détecté
 } else {
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");                           // Afficher la distance mesurée
 }
}
```

IV.3 Travail à réaliser

Le travail demande est de concevoir un système de surveillance de la chaîne logistique en utilisant des capteurs de distance pour suivre la position et le mouvement d'objets simulés sur

une chaîne de production ou une ligne de transport. L'objectif est d'intégrer un écran LCD pour afficher les informations en temps réel.

IV.3.1 Matériel requis

Pour réaliser le projet de système de surveillance de la chaîne logistique, vous aurez besoin des éléments suivants :

- Carte Arduino Mega
- Capteurs de Distance (Ultrasons)
- Écran LCD 16x2 avec Interface I2C
- Câbles de Connexion
- Plaque d'Essai (Breadboard)
- Objets Simulés
- Ordinateur avec IDE Arduino

IV.3.2 Étapes du projet

Voici les étapes pour la mise en place du système de surveillance de la chaîne logistique basé sur le capteur ultrason, ainsi que son schéma électronique (figure 26) :

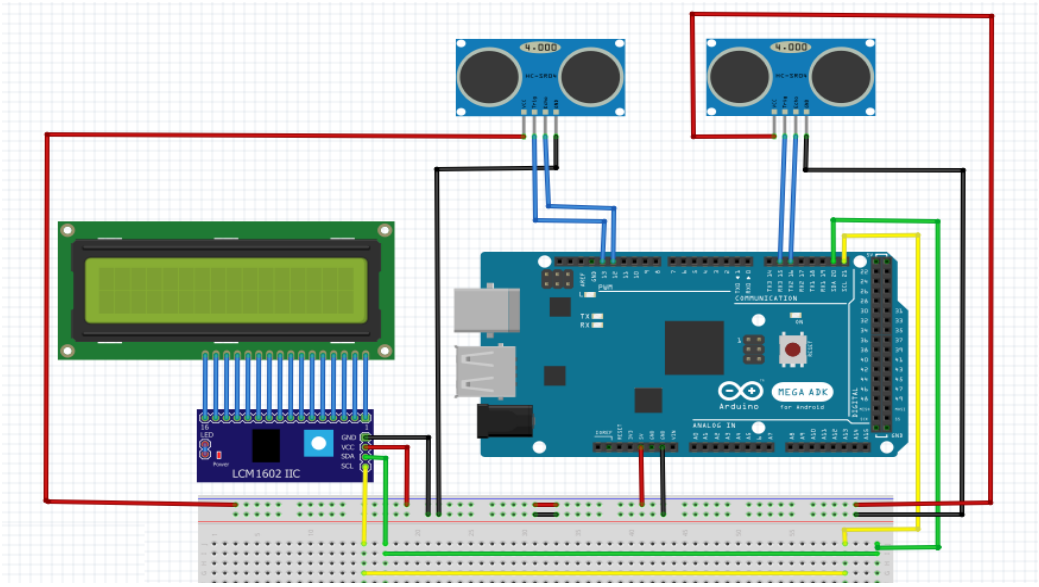


Figure 26. Schéma électronique «Système de Surveillance de la Chaîne basé sur Capteurs Ultrason»

Étape 1 : Préparation du Matériel

Assurez-vous d'avoir tous les composants nécessaires et vérifiez leur fonctionnement. Connectez l'écran LCD avec interface I2C à l'Arduino Mega selon le schéma de raccordement fourni.

Étape 2 : Montage des Capteurs

Placez les capteurs de distance (ultrasons) sur la chaîne logistique (ou sur le tapi roulons du laboratoire) pour surveiller les objets en mouvement. Assurez-vous que les capteurs sont correctement positionnés pour mesurer la distance entre le capteur et les objets simulés.

Étape 3 : Programmation Arduino

Développez le code Arduino pour lire les données des capteurs de distance. Utilisez les bibliothèques appropriées pour les capteurs et l'écran LCD. Programmez l'Arduino pour afficher les informations de distance mesurées sur l'écran LCD en temps réel.

Étape 4: Affichage en Temps Réel

Configurez l'écran LCD pour afficher les données de distance mesurées par les capteurs. Assurez-vous que les informations sont mises à jour régulièrement à mesure que les objets simulés se déplacent sur le tapi roulons.

Étape 5 : Simulation de Mouvement

Faites glisser les objets simulés le long de tapis roulon pour simuler leur mouvement. Observez comment les capteurs de distance détectent les objets et affichent les distances correspondantes sur l'écran LCD.

Étape 6 : Tests et Vérifications

Testez le système dans différentes situations pour vous assurer que les capteurs de distance fonctionnent correctement et que les informations affichées sur l'écran LCD sont précises et en temps réel.

Travaux Pratiques N° 5:

Contrôle Automatisé de la Température et de l'Humidité Actionné par un Ventilateur/Moteur à Courant Continu

V.1 Objectif

- Se familiariser avec les capteurs de température et d'humidité tels que le DHT11.
- Apprendre le contrôle d'un moteur en courant continu en utilisant l'Arduino Mega et le shield L293D.
- Comprendre l'utilisation de l'afficheur LCD en mode I2C avec la carte Arduino Mega.
- La régulation environnementale et du contrôle automatique pour maintenir les paramètres dans des plages définies.

V.2 Introduction

V.2.1 Capteurs DHT11

Les capteurs DHT11 Arduino permettent une mesure précise de la température et de l'humidité ambiante. Ces capteurs sont faciles à utiliser et fournissent des données en temps réel, ce qui les rend idéaux pour la surveillance environnementale. Nous intégrerons ces capteurs à l'Arduino Mega, qui jouera le rôle de cerveau pour le contrôle automatisé.

A. Schéma de Raccordement de DHT11 avec Arduino Mega

Le Capteur DHT11 dispose de trois broches : VCC (Alimentation), DATA (Données) et GND (Masse). Pour le raccorder à l'Arduino Mega, effectuez les connexions suivantes (figure 27):

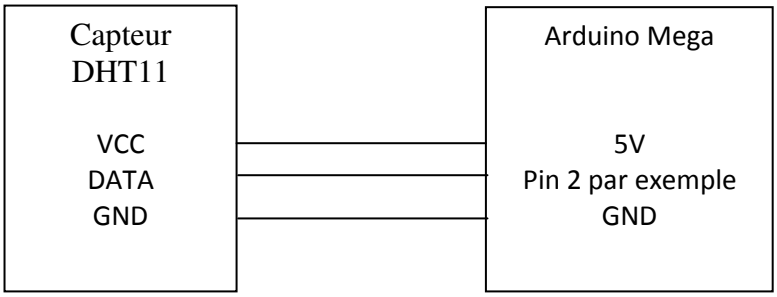


Figure 27. Schéma de Raccordement de capteur DHC11 avec Arduino Mega

B. Téléchargement de la Bibliothèque pour l'utilisation de capteurDHT11

Avant d'utiliser le capteur DHT11, vous devez télécharger la bibliothèque DHT correspondante dans l'IDE Arduino. Voici les étapes à suivre :

- Ouvrez l'IDE Arduino sur votre ordinateur.
- Allez dans "Croquis" (Sketch) > "Inclure une Bibliothèque" (Include Library) > "Gérer les Bibliothèques" (Manage Libraries).
- Recherchez "DHT" dans la boîte de recherche.
- Cliquez sur la bibliothèque "DHT sensor library by Adafruit" et cliquez sur le bouton "Installer" (Install).

C. Exemple de Code pour Utiliser le DHC11 dans Arduino

Voici un exemple de code pour utiliser le DHC11 :

```
#include <DHT.h>
#define DHTPIN 2       // Définir la broche à laquelle est connecté le capteur DHT11 (ici, broche 2)
#define DHTTYPE DHT11   // Définir le type de capteur (DHT11 dans ce cas)
DHT dht(DHTPIN, DHTTYPE);
void setup() {
 Serial.begin(9600); // Initialisation de la communication série
 dht.begin();       // Initialisation du capteur DHT11
}
void loop() {
 // Lecture des valeurs de température et d'humidité
 float temperature = dht.readTemperature();
 float humidity = dht.readHumidity();
 // Vérifier si la lecture est réussie
 if (isnan(temperature) || isnan(humidity)) {
  Serial.println("Erreur lors de la lecture du capteur DHT11 !");
  return;
 }
 // Affichage des valeurs lues
 Serial.print("Température: "); Serial.print(temperature); Serial.print(" °C\t");
 Serial.print("Humidité: "); Serial.print(humidity); Serial.println(" %");
 delay(2000); // Attendre 2 secondes avant la prochaine lecture
}
```

V.2.2 Moteur a courant continu et le Shield L293D:

Le moteur à courant continu associé au shield L293D joue un rôle essentiel dans le Contrôle Automatisé de la Température et de l'Humidité. Cette configuration permet de réguler la vitesse du ventilateur ou du moteur en fonction des mesures de température et d'humidité ambiante, assurant ainsi des conditions optimales dans l'environnement ciblé.

A. Schéma de Raccordement le Shield L293D avec les moteurs courant continu et un cerveau moteur

Voici le schéma (figure 28) de raccordement du shield L293D avec les moteurs courant continus et un cerveau moteur:

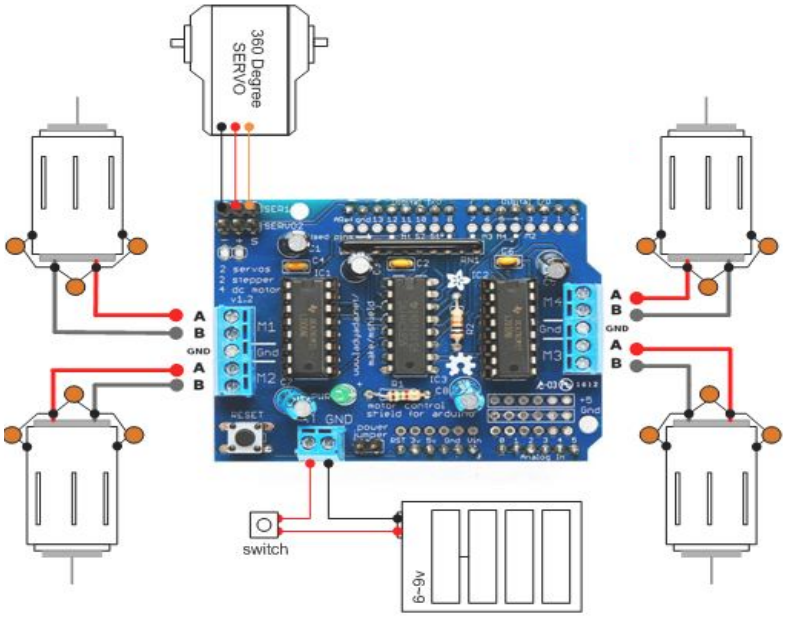


Figure 28. Schéma de Raccordement le Shield L293D avec les moteurs CC

B. Téléchargement de la Bibliothèque pour l'utilisation de shield L293D

Pour télécharger la bibliothèque AFMotor.h, suivez les étapes suivantes :

- Dans l'IDE Arduino, allez dans "Croquis" (Sketch) dans la barre de menu en haut de l'écran.
- Sélectionnez "Inclure une bibliothèque" (Include Library) dans le menu déroulant. Une liste de bibliothèques déjà installées sur votre système s'affichera.

- Si vous ne trouvez pas la bibliothèque AFMotor.h dans la liste, sélectionnez "Gérer les bibliothèques" (Manage Libraries) dans le menu déroulant. Cela ouvrira la bibliothèque de gestion des bibliothèques Arduino.
- Dans la bibliothèque de gestion des bibliothèques, vous verrez un champ de recherche en haut à droite. Tapez "Adafruit Motor Shield" dans ce champ de recherche.
- La bibliothèque "Adafruit Motor Shield" devrait apparaître dans les résultats de recherche. Cliquez sur le bouton "Installer" pour télécharger et installer la bibliothèque.

C. Exemple de Code pour Utiliser le Sheald L293D dans Arduino

Voici un exemple de code pour contrôler le moteur à courant continu :

```
// Inclure la bibliothèque de contrôle du shield L293D
#include <AFMotor.h>
// Créer un objet pour contrôler le moteur
AF_DCMotor motor(1); // Vous pouvez ajuster le numéro (1, 2, 3, ou 4) en fonction du port du L293D auquel le moteur est connecté

void setup() {  /*Rien à faire ici pour cet exemple*/ }
void loop() {
  // Faire tourner le moteur dans un sens pendant 2 secondes
  motor.setSpeed(200); // Définir la vitesse du moteur (0-255)
  motor.run(FORWARD);  // Faire tourner le moteur dans le sens horaire
  delay(2000);        // Attendre 2 secondes

  // Arrêter le moteur pendant 1 seconde
  motor.run(RELEASE);  // Arrêter le moteur
  delay(1000);        // Attendre 1 seconde

  // Faire tourner le moteur dans l'autre sens pendant 2 secondes
  motor.setSpeed(200); // Définir la vitesse du moteur (0-255)
  motor.run(BACKWARD); // Faire tourner le moteur dans le sens antihoraire
  delay(2000);        // Attendre 2 secondes

  // Arrêter le moteur pendant 1 seconde
  motor.run(RELEASE);  // Arrêter le moteur
  delay(1000);        // Attendre 1 seconde
}
```

Ce code utilise la bibliothèque AFMotor.h pour contrôler le shield L293D et le moteur à courant continu. Vous pouvez ajuster la vitesse du moteur en modifiant la valeur passée à la

fonction setSpeed (0-255). Assurez-vous de choisir le bon numéro de moteur (1, 2, 3 ou 4) en fonction du port du L293D auquel le moteur est connecté.

V.3 Travail à réaliser

Le travail vise à développer un système avancé de Contrôle Automatisé de la Température et de l'Humidité en utilisant le capteur DHT11 et un moteur à courant continu pour actionner un ventilateur. Le système ajustera automatiquement la vitesse du moteur en fonction des variations de température et d'humidité détectées par le capteur. De plus, un écran LCD-I2C affichera en temps réel les valeurs de température et d'humidité, et un menu interactif permettra à l'utilisateur d'ajuster les plages de température et d'humidité souhaitées en utilisant des boutons poussoirs.

V.3.1 Matériel requis

Pour réaliser ce projet de Contrôle Automatisé de la Température et de l'Humidité Actionné par un Ventilateur/Moteur à Courant Continu avec Affichage en Temps Réel sur un LCD-I2C et Menu Interactif par Boutons Poussoirs, vous aurez besoin du matériel suivant :

- Arduino Mega (ou tout autre microcontrôleur compatible)
- Capteur DHT11 (ou DHT22) pour la mesure de la température et de l'humidité
- Shield L293D ou autre module de contrôle de moteur à courant continu
- Moteur à courant continu avec hélice (pour le ventilateur)
- Écran LCD I2C 16x2 (ou 20x4) pour l'affichage en temps réel
- Boutons poussoirs (au moins 2) pour l'interaction du menu
- Résistances (pour les boutons poussoirs, si nécessaire)
- Breadboard ou plaque de prototypage
- Fils de connexion (jumpers)
- Alimentation externe pour le moteur à courant continu (si nécessaire)

V.3.2 Étapes du projet

Voici les étapes détaillées du projet en respectant le principe de tester chaque partie séparément :

Étape 1 : Montage Initial et Tests Individuels des Composants

1. Assemblez le capteur DHT11, le moteur à courant continu, le shield L293D, l'écran LCD I2C et les boutons poussoirs sur la breadboard.
2. Testez individuellement chaque composant pour s'assurer qu'ils fonctionnent correctement:
3. Vérifiez la lecture des données du capteur DHT11 sur le moniteur série.
4. Testez la commande du moteur à courant continu à l'aide du L293D.
5. Assurez-vous que l'écran LCD I2C affiche les informations correctement.
6. Vérifiez la détection des appuis sur les boutons poussoirs.

Le schéma de montage est présenté comme suite (figure 29) :

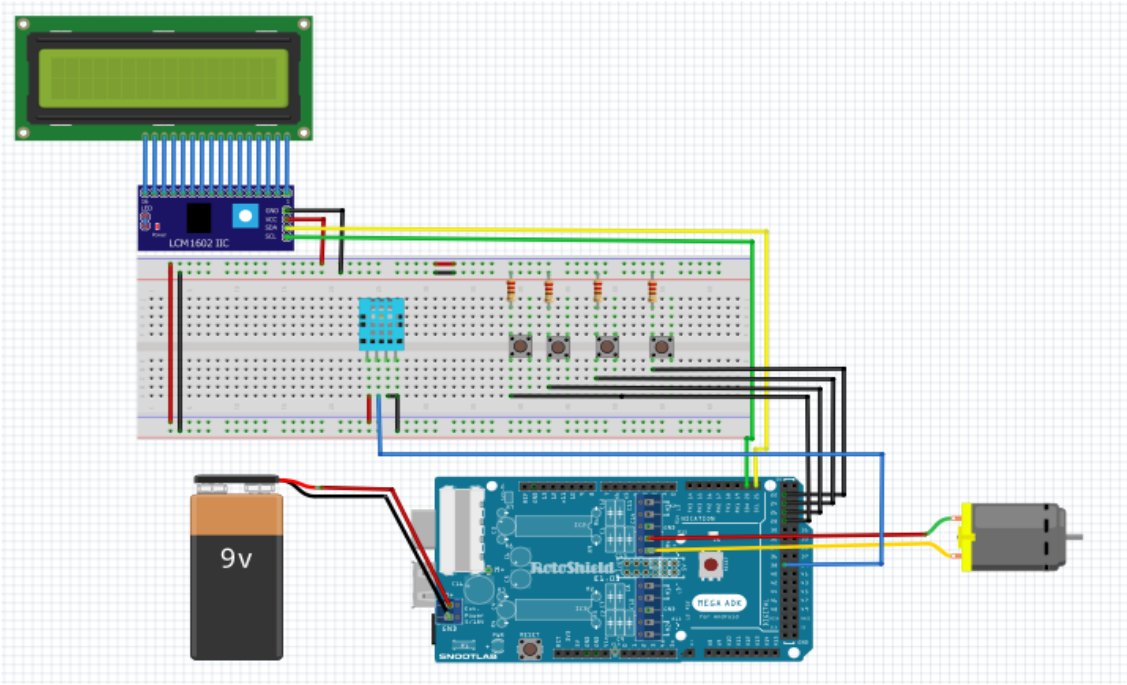


Figure 29. Schéma électronique «Contrôle Automatisé de la Température et de l'Humidité par Moteur CC »

»

Étape 2 : Programmation et Tests Initiaux

1. Ouvrez l'Arduino IDE et commencez à écrire le code en divisant les fonctionnalités en sections distinctes.
2. Programmez la lecture des données du capteur DHT11 et l'affichage sur le moniteur série.
3. Programmez le contrôle manuel du moteur à courant continu via le L293D.
4. Assurez-vous que chaque fonctionnalité est testée et fonctionne individuellement.

Étape 3 : Affichage en Temps Réel sur l'Écran LCD-I2C

1. Intégrez le code pour afficher les données de température et d'humidité sur l'écran LCD I2C
2. Testez cette fonctionnalité pour vous assurer que les valeurs sont correctement affichées en temps réel.

Étape 4 : Menu Interactif par Boutons Poussoirs

1. Programmez le système de menu interactif pour ajuster les plages de température et d'humidité souhaitées à l'aide des boutons poussoirs.
2. Testez chaque option du menu pour vous assurer que les ajustements se font correctement et que les valeurs sont reflétées sur l'écran LCD.

Voici une description des fonctions de ces boutons :

Bouton "+" (Augmenter) : Ce bouton permet à l'utilisateur d'augmenter la valeur des paramètres, tels que la plage de température ou d'humidité souhaitée. En appuyant sur ce bouton, l'utilisateur peut progressivement augmenter la valeur affichée sur l'écran LCD, ce qui ajustera les seuils de contrôle automatique.

Bouton "-" (Diminuer) : Contrairement au bouton "+", ce bouton permet de réduire la valeur des paramètres. En appuyant sur ce bouton, l'utilisateur peut abaisser progressivement la valeur affichée sur l'écran LCD, ce qui ajustera les seuils de contrôle automatique vers le bas.

Bouton "Valider" : Ce bouton est utilisé pour confirmer et enregistrer les ajustements effectués aux plages de température et d'humidité. Lorsque l'utilisateur est satisfait de la valeur affichée sur l'écran LCD après l'ajustement à l'aide des boutons "+" et "-", il peut appuyer sur ce bouton pour valider et enregistrer les nouvelles plages.

Bouton "Annuler" : Si l'utilisateur souhaite annuler les modifications apportées aux paramètres de contrôle sans enregistrer les changements, il peut appuyer sur ce bouton. Cela ramènera le système aux valeurs précédentes et annulera les ajustements en cours.

Étape 5 : Intégration Complète et Tests Finaux

1. Intégrez toutes les fonctionnalités du projet en combinant le code de lecture du capteur, de contrôle du moteur, d'affichage LCD et de menu interactif.
2. Effectuez des tests complets pour vous assurer que le système automatisé fonctionne harmonieusement :
3. Vérifiez que le moteur est contrôlé automatiquement en fonction des conditions de température et d'humidité.
4. Assurez-vous que les ajustements du menu sont reflétés correctement sur l'affichage LCD.

En suivant ces étapes méthodiques et en testant chaque composant et fonctionnalité individuellement, vous pouvez garantir que votre projet de Contrôle Automatisé de la Température et de l'Humidité sera fonctionnel et efficace dans son ensemble.

Liste des abréviations

CAN	Convertisseur Analogique Numérique
DC	Direct Current
E/S	entre / Sortie
EEPROM	Electrically Erasable Programmable Read-Only Memory
FTDI	Future Technology Devices International
Gnd	ground
I/O	Input/Output
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
MISO	Master In Slave Out
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MOSI	Master Out Slave In
PWM	Pulse Width Modulation
RF	radio frequency
RFID	Radio-Frequency Identification
RISC	Reduced Instruction Set Computer
SCK	Serial Clock
SCL	Serial Clock
SDA	Serial Data
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SS	Slave Select
USART	Universal Synchronous & Asynchronous Receiver Transmitter
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity

Références

John Boxall ,Arduino Workshop: A Hands-On Introduction with 65 Projects.2013.

Mark Geddes, Arduino Project Handbook: 25 Practical Projects to Get You Started.2016.

Pierre-Yves Rochat, EPFL.introduction au microcontrôleur. Ecole Polytechnique fédérale

de Lausanne. 2016

Massimo Banzi ,Getting Started with Arduino2015..

MEGNAFI, Hicham, ABDELLAOUI, Ghouti, et MR BRAHAMI, Mustapha Anwar. Systèmes à Microcontrôleur. 2019.

ABDELLAOUI, Ghouti et MEGNAFI, Hicham. Systèmes embarqés et temps réel (RaspberryPi). 2019.

BRAHAMI, Mustapha Anwar et MEGNAFI, Hicham. Polycopie TP Réseaux et Protocoles. 2022.

H. Megnafi, A. Ayad, W. Tabib, A. A Mouaziz, R. Ould Babaali, I. Medjhoud, Improved printing time by changing the mechanical part of the 3D printer, embedded system application, NewMat'21–1st International Conférence: New Trends on Innovative Construction Materials-ESSA-Tlemcen (Algeria)–22, 23 March, 2022.

M.A. Brahami, H. Megnafi, H. Boukeffous, O. Benlaldj, Design and implementation of an embedded system for effective attendance management, NewMat'21–1st International Conférence: New Trends on Innovative Construction Materials-ESSA-Tlemcen (Algeria)–22, 23 March, 2022.

https://www.arduino.cc/

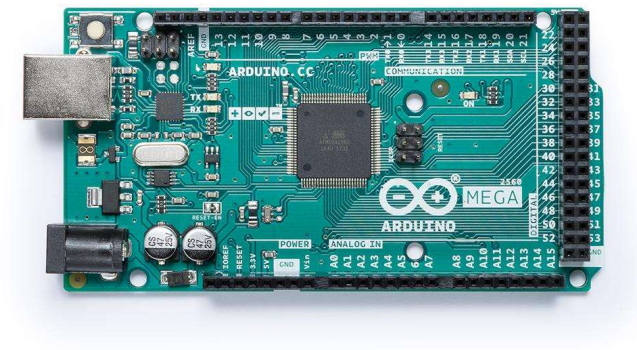
https://www.instructables.com/

https://learn.sparkfun.com/

https://www.cours-gratuit.com/cours-arduino/cours-et-exercices-corriges-arduino-pdf

http://handsontec.com/

Product Reference Manual
SKU: A000067



Description

Arduino® Mega 2560 is an exemplary development board dedicated for building extensive applications as compared to other maker boards by Arduino. The board accommodates the ATmega2560 microcontroller, which operates at a frequency of 16 MHz. The board contains 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a USB connection, a power jack, an ICSP header, and a reset button.

Target Areas

3D Printing, Robotics, Maker

Features

- **ATmega2560 Processor**
  - Up to 16 MIPS Throughput at 16MHz
  - 256k bytes (of which 8k is used for the bootloader)
  - 4k bytes EEPROM
  - 8k bytes Internal SRAM
  - 32 × 8 General Purpose Working Registers
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Four Programmable Serial USART
  - Controller/Peripheral SPI Serial Interface
- **ATmega16U2**
  - Up to 16 MIPS Throughput at 16 MHz
  - 16k bytes ISP Flash Memory
  - 512 bytes EEPROM
  - 512 bytes SRAM
  - USART with SPI master only mode and hardware flow control (RTS/CTS)
  - Master/Slave SPI Serial Interface
- **Sleep Modes**
  - Idle
  - ADC Noise Reduction
  - Power-save
  - Power-down
  - Standby
  - Extended Standby
- **Power**
  - USB Connection
  - External AC/DC Adapter
- **I/O**
  - 54 Digital
  - 16 Analog
  - 15 PWM Output

Contents

1 The Board

Arduino® Mega 2560 is a successor board of Arduino Mega, it is dedicated to applications and projects that require large number of input output pins and the use cases which need high processing power. The Arduino® Mega 2560 comes with a much larger set of IOs when we compare it with traditional Uno board considering the form factor of both the boards.

1.1 Application Examples

- **Robotics**: Featuring the high processing capacitity, the Arduino Mega 2560 can handle the extensive robotic applications. It is compatible with the motor controller shield that enables it to control multiple motors at an instance, thus making it perfect of robotic applications. The large number of I/O pins can accommodate many robotic sensors as well.
- **3D Printing**: Algorithms play a significant role in implementation of 3D printers. Arduino Mega 2560 has the power to process these complex algorithms required for 3D printing. Additionally, the slight changes to the code is easily possible with the Arduino IDE and thus 3D printing programs can be customized according to user requirements.
- **Wi-Fi**: Integrating wireless functionality enhances the utility of the applications. Arduino Mega 2560 is compatible with WiFi shields hence allowing the wireless features for the applications in 3D printing and Robotics.

1.2 Accessories

1.3 Related Products

- Arduino® Uno Rev 3
- Arduino® Nano
- Arduino® DUE without headers

2 Ratings

2.1 Recommended Operating Conditions

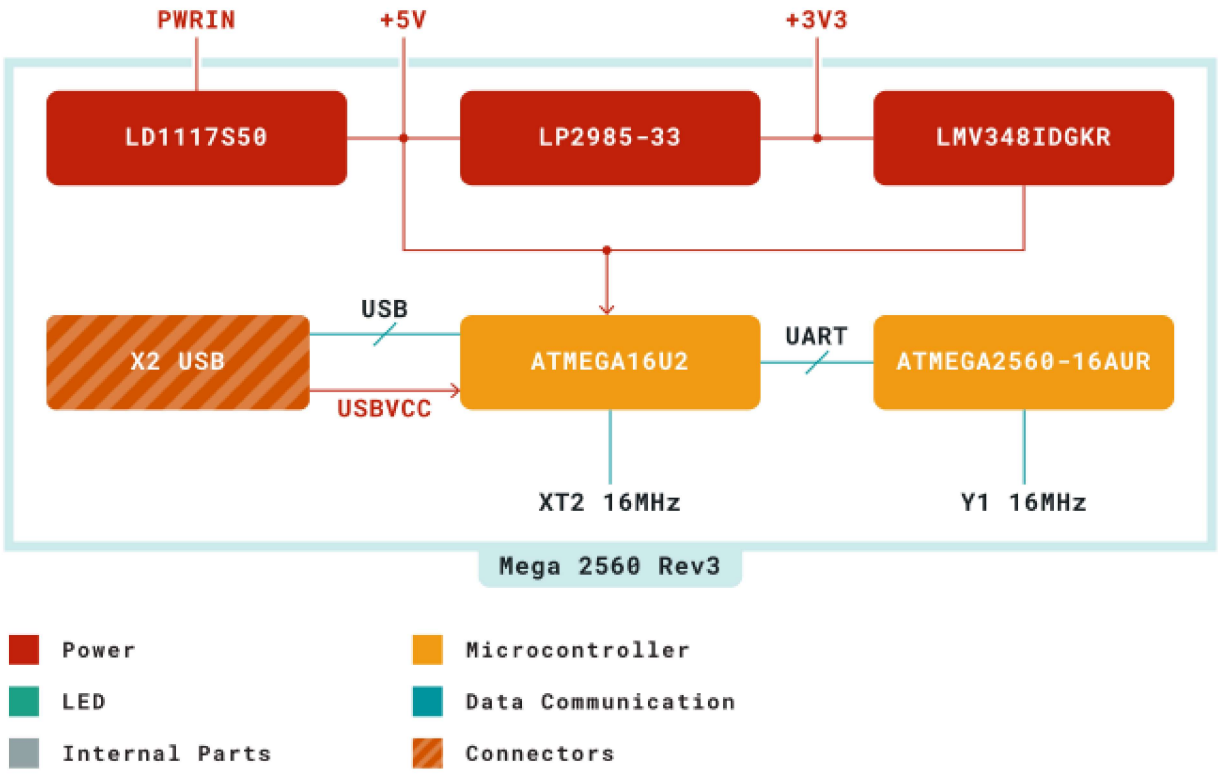
Symbol	Description	Min	Max
TOP	Operating temperature:	-40 °C	85 °C

2.2 Power Consumption

Symbol	Description	Min	Typ	Max	Unit
PWRIN	Input supply from power jack		TBC		mW
USB VCC	Input supply from USB		TBC		mW
VIN	Input from VIN pad		TBC		mW

3 Functional Overview

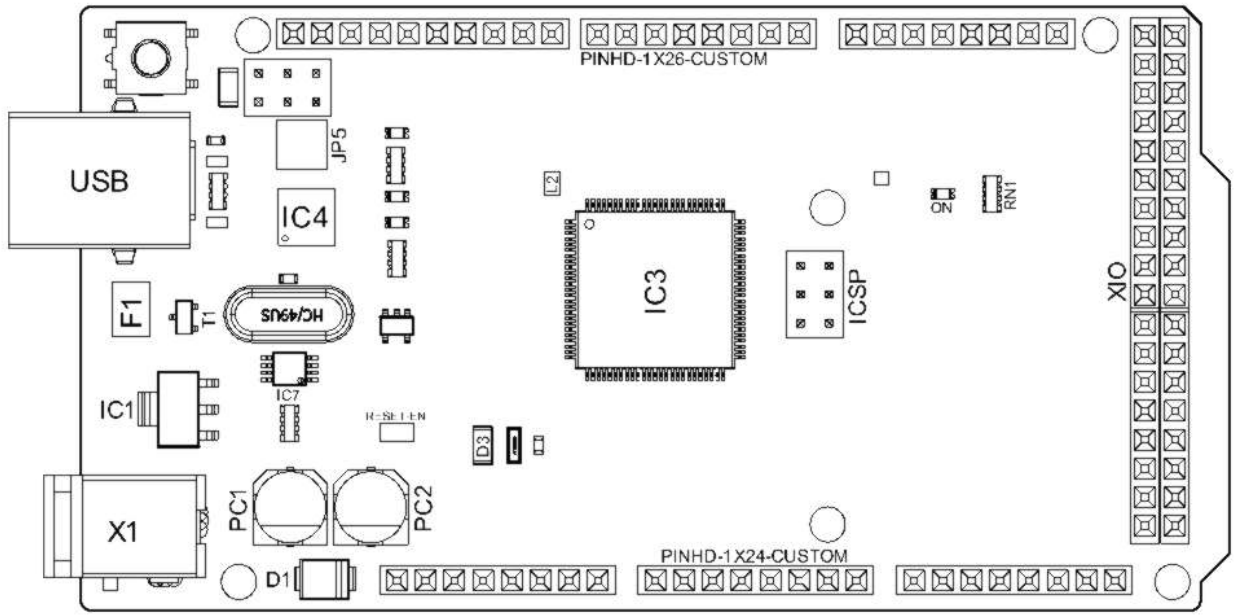
3.1 Block Diagram



Arduino MEGA Block Diagram

3.2 Board Topology

Front View



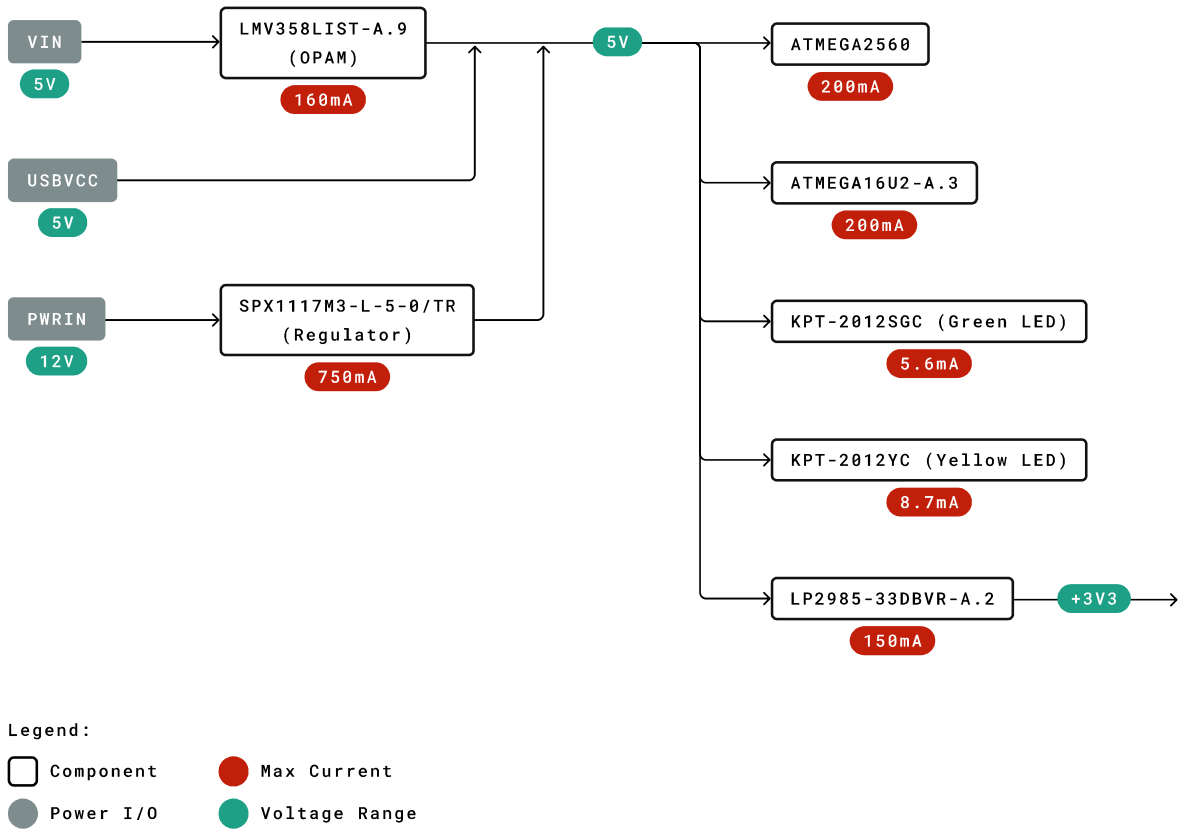
Arduino MEGA Top View

Ref.	Description	Ref.	Description
USB	USB B Connector	F1	Chip Capacitor
IC1	5V Linear Regulator	X1	Power Jack Connector
JP5	Plated Holes	IC4	ATmega16U2 chip
PC1	Electrolytic Alumninum Capacitor	PC2	Electrolytic Alumninum Capacitor
D1	General Purpose Rectifier	D3	General Purpose Diode
L2	Fixed Inductor	IC3	ATmega2560 chip
ICSP	Connector Header	ON	Green LED
RN1	Resistor Array	XIO	Connector

3.3 Processor

Primary processor of Arduino Mega 2560 Rev3 board is ATmega2560 chip which operates at a frequency of 16 MHz. It accommodates a large number of input and output lines which gives the provision of interfacing many external devices. At the same time the operations and processing is not slowed due to its significantly larger RAM than the other processors. The board also features a USB serial processor ATmega16U2 which acts an interface between the USB input signals and the main processor. This increases the flexibility of interfacing and connecting peripherals to the Arduino Mega 2560 Rev 3 board.

3.4 Power Tree



Power Tree

4 Board Operation

4.1 Getting Started - IDE

If you want to program your Arduino® MEGA 2560 while offline you need to install the Arduino® Desktop IDE **[1]** To connect the Arduino® MEGA 2560 to your computer, you'll need a Type-B USB cable. This also provides power to the board, as indicated by the LED.

4.2 Getting Started - Arduino Web Editor

All Arduino® boards, including this one, work out-of-the-box on the Arduino® Web Editor **[2]**, by just installing a simple plugin.

The Arduino® Web Editor is hosted online, therefore it will always be up-to-date with the latest features and support for all boards. Follow **[3]** to start coding on the browser and upload your sketches onto your board.

4.3 Sample Sketches

Sample sketches for the Arduino® MEGA 2560 can be found either in the "Examples" menu in the Arduino® IDE

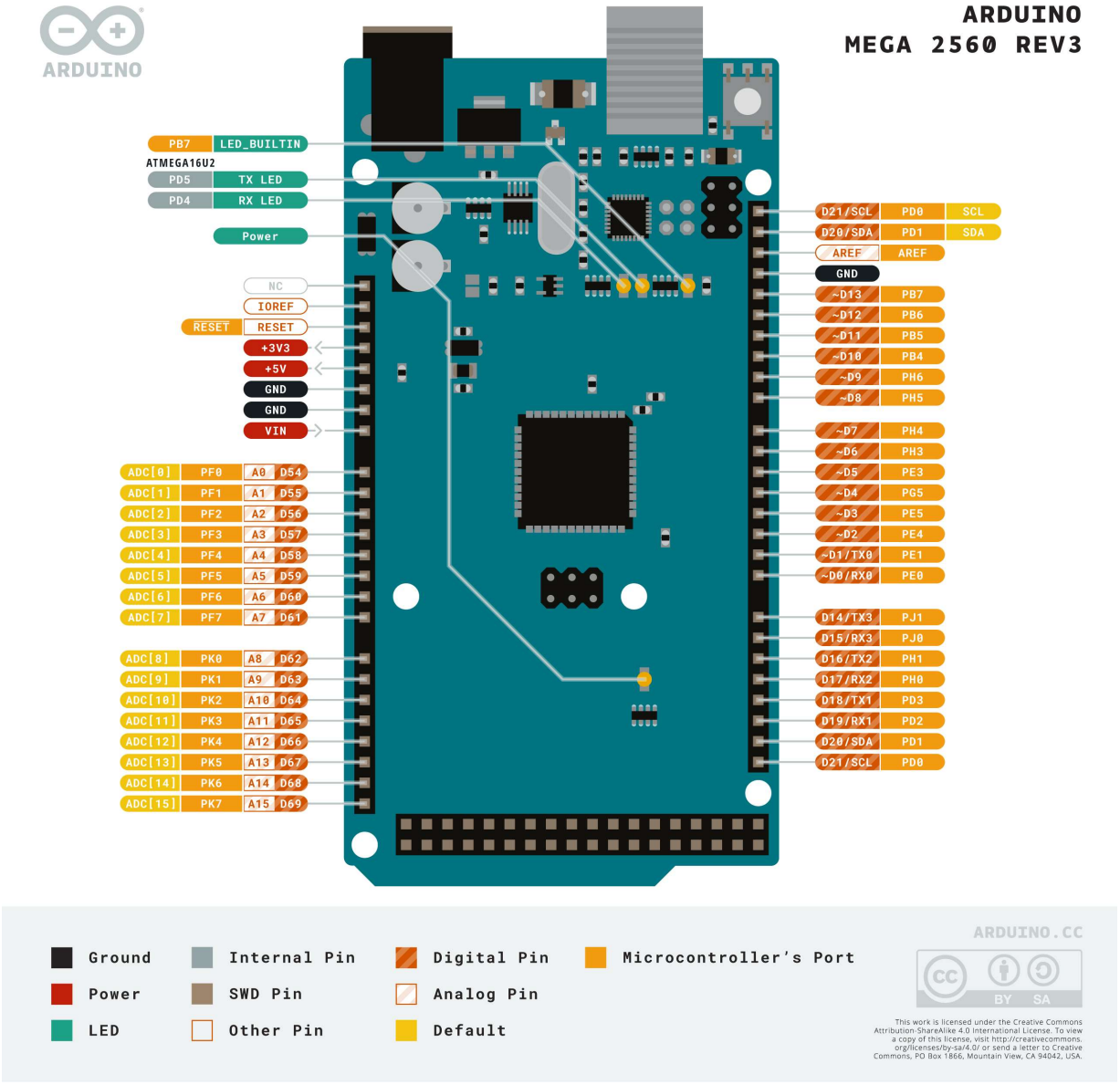
4.4 Online Resources

Now that you have gone through the basics of what you can do with the board you can explore the endless possibilities it provides by checking exciting projects on ProjectHub **[5]**, the Arduino® Library Reference **[6]** and the online store **[7]** where you will be able to complement your board with sensors, actuators and more.

4.5 Board Recovery

All Arduino boards have a built-in bootloader which allows flashing the board via USB. In case a sketch locks up the processor and the board is not reachable anymore via USB it is possible to enter bootloader mode by double-tapping the reset button right after power up.

5 Connector Pinouts



Arduino Mega Pinout

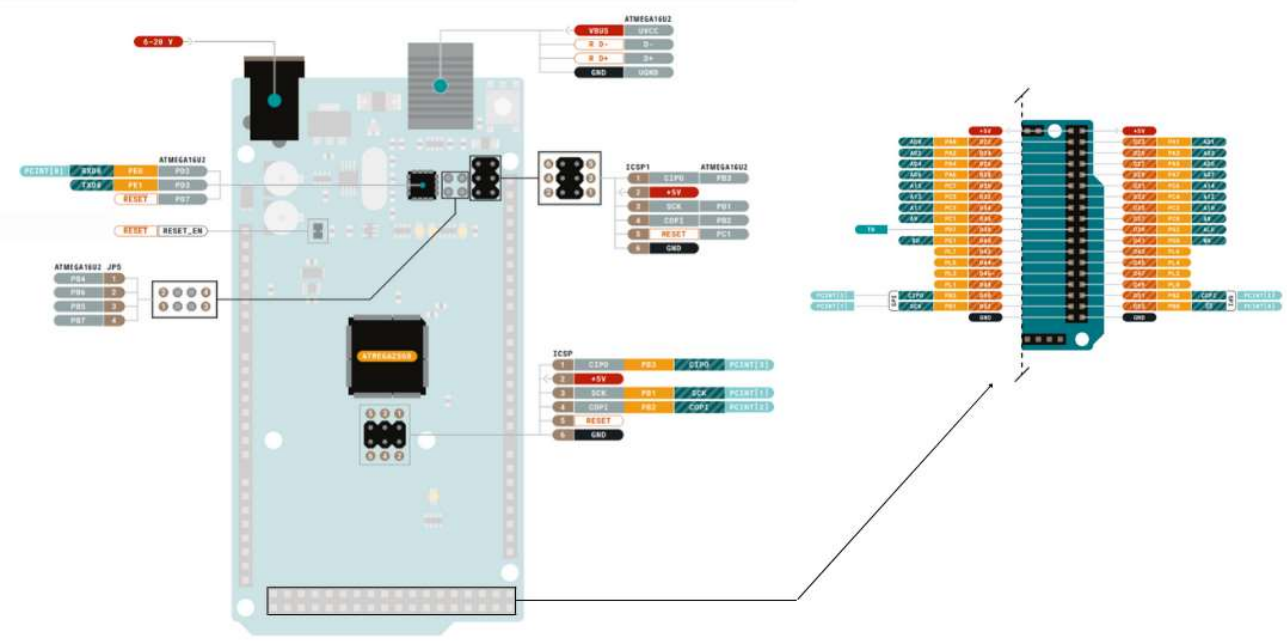
5.1 Analog

Pin	Function	Type	Description
1	NC	NC	Not Connected
2	IOREF	IOREF	Reference for digital logic V - connected to 5V
3	Reset	Reset	Reset
4	+3V3	Power	+3V3 Power Rail
5	+5V	Power	+5V Power Rail
6	GND	Power	Ground
7	GND	Power	Ground
8	VIN	Power	Voltage Input
9	A0	Analog	Analog input 0 /GPIO
10	A1	Analog	Analog input 1 /GPIO
11	A2	Analog	Analog input 2 /GPIO
12	A3	Analog	Analog input 3 /GPIO
13	A4	Analog	Analog input 4 /GPIO
14	A5	Analog	Analog input 5 /GPIO
15	A6	Analog	Analog input 6 /GPIO
16	A7	Analog	Analog input 7 /GPIO
17	A8	Analog	Analog input 8 /GPIO
18	A9	Analog	Analog input 9 /GPIO
19	A10	Analog	Analog input 10 /GPIO
20	A11	Analog	Analog input 11 /GPIO
21	A12	Analog	Analog input 12 /GPIO
22	A13	Analog	Analog input 13 /GPIO
23	A14	Analog	Analog input 14 /GPIO
24	A15	Analog	Analog input 15 /GPIO

5.2 Digital

Pin	Function	Type	Description
1	D21/SCL	Digital Input/I2C	Digital input 21/I2C Dataline
2	D20/SDA	Digital Input/I2C	Digital input 20/I2C Dataline
3	AREF	Digital	Analog Reference Voltage
4	GND	Power	Ground
5	D13	Digital/GPIO	Digital input 13/GPIO
6	D12	Digital/GPIO	Digital input 12/GPIO
7	D11	Digital/GPIO	Digital input 11/GPIO
8	D10	Digital/GPIO	Digital input 10/GPIO
9	D9	Digital/GPIO	Digital input 9/GPIO
10	D8	Digital/GPIO	Digital input 8/GPIO
11	D7	Digital/GPIO	Digital input 7/GPIO
12	D6	Digital/GPIO	Digital input 6/GPIO
13	D5	Digital/GPIO	Digital input 5/GPIO
14	D4	Digital/GPIO	Digital input 4/GPIO

Pin	Function	Type	Description
15	D3	Digital/GPIO	Digital input 3/GPIO
16	D2	Digital/GPIO	Digital input 2/GPIO
17	D1/TX0	Digital/GPIO	Digital input 1 /GPIO
18	D0/Tx1	Digital/GPIO	Digital input 0 /GPIO
19	D14	Digital/GPIO	Digital input 14 /GPIO
20	D15	Digital/GPIO	Digital input 15 /GPIO
21	D16	Digital/GPIO	Digital input 16 /GPIO
22	D17	Digital/GPIO	Digital input 17 /GPIO
23	D18	Digital/GPIO	Digital input 18 /GPIO
24	D19	Digital/GPIO	Digital input 19 /GPIO
25	D20	Digital/GPIO	Digital input 20 /GPIO
26	D21	Digital/GPIO	Digital input 21 /GPIO



Arduino Mega Pinout

5.3 ATMEGA16U2 JP5

Pin	Function	Type	Description
1	PB4	Internal	Serial Wire Debug
2	PB6	Internal	Serial Wire Debug
3	PB5	Internal	Serial Wire Debug
4	PB7	Internal	Serial Wire Debug

5.4 ATMEGA16U2 ICSP1

Pin	Function	Type	Description
1	CIPO	Internal	Controller In Peripheral Out
2	+5V	Internal	Power Supply of 5V
3	SCK	Internal	Serial Clock
4	COPI	Internal	Controller Out Peripheral In
5	RESET	Internal	Reset
6	GND	Internal	Ground

5.5 Digital Pins D22 - D53 LHS

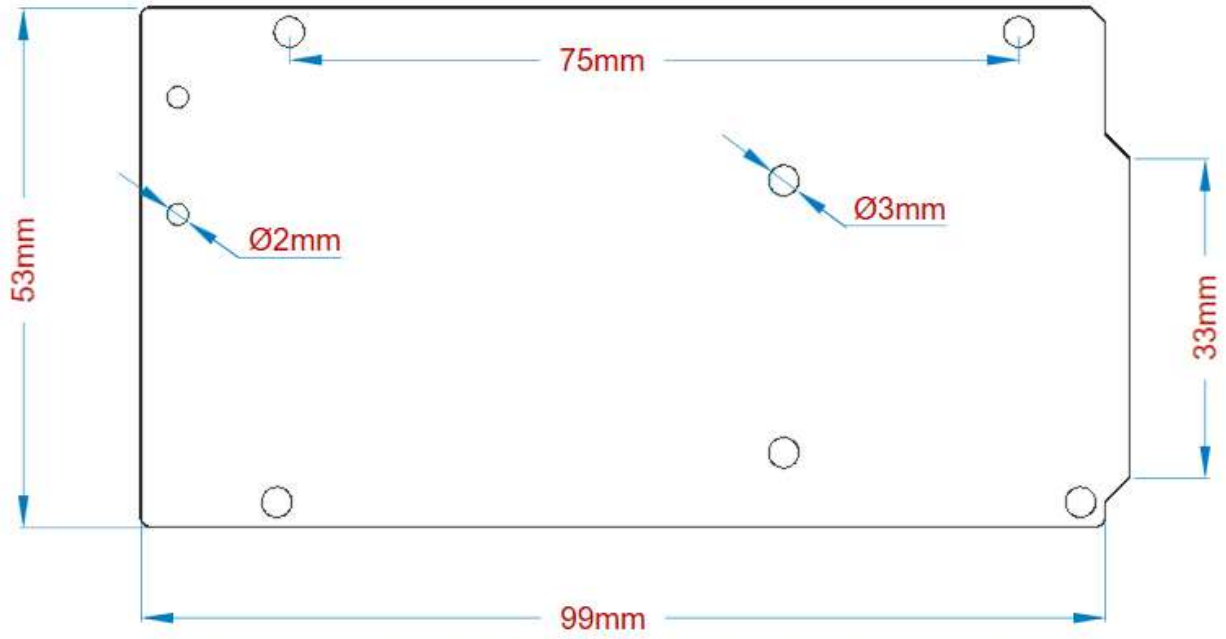
Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D22	Digital	Digital input 22/GPIO
3	D24	Digital	Digital input 24/GPIO
4	D26	Digital	Digital input 26/GPIO
5	D28	Digital	Digital input 28/GPIO
6	D30	Digital	Digital input 30/GPIO
7	D32	Digital	Digital input 32/GPIO
8	D34	Digital	Digital input 34/GPIO
9	D36	Digital	Digital input 36/GPIO
10	D38	Digital	Digital input 38/GPIO
11	D40	Digital	Digital input 40/GPIO
12	D42	Digital	Digital input 42/GPIO
13	D44	Digital	Digital input 44/GPIO
14	D46	Digital	Digital input 46/GPIO
15	D48	Digital	Digital input 48/GPIO
16	D50	Digital	Digital input 50/GPIO
17	D52	Digital	Digital input 52/GPIO
18	GND	Power	Ground

5.6 Digital Pins D22 - D53 RHS

Pin	Function	Type	Description
1	+5V	Power	Power Supply of 5V
2	D23	Digital	Digital input 23/GPIO
3	D25	Digital	Digital input 25/GPIO
4	D27	Digital	Digital input 27/GPIO
5	D29	Digital	Digital input 29/GPIO
6	D31	Digital	Digital input 31/GPIO
7	D33	Digital	Digital input 33/GPIO
8	D35	Digital	Digital input 35/GPIO
9	D37	Digital	Digital input 37/GPIO
10	D39	Digital	Digital input 39/GPIO
11	D41	Digital	Digital input 41/GPIO
12	D43	Digital	Digital input 43/GPIO
13	D45	Digital	Digital input 45/GPIO
14	D47	Digital	Digital input 47/GPIO
15	D49	Digital	Digital input 49/GPIO
16	D51	Digital	Digital input 51/GPIO
17	D53	Digital	Digital input 53/GPIO
18	GND	Power	Ground

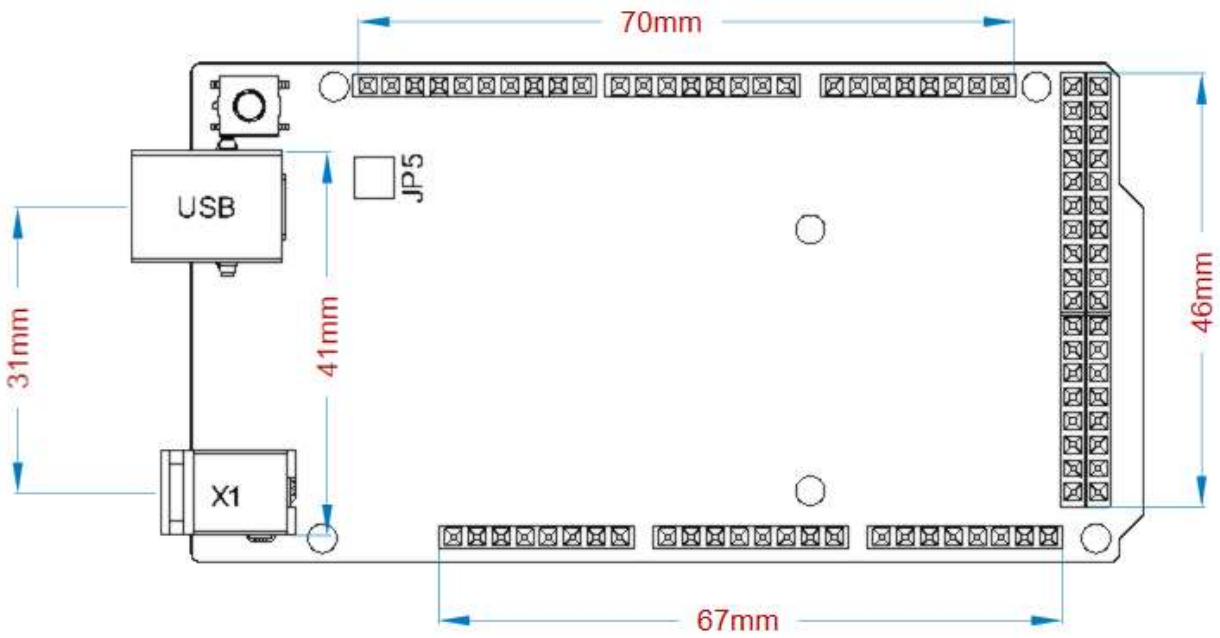
6 Mechanical Information

6.1 Board Outline



Arduino Mega Outline

6.2 Board Mount Holes



Arduino Mega Mount Holes

Certifications

7 Declaration of Conformity CE DoC (EU)

We declare under our sole responsibility that the products above are in conformity with the essential requirements of the following EU Directives and therefore qualify for free movement within markets comprising the European Union (EU) and European Economic Area (EEA).

8 Declaration of Conformity to EU RoHS & REACH 211 01/19/2021

Arduino boards are in compliance with RoHS 2 Directive 2011/65/EU of the European Parliament and RoHS 3 Directive 2015/863/EU of the Council of 4 June 2015 on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

Substance	Maximum Limit (ppm)
Lead (Pb)	1000
Cadmium (Cd)	100
Mercury (Hg)	1000
Hexavalent Chromium (Cr6+)	1000
Poly Brominated Biphenyls (PBB)	1000
Poly Brominated Diphenyl ethers (PBDE)	1000
Bis(2-Ethylhexyl} phthalate (DEHP)	1000
Benzyl butyl phthalate (BBP)	1000
Dibutyl phthalate (DBP)	1000
Diisobutyl phthalate (DIBP)	1000

Exemptions : No exemptions are claimed.

Arduino Boards are fully compliant with the related requirements of European Union Regulation (EC) 1907 /2006 concerning the Registration, Evaluation, Authorization and Restriction of Chemicals (REACH). We declare none of the SVHCs (https://echa.europa.eu/web/guest/candidate-list-table), the Candidate List of Substances of Very High Concern for authorization currently released by ECHA, is present in all products (and also package) in quantities totaling in a concentration equal or above 0.1%. To the best of our knowledge, we also declare that our products do not contain any of the substances listed on the "Authorization List" (Annex XIV of the REACH regulations) and Substances of Very High Concern (SVHC) in any significant amounts as specified by the Annex XVII of Candidate list published by ECHA (European Chemical Agency) 1907 /2006/EC.

9 Conflict Minerals Declaration

As a global supplier of electronic and electrical components, Arduino is aware of our obligations with regards to laws and regulations regarding Conflict Minerals, specifically the Dodd-Frank Wall Street Reform and Consumer Protection Act, Section 1502. Arduino does not directly source or process conflict minerals such as Tin, Tantalum, Tungsten, or Gold. Conflict minerals are contained in our products in the form of solder, or as a component in metal alloys. As part of our reasonable due diligence Arduino has contacted component suppliers within our supply chain to verify their continued compliance with the regulations. Based on the information received thus far we declare that our products contain Conflict Minerals sourced from conflict-free areas.

10 FCC Caution

Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

(1) This device may not cause harmful interference

(2) this device must accept any interference received, including interference that may cause undesired operation.

FCC RF Radiation Exposure Statement:

1. This Transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.
2. This equipment complies with RF radiation exposure limits set forth for an uncontrolled environment.
3. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.

English: User manuals for licence-exempt radio apparatus shall contain the following or equivalent notice in a conspicuous location in the user manual or alternatively on the device or both. This device complies with Industry Canada licence-exempt RSS standard(s). Operation is subject to the following two conditions:

(1) this device may not cause interference

(2) this device must accept any interference, including interference that may cause undesired operation of the device.

French: Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes :

(1) l' appareil nedoit pas produire de brouillage

(2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

IC SAR Warning:

English This equipment should be installed and operated with minimum distance 20 cm between the radiator and your body.

French: Lors de l' installation et de l' exploitation de ce dispositif, la distance entre le radiateur et le corps est d 'au moins 20 cm.

Important: The operating temperature of the EUT can't exceed 85°C and shouldn't be lower than -40°C.

Hereby, Arduino S.r.l. declares that this product is in compliance with essential requirements and other relevant provisions of Directive 201453/EU. This product is allowed to be used in all EU member states.

11 Company Information

Company name	Arduino S.r.l.
Company Address	Arduino SRL, Via Andrea Appiani 25, 20900 Monza MB, Italy

12 Reference Documentation

Ref	Link
Arduino IDE (Desktop)	https://www.arduino.cc/en/Main/Software
Arduino IDE (Cloud)	https://create.arduino.cc/editor
Cloud IDE Getting Started	https://create.arduino.cc/projecthub/Arduino_Genuino/getting-started-with-arduino-web-editor-4b3e4a
Arduino Pro Website	https://www.arduino.cc/pro
Project Hub	https://create.arduino.cc/projecthub?by=part&part_id=11332&sort=trending
Library Reference	https://www.arduino.cc/reference/en/libraries/
Online Store	https://store.arduino.cc/

13 Revision History

Date	Revision	Changes
29/09/2020	1	First Release