République Algérienne Démocratique et Populaire

Ministère de L'enseignement Supérieur et de la Recherche Scientifique

Université Echahid Hamma Lakhdar - El Oued

Faculté de la Technologie

Département de Génie Mécanique

MÉMOIRE

Présenté en vue de l'obtention du diplôme de

MASTER ACADEMIQUE

Spécialité : Électromécanique

Présenté par :

1. **BOUHOREIRA Abdelmalek**
2. **ZAID Messaoud**
3. **KHEDIR Nourredine**

Intitulé :

Réalisation d'un Système de contrôle et commande de remplissage des Réservoirs

Soutenue le :28/06 /2025

Devant le jury composé de :

Dr : Djokhrab Alaeddine	**Président**
Dr : ZIN Bachir	**Examinateur**
Dr : AROUDJ Mohammed	**Encadreur**

Année académique : 2024/2025

Résumé :

Nos travaux s'inscrivent dans le domaine de l'automatisation. Nous avons réalisé un projet de système automatisé de surveillance et de contrôle du remplissage des réservoirs à l'aide d'Arduino.

Nous avons commencé une étude exploratoire sur les causes des catastrophes et des accidents résultant de la défaillance des systèmes de remplissage et de surveillance des réservoirs de liquide dans le monde entier, ainsi que sur l'étude des conséquences économiques, humaines et environnementales de ces accidents.

En analysant ces raisons, nous avons choisi l'idée de mettre en œuvre ce système basé sur la surveillance du niveau de liquide à l'intérieur du réservoir à l'aide d'un capteur à ultrasons, nous prenons en considération le faible coût de réalisation du projet, ainsi que la précision et la fiabilité des performances du système, tout en tenant compte du fait que le système peut être utilisé dans les domaines industriels et domestiques et pour différents types de liquides.

Les mots-clés : Arduino, réservoir, système de commande, contrôle, remplissage.

Abstract :

Our work is focused on automation. We released a project for an automated tank filling monitoring and control system using Arduino.

We began an exploratory study on the causes of disasters and accidents resulting from the failure of liquid tank filling and monitoring systems worldwide, as well as the economic, human, and environmental consequences of these accidents.

After analyzing these reasons, we chose the idea of implementing this system based on monitoring the liquid level inside the tank using an ultrasonic sensor. We took into consideration the low cost of implementing the project, as well as the accuracy and reliability of the system's performance, while also considering that the system can be used in industrial and domestic applications and for various types of liquids.

Keywords: Arduino, tank, control system, control, filling

الملخص:

يندرج عملنا هدا ضمن مجال الاتمتة فقد قمنا بإنجاز مشروع نظام آلي لمراقبة وتحكم في ملء الخزانات باستخدام الأردوينو.

بدانا العمل بدراسة استكشافية حول أسباب الكوارث والحوادث الناجمة عن فشل أنظمة مراقبة وتحكم في ملء خزانات السوائل في العالم وكدلك دراسة أسباب ونتائج هده الحوادث اقتصاديا وبشريا وبيئيا.

من خلال تحليل هده الأسباب اخترنا فكرة انجاز هدا النظام القائم على مراقبة مستوى السائل داخل الخزان عن طريق مستشعر الأمواج فوق صوتية. بحيث نأخذ بعين الاعتبار التكلفة المنخفضة لإنجاز المشروع وكدلك الدقة والموثوقية في أداء النظام مع الأخذ بعين الاعتبار ايضا ان النظام يمكن استخدامه في المجالين الصناعي والمنزلي ولمختلف أنواع السوائل.

الكلمات المفتاحية: أردوينو، خزان، نظام تحكم، تحكم، تعبئة.

DÉDICACE

Je dédie ce travail :

À la mémoire de mon père Hadji, que Dieu lui accorde sa miséricorde et l'accueille en son vaste paradis. Ton souvenir m'accompagne à chaque étape de ma vie.

À ma mère B. Khadidja, pour son amour infini, ses prières silencieuses et son soutien indéfectible.

À mon épouse R. Khadidja, pour sa patience, son amour et son soutien constant tout au long de ce parcours.

À mes frères et sœurs, pour leurs encouragements.

À mes ami, collègues Imad Gharbi, Zaid messaoud, khedir Noureddine et Abd Elhak Sousa et ma promotion et tous les enseignants d'électromécanique.

BOUHOREIRA Abdelmalek

DÉDICACE

À celle dont les prières ont été le secret de ma réussite, et dont la tendresse est la source de ma force, À ma chère mère, toi qui m'as appris comment donner sans limites, et être sincère sans attendre en retour.

Et à celui qui a été mon premier soutien et ma source de fierté, À mon cher père, toi qui as planté dans mon âme le sens de la persévérance, du travail et de l'acharnement.

Et à mes chers frères et sœurs, Vous êtes mon secours après Dieu, et le dos qui ne fléchit pas, À vous tout mon amour et ma gratitude pour votre soutien constant.

Et à mes amis qui ont partagé mon chemin dans ses bons et mauvais moments, C

ompagnons de la science, du rire et du défi, Vous avez une place dans mon cœur,

Et à l'amie et frère chère Abd Elhak Sousa,

À toi toute mon estime et ma reconnaissance pour ton soutien et tes positions sincères.

À vous tous, j'offre le fruit de ce parcours, en signe de reconnaissance et de gratitude.

ZAIDE Messaoud

DÉDICACE

Je dédie ce travail à ma chère famille**.**

Je dédie également cette réussite à mes collègues de l'université, qui ont partagé avec moi le chemin du savoir et du défi, Bouhoreira Abdelmalek et Zaide Messaoude et Abd Elhak Sousa.

À vous tous, j'adresse mes plus sincères remerciements et ma profonde gratitude.

KHEDIR Noureddine

Remerciements

Nous tenons tout d'abord à remercier le Dr AROUDJ Mohammed, superviseur et encadrant de ce travail, pour toute l'aide qu'il nous a apportée.

Nous adressons également nos remerciements aux membres du jury qui nous ont honorés en acceptant d'examiner notre travail et en nous faisant part de leurs remarques précieuses et utiles commentaires.

Nous remercions toutes les personnes qui nous ont aidés à réaliser ce projet et qui nous ont soutenus pour surmonter les nombreuses difficultés rencontrées tout au long de sa réalisation.

Enfin, nous exprimons notre profonde reconnaissance à tous les professeurs et enseignants de la Faculté de Technologie de l'Université Hamma Lakhdar El Oued, qui nous ont enrichis dans tous les domaines au cours de nos études universitaires.

I Table des matières

Liste des Tableaux :

Liste des figures

LISTE DES ABBREVATIONS

LED: Light Emetting Diode.

USB: Universal Serial Bus.

GND: Ground.

CAN: convertisseur analogique numérique

MCU: Microcontrôleur

IDE: Integrated Development Environnement.

UART : Universal Asynchronous Receiver and Transmitter.

ICSP: In Circuit Serial Programming.

VCC : Alimentation Tension Continue.

C, C++ : Langage de programmation

WiFi : Wireless Fidelity

MLI : Modulation de Largeur d'Impulsion.

LCD: Liquid Crystal Display.

MHz: Megahertz

SRAM: Static Random Access Memory

EEPROM: Electronically Erasable Programmable Read-Only Memory

RAM : Random Access Memory

L'I2C: Inter-Integrated Circuit

VCC: Voltage Common Collector

PWM : Pulse Width Modulation

E/S: Entrées/Sorties

ABS : Anti-Blockier-System en allem

IoT : Internet of Things

Introduction générale :

Les opérations de remplissage sécurisé des réservoirs revêtent une importance capitale dans diverses industries. En raison des risques inhérents liés au stockage des liquides, notamment des matériaux dangereux, les systèmes de contrôle et de commande jouent un rôle essentiel pour garantir la sécurité de ces opérations. Ils contribuent à prévenir les accidents et les catastrophes potentielles.

Dans un contexte de progrès technologique rapide, les systèmes automatisés sont devenus une composante incontournable de notre quotidien, améliorant l'efficacité et réduisant l'effort humain dans de nombreux domaines. Parmi ces systèmes, les dispositifs de commande et de contrôle se distinguent comme des solutions efficaces pour une gestion intelligente et précise des ressources, particulièrement dans le domaine de la gestion et du stockage des liquides. Ces derniers, constituant une ressource vitale, exigent une préservation rigoureuse et une utilisation optimale. C'est dans cette Mémoire que s'inscrit notre projet de fin d'études intitulé "Réalisation d'un système de commande et de contrôle du remplissage des réservoirs" à l'aide d'Arduino, qui vise à proposer une solution innovante combinant technologie moderne et simplicité d'utilisation pour optimiser la gestion du processus de remplissage des réservoirs.

Ce projet a pour objectif de concevoir un système automatisé basé sur la plateforme Arduino, une plateforme open-source permettant un contrôle précis et flexible des appareils électroniques. Le système cherche à contrôler le niveau des liquides dans les réservoirs et à commander le processus de remplissage de manière automatique, contribuant ainsi à réduire le gaspillage, à économiser l'énergie et à assurer la durabilité des ressources. De plus, ce système se caractérise par son faible coût, le rendant adapté aussi bien aux applications domestiques qu'industrielles.

Dans son premier chapitre, cette thèse aborde les catastrophes majeures découlant de l'échec des systèmes de surveillance et de contrôle, en analysant leurs causes et leurs conséquences. Ensuite, les deuxième et troisième chapitre détaillent de manière méthodique les étapes de réalisation du projet, depuis l'étude du problème et la définition des objectifs, en passant par la conception du système et le choix des composants, jusqu'à la mise en œuvre pratique et les tests de performance. La thèse met également en lumière les aspects théoriques et pratiques du projet.

À travers ce travail, nous aspirons à apporter une contribution modeste au domaine de l'ingénierie électronique et des systèmes de contrôle, en espérant que ce projet constitue une avancée vers la promotion de l'innovation technologique et le soutien au développement durable. Nous souhaitons que cette thèse serve de référence utile aux chercheurs et aux

personnes intéressées par le développement de systèmes de surveillance et de contrôle utilisant des technologies modernes à des coûts abordables.

I CHAPITRE I : HISTORIQUE DES ACCIDENTS DES RESERVOIRS DE LIQUIDES.

I.1 Introduction :

Dans un monde en constante évolution technologique et industrielle, les défis liés à la sécurité et à la prévention des risques deviennent de plus en plus complexes. L'expansion des infrastructures, la densification des activités humaines, et l'augmentation de la manipulation de matières dangereuses rendent indispensable la mise en place de systèmes efficaces et intelligents de gestion des risques. Les accidents majeurs ne sont plus des événements rares et isolés, mais représentent des menaces sérieuses et récurrentes pouvant avoir des conséquences humaines, économiques et environnementales graves.

Dans ce contexte, il devient impératif de repenser les approches traditionnelles de la gestion des urgences et de se diriger vers la conception d'un système intégré, préventif et intelligent, capable de détecter les dangers à un stade précoce, d'analyser les données en temps réel, et d'activer les réponses appropriées rapidement. Un tel système repose non seulement sur la technologie de pointe, mais également sur une coordination institutionnelle efficace et une culture de la sécurité partagée à tous les niveaux de la société.

I.2 Historiques de quelque Incidents, catastrophes majeures :

I.2.1 Boston, États-Unis (1919) :

L'accident dont tu parles est connu sous le nom de la Grande inondation de mélasse de Boston (Great Molasses Flood), un événement aussi tragique qu'étrange qui s'est produit le 15 janvier 1919 à Boston, dans le quartier de North End.

Voici les faits principaux:

- Lieu : Boston, Massachusetts, États-Unis.
- Date : 15 janvier 1919.
- Produit impliqué : Mélasse (un sirop épais, sous-produit de la fabrication du sucre).
- Réservoir : Un énorme réservoir appartenant à la société Purity Distilling Company. Il mesurait environ 15 mètres de haut et contenait plus de 8,7 millions de litres de mélasse.
- Que s'est-il passé? le réservoir a explosé soudainement, libérant une vague de mélasse de plus de 4 mètres de haut, se déplaçant à une vitesse estimée à 56 km/h.
- Cette masse visqueuse a détruit des bâtiments, renversé des wagons de tramway et tué des personnes et des animaux par asphyxie, noyade ou blessures. [1]

I.2.1.1 Bilan humain et matériel*:*

a) 21 morts
b) Environ 150 blessés
c) D'importants dégâts matériels : bâtiments écrasés, rues impraticables, etc.

I.2.1.2 Causes de l'explosion*:*

a) Mauvaise conception du réservoir : il avait des défauts structurels (les rivets étaient mal posés, l'acier était de mauvaise qualité).
b) Pression intérieure excessive : à cause de la fermentation de la mélasse, qui produisait des gaz.
c) Changement brusque de température : il avait fait très froid puis soudainement plus chaud, ce qui a pu augmenter la pression dans la cuve.

I.2.1.3 Conséquences*:*

a) Le nettoyage a pris des semaines.
b) Une odeur de mélasse aurait persisté dans certaines rues pendant des années.
c) Cette catastrophe a entraîné une réforme des normes de construction et des inspections plus strictes pour les structures industrielles aux États-Unis.

I.2.2 La catastrophe de Bhopal (Inde, 1984) :

Aux premières heures du 3 décembre 1984, la ville de Bhopal, située au centre de l'Inde, a été le théâtre d'un drame humanitaire sans précédent, considéré aujourd'hui comme la pire catastrophe industrielle de l'histoire. Cette tragédie s'est produite lorsqu'environ 42 tonnes de gaz méthyl-isocyanate (MIC), un produit extrêmement toxique utilisé dans la fabrication de pesticides, se sont échappées d'une usine appartenant à Union Carbide India Limited (UCIL), filiale de la société américaine Union Carbide Corporation. Le gaz s'est répandu rapidement, formant un nuage mortel qui a enveloppé les quartiers densément peuplés autour de l'usine, habités principalement par des familles pauvres vivant dans des habitations précaires. [2]

I.2.2.1 Les causes de la catastrophe étaient multiples*:*

a) Défaillance totale des systèmes de sécurité, notamment le système de refroidissement et le laveur de gaz, tous deux désactivés ou non fonctionnels.
b) Infiltration d'eau dans un réservoir de MIC, entraînant une réaction chimique violente et une montée de pression qui a fait exploser les soupapes de sécurité.
c) Manque d'entretien, couplé à une réduction volontaire des coûts au détriment de la sécurité.
d) Personnel mal formé, sans réelle préparation aux situations d'urgence.
e) Et surtout, la localisation de l'usine au cœur d'une zone résidentielle pauvre, une erreur majeure de planification industrielle.

I.2.2.2 Les conséquences furent catastrophiques*:*

a) Plus de 3 000 personnes sont mortes dans les premières heures, asphyxiées par le gaz.
b) Environ 15 000 autres décès sont survenus dans les années qui ont suivi à cause de maladies Plus de 500 000 individus ont été exposés, souffrant de divers maux : troubles

respiratoires, cancers, maladies chroniques, cécité, et malformations congénitales chez les nouveau-nés.

c) Malgré l'ampleur de la tragédie, la justice reste largement incomplète. En 1989, Union Carbide a versé 470 millions de dollars en compensation – un montant largement insuffisant. Le PDG de l'époque, Warren Anderson, n'a jamais été jugé et a fui l'Inde. En 2010, seuls sept responsables indiens ont été condamnés à deux ans de prison avec de simples amendes, provoquant l'indignation nationale.

I.2.2.3 Les effets à long terme persistent encore aujourd'hui*:*

a) Le site de l'usine reste hautement contaminé, avec des déchets chimiques qui polluent les nappes phréatiques et les sols.

b) De nombreux enfants nés après la catastrophe souffrent de malformations graves et de troubles neurologiques.

c) Les survivants continuent de se battre pour la dépollution du site, des soins médicaux adaptés, et une reconnaissance réelle de leurs droits.

d) La catastrophe de Bhopal n'était pas seulement un accident, mais le résultat tragique d'une négligence systémique, dont les cicatrices sont encore visibles des décennies plus tard.

Figure (I.1) : Vue du quartier après la vague de mélasse

I.2.3 La catastrophe de Guadalajara, Mexique – 22 avril 1992 :

Le 22 avril 1992, la ville de Guadalajara, deuxième plus grande métropole du Mexique, a été frappée par l'une des catastrophes urbaines les plus tragiques de l'histoire du pays, lorsqu'une série d'explosions massives a ravagé le quartier d'Analco, transformant des rues entières en cratères et causant des centaines de morts et de blessés. Les jours précédant le drame, les habitants avaient signalé la présence d'odeurs fortes d'essence émanant des égouts, mais

malgré ces alertes répétées, aucune mesure préventive sérieuse n'a été prise par les autorités locales [3]

I.2.3.1 Les causes principales de la catastrophe sont les suivantes :

a) Une fuite prolongée de carburant provenant d'un pipeline souterrain appartenant à l'entreprise pétrolière nationale PEMEX.
b) La fuite était due à un problème de corrosion et à des défauts de conception des conduites, posées trop près du réseau d'égouts.
c) Le gaz inflammable s'est accumulé dans les égouts, formant une concentration de vapeurs hautement explosives.
d) Le manque de réaction adéquate des autorités locales malgré les alertes répétées des résidents, ce qui a permis aux fuites de persister sans intervention.
e) Le déclenchement de l'explosion a été causé par une étincelle, probablement due à un court-circuit ou à une friction électrique.

Le matin du 22 avril, une étincelle a provoqué l'explosion des vapeurs d'essence accumulées, déclenchant huit détonations successives qui ont secoué environ huit kilomètres de rues. Le bilan fut dévastateur : au moins 200 morts, bien que certaines sources évoquent plus de 250 victimes, environ 1 500 blessés, ainsi que des milliers de personnes déplacées. Plus de 1 000 maisons et 100 véhicules furent totalement détruits, et de nombreux corps furent retrouvés à des dizaines de mètres du lieu de l'explosion, projetés par la puissance du souffle.

I.2.3.2 Les conséquences de cette tragédie ont été multiples :

a) Une indignation massive de la population mexicaine face à l'ampleur de l'impréparation et de la négligence.
b) Peu de responsables ont été sanctionnés, malgré les enquêtes ouvertes.
c) Les compensations financières versées aux victimes ont été jugées largement insuffisantes.
d) L'événement a conduit à une révision des normes de sécurité urbaines et industrielles au Mexique pour éviter la répétition de telles tragédies.
e) Trente ans plus tard, la catastrophe de Guadalajara reste dans la mémoire collective mexicaine comme un symbole tragique de l'inefficacité des mesures de sécurité et de la mauvaise gestion des risques.

I.2.4 Catastrophe de Jaipur, Inde (13 mai 2009) :

Le 13 mai 2009, la ville de Jaipur, en Inde, a été secouée par une explosion dévastatrice dans une usine chimique appartenant à la société Indian Explosives Ltd., qui a provoqué une véritable tragédie industrielle. L'explosion a tué plusieurs travailleurs et causé des blessures à

de nombreuses personnes dans la zone environnante. L'incident s'est produit à la suite d'une gestion inadéquate des matières chimiques hautement inflammables dans le site de production. [4]

I.2.4.1 Les causes principales*:*

a) Négligence des mesures de sécurité : L'usine manquait de procédures de sécurité de base, avec des pratiques non conformes concernant la manipulation des matières chimiques dangereuses.
b) Explosion de produits chimiques : Un produit chimique a explosé dans l'un des entrepôts de stockage, probablement en raison de températures excessives ou d'une réaction chimique incontrôlée.
c) Installations inappropriées : L'usine ne possédait pas de systèmes modernes pour garantir la sécurité industrielle, comme un système d'extinction automatique des incendies, qui aurait pu limiter les dégâts.
d) Manque de formation des employés : Les travailleurs n'avaient pas reçu de formation adéquate pour travailler en toute sécurité avec des substances chimiques dangereuses, ce qui a contribué à l'accident.

I.2.4.2 Les conséquences de l'explosion:

a) Décès de 12 personnes au moins, principalement des travailleurs et des habitants des environs, et plus de 30 blessés, certains gravement.
b) Destruction partielle de l'usine, avec des dégâts importants aux infrastructures et aux équipements, augmentant l'impact de l'explosion.
c) Conséquences environnementales : La fuite de produits chimiques dans l'air et dans les eaux a provoqué une contamination de l'environnement, entraînant des effets à long terme sur la santé publique.
d) Enquête et poursuites judiciaires : Les autorités indiennes ont ouvert une enquête pour déterminer les responsabilités, ce qui a conduit à la poursuite des responsables pour négligence grave.
e) Renforcement des régulations de sécurité : Après l'incident, des lois de sécurité plus strictes ont été mises en place dans le secteur industriel en Inde, avec une attention particulière à la gestion des matières dangereuses.

I.2.4.3 Leçons apprises*:*

Cet incident a conduit à la mise en œuvre de régulations renforcées pour garantir la sécurité dans les usines chimiques en Inde.

Il a également mis en évidence l'importance de la formation des travailleurs et du respect des normes de sécurité pour éviter de tels accidents dans l'avenir.

I.2.5 Catastrophe de Beyrouth, Liban (4 août 2020) :

Le 4 août 2020, une explosion dévastatrice a secoué la capitale du Liban, Beyrouth, causant des destructions massives dans la ville. L'explosion a eu lieu dans le port de Beyrouth, où des tonnes de nitrate d'ammonium étaient stockées depuis 2013 dans des conditions dangereuses. L'incident est considéré comme l'un des plus grandes explosions non nucléaires de l'histoire moderne. [5]

I.2.5.1 Les causes principales :

a) Stockage de matières dangereuses : L'explosion a été causée par le stockage de nitrate d'ammonium, une substance hautement explosive, dans des conditions inappropriées et sans mesures de sécurité adéquates.

b) Négligence et corruption : Les autorités libanaises étaient au courant de la présence de cette substance dangereuse, mais aucune mesure n'a été prise pour la sécuriser. L'inefficacité administrative et la corruption au sein des institutions ont contribué à la tragédie.

c) Ignition par une étincelle : Le feu aurait été déclenché par une étincelle électrique ou un incendie dans le port, provoquant une réaction en chaîne avec les stocks de nitrate d'ammonium.

I.2.5.2 Les conséquences :

a) Plus de 200 morts : Selon les rapports officiels, l'explosion a tué plus de 200 personnes et blessé plus de 6 000 autres. Des centaines d'autres ont été tuées par les débris et les secousses.

b) Destruction massive à Beyrouth : 50 % des bâtiments de la ville ont été endommagés, y compris des maisons, des hôpitaux et des bureaux, provoquant le déplacement de plus de 300 000 personnes.

c) Destruction complète du port : Le port de Beyrouth, un point clé pour l'économie libanaise, a été presque complètement détruit, perturbant ainsi les importations et les exportations, ainsi que l'acheminement des aides internationales.

d) Crise économique et humanitaire exacerbée : L'explosion est survenue dans un contexte de crise économique sévère au Liban, aggravant la situation déjà précaire du pays.

e) Enquête et poursuites légales : Suite à la catastrophe, des enquêtes ont été ouvertes pour identifier les responsables. Cependant, ces enquêtes ont été marquées par des obstacles politiques, et les responsables n'ont pas été suffisamment tenus pour responsables.

I.2.5.3 Leçons tirées :

a) Renforcement des protocoles de sécurité : Cette tragédie a mis en lumière la nécessité de renforcer les protocoles de sécurité dans les ports et les installations de stockage de matières dangereuses.

b) Réformes politiques urgentes : La catastrophe a révélé l'ampleur du manque de gouvernance et de corruption au Liban, et a entraîné des appels à une réforme politique majeure.

c) Aide internationale : Bien que la communauté internationale ait fourni une aide d'urgence pour soutenir le Liban, la catastrophe a démontré la nécessité d'un soutien à long terme pour reconstruire le pays et aider les victimes.

Figure (I.2) : Vue après la catastrophe

I.2.6 Éclaté dans un terminal pétrolier à Matanzas :

Le 5 août 2022, un incendie dévastateur a éclaté dans un terminal pétrolier à Matanzas, une ville située sur la côte nord de Cuba. L'incendie a été provoqué par une fuite de pétrole dans un des réservoirs de stockage, et il a rapidement dégénéré en une explosion massive, entraînant la destruction partielle de l'installation. Ce sinistre a eu un impact considérable sur la sécurité, l'environnement et l'économie locale. [6]

I.2.6.1 Les causes principales :

a) Fuite de pétrole : L'incendie a commencé dans un réservoir de stockage de pétrole brut, suite à une fuite dans l'une des conduites. L'ignition de la fuite par un éclat de feu ou une étincelle a rapidement provoqué l'explosion.

b) Conditions climatiques défavorables : Des conditions météorologiques difficiles, notamment des ventes fortes, ont facilité la propagation de l'incendie et ont rendu l'extinction des flammes extrêmement difficile.

c) Vieux équipements et mauvaises pratiques de sécurité : Le terminal pétrolier présentait des manquements aux normes de sécurité. Des équipements vétustes et une gestion insuffisante des risques ont contribué à l'intensité de la catastrophe.
d) Manque de préparation aux situations d'urgence : Les équipes de secours n'étaient pas suffisamment préparées pour gérer un incendie de cette ampleur, ce qui a retardé l'intervention et la gestion de la crise.

I.2.6.2 Les conséquences :

a) Destruction du terminal pétrolier : L'explosion a ravagé une grande partie du terminal pétrolier, détruisant plusieurs réservoirs de stockage et endommageant les infrastructures locales.
b) Victimes humaines : Au moins 16 personnes ont été tuées, et 74 autres ont été blessées, dont certaines gravement. Plusieurs travailleurs ont été piégés par les flammes.
c) Effets environnementaux : L'incendie a causé des dégâts environnementaux considérables. Des quantités importantes de pétrole brut se sont échappées, menaçant les écosystèmes marins et les zones environnantes.
d) Perturbation de l'approvisionnement énergétique : Le terminal de Matanzas joue un rôle crucial dans l'approvisionnement en énergie de Cuba. La catastrophe a donc affecté l'approvisionnement en carburant, exacerbant ainsi les difficultés énergétiques du pays.
e) Aide internationale : L'incendie a mobilisé des équipes de secours internationales, notamment du Mexique et de la Russie, pour aider à éteindre les flammes et limiter les dégâts.

I.2.6.3 Les leçons tirées :

a) Renforcement des normes de sécurité : La catastrophe a mis en évidence la nécessité de revoir les normes de sécurité dans les infrastructures critiques de Cuba, notamment dans le secteur pétrolier.
b) Amélioration de la gestion des urgences : Des protocoles de gestion des crises doivent être renforcés, avec une meilleure préparation pour faire face à des incidents de grande envergure.
c) Régénération des infrastructures : La catastrophe a souligné l'importance de moderniser les équipements et les installations vieillissantes afin de minimiser les risques d'incidents similaires.

La tragédie de Matanzas a laissé des cicatrices profondes sur la ville et ses habitants, tout en soulignant les vulnérabilités auxquelles sont confrontées de nombreuses infrastructures vieillissantes à travers le monde.

Figure (I.3) : Vue après la catastrophe

I.3 Conclusion :

À travers l'analyse des catastrophes industrielles, il apparaît clairement qu'un point commun relie tous ces drames : l'absence de systèmes d'alerte précoce efficaces, le manque de surveillance continue, une mauvaise gestion du stockage des matières dangereuses, et une coordination insuffisante entre les entités responsables. Bien que les contextes géographiques et politiques soient différents, les conséquences ont été, à chaque fois, tragiques : des dizaines de morts, des destructions massives, des crises environnementales graves, et d'importantes pertes économiques.

Ces constats nous poussent à poser une question essentielle : ces catastrophes auraient-elles pu être évitées grâce à un système unifié, intelligent et proactif ? La réponse nous conduit naturellement à la nécessité de concevoir un système préventif intégré, basé sur les enseignements des événements passés, et capable d'interagir de manière dynamique avec les risques dès leurs premiers signes. Un système qui combine analyse numérique, capteurs intelligents, intelligence artificielle et gestion coordonnée des urgences, le tout dans une structure centralisée et réactive.

C'est à partir de cette réflexion que nous présenterons, dans la section suivante, une proposition détaillée d'un système intelligent et global de prévention des catastrophes, conçu à la lumière des leçons tirées des accidents précédents, dans le but d'offrir une solution concrète et adaptable aux environnements à haut risque.

II Chapitre II : DESCRIPTION DU PROCESSUS ET IDENTIFICATION DU MATÉRIEL UTILISÉ

II.1 Introduction :

Ce chapitre propose une introduction générale sur le système Arduino et les capteurs, dans ce chapitre, nous avons exposé une vue d'ensemble du système Arduino et de ses diverses variantes, suivie d'une description de leurs caractéristiques essentielles. Par la suite, nous nous concentrerons sur l'Arduino Uno et ses spécificités, puis sur son aspect logiciel et son langage de programmation.

Nous avons enfin présenté les divers capteurs utilisés pour mener à bien ce projet.

II.2 Système Arduino :

II.2.1 Définition :

Arduino permet de combiner les capacités de la programmation avec celles de l'électronique. Plus spécifiquement, pour la programmation de systèmes électroniques. L'électronique programmée offre l'énorme bénéfice de simplifier considérablement les circuits électroniques, ce qui entraîne une réduction des coûts de fabrication ainsi qu'une diminution du travail nécessaire à la conception d'un circuit électronique.

Le système Arduino se compose de deux éléments principaux : le matériel et le programme logiciel.

II.2.2 Le matériel :

C'est une carte électronique conçue autour d'un microcontrôleur Atmega d'Atmel, dont le coût reste modique à la considération de la diversité des applications possibles.

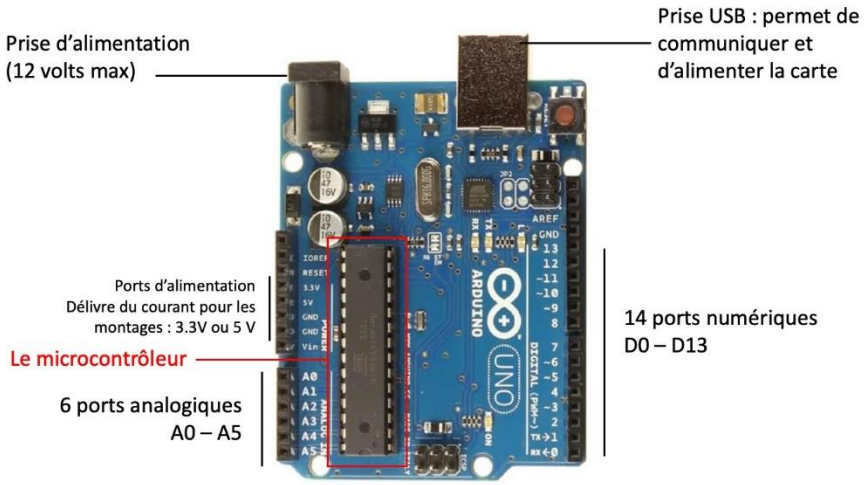


Figure (II.1) : Carte Arduino UNO

II.2.3 Le logiciel :

Logiciel permet de programmer la carte Arduino. Il offre une multitude de fonctionnalités. [7]

II.2.4 Arduino IDE :

L'environnement de développement intégré Arduino est un programme qui assure la liaison entre votre carte Arduino et le logiciel. L'IDE Arduino intègre un assembleur qui transformera votre programme en code machine interprétable par la carte Arduino.

Arduino IDE est une variante de Processing. C'est un logiciel open source qui permet de programmer des cartes autres que l'Arduino.

Vous pouvez télécharger gratuitement l'environnement de programmation open source pour Arduino (compatible avec Mac OS, Windows et Linux) directement depuis le site officiel d'Arduino. [8] https://www.arduino.cc/en/software/

II.2.4.1 Découverte de l'IDE Arduino :

L'IDE Arduino est particulièrement conviviale et propose une interface claire et intuitive pour la programmation des cartes Arduino. Il inclut un éditeur de code offrant une mise en couleur syntaxique et une barre d'outils accès rapide. Ces deux éléments sont les plus essentiels et les plus fréquemment utilisés de l'interface. Il est également possible d'accéder aux fonctionnalités avancées de l'IDE via une barre de menu plus traditionnelle. Pour finir, une console offre la possibilité de montrer les résultats de la compilation du code source, les opérations de mapping, et plus encore.

L'IDE Arduino (logiciel) permet de rédiger un programme (sous la forme d'un ou plusieurs modèles) en C, le compiler pour le transformer en code « machine » puis l'installer dans la mémoire de l'Arduino.

Le microcontrôleur conserve le programme Arduino dans une partie de sa mémoire (zone réservée au programme) où il réside de manière permanente. [9]

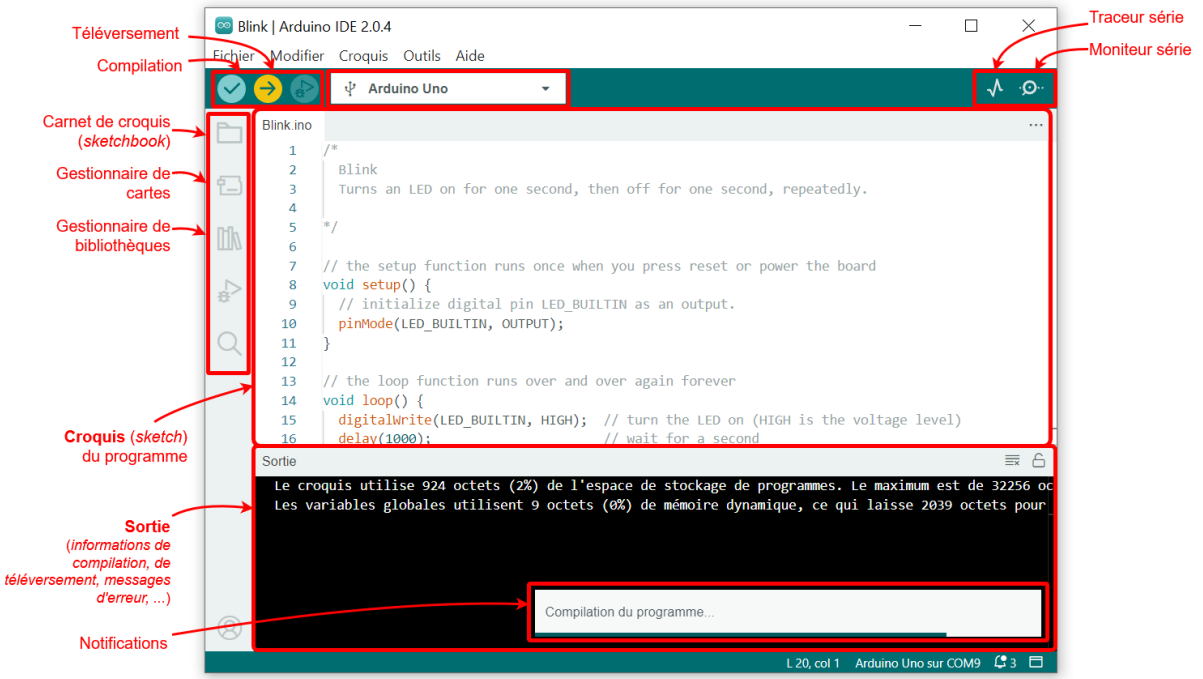


Figure (II.2) : Le logiciel de programmation IDE

II.2.5 Langage de programmation :

Arduino puise sa référence dans plusieurs langages. On a décelé des ressemblances avec C, C++, Ce langage impose une structure particulière, caractéristique de l'informatique embarquée.

Un langage de programmation offre la possibilité de rédiger un ensemble d'instructions (code source) qui est directement traduit en langage machine grâce à l'utilisation d'un compilateur (on le nomme alors assembleur). Les programmes Arduino s'exécutent de manière séquentielle, c'est-à-dire que les commandes sont exécutées successivement. [10]

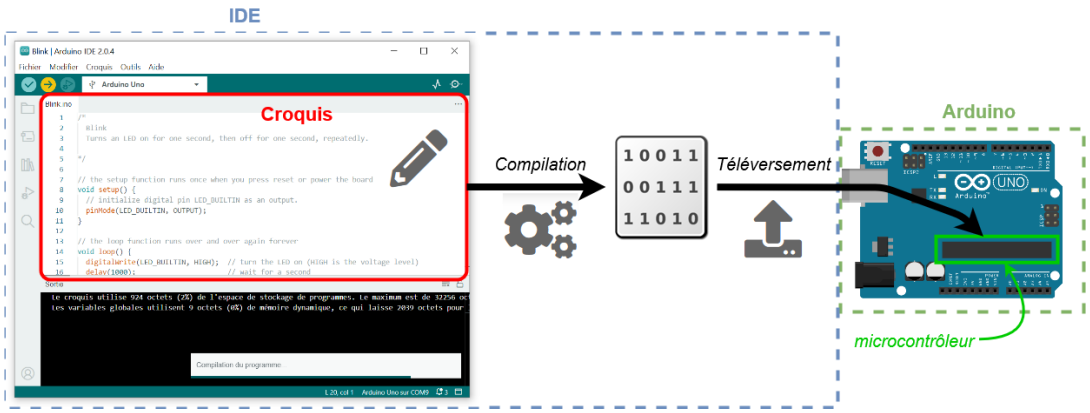


Figure (II.3) : Uploader programme sur carte Arduino

II.2.6 La structure d'un programme :

a) La fonction « Setup » inclura toutes les opérations requises pour configurer la carte (directions des entrées et sorties, débits de communication série, et ainsi de suite).

b) La fonction « Loop » est exécutée en continu suite à l'exécution de la fonction setup. Elle restera en boucle tant que la carte ne sera pas éteinte, réinitialisée (via le bouton de réinitialisation). Il est indispensable d'avoir cette boucle sur les microcontrôleurs, car ils ne disposent pas de système d'exploitation. En effet, sans cette boucle, à la fin du programme généré, il serait impossible de reprendre le contrôle de la carte Arduino qui commencerait alors à exécuter un code aléatoire. [11]

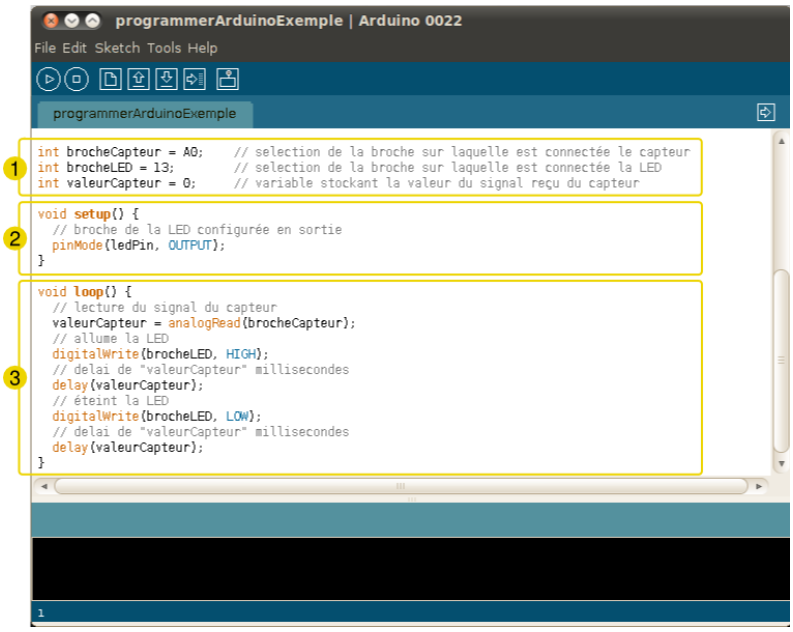


Figure (II.4) : structure d'un programme

II.2.7 Historique :

Hernando Barragan a conçu un langage de programmation simplifié au début des années 2000, qu'il a nommé Wiring. Massimo Banzi et David Cuartielles ont créé en 2005 un dispositif programmable simple d'utilisation destiné à la conception et la réalisation de divers projets artistiques interactifs. Ce projet, conçu à l'institut de design interactif d'Ivrea en Italie, a été appelé Arduino. Le logiciel de l'atelier Arduino a été rédigé par David Mellis, directement basé sur Wiring. Peu après, Gianluca Martino et Tom Igoe ont rejoint leurs confrères, portant l'équipe initiale à cinq membres. Le produit était censé être intuitif, aisément raccordable à une variété de capteurs et d'actionneurs (relais, moteurs, capteurs), et surtout simple à coder, tout en demeurant abordable financièrement, car les étudiants et les artistes ne sont pas souvent fortunés.

Nous avons opté pour la gamme de microcontrôleurs 8 bits AVR d'Atmel pour notre sélection. Un microcontrôleur (MCU) est une puce intégrant toutes les fonctionnalités nécessaires avec des liaisons simplifiées. Les fondateurs ont donc développé le microprogramme de démarrage (bootloader), qui autorise le téléchargement du programme et de l'outil de création fonctionnant sur une machine hôte sous Windows, macOS ou Linux pour

la rédaction et la compilation des programmes. Cet instrument est l'environnement de développement intégré (IDE) d'Arduino. Les créateurs ont choisi de nommer les fichiers contenant le code source « croquis », afin d'illustrer clairement que cette écriture devait demeurer simple comme un dessin et être améliorée graduellement. C'est tout cela qui constitue l'environnement Arduino.

Durant ses dix premières années d'existence, Arduino a connu une expansion dans diverses directions. Il existe de nouvelles cartes qui sont plus petites que la première, tandis que d'autres sont de plus grande taille. Chaque carte est associée à un domaine d'utilisation. Chaque carte utilise la même bibliothèque de fonctions open source, le même ensemble d'outils avr-gcc et le même logiciel de démarrage/chargeur préinstallé intégré dans le microcontrôleur de chaque carte Arduino.

Les microcontrôleurs proviennent de l'entreprise Atmel Corporation basée à San José, en Californie. La majorité des cartes Arduino reposent sur une architecture 8 bits. (Le modèle Due intègre un processeur distinct, à savoir un Cortex ARM 32 bits).

D'après la définition de l'équipe fondatrice Arduino, une carte Arduino se présente comme une carte de circuit imprimé sur laquelle est fixée un microcontrôleur appartenant à la série AVR Atmel. L'environnement logiciel constitue ce qui différencie une carte Arduino des autres nombreuses cartes dites de développement. Tous les utilisateurs Arduino bénéficient du même environnement, qui est le fondement et la raison principale de la réussite d'Arduino. [12]

II.2.8 Utilisation d'un Arduino :

Voici quelques utilisations possibles pour une Arduino :

- **Mesure et détection :**
  - Station météorologique automatisée.
  - Détecteur de foudre.
  - Suivi du soleil pour orientation des panneaux solaires.
  - Moniteur de radiation.
  - Détecteur automatique de la faune.
  - Système de sécurité domestique ou professionnel.
  - Électronique industrielle et embarquée.
- **Contrôle :**
  - Petits robots.
  - Maquette de fusée ou d'avion.
  - Drones multi-rotor.
  - CNC simple pour petites machines-outils.
  - Contrôler les appareils domestiques.

- **Automatisation :**
  - Serre automatisée.
  - Aquarium automatisé.
  - Robot navette d'échantillon de laboratoire.
  - Enceinte thermique de précision (couveuse, yaourtière, étuve, séchoir…).
  - Système de test électronique automatisé.
- **Art** :
  - Contrôle d'éclairage et sonore dynamique.
  - Structures cinématiques.
  - Œuvre d'art.

II.2.9 Modules Arduino :

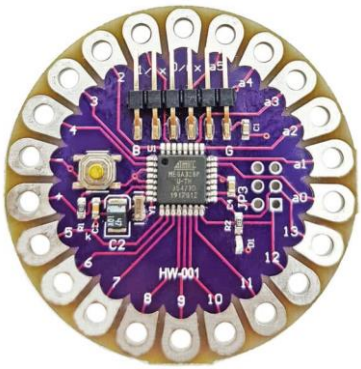
L'Arduino est une plateforme de microcontrôleurs Atmel reposant sur un logiciel libre et ouvert. Ces modules intègrent des éléments qui favorisent une manipulation simple et rapide du Microcontrôleur. Arduino facilite la programmation d'objets ou de dispositifs interactifs. Les objets peuvent être équipés de divers capteurs, témoins lumineux ou interrupteurs à actionner.

La carte Arduino est dotée de connecteurs standard qui permettent la connexion de modules compatibles, connus sous le nom de Shields. Ce sont des circuits de dimensions approximativement équivalentes à celles de l'Arduino qui viennent se superposer sur ces connecteurs. Ils offrent des ajouts matériels qui permettent d'intégrer des caractéristiques uniques à son projet. Outre ces connecteurs, chaque carte est équipée d'une interface USB qui facilite la programmation du microcontrôleur intégré. [13]

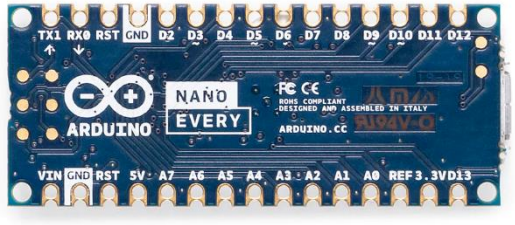
II.2.10 Les différentes cartes Arduino :

Au fil des ans, les développeurs d'Arduino créent une quantité considérable de cartes distinctes disponibles sur le marché. La liste que nous suggérons, C'est une représentation des cartes les plus prisées et couramment utilisées. [14]

Arduino Méga 2560

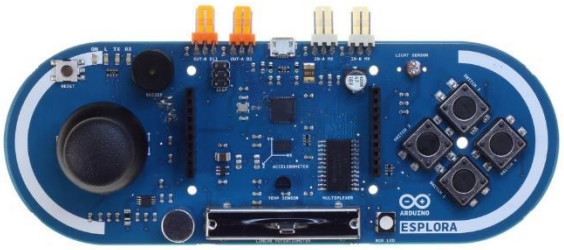


Arduino Lily Pad18



Arduino Nano 19

Arduino Leonardo



Arduino Esplora

Arduino Yun

Figure (II.5) : les différents types de cartes Arduino

II.2.10.1 Arduino Méga 2560 :

Catégories	**Valeur**
Microcontrôleur	ATmega 2560
Fréquence d'horloge	16 MHz
Tension de service	5 V
Tension d'entrée (recommandée)	7-12V
Tension d'entrée (limites)	6-20V
Ports numériques	54 entrées et sorties (15 sorties commutables en MLI)
Ports analogiques	16 entrées analogiques
Courant maxi par broche d'E/S (c.c.)	40 mA
Courant maxi par broche 3,3 V	50 mA
Mémoire	256 Ko Flash, 8 Ko SRAM, 4 Ko EEPROM
Chargeur d'amorçage	8 Ko (en mémoire Flash)
Interface	USB
Dimensions	10,16 cm x 5,3 cm

Tableau N 01 Caractéristique Arduino Méga 2560

II.2.10.2 Arduino Lily Pad :

Catégories	Valeur
Microcontrôleur	ATmega 168v ou 328
Fréquence d'horloge	8 MHz
Tension de service	2.7-5.5v
Tension d'entrée (recommandée)	2.7-5.5v
Ports numériques	14 entrées et sorties (5 sorties commutables en MLI)
Ports analogiques	6 entrées analogiques
Courant maxi par broche d'E/S (c.c.)	40 mA
Mémoire	16 Ko Flash, 1 Ko SRAM, 512 Ko EEPROM
Chargeur d'amorçage	2 Ko (en mémoire Flash)
Dimensions	5 cm de diamètre

Tableau N 02 Caractéristique Arduino Lily Pad

II.2.10.3 Arduino Nano :

Catégories	Valeur
Microcontrôleur	ATmega 168 ou 328
Fréquence d'horloge	16 MHz
Tension de service	5 V
Tension d'entrée (recommandée)	7-12V
Tension d'entrée (limites)	6-20V
Ports numériques	14 entrées et sorties (6 sorties commutables en MLI)
Ports analogiques	8 entrées analogiques
Courant maxi par broche d'E/S (c.c.)	40 mA
Mémoire	16 Ko (en mémoire Flash), 1 ko SRAM, ATmega 168
Mémoire	32 Ko mémoire Flash, 2 Ko SRAM, 1 Ko EEPROM
Chargeur d'amorçage	
Interface	USB
Dimensions	10,16 cm x 5,3 cm

Tableau N 03 Caractéristique Arduino Nano

II.2.10.4 Arduino Leonardo :

Catégories	Valeur
Microcontrôleur	ATmega 32U4
Fréquence d'horloge	16 MHz
Tension de service	5 V
Tension d'entrée (recommandée)	7-12V
Tension d'entrée (limites)	6-20V
Ports numériques	20 entrées et sorties (7 sorties commutables en MLI)
Ports analogiques	12 entrées analogiques
Courant maxi par broche d'E/S (c.c.)	40 mA
Courant maxi par broche 3,3 V	50 mA
Mémoire	32 Ko Flash, 2.5 Ko SRAM, 1 Ko EEPROM
Chargeur d'amorçage	4 Ko (en mémoire Flash)
Interface	USB
Dimensions	6.86 cm x 5,3 cm

Tableau N 04 Caractéristique Arduino Leonardo

II.2.10.5 Arduino Esplora :

Catégories	Valeur
Microcontrôleur	ATmega 32U4
Fréquence d'horloge	16 MHz
Tension de service	5 V
Mémoire	32 Ko Flash, 2.5 Ko SRAM, 1 Ko EEPROM
Chargeur d'amorçage	4 Ko (en mémoire Flash)
Interface	USB
Dimensions	16,51 cm x 5,3 cm

Tableau N 05 Caractéristique Arduino Esplora

II.2.10.6 Arduino Yun :

Catégories	Valeur
Microcontrôleur	ATmega 32U4
Fréquence d'horloge	16 MHz
Tension de service	5 V
Tension d'entrée	5 V

Ports numériques	20 entrées et sorties (7 sorties commutables en MLI)
Ports analogiques	12 entrées analogiques
Courant maxi par broche 3,3v (c.c.)	50 mA
Courant maxi par broche 5 V	50 mA
Mémoire	32 Ko Flash, 2.5 Ko SRAM, 1 Ko EEPROM
Dimensions	7 cm x 5,3 cm

Tableau N 06 Caractéristique Arduino Yun :

II.3 Pourquoi Arduino UNO ?

Un grand nombre de cartes électroniques basées sur microcontrôleur sont réalisées à partir de la plateforme proposée pour l'électronique programmable. Ces outils intègrent les subtilités de la programmation et de l'intégration dans une interface aisée. De même, Arduino facilite la manipulation des microcontrôleurs et propose à ceux qui s'y intéressent divers bénéfices tels que :

- Coût (réduit) : les cartes Arduino sont largement économiques comparativement aux autres plateformes. On peut assembler manuellement les versions les plus économiques du module Arduino.
- Compatible avec plusieurs plateformes : le programme Arduino, développé en C++, est opérationnel sur les systèmes d'exploitation Windows, Macintosh et Linux. La majorité des systèmes de microcontrôleurs sont confinés à l'environnement Windows.
- Un environnement de codage clair et accessible : l'environnement de codage Arduino (le logiciel Arduino IDE) est conçu pour être simple d'utilisation pour les novices, tout en offrant la flexibilité nécessaire pour que les utilisateurs plus expérimentés puissent également l'apprécier.
- Logiciel extensible et open source : le logiciel Arduino ainsi que son langage sont mis à disposition sous une licence open source, permettant aux programmeurs chevronnés de les enrichir. L'application de développement Arduino est une application JAVA compatible avec plusieurs plateformes (elle peut fonctionner sur divers systèmes d'exploitation), servant d'éditeur de code et de compilateur, et a la capacité de transférer le programme par le biais de la connexion série (RS232, Bluetooth ou USB en fonction du module).
- Équipement open source et évolutif : les cartes Arduino se basent sur les microcontrôleurs Atmel ATMEGA8, ATMEGA168 et ATMEGA328. Les modèles des modules sont diffusés sous une licence Creative Commons. Les concepteurs de circuits chevronnés ont la capacité de

concevoir, d'achever et d'améliorer leurs propres variantes de cartes Arduino. Même ceux qui n'ont pas d'expérience peuvent concevoir la version de test de la carte Arduino. [15]

II.4 Les composants de la carte Arduino UNO :

Un module Arduino est généralement conçu autour d'un microcontrôleur ATMEL AVR et d'éléments supplémentaires qui simplifient la programmation et l'interaction avec d'autres dispositifs. Chaque module possède au minimum un régulateur linéaire de 5 V et un oscillateur, Quartz à 16 MHz (ou résonateur en céramique sur quelques modèles). Le microcontrôleur est déjà équipé d'un chargeur de démarrage, il n'est donc pas nécessaire d'utiliser un programmateur spécifique.

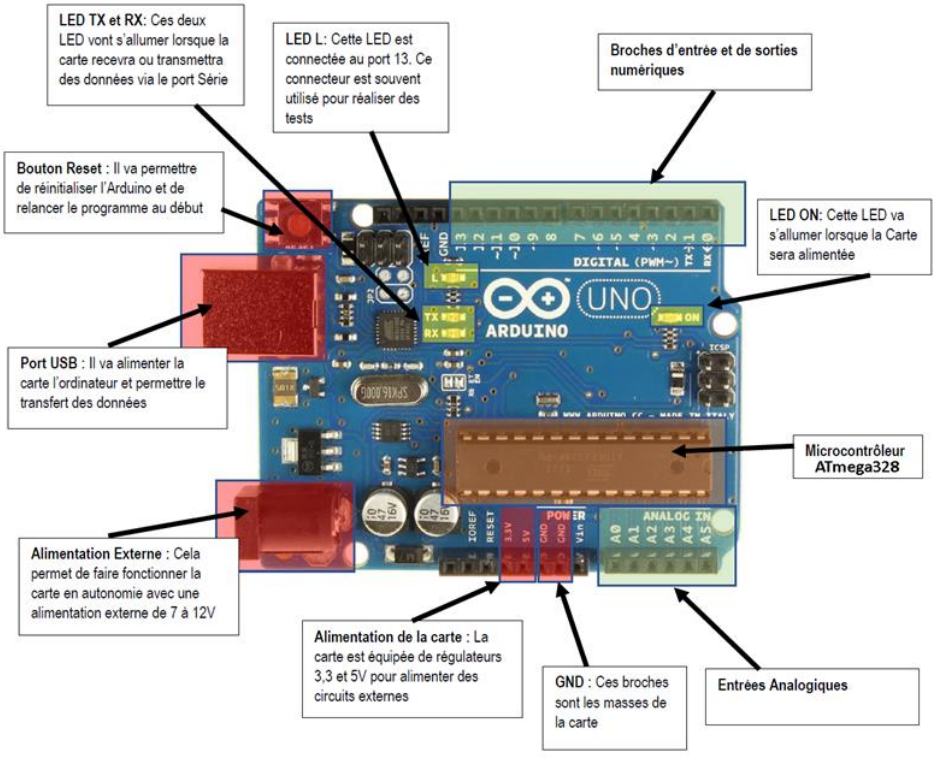


Figure (II.6) : les composants de la carte Arduino UNO

II.4.1 Microcontrôleur ATmega328 :

Atmel, actuellement une division de Microchip Technology, a créé le ATmega328, un microcontrôleur largement prisé. On l'utilise fréquemment dans différents projets et applications électroniques grâce à sa simplicité, sa polyvalence et son accessibilité répandue.

L'ATmega328 est un microcontrôleur 8 bits qui repose sur l'architecture RISC AVR, proposant une multitude de fonctionnalités. Il est équipé d'une mémoire flash de 32 Ko pour le code du programme, de 2 Ko de SRAM et de 1 Ko d'EEPROM pour le stockage des données.

Doté de 23 broches d'entrée/sortie, avec 14 numériques et 6 analogiques, il offre une large possibilité de connexion. Il comprend un convertisseur analogique-numérique de 10 bits ainsi que des temporisateurs pour la gestion du temps. En outre, il prend en charge les interfaces de

communication telles que UART, SPI et I2C. L'ATmega328 fonctionne avec une tension allant de 1,8 V à 5,5 V, ce qui le rend flexible pour diverses applications. [16]

II.4.1.1 Applications des microcontrôleurs :

- ❖ Informatique (souri, modem…).
- ❖ Vidéo (appareil photos numérique, caméra numérique…).
- ❖ Contrôle des processus industriels (régulation, pilotage, supervision…).
- ❖ Appareil de mesure (affichage, calcul statistique, mémorisation…).
- ❖ Automobile (ABS, injection, GPS, airbag …).
- ❖ Multimédia (téléviseur, carte audio, carte vidéo, MP3, magnétoscope…).
- ❖ Téléphone (fax, téléphone portable, modem …).
- ❖ Électroménager (Lave-vaisselle, lave-linge, four micro-onde …). [13]

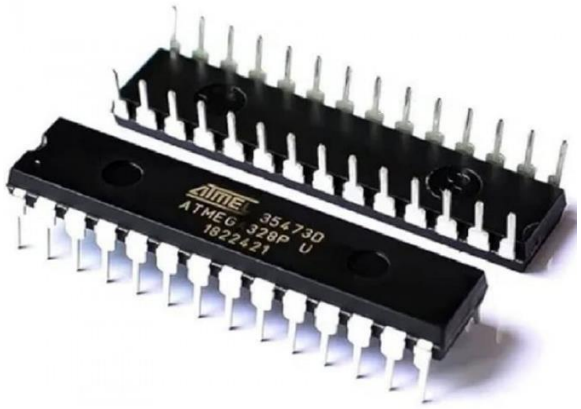


Figure (II.7) : Le microcontrôleur ATmega 328

II.4.1.2 Alimentation :

Pour assurer le fonctionnement de la carte, une alimentation est indispensable. Le microcontrôleur étant alimenté par une tension de 5V, on peut l'alimenter en branchant la carte au port USB de l'ordinateur via un câble USB de type A à type B.

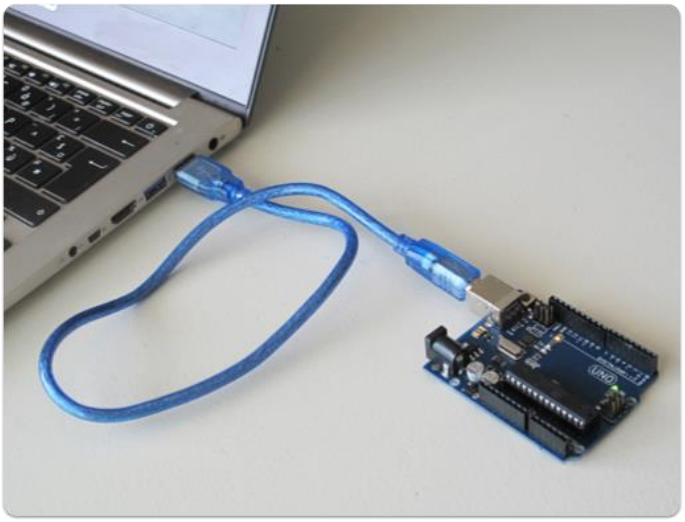


Figure (II.8) : L'alimentation de la carte Arduino par le port USB

Ou par une source d'alimentation externe dont la tension varie entre 7 et 12 volts. Cette tension doit être de nature continue et peut, par exemple, être assurée par une batterie de 9V.

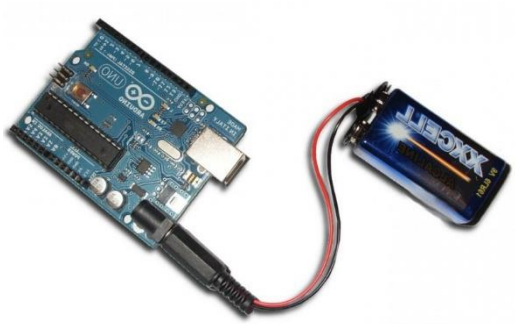


Figure (II.9) : L'alimentation de la carte Arduino par une pile 9v

Ou bien par une alimentation continue stabilisée :

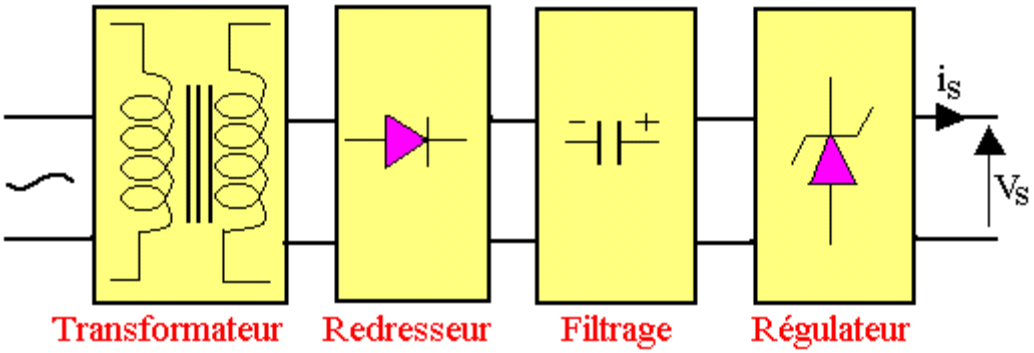


Figure (II.10) : Principe d'une alimentation stabilisée

Quand la carte Arduino est alimentée par une source externe, un régulateur intégré sur la carte prend en charge la réduction de la tension à 5V afin d'assurer le fonctionnement correct de la carte Arduino. [16]

II.4.2 Le bouton Reset :

Permet le redémarrage de la carte ainsi que la réinitialisation du programme qu'elle héberge.

II.4.3 Visualisation :

Les trois « points blancs » correspondent en réalité à des LED d'approximativement un millimètre. Ces indicateurs possèdent deux rôles : Ils scintillent quelques instants lorsque la carte est branchée à l'ordinateur, LED située en bas du cadre : utilisée pour le transfert du programme vers le microcontrôleur. Figure (II.6). [17]

II.4.4 Les entrées/sorties numériques : de D0 à D13 :

Quatorze broches numériques, allant de 0 à 13, sont disponibles pour une utilisation en tant qu'entrées ou sorties numériques, en fonction du paramétrage désiré. L'annonce de ces broches doit être faite en début de programme si l'on souhaite les utiliser. Selon l'API employée, cette configuration pourrait être déjà incluse où exiger une déclaration explicite. Les pins

numériques de la carte Uno opèrent à 5 V et sont capables de délivrer ou d'absorber jusqu'à 40 mA de courant. Cela doit être pris en considération lors de l'élaboration de projets électroniques afin d'éviter toute surcharge ou court-circuit. [18]

II.4.5 Les entrées analogiques A0 à A5 :

Il est équipé de six entrées analogiques, identifiées de A0 à A5, aptes à mesurer la tension analogique. Chaque entrée analogique dispose d'un convertisseur analogique-numérique (CAN) de 10 bits, signifiant que la tension mesurée peut être transformée en une valeur numérique allant de 0 à 1023. Cette résolution de 10 bits offre une précision adéquate pour diverses applications comme l'évaluation de la luminosité ambiante, de la température ou du niveau de charge de la batterie. Concernant la tension, la sensibilité s'établit à 5/1024, soit 4,88 mV. [18]

II.5 Installation de code sur une carte Arduino :

Pour télécharger du code sur une carte Arduino via un PC, voici les étapes à suivre :

1. Mise en place du logiciel Arduino IDE : télécharger et installez le logiciel Arduino IDE à partir du site officiel d'Arduino.
2. Liaison de la carte Arduino : Branchez votre carte Arduino à votre ordinateur en utilisant un câble USB.
3. Sélection de la carte et du port : Dans l'environnement de développement Arduino IDE, choisissez le modèle de votre carte Arduino ainsi que le port COM associé à cette dernière dans les sections « Outils ».
4. Télécharger le code : Accédez au code que vous avez programmé dans l'environnement de développement Arduino IDE, puis appuyez sur le bouton « Téléverser » pour compiler et transférer le code vers la carte.
5. Vérification du téléchargement : Lorsque le téléchargement est complet, une notification de confirmation apparaît dans le programme.

II.6 Les capteurs :

II.6.1 Définitions :

On définit un « CAPTEUR » comme un appareil qui convertit une mesure physique d'entrée, nommée mesurande, en une grandeur généralement électrique (Elle peut être : une charge, une tension (V), un courant (I), une impédance (R, L, C)) qu'on appelle réponse. [19]

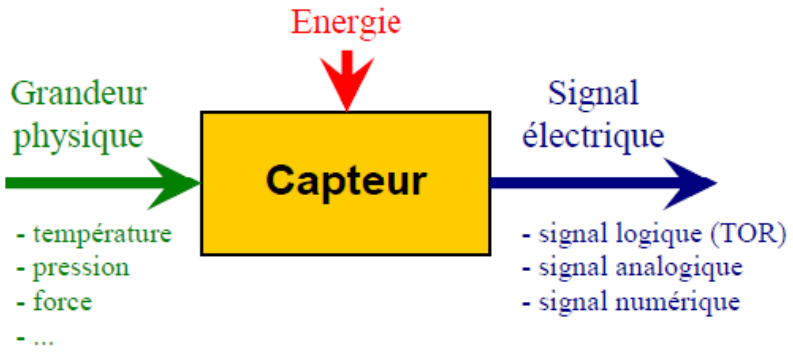


Figure (II.11) : Schéma de principe d'un capteur

II.6.2 Différents types de capteurs :

On trouve une vaste sélection de capteurs, chacun étant conçu pour répondre à des exigences particulières. Parmi eux, les plus importants sont :

II.6.2.1 Capteurs mécaniques et physiques :

- Capteur de force : évalue la force appliquée sur un objet.
- Capteur de masse ou cellule de pesée : employé dans les balances pour quantifier la masse.
- Capteur de pression : Évalue la pression d'un fluide ou d'un gaz dans les milieux industriels. Dispositif de détection de mouvement : il perçoit les actions physiques dans un espace donné.
- Sonde de position : Identifie la localisation exacte d'un objet dans l'espace.
- Capteur de chocs : Il identifie les vibrations ou les impacts sur le matériau.
- Accéléromètre : évalue la rapidité d'un élément en déplacement.

II.6.2.2 Capteurs électriques et électroniques :

- Détecteur de courant : Évalue la force du courant électrique.
- Capteur électrique : il transforme les mesures électriques en informations exploitables.
- Capteur de puissance : Évalue l'énergie électrique utilisée ou générée.
- Mesure de tension : Permet d'évaluer le degré de tension dans un circuit électrique.

II.6.2.3 Capteurs environnementaux :

- Détecteur de température : évalue la chaleur ou le froid dans un espace ou un dispositif.
- Capteur d'humidité : il mesure le taux d'humidité.
- Détecteur de lumière : Évalue l'intensité de la lumière ambiante.
- Capteur de qualité de l'air : mesure la concentration de substances polluantes dans l'atmosphère.

II.6.2.4 Détecteurs pour la sécurité et la domotique :

- Détecteur d'ouverture de porte ou switch de porte : employé pour déterminer si la porte est en position ouverte ou fermée.

❖ Détecteur d'ouverture de porte intelligent : incorpore les capacités IoT pour la supervision à distance.

❖ Capteur de choc : pratique pour détecter les tentatives d'intrusion.

II.6.2.5 Capteurs spécialisés :

❖ Girouette : Évalue l'orientation du vent.

❖ Anémomètre : Évalue la rapidité du vent.

❖ Détecteur de niveau : surveille le niveau des fluides dans les contenants. [20]

II.6.3 Classification des capteurs :

La classification est réalisée par :

- L'instrument de mesure qu'ils représentent (capteur de température, capteur de pression, etc.)
- Leur fonction dans le processus industriel (vérification des produits finis, sécurité, etc.)
- Le signal émis par (capteur analogique, capteur logique, capteurs numériques)
- Leur méthode de conversion de la grandeur (capteur de résistance, effet Hall, etc.)
- Mode d'opération : Les capteurs opèrent sur deux principes fondamentaux : Selon l'origine du signal électrique produit. Nous faisons la distinction.

❖ Capteur actif : Ce type de capteur agit comme un générateur. Un capteur actif s'appuie généralement sur les principes physiques qui garantissent la conversion en énergie électrique de diverses formes de mesures : thermique, mécanique ou radiante.

❖ Capteur passif : C'est une impédance où l'un des critères spécifiés est la sensibilité de la mesure. [21]

II.6.4 Les capteurs utilisés :

II.6.4.1 Capteur de Température et d'humidité DHT22 :

II.6.4.1.1 Description :

Le DHT22 est un capteur économique qui permet d'obtenir numériquement la température et l'humidité ambiantes. Ce dispositif, doté d'un capteur capacitif d'humidité et d'une thermistance, détermine la température et l'humidité de l'air et transmet ces données de manière numérique via un bus série. Les données sont mises à jour toutes les deux secondes. La mise en place de ce capteur est très facile, il suffit de brancher la première broche à gauche à la source d'alimentation (3V à 5V), le pin central à une broche Arduino configurée en entrée (INPUT) et le pin de droite à la terre (GND).

Ce détecteur est proposé avec une résistance de tirage variant entre 4,7KΩ et 10KΩ pour établir la connexion entre la broche DATA et le VCC. [22]

Figure (II.12) : Capteur de température et d'humidité DHT22

II.6.4.1.2 Caractéristiques techniques :

- Tension d'alimentation : 3 à 6V DC
- Courant de fonctionnement : < 2.5mA
- Plage de température : -40 à +80°C
- Précision de température : ± 0.5°C
- Humidité : de 0 à 100% RH
- Précision d'humidité : ± 2% HR
- Temps de réponse : <1s
- Dimension : 27mm x 59mm x 13.5mm, (1.05" x 2.32" x 0.53")
- Type de connexion : 3 fils (Vcc, Signal, GND)
- Broche 1 à gauche à la tension d'alimentation comprise entre 3 et 5V.
- Broche 2 à votre broche d'entrée de données (doit être connecté à une résistance de pull up, interne ou externe).
- Broche 3 non connecté.
- Broche 4 Masse. [22]

II.6.4.2 Domaines d'utilisation de capteur de température et d'humidité DHT22 :

Le capteur DHT22, grâce à sa précision exceptionnelle dans le suivi de la température et de l'humidité relative, est couramment employé dans une multitude de projets d'automatisation et de contrôle environnemental. Doté d'une capacité de fonctionnement allant de -40 à 80 degrés Celsius et d'une exactitude de ±0,5 degré pour la température, il est également capable de mesurer l'humidité relative de 0 à 100% avec une précision de ±2%. Grâce à son interface de communication conviviale, généralement fondée sur le protocole numérique à un seul fil, il est compatible avec une variété de plateformes de microcontrôleurs et de capteurs, simplifiant ainsi son incorporation dans différents systèmes.

En tant que capteur passif, il répond aux variations de température et d'humidité sans avoir besoin d'une source d'énergie externe, tout en se comportant comme un capteur numérique qui traduit ces changements en signaux numériques utilisables avec des systèmes numériques et des microcontrôleurs. Le capteur DHT22 est couramment choisi pour la régulation

climatique, le suivi environnemental, les systèmes de maison intelligente et l'agriculture intelligente grâce à ses caractéristiques remarquables. [23]

II.6.4.3 Le capteur de distance ultrason HC-SR04 :

Le capteur HC-SR04 fait appel à des ondes ultrasonores pour estimer la distance d'un objet. Il propose une large portée de détection sans contact, alliant précision élevée et mesures constantes. Son fonctionnement n'est pas influencé par la lumière solaire ou les matériaux de couleur sombre. [24]

Avec une exactitude de 3mm, le capteur à ultrasons HC-SR04 peut évaluer la distance des objets se trouvant entre 2cm et 400cm du dispositif. Le capteur comprend un émetteur d'ultrasons, un récepteur et le circuit de contrôle. [25]

Figure (II.13) : Capteur ultrason HC-SR04

II.6.4.3.1 Le principe de fonctionnement :

1. Transmettre un signal numérique à l'état haut sur l'émetteur durant 10 µs.
2. Le capteur émet automatiquement 8 impulsions d'ultrasons à 40 kHz et enregistre les signaux de retour. Si le signal est rétabli, la durée pendant laquelle le signal est élevé correspond au temps écoulé entre l'émission des ultrasons et leur réception.
3. Calcul de la distance : Distance = (temps en état haut du signal reçu * vitesse du son) / 2 (vitesse du son dans l'air : 340 m/s) Figure (II.14). [26]

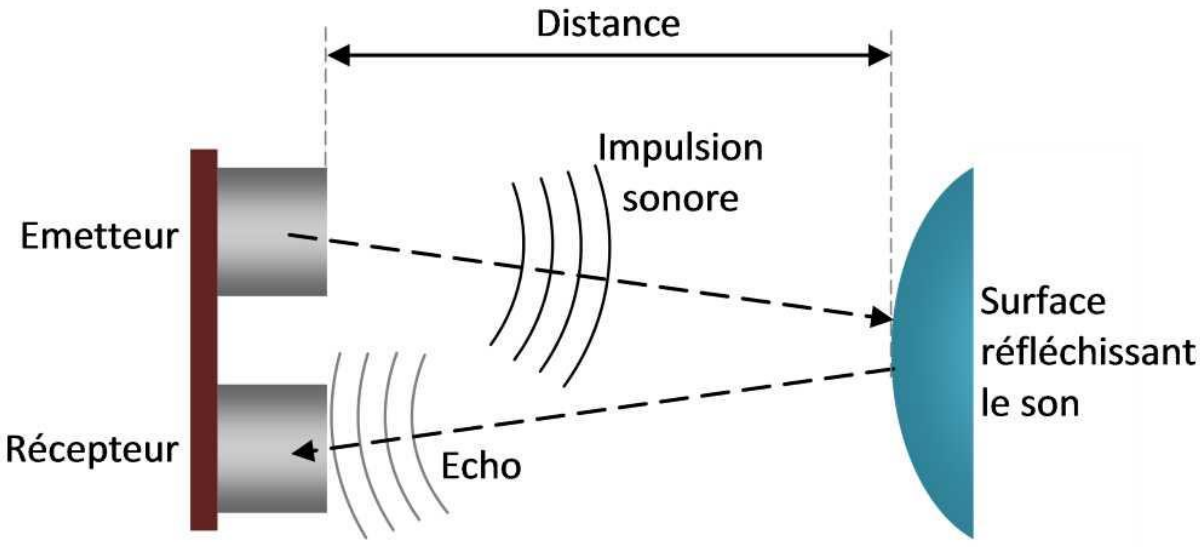


Figure (II.14) : Fonctionnement capteur ultrason HC-SR04

II.6.4.3.2 Caractéristiques : [25]

Tension d'alimentation	5V DC
Courant d'alimentation	15mA
Fréquence de travail	40Hz
Distance maximale de détection	4m
Distance minimale de détection	2cm
Angle de détection	15 degrés
Résolution de la mesure	0.3 cm
Signal d'entrée de l'émetteur	Impulsion à l'état haut de 10µs
Signal de sortie du récepteur	Signal numérique à l'état haut et la distance proportionnellement
Dimension	45*20*15mm

Tableau N 07 Caractéristiques ultrason HC-SR04

II.6.4.3.3 Broches de connexion :

- Vcc = Alimentation +5 V DC
- Trig = Entrée de déclenchement de la mesure (Trigger input)
- Echo = Sortie de mesure donnée en écho (Echo output)
- GND = Masse de l'alimentation. [24]

II.6.4.4 Capteur de débit d’eau YF-S201 :

II.6.4.4.1 Description :

Le capteur de débit d'eau YF-S201 est un dispositif servant à évaluer le volume d'eau qui traverse un tuyau ou une conduite. C'est un détecteur sûr et exact qui peut servir dans diverses applications, y compris le suivi de la consommation en eau, la gestion de l'irrigation, le contrôle du flux d'eau dans les systèmes de refroidissement ou encore la détection des fuites d'eau. On l'utilise aussi fréquemment pour la facturation de l'eau concernant les logements d'immeubles ou les édifices industriels. [27]

Figure (II.15) : Capteur de débit d’eau YF-S201

II.6.4.4.2 Principe du fonctionnement :

Le débitmètre d'eau contient une roue turbine qui s'active lorsqu'elle est traversée par l'eau. La rapidité de rotation de la turbine est corrélée à la vitesse d'écoulement de l'eau. Un capteur à effet Hall génère aussi une impulsion à chaque tour complet de la roue de turbine, et le nombre d'impulsions est proportionnel à la vitesse de rotation de cette dernière. [27]

II.6.4.4.3 Choix d'un capteur :

On choisit parmi diverses catégories de détecteurs de présence, en considérant.

• Type d'objet à identifier : solide, liquide, gazeux, métallique ou non.

• possibilité de contact avec l'objet.

• distance entre l'objet et le détecteur, poids de l'objet.

• vitesse de déroulement.

• rythmes de manœuvres.

• espace pour l'intégration du capteur dans l'appareil.

• l'environnement : conditions de température, taux d'humidité, présence de poussières, éclaboussures variées, et ainsi de suite.

• la source d'énergie : alternative ou continue.

• signal de sortie : électromécanique, statique.

• Mode de connexion : fil, terminal, connecteur. [28]

II.7 Conclusion :

Dans ce chapitre, nous avons mené une analyse théorique de notre projet et discuté d'Arduino dans son ensemble. Grâce à sa simplicité d'emploi et son coût modique, Arduino trouve de nombreuses applications dans des domaines tels que l'électronique intérieure et industrielle ainsi que la domotique.

Par la suite, nous avons donné une explication détaillée des deux composantes majeures: le système Arduino et les capteurs employés pour déterminer le niveau et la condition du fluide dans le réservoir.

Ainsi, nous avons cité les différents équipements employés dans ce projet (capteurs, actionneurs) qui nous permettent de saisir les fondements et les spécificités des composants et des unités intégrés au projet, ainsi que de comprendre les méthodes opérationnelles et le logiciel mis en œuvre.

III Chapitre III : RÉALISATION DE PROJET

III.1 Introduction

L'objectif de ce projet est de mettre en place un système automatique fiable et précis pour le remplissage des réservoirs grâce à Arduino, en gérant et surveillant le niveau des liquides. L'objectif est de satisfaire aux besoins industriels et domestiques pour protéger nos ressources et notre environnement, tout en régulant leur utilisation afin d'éviter le gaspillage et les désastres liés à la défaillance des systèmes classiques de supervision et de contrôle.

III.2 Système intelligent de remplissage de liquide contrôlé par Arduino :

Le système intelligent de remplissage de liquide géré par Arduino est un équipement automatique qui se sert de capteurs, d'algorithmes de commande et d'actionneurs pour effectuer le remplissage précis et répétitif de réservoirs avec des liquides. Il trouve son utilité dans une multitude d'applications domestiques et industrielles, comme la fabrication de boissons, la production de produits chimiques, ou encore la création de réservoirs pour l'eau et les carburants. Ce dispositif peut être adapté pour recevoir aisément une variété de liquides et de contenants. Figure (III.1).

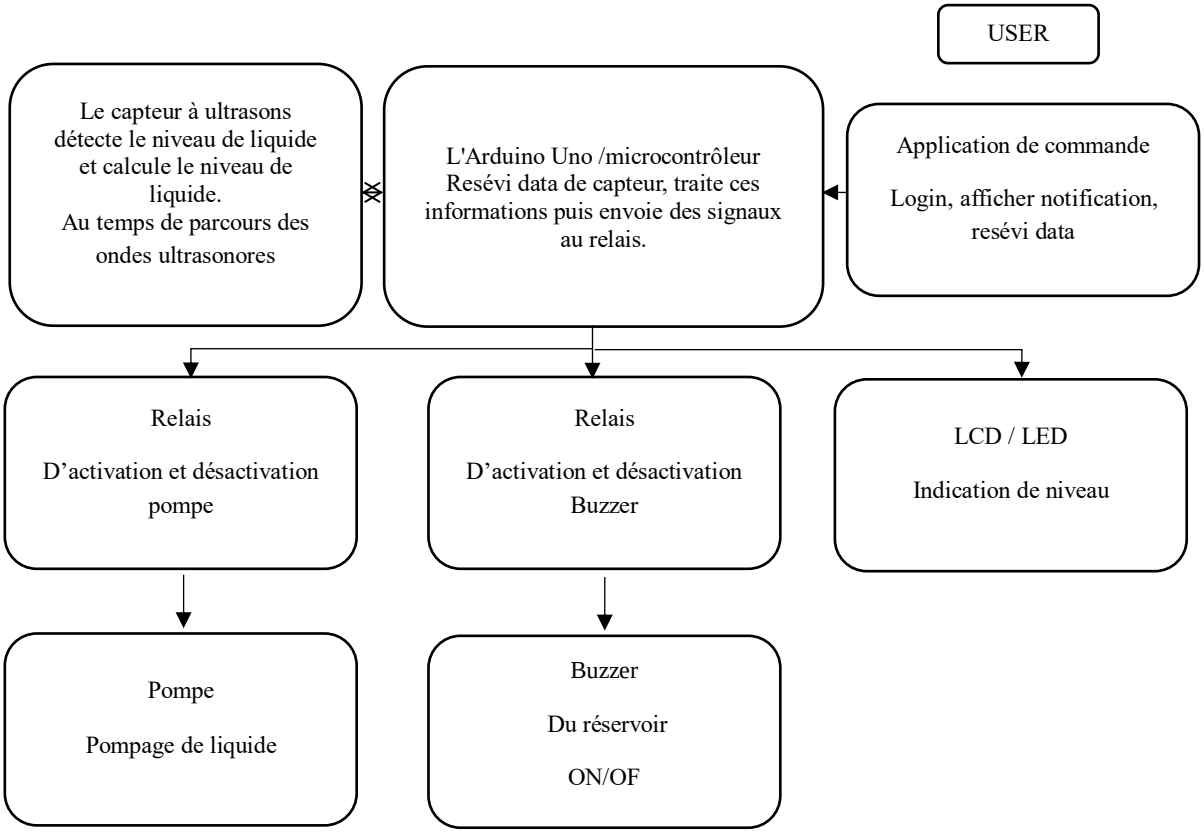


Figure (III.1) : Organigramme de système [29]

III.3 Objectif du projet :

a) Dans ce projet, nous allons réaléser un système qui remplit automatiquement des réservoirs en utilisant la carte Arduino.

b) La carte Arduino prend en charge toutes les fonctionnalités de contrôle.

c) L'utilisateur saisit d'abord la quantité de liquide à remplir.

d) La pompe à eau tire l'eau du réservoir, qui est ensuite dirigée vers le réservoir de stockage.

e) Le détecteur à ultrasons de liquide assiste dans la transmission de la mesure du niveau d'eau vers la carte Arduino.

f) En fin de compte, le relais coupe la pompe à eau lorsque le niveau du liquide atteint l'altitude spécifiée par l'utilisateur.

III.4 Fonctionnement de système :

1. Le système est paramétré selon les critères de remplissage nécessaires, comme le niveau de remplissage, la température et l'humidité à l'intérieur du réservoir.

2. Le dispositif emploie des détecteurs pour évaluer le niveau de liquide, la température et l'humidité dans le réservoir.

3. Le dispositif fait appel à des programmes de gestion et de régulation afin d'évaluer le niveau de remplissage et pour indiquer la température ainsi que l'humidité présentes dans le réservoir, en vue d'atteindre les critères de remplissage désirés.

4. Le dispositif emploie des actionneurs de remplissage et des vannes pour le contrôle du débit afin de remplir le récipient avec exactitude et de façon répétée.

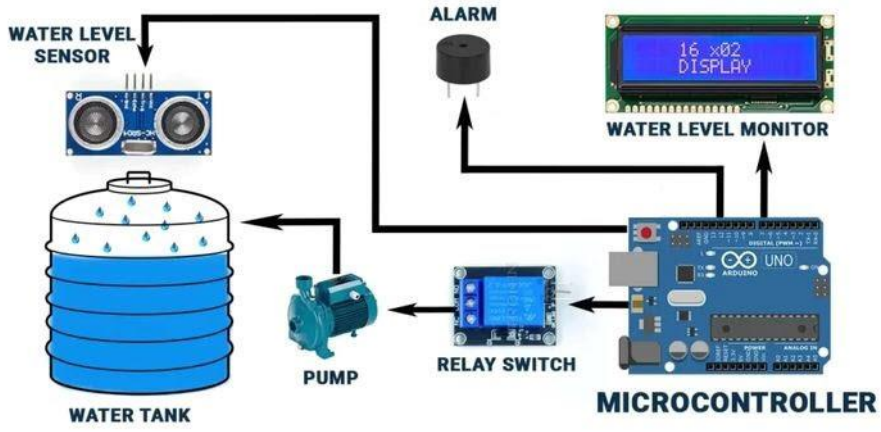


Figure (III.2) : Schéma de principe de système [25]

III.5 Le matériel utilisé dans ce projet

III.5.1 Afficheur LCD 2 x16C vert :

LCD est l'acronyme de « Liquid Crystal Display » en anglais, qui se traduit par « Écran à Cristaux Liquides » en français. Il s'agit d'un affichage alphanumérique 2 x 16 caractères avec un rétroéclairage bleu. Un microcontrôleur ou un Arduino contrôle l'écran LCD 16x2 par le

biais d'un protocole de communication série. Il est doté de 16 caractères répartis sur deux lignes et est fréquemment employé dans divers projets électroniques. [30]

Figure (III.3) : Afficheur LCD 16*2

III.5.2 Relais :

Un relais électromécanique est un dispositif électrique qui assure la distribution de l'énergie sur la base d'un signal émis par l'organe de commande. Par conséquent, un relais offre la possibilité d'ouvrir et de fermer un circuit électrique de puissance à partir d'un signal logique.

Les deux circuits, l'un pour la puissance et l'autre pour l'information, sont totalement séparés par une isolation galvanique et peuvent présenter des spécificités distinctes en matière d'alimentation électrique.

La fonction principale des relais consiste généralement à isoler les circuits de commande des circuits de puissance, par exemple pour contrôler une tension ou un courant important à partir d'une commande moins puissante. Dans certaines situations, cela garantit également la sécurité de l'opérateur. [31]

Figure (III.4) : Relais

III.5.3 Pompe :

Toutes les pompes servent le même objectif, à savoir transporter un liquide d'un endroit à un autre. Pour effectuer ce transfert, il est nécessaire d'apporter de l'énergie au liquide, et c'est précisément là que les pompes interviennent.

Généralement, l'usage de pompes est privilégié, permettant ainsi d'obtenir des débits constants et maîtrisés, sur des distances et altitudes significatives. [32]

Figure (III.5) : Pompe

III.5.4 Résistances :

Une résistance est un élément électrique ou électronique qui se définit principalement par sa capacité à fournir une opposition variable au flux du courant électrique. (Évaluée en ohms : Ω). [32]

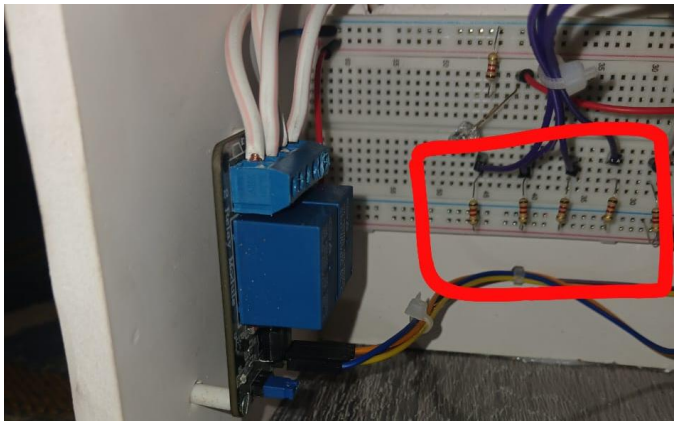


Figure (III.6) : Résistance

III.5.5 Transformateur 220/12 V :

Le transformateur 220 V vers 12 V convertit les 220 volts de la source électrique en une tension sécurisée de 12 V pour alimenter les éléments de ce projet.

Figure (III.7) : Transformateur

III.5.6 Cable USB :

Le câble USB est un élément crucial pour l'échange d'informations entre l'ordinateur et la carte Arduino. Il fournit non seulement l'énergie à la carte, mais également permet sa programmation grâce au logiciel Arduino IDE. De plus, il offre la possibilité d'utiliser le Moniteur Série, un instrument de diagnostic et de débogage qui permet d'observer les informations transmises entre la carte et l'ordinateur.

Figure (III.8) : Cable USB type A-B

III.5.7 Les LED :

Une diode (jonction PN) est un élément qui génère de la lumière quand elle est traversée par un courant électrique dans le sens direct.

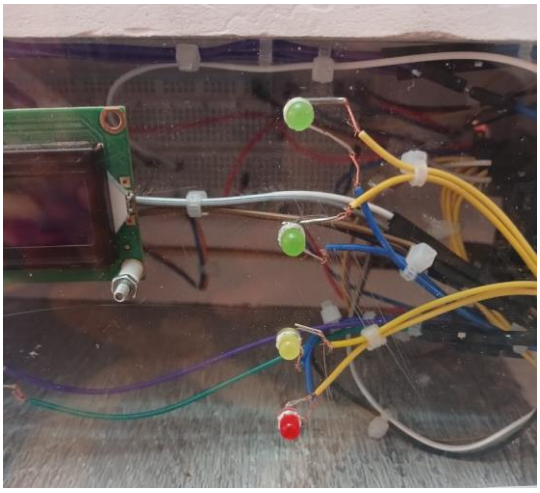


Figure (III.9) : Les LED

III.5.8 Fils de pin d'Arduino :

Les fils mâles sont des connexions qui facilitent la liaison entre l'Arduino et d'autres éléments électroniques, ou encore la réalisation de tests. Elle offre la possibilité d'effectuer des liaisons provisoires entre les divers éléments sans nécessiter de soudure des câbles.

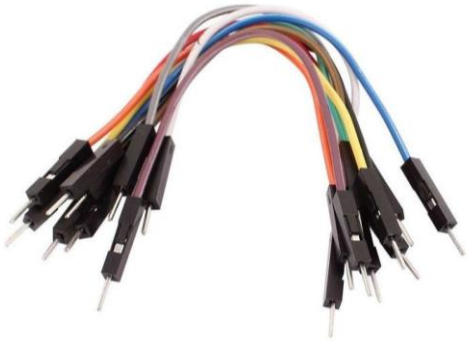


Figure (III.10) : Fils de pin d'Arduino

III.5.9 La plaque d'essai :

C'est un instrument essentiel pour la conception d'un prototype de circuit électronique. La plaque d'essai, couramment utilisée pour les expérimentations avec Arduino, simplifie considérablement le processus de prototypage. [33]

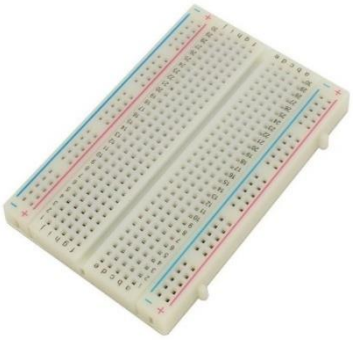


Figure (III.11) : La plaque d'essai

III.5.10 Le Buzzer :

Le buzzer est un appareil acoustique capable de transformer des signaux audios en signaux sonores, et il peut être de type électromagnétique ou piézoélectrique. Ce buzzer passif produit un son d'environ 85 db lorsqu'il est branché à une tension variante entre 4 V et 8 V. Ce buzzer passif a besoin d'un microcontrôleur pour émettre un son à l'aide d'un signal PWM. Le buzzer a une consommation d'environ 30 mA. [34]

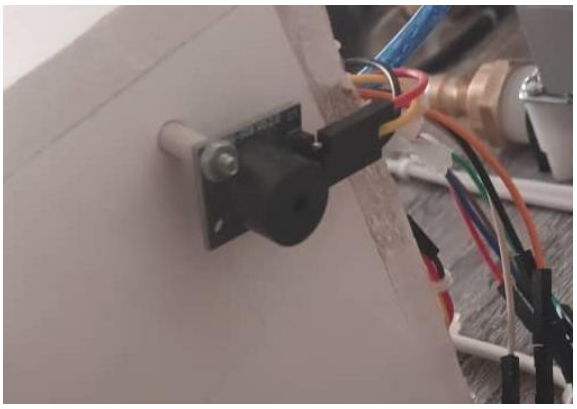


Figure (III.12) : Buzzer

III.5.11 Électrovanne :

Cette valve électromagnétique d'eau reste fermée sans 12V, cependant dès l'application de 12V, la valve s'ouvre permettant à l'eau (ou tout autre liquide) de s'écouler. Peut également servir de soupape d'air. [35]

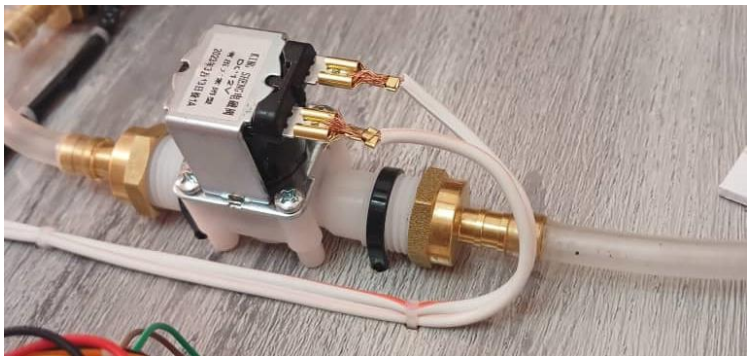


Figure (III.13) : Électrovanne et tuyaux de raccordement

III.5.12 Réservoir de liquide et tuyaux de raccordement :

Figure (III.14) : Réservoir de liquide

III.6 Test du système :

La phase de test représente l'étape ultime où tous les processus sont soumis à une évaluation afin de s'assurer que le système élaboré répond bien aux résultats prévus. On peut tester le système employé dans ce projet en respectant les étapes suivantes : [36]

Nous établissons une liaison entre la source d'alimentation et le système de contrôle du niveau d'eau.

Le système de contrôle du niveau d'eau, ainsi que les équipements de soutien tels qu'Arduino Uno, LCD LM016L, HC-SR04 et module relais, seront ensuite alimentés en énergie.

Lorsque le mécanisme de vérification du niveau de liquide est enclenché, l'appareil à ultrasons HC-SR04 évalue la hauteur de l'eau dans le réservoir. [37]

Quand le capteur à ultrasons HC-SR04 constate que le réservoir est vide ou qu'il est rempli à 10 %, la pompe s'active d'elle-même, cet instrument à ultrasons agit en tant que régulateur de niveau de liquide. Le signal diffusé est envoyé au liquide sous forme d'onde, puis ce dernier est réfléchi et capté par le récepteur à ultrasons. Dès que le signal parvient au récepteur, il est analysé afin de déterminer la distance jusqu'au niveau du liquide dans le réservoir.

Lorsque le réservoir est entièrement plein, la pompe se coupe d'elle-même, l'écran LCD montre et signale le niveau de liquide à diverses fréquences. Le programme s'initialise dès que le circuit est alimenté, établissant ainsi une connexion entre le programme et le circuit du capteur à ultrasons. Par la suite, quand le réservoir de liquide est épuisé, la pompe se met en marche pour le remplir. Lorsque le niveau d'eau atteint 100%, la pompe se coupe et les indications du niveau de liquide dans le réservoir sont présentées sur l'écran LCD, suite à cela, le capteur à ultrasons persiste à fonctionner et, une fois que le niveau de l'eau atteint 10%, la pompe s'active. Parallèlement, les résultats du niveau de liquide dans le réservoir se visualisent

sur l'écran LCD. De ce fait, Le réservoir que nous utilisons ne se videra pas de lui-même, car une fois que le niveau du liquide chute en dessous de 10%, la pompe se met à fonctionner d'elle-même. [38]

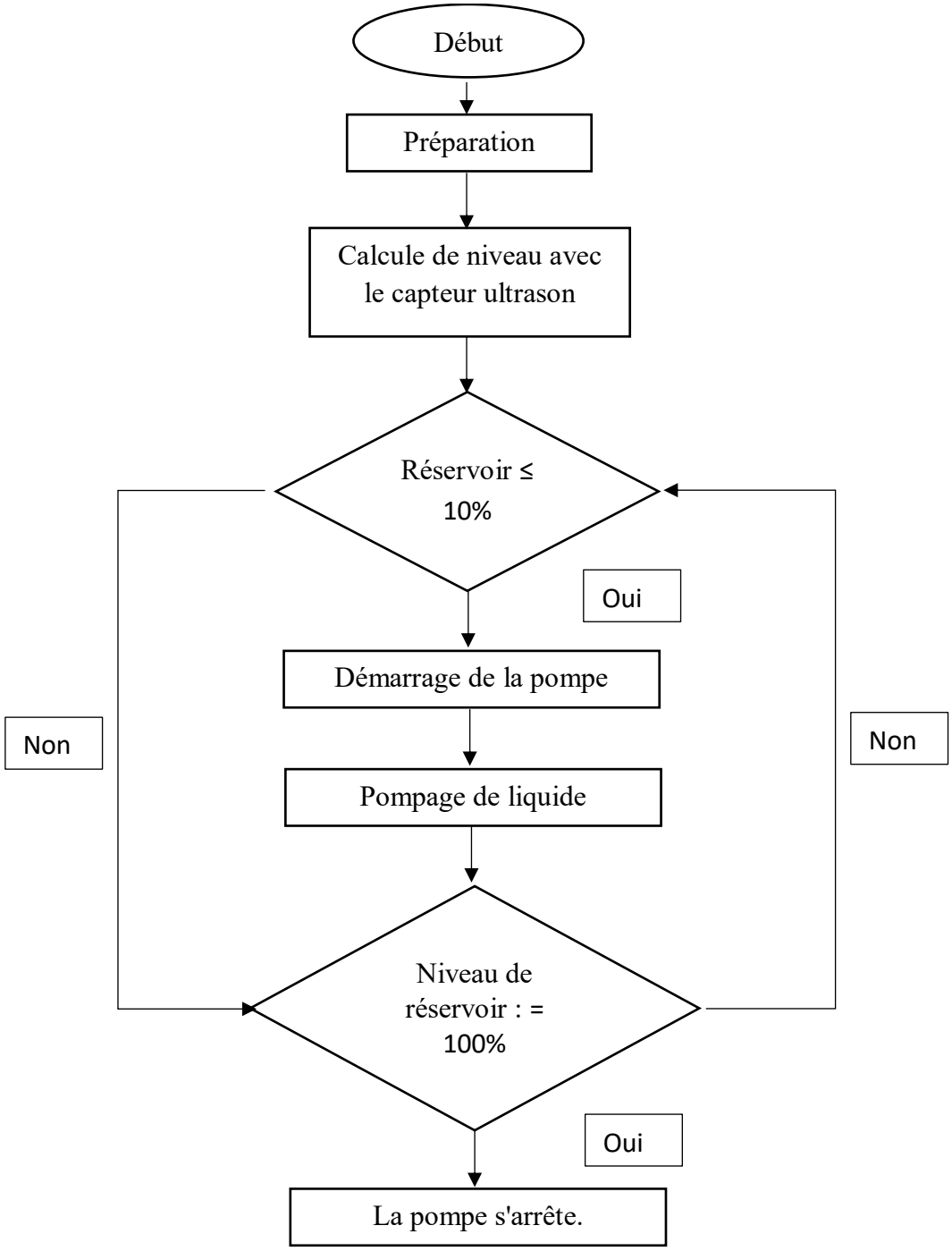


Figure (III.15) : Organigramme de système [39]

III.7 Résultats des tests :

Ce test fait appel à la technique de test en boîte noire, mettant l'accent sur les besoins fonctionnels de ce système pour tenter d'identifier les défauts, qu'il s'agisse d'une anomalie, d'un problème de structure de données ou d'un problème de performance. [40]

No	Procédure	Résultats attendus	Validation
1	Liquide dans le réservoir contient moine de 10%	LED allumée et La pompe démarre automatiquement et s'affiche sur l'écran LCD, LED rouge allumé + Buzzer	Valide
2	Liquide dans un réservoir plein 100%	La pompe s'arrête automatiquement et s'affiche sur l'écran LCD, LED vert on + Buzzer	Valide
3	Liquide dans le réservoir contient 80%	La pompe est arrêtée	Valide
4	Liquide dans le réservoir contient 60%	La pompe est arrêtée	Valide
5	Liquide remplit progressivement le réservoir	Le capteur à ultrasons peut mesurer le niveau de liquide et l'afficher sur l'écran LCD	Valide
6	Teste en cas de fuite	La pompe s'arrête automatiquement et s'affiche sur l'écran LCD, LED bleu on + Buzzer	Valide

Tableau N 8 Résulta de teste

Figure (III.16) : le Teste de système

Figure (III.17) : Résultat : Liquide dans le réservoir contient 100%

Figure (III.18) : Résultat de Teste en cas de fuite

III.8 Développement d'une Application de Commande :

III.8.1 Utilisation du module Node MCU ESP8266 :

La carte Node MCU ESP8266 V3 Lolin est un outil de développement IoT polyvalent et performant, parfaitement adapté aux projets de suivi et de gestion en temps réel. Elle comprend un module Wi-Fi ESP8266, offrant une connectivité sans fil fiable, indispensable pour l'envoi de données à distance.

III.8.2 Caractéristique :

Cette version V3 Lolin se caractérise par une compatibilité renforcée et une conception améliorée. Le processeur 32 bits Tensilica L106, fonctionnant à une fréquence de 80 MHz, offre des performances adéquates pour le traitement et la manipulation des données. Dotée de 4 Mo de mémoire flash et 96 Ko de mémoire de données, elle propose une capacité suffisante pour stocker les programmes et les informations requises. La carte est compatible avec les environnements de développement Arduino IDE et Lua, ce qui facilite la programmation et l'implémentation des applications. En outre, elle possède de nombreuses broches GPIO pour la liaison avec divers capteurs et dispositifs, ce qui simplifie l'intégration avec les capteurs.

La carte, grâce à sa petite taille et son faible besoin en énergie, s'intègre aisément dans des systèmes de surveillance à consommation énergétique restreinte tels que les systèmes de

contrôle et commande de remplissage. Ces attributs positionnent le Node MCU ESP8266 V3 Lolin comme une option idéale pour les applications IoT qui exigent fiabilité, souplesse et performance.

Figure (III.19) : La carte Node MCU ESP8266 V3 Lolin.

III.8.3 La commande :

III.8.3.1 L'interface de l'application développé :

Figure (III.20) : L'interface de l'application développé

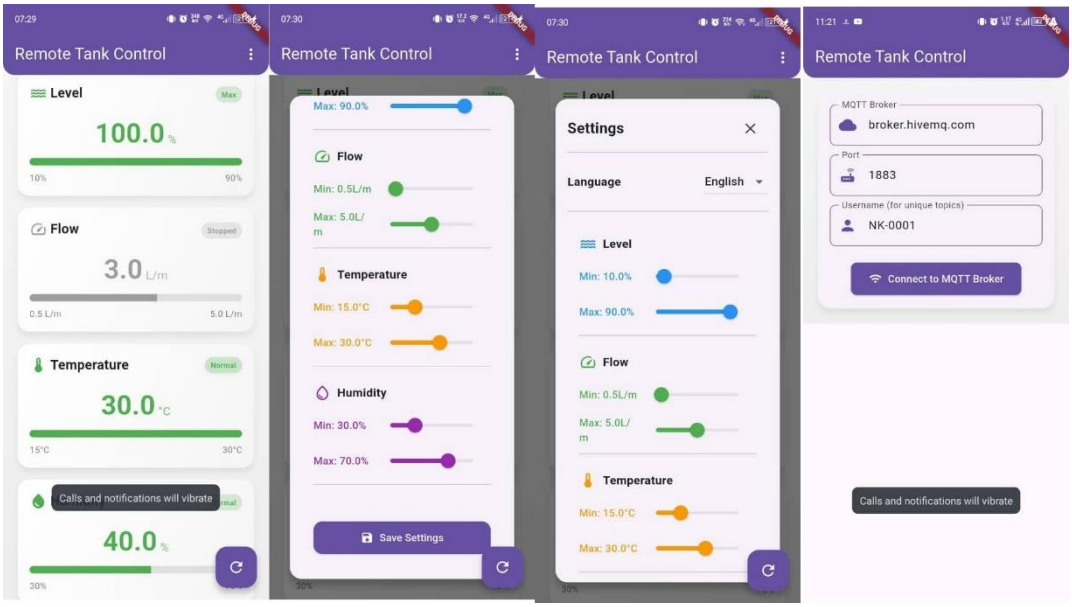


Figure (III.21) : L'interface de informations reçues

III.8.3.2 Le fonctionnement de L'App :

Au moment où l'utilisateur se connecte à l'application, il peut contrôler à distance les fonctions du système, telles que l'activation et l'arrêt de la pompe, ainsi que l'ouverture et la fermeture de la vanne de vidange en cas de besoin. L'utilisateur peut également choisir à tout moment entre le mode de contrôle automatique ou manuel du système. L'interface de l'application permet de consulter les données reçues du système et captées par les capteurs, telles que le niveau de liquide dans le réservoir, ainsi que la température et l'humidité dans le réservoir, en plus de l'état des équipements en fonctionnement ou à l'arrêt, et tout cela en temps réel et avec une mise à jour continue. [13]

III.9 Conclusion :

Grâce aux résultats de l'analyse, de la conception et de la mise en œuvre, plusieurs conclusions peuvent être tirées, notamment les suivantes :

1. Il est possible d'implémenter un dispositif de suivi et de gestion du niveau du réservoir, et l'information sur la hauteur du liquide pourrait être présentée par le biais d'un écran à cristaux liquides LCD. En outre, l'utilisateur peut ajuster le niveau de remplissage du liquide selon ses préférences.

2. On peut réaliser et mettre en place un dispositif avec un système de contrôle du niveau de liquide pour surveiller le processus de remplissage du réservoir. Cet appareil active et désactive la pompe de manière automatique, minimisant de ce fait le gaspillage d'énergie électrique et procurant un contrôle exact et instantané.

IV Conclusion générale

Dans cette étude, nous avons abordé la réalisation et la mise en œuvre d'un système de contrôle et de surveillance du remplissage des réservoirs (en utilisant l'eau comme liquide d'expérimentation dans cette étude). Ce projet a été réalisé comme un prototype initial, pouvant être développé ultérieurement en utilisant des dispositifs de mesure et de détection plus récents et plus performants, afin de s'adapter à tous les types de liquides.

Notre système repose sur un capteur mesurant le niveau du liquide dans le réservoir à l'aide de la technologie à ultrasons, connecté principalement à une carte Arduino servant de base au système. Le système peut surveiller le niveau du liquide et fournir des données à travers des alertes visuelles et sonores (voyants LED et signal sonore). Nous avons également développé une application qui permet de contrôler la mise en marche et l'arrêt du système à distance via le réseau, ce qui confère au système davantage d'efficacité et de performance.

Nous aspirons à développer ce système à l'avenir en intégrant l'intelligence artificielle, ce qui nous permettra de diagnostiquer et d'identifier les pannes, ainsi que de modifier les codes de fonctionnement selon les besoins, sans intervention humaine. Cela augmentera la rapidité de réaction, la fiabilité et l'efficacité de ce système, qui pourra être utilisé aussi bien dans le domaine industriel que domestique.

Nous envisageons également d'utiliser des capteurs variés, selon les besoins et les caractéristiques de chaque liquide, afin qu'ils soient plus précis et plus réactifs, pour répondre aux exigences de l'utilisateur et à la sensibilité ou la dangerosité de certains liquides pour l'environnement et les personnes en cas de fuite ou de perte.

V Références

[1] P. S. Beacon, "Process Safety Beacon," 17 04 2012. [Online]. Available: https://www.aiche.org/ccps/resources/process-safety-beacon/archives/2007/may/english. [Accessed 03 05 2025].

[2] J. d. professeur, "Bhopal: causes, conséquences et leçons dix ans après," *HYGIÈNE SÉCURITÉ,* p. 45, 02 12 1995.

[3] https://www.humanite.fr/-/-/avenue-de-la-mort, "AVENUE DE LA MORT - L'Humanité," https://www.humanite.fr, 1992.

[4] DGPR/SRT/SDRA/BARPI, "Synthèse relative à l'accidentologie des stockages de liquides inflammables," 2 9 2010. [Online]. Available: https://www.aria.developpement-durable.gouv.fr/wp-content/uploads/2013/08/sy_stockage_li_ddgc_vfinb_02092010.pdf.

[5] M. d. l. T. é. -. D. /. S. /. BARPI, "Rapport d'accident," Beyrouth - Liban, 2024.

[6] L. BARPI, "Incendie dans un dépôt pétrolier à la suite d'un impact de foudre," 05 08 2022. [Online]. Available: https://www.aria.developpement-durable.gouv.fr/accident/59453/. [Accessed 17 04 2025].

[7] F. D. Ahlam BENAMARA, "Etude et réalisation d'un système de commande acquisition et régulation PID a base d'un api /pic Memoire de Master," UNIVERSITE MOHAMED BOUDIAF - M'SILA, 2019/2020.

[8] ArduinoFactory, "ArduinoFactory – Guide de démarrage," 05 05 2025. [Online]. Available: https://arduinofactory.fr/.

[9] L. –. Arduino, "Logiciel – Arduino," [Online]. Available: https://arduino.blaisepascal.fr/logiciel/.

[10] Developpez.com, "Cours complet Arduino," 05 05 2025. [Online]. Available: https://arduino.developpez.com/tutoriels/cours-complet-arduino/.

[11] F. B. e. J. Bobroff, MICROCONTROLEUR ARDUINO, Paris Sud: Université Paris Sud, 2015.

[12] J.Hughes, Aduino : le guide complet, 75013 Paris – France: O'Reilly Media, janvier 2018.

[13] B. R. BOUHEBEL Samira, "Commande d'une pompe immergée par Arduino," Centre Universitaire AbdelhafidBoussouf -Mila, 2020/2021.

[14] E. Bartmann, LE GRAND LIVRE D'ARDUINO Deuxième édition, Paris: ÉDITION EYROLLES, 2018.

[15] G. A. KRAMA Abdelbasset, "Etude et réalisation d'une carte de contrôle par Arduino via le système Androïde," UNIVERSITE KASDI MERBAH OUARGLA, 2014/2015.

[16] B. Aboubakr, "COMMANDE VOCALE POUR ÉQUIPEMENTS DOMESTIQUES À BASE D'UNE CARTE ARDUINO," Université Aboubakr Belkaïd Tlemcen , Mémoire de Master, 2022 /2023.

[17] C. B. –. F. C. –. R. Radoux, Le langage C pour Arduino, 2017-2018.

[18] B. Mohamed, "Télésurveillance d'une serre agricole," UNIVERSITE BADJI MOKHTAR - ANNABA, Mémoire de Master, 2020/2021.

[19] D. M. Abdelmalek, "Capteurs et Instrumentation de mesure," UNIVERSITE M'HAMED BOUGARA-BOUMERDES, Polycopié du Cours, 2022-2023.

[20] Symes, "Les Capteurs : Types et Applications dans l'Industrie Moderne," 05 mai 2025. [Online]. Available: https://www.symes.fr/fr/blog/post/les-capteurs-definition-types-et-applications-dans-lindustrie-moderne.

[21] G. TOULMINET, "Généralités sur la chaîne d'acquisition des données et sur les capteurs," Cours 12003 | PDF, https://fr.scribd.com/document/54530805/cours12003, 2002-2003.

[22] P. 1.5, "DHT22- Capteur de temperature et humidite digital - Boutique Semageek," Boutique Semageek, 05 mai 2025. [Online]. Available: https://boutique.semageek.com/fr/416-dht22-capteur-de-temperature-et-humidite-digital-3008943768541.html.

[23] M. B. A. Mlle. BEN ACHOUR Roua, "Conception et réalisation d'une serre agricole intelligente à base d'API S7-1200," Université SAAD DAHLAB de BLIDA, Memoire de Master, 2023/2024.

[24] A. M. WALID, "Conception et Réalisation d'un Radar de Recul Automobile à base d'Arduino et Processing," Université Mouloud Mammeri De Tizi-Ouzou, MASTER PROFESSIONNEL, 2017/2018.

[25] robot-maker, "Capteur à Ultrasons HC-SR04," robot-maker, 05 mai 2025. [Online]. Available: http://www.robot-maker.com/shop/capteurs/13-telemetre-a-ultrasons-hc-sr04.html.

[26] GoTronic, "GoTronic | Robotique et composants électroniques," 07 05 2025. [Online]. Available: https://www.gotronic.fr/.

[27] B. -. Moussasoft, "Boutique - Moussasoft," Comment utiliser YF-S201 capteur de débit d'eau avec Arduino - Moussasoft, 05 05 2025. [Online]. Available: https://www.moussasoft.com/boutique/.

[28] "Critères de Choix D'un Capteur | PDF | Capteur | Électromagnétisme," Scribd, 07 05 2025. [Online]. Available: https://fr.scribd.com/document/365079212/Criteres-de-Choix-d-Un-Capteur.

[29] U. o. K. –. S. Khalifa Dai Elnur Mohamed, "Liquid level Indication and Control System using Arduino," *Petroleum and Chemical Industry International,* vol. 6, no. 2, 2023.

[30] "Power Lab - Power Lab," 06 05 2025. [Online]. Available: https://powerlab.dz/.

[31] F. F. BENNACER ABDENNOUR, "Etude et simulation d'un régulateur de contrôle de niveau d'eau," Centre Universitaire BOUSSOOUF Abdelhafid -Mila, 2021/2022.

[32] Techno-Science.net, "Pompe : définition et explications," 06 05 2025. [Online]. Available: https://www.techno-science.net/definition/5799.html.

[33] B. Mohamed, "Teleservaillance d'une serre agrigole," UNIVERSITE BADJI MOKHTAR - ANNABA, 2020/2021.

[34] OTRONIC, "Module buzzer passif pour Arduino," 06 05 2025. [Online]. Available: https://www.otronic.nl/fr/module-buzzer-passif-pour-arduino.html?source=facebook.

[35] OTRONIC, "Vanne à eau ou électrovanne à air 12V Plastique NC," 06 05 2025. [Online]. Available: https://www.otronic.nl/fr/vanne-a-eau-ou-electrovanne-a-air-12v-plastique-nc.html?source=facebook.

[36] N. Tanwar, "Arduino Based Automatic Water Tank Filling System," 05 04 2023. [Online].

[37] J. T. S. M. Y. I. 4. Refni Wahyuni1, "Water Level Control Monitoring Based On Arduino Uno R3 ATMega 238p Using Lm016l LCD at STMIK Hang Tuah Pekanbaru," *Journal of Robotics and Control (JRC),* vol. 2, no. 4, July 2021.

[38] Y. A. I. R. F. S. F. K. D. R. Ritzkal1*, "Water Tank Wudhu and Monitoring System Design using Arduino and Telegram," *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 14, no. 1, 2023.

[39] O. O. B. J.-O. Lambe Mutalub ADESINA1, "Development of Water Level Controller with SMS Capability," *ABUAD Journal of Engineering Research and Development (AJERD),* vol. 7, no. 1, pp. 195-207, 2024.

[40] A. M. 2. N. S. A. 3. Bagas Putra Anggara 1, "Design of a Liquid Tank Filling Control System Using PID," vol. 5, no. 3, pp. pp. 367-375, 2023, .

[41] V. Mittal, "Automated Water Level Controlling and Detection Using Arduino and GSM Sim Module," *International Journal of Science and Research (IJSR),* vol. 7, no. 6, p. 79.57 , June 2018.

[42] *. M. T. 2. a. S. P. 1. Livinti Petru 1, "Water Level Control and Monitoring System in a Tank made with Arduino Uno and NodeMCU ESP8266 development boards," p. Preprints (www.preprints.org), 23 04 2023.

[43] S. E. P. Marius V Tivlea, "water level control and monitoring system in a tank made withe arduino uno and nodeMCU ESP8266 development boards," 23 04 2023.

[44] B. M.-C. A. D. Christian Baldeon-Perez1, "Water Level Monitoring and Control System in Elevated Tanks to Prevent Water Leaks," *(IJACSA) International Journal of Advanced Computer Science and Applications,,* vol. 12, no. 2, 2021.

[45] R. T. Tafitasoa, "DOMOTIQUE COMMANDEE PAR ANDROID," UNIVERSITE D'ANTANANARIVO, 2018/2019.

[46] lefigarofr, "Liban: double explosion dans le port de Beyrouth, un drame national," pp. https://www.lefigaro.fr/international/dossier/liban-explosions-4-aout-nitrate-d-ammonium-port-beyrouth, 2020.

[47] Debord, "EUROPETROLE," revue de presse sur les risques industriels en Algérie, 10 02 2024. [Online]. Available: https://www.euro-petrole.com/accident.

[48] "Yémen: 85 morts et des centaines de blessés dans une bousculade à Sanaa," 17 05 2025. [Online]. Available: https://www.bfmtv.com/international/moyen-orient/yemen-85-morts-et-des-centaines-de-blesses-dans-une-bousculade-a-sanaa_AD-202304200319.html.

VI ANNEXES

Code Arduino utilisé :

```
* Tank Control System - Arduino Component
 *
 * Features:
 * - Water level monitoring with ultrasonic sensor
 * - Flow rate monitoring
 * - Temperature and humidity monitoring
 * - Pump and valve control (manual and automatic)
 * - LED status indicators
 * - LCD display
 * - Serial communication with ESP8266 for remote monitoring and control
 * - Leak detection with alerts
 */

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <SoftwareSerial.h>

// Define software serial pins for ESP8266 communication
#define ESP_RX 0  // Connect to ESP8266 TX
#define ESP_TX 1  // Connect to ESP8266 RX

SoftwareSerial espSerial(ESP_RX, ESP_TX);

// Pin definitions
#define PUMP_PIN 7
#define VALVE_PIN 6
#define TRIG_PIN 3
#define ECHO_PIN 4
#define FLOW_PIN 5
#define DHT_PIN 2
#define BUZZER_PIN 8

// LEDs
#define LED_BLUE 9      // Leak indicator
#define LED_RED 10      // Low level warning
#define LED_ORANGE 11   // Mid level indicator
#define LED_GREEN1 12   // High level indicator
#define LED_GREEN2 13   // Full level indicator

// Sensor configuration
#define DHT_TYPE DHT22
#define FLOW_CALIBRATION 450.0  // Pulses per liter

// Tank parameters
const float MIN_LEVEL = 10.0;   // Empty level threshold (%)
const float LOW_LEVEL = 15.0;   // Low level threshold (%)
const float MID_LEVEL = 50.0;   // Mid level threshold (%)
const float HIGH_LEVEL = 75.0;  // High level threshold (%)
const float FULL_LEVEL = 98.0;  // Full level threshold (%)
const float MAX_LEVEL = 100.0;  // Maximum level (%)
const float MAX_DISTANCE = 32.0; // Maximum distance reading (cm)
const float MIN_DISTANCE = 7.0;  // Minimum distance reading (cm)
const float LEAK_THRESHOLD = 3.0; // Level change to trigger leak detection (%)

// Initialize objects
DHT dht(DHT_PIN, DHT_TYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

// State variables
bool pumpRunning = false;
bool valveOpen = false;
bool autoMode = true;
bool leakDetected = false;
```

```
bool buzzerNotified = false;

// LED status variables
bool redLedBlinking = false;
bool blueLedBlinking = false;
bool blueLedState = false;

// Sensor readings
float level = 0.0;
float flow = 0.0;
float temperature = 0.0;
float humidity = 0.0;
float previousLevel = 0.0;

// Flow meter variables
volatile int pulseCount = 0;

// Timing variables
unsigned long lastSensorRead = 0;
unsigned long lastFlowCalc = 0;
unsigned long lastBlinkTime = 0;
unsigned long lastBlueBlink = 0;
unsigned long lastLevelCheckTime = 0;
unsigned long lastDataSend = 0;
const unsigned long blinkInterval = 500;
const unsigned long blueBlinkInterval = 500;
const unsigned long dataSendInterval = 2000; // Send data to ESP every 2 seconds

void setup() {
  Serial.begin(9600);
  espSerial.begin(115200);  // ESP8266 communication

  // Initialize pins
  pinMode(PUMP_PIN, OUTPUT);
  pinMode(VALVE_PIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(FLOW_PIN, INPUT_PULLUP);
  pinMode(BUZZER_PIN, OUTPUT);

  pinMode(LED_BLUE, OUTPUT);
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_ORANGE, OUTPUT);
  pinMode(LED_GREEN1, OUTPUT);
  pinMode(LED_GREEN2, OUTPUT);

  // Initialize outputs to default state
  digitalWrite(PUMP_PIN, HIGH);       // HIGH = OFF for most relay modules
  digitalWrite(VALVE_PIN, HIGH);      // HIGH = OFF for most relay modules
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(LED_BLUE, LOW);
  digitalWrite(LED_RED, LOW);
  digitalWrite(LED_ORANGE, LOW);
  digitalWrite(LED_GREEN1, LOW);
  digitalWrite(LED_GREEN2, LOW);

  // Initialize flow sensor
  attachInterrupt(digitalPinToInterrupt(FLOW_PIN), countPulse, FALLING);

  // Initialize display and sensors
  lcd.begin(16, 2);
  lcd.backlight();
  dht.begin();

  // Display startup message
  lcd.setCursor(0, 0);
  lcd.print("Tank Control System");
  lcd.setCursor(0, 1);
  lcd.print("Initializing...");
```

```
  delay(1500);
  lcd.clear();
}

void startPump() {
  digitalWrite(PUMP_PIN, LOW);  // LOW = ON for most relay modules
  pumpRunning = true;
  Serial.println("Pump started");
  sendStatusToESP();
}

void stopPump() {
  digitalWrite(PUMP_PIN, HIGH);  // HIGH = OFF for most relay modules
  pumpRunning = false;
  Serial.println("Pump stopped");
  sendStatusToESP();
}

void openValve() {
  digitalWrite(VALVE_PIN, LOW);  // LOW = ON for most relay modules
  valveOpen = true;
  Serial.println("Valve opened");
  sendStatusToESP();
}

void closeValve() {
  digitalWrite(VALVE_PIN, HIGH);  // HIGH = OFF for most relay modules
  valveOpen = false;
  Serial.println("Valve closed");
  sendStatusToESP();
}

void enableAutoMode(bool send = true) {
  if (!autoMode) {
    autoMode = true;
    Serial.println("Auto mode enabled");
    if (send) {
      sendStatusToESP();
    }
  }
}

void disableAutoMode(bool send = true) {
  if (autoMode) {
    autoMode = false;
    Serial.println("Auto mode disabled");
    if (send) {
      sendStatusToESP();
    }
  }
}

void readSensors() {
  // Read ultrasonic sensor for water level
  level = calculateLevel();

  // Read temperature and humidity
  float newHumidity = dht.readHumidity();
  float newTemperature = dht.readTemperature();

  // Only update if readings are valid
  if (!isnan(newHumidity) && !isnan(newTemperature)) {
    humidity = newHumidity;
    temperature = newTemperature;
  } else {
    Serial.println("DHT sensor read failed!");
  }

  // Read flow
```

```
  // unsigned long currentTime = millis();
  // if (currentTime - lastFlowCalc >= 1000) {
    detachInterrupt(digitalPinToInterrupt(FLOW_PIN));
    flow = pulseCount / FLOW_CALIBRATION;
    pulseCount = 0;
    // lastFlowCalc = currentTime;
    attachInterrupt(digitalPinToInterrupt(FLOW_PIN), countPulse, FALLING);// RISING);
    //espSerial.print(",QQQQQQQQQQQQQQQQQQQQ:");
    //espSerial.println(flow);
  //}

  // Check for leaks
  checkForLeaks();

  // Update LED indicators
  updateLEDs();

  // Update display
  updateDisplay();

  // Send data to serial for debugging
  sendSerialData();
}

float readUltrasonic() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Read the echo pulse
  long duration = pulseIn(ECHO_PIN, HIGH, 30000);

  // Calculate distance in cm
  float distance = duration * 0.034 / 2;

  // Constrain to valid range
  distance = constrain(distance, MIN_DISTANCE, MAX_DISTANCE);
  return distance;
}

float calculateLevel() {
  float distance = readUltrasonic();

  // Convert distance to level percentage
  float level = ((distance - MIN_DISTANCE) / (MAX_DISTANCE - MIN_DISTANCE));
  level = (1.0 - level) * 100;
  level = constrain(level, 0, 100);
  return level;
}

void checkForLeaks() {
  unsigned long currentTime = millis();

  // Check for leaks every 10 second
  if (currentTime - lastLevelCheckTime >= 5000) {
    float levelChange = previousLevel - level;
    previousLevel = level;
    lastLevelCheckTime = currentTime;

    // Detect leak: level decreasing without valve open and no flow
    if (!valveOpen && flow == 0.0 && levelChange > LEAK_THRESHOLD) {
      leakDetected = true;
      blueLedBlinking = true;
      digitalWrite(LED_BLUE, HIGH);
      sendStatusToESP();
    } else if (levelChange <= 0.0) {
      // Reset leak detection if level is stable or increasing
```

```
      if (leakDetected) {
        leakDetected = false;
        blueLedBlinking = false;
        digitalWrite(LED_BLUE, LOW);
        sendStatusToESP();
      }
    }
  }
}

void updateLEDs() {
  // Low level warning (blinking red)
  if (level <= MIN_LEVEL) {
    redLedBlinking = true;
    digitalWrite(LED_RED, HIGH);
  } else if (level >= LOW_LEVEL) {
    redLedBlinking = false;
    digitalWrite(LED_RED, LOW);
  }

  // Level indicators
  digitalWrite(LED_ORANGE, (level >= MID_LEVEL-2 && level <= MID_LEVEL+2));
  digitalWrite(LED_GREEN1, (level >= HIGH_LEVEL-2 && level <= HIGH_LEVEL+2));
  digitalWrite(LED_GREEN2, (level >= FULL_LEVEL && level <= MAX_LEVEL));

  // Handle blinking LEDs
  unsigned long currentTime = millis();

  // Red LED blinking (low level)
  if (redLedBlinking && currentTime - lastBlinkTime >= blinkInterval) {
    digitalWrite(LED_RED, !digitalRead(LED_RED));
    lastBlinkTime = currentTime;
  }

  // Blue LED blinking (leak detected)
  if (blueLedBlinking && currentTime - lastBlueBlink >= blueBlinkInterval) {
    tone(BUZZER_PIN, 1000, 200);
    delay(250);
    blueLedState = !blueLedState;
    digitalWrite(LED_BLUE, blueLedState);
    digitalWrite(BUZZER_PIN, blueLedState);
    lastBlueBlink = currentTime;
  } else if (!blueLedBlinking) {
    digitalWrite(BUZZER_PIN, LOW);
  }
}

void updateDisplay() {
  // Update LCD display with current readings
  lcd.clear();

  // First row: Temperature and Humidity
  lcd.setCursor(0, 0);
  lcd.print("T:");
  lcd.print(temperature, 0);
  lcd.print("C H:");
  lcd.print(humidity, 0);
  lcd.print("%");

  // Second row: Level and Flow
  lcd.setCursor(0, 1);
  lcd.print("Lv=");
  lcd.print(level, 0);
  lcd.print("% Q=");
  lcd.print(flow, 2);
  lcd.print("L/s");
}

void sendSerialData() {
```

```
  // Send data to serial port for debugging
  Serial.print("T:");
  Serial.print(temperature, 0);
  Serial.print(" H:");
  Serial.print(humidity, 0);
  Serial.print(" L:");
  Serial.print(level, 0);
  Serial.print(" Q:");
  Serial.println(flow, 2);
}

void sendStatusToESP() {
  // Send data to ESP8266 via SoftwareSerial
  espSerial.print("T:");
  espSerial.print(temperature, 1);
  espSerial.print(",H:");
  espSerial.print(humidity, 1);
  espSerial.print(",L:");
  espSerial.print(level, 1);
  espSerial.print(",Q:");
  espSerial.print(flow, 2);
  espSerial.print(",P:");
  espSerial.print(pumpRunning ? "1" : "0");
  espSerial.print(",V:");
  espSerial.print(valveOpen ? "1" : "0");
  espSerial.print(",K:");
  espSerial.print(leakDetected ? "1" : "0");
  espSerial.print(",A:");
  espSerial.println(autoMode ? "1" : "0");
}

void checkAutoControl() {
  if (autoMode) {
    // Auto control logic for pump based on level
    if (!pumpRunning && level <= MIN_LEVEL) {
      startPump();
    }
    if (pumpRunning && level >= MAX_LEVEL) {
      stopPump();
    }
  }

  if (!autoMode && (level <= MIN_LEVEL || level >= MAX_LEVEL)) {
    enableAutoMode();
  }
}

void handleSerialCommands() {
  if (espSerial.available()) {
    String command = espSerial.readStringUntil('\n');
    command.trim();

    if (command == "P1") {  // Start pump
      disableAutoMode(false);
      startPump();
    } else if (command == "P0") {  // Stop pump
      enableAutoMode(false);
      stopPump();
    } else if (command == "V1") {  // Open valve
      openValve();
    } else if (command == "V0") {  // Close valve
      closeValve();
    } else if (command == "A1") {  // Auto mode on
      enableAutoMode();
    } else if (command == "A0") {  // Auto mode off
      disableAutoMode();
    } else if (command == "S") {  // Status request
      sendStatusToESP();
    }
```

```
  }
}

void handleIntrSerialCommands() {
  if (Serial.available()) {
    String command = Serial.readStringUntil('\n');
    command.trim();

    if (command == "P1") {  // Start pump
      disableAutoMode(false);
      startPump();
    } else if (command == "P0") {  // Stop pump
      enableAutoMode(false);
      stopPump();
    } else if (command == "V1") {  // Open valve
      openValve();
    } else if (command == "V0") {  // Close valve
      closeValve();
    } else if (command == "A1") {  // Auto mode on
      enableAutoMode();
    } else if (command == "A0") {  // Auto mode off
      disableAutoMode();
    } else if (command == "S") {  // Status request
      sendStatusToESP();
    }
  }
}

void countPulse() {
  pulseCount++;
}

void checkTankAlerts() {
  // Full tank alert
  /*if (level >= MAX_LEVEL && !buzzerNotified) {
    tone(BUZZER_PIN, 1500, 100);
    delay(100);
    digitalWrite(BUZZER_PIN, HIGH);
    buzzerNotified = true;
  } else if (level < MAX_LEVEL) {
    buzzerNotified = false;
    digitalWrite(BUZZER_PIN, LOW);
  }

  // Empty tank alert
  if (level <= MIN_LEVEL) {
    tone(BUZZER_PIN, 1500, 100);
    delay(100);

    digitalWrite(BUZZER_PIN, HIGH);
    buzzerNotified = true;
  } else {
    digitalWrite(BUZZER_PIN, LOW);
    buzzerNotified = false;
  }*/
  if (level <= MIN_LEVEL)  {
    tone(BUZZER_PIN, 1500, 100);
    delay(100);
    digitalWrite(BUZZER_PIN, HIGH);
    buzzerNotified = true;
  } else if (level >= MAX_LEVEL) {
    if (!buzzerNotified) {
      tone(BUZZER_PIN, 1000, 200);
      delay(250);
      tone(BUZZER_PIN, 1000, 200);
      digitalWrite(BUZZER_PIN, HIGH);
      buzzerNotified = true;
    } else {
      digitalWrite(BUZZER_PIN, LOW);
```

```
    }
  } else {
    buzzerNotified = false;
    digitalWrite(BUZZER_PIN, LOW);
  }
}

void loop() {
  unsigned long currentTime = millis();

  // Handle serial commands from ESP8266
  handleSerialCommands();
  //handleIntrSerialCommands();

  // Read sensors every second
  if (currentTime - lastSensorRead >= 1000) {
    lastSensorRead = currentTime;
    readSensors();
    checkAutoControl();
    checkTankAlerts();
  }

  // Send data to ESP8266 periodically
  /*if (currentTime - lastDataSend >= dataSendInterval) {
    lastDataSend = currentTime;
    sendStatusToESP();
  }*/
}
```

Code NodeMCU v3 MQTT Bridge :

```
* NodeMCU v3 MQTT Bridge for Tank Control System
 *
 * This code runs on a NodeMCU v3 and:
 * 1. Connects to WiFi
 * 2. Establishes MQTT connection
 * 3. Receives sensor data from Arduino via Serial
 * 4. Publishes data to MQTT broker
 * 5. Subscribes to command topics and forwards commands to Arduino
 */

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// WiFi credentials - Replace with your actual credentials
const char* ssid = "ZTE";
const char* password = "Rhe4NHmF";

// MQTT Broker settings
const char* mqtt_server = "broker.hivemq.com";
const int   mqtt_port = 1883;
const char* client_id = "nodemcu_tank_controller";
const char* username = "NK-0001";  // Device identifier

// MQTT Topics
const char* level_topic = "NK-0001/tank/level";
const char* flow_topic = "NK-0001/tank/flow";
const char* humidity_topic = "NK-0001/tank/humidity";
const char* temperature_topic = "NK-0001/tank/temperature";
const char* pump_status_topic = "NK-0001/tank/pump_status";
const char* valve_status_topic = "NK-0001/tank/valve_status";
const char* leak_status_topic = "NK-0001/tank/leak_status";
const char* auto_mode_topic = "NK-0001/tank/auto_mode";
const char* pump_command_topic = "NK-0001/tank/pump/command";
const char* valve_command_topic = "NK-0001/tank/valve/command";
const char* auto_command_topic = "NK-0001/tank/auto/command";
const char* update_request_topic = "NK-0001/tank/request_update";
```

```
// Variables to store sensor data
float temperature = 0.0;
float humidity = 0.0;
float level = 0.0;
float flow = 0.0;
bool pumpRunning = false;
bool valveOpen = false;
bool leakDetected = false;
bool autoMode = true;

// Serial buffer
String serialData = "";

// MQTT client
WiFiClient espClient;
PubSubClient mqtt(espClient);

// Timing variables
unsigned long lastMqttPublish = 0;
const unsigned long mqttPublishInterval = 10000; // 10 seconds

// Status LED pin (built-in LED on NodeMCU)
const int LED_PIN = LED_BUILTIN; // D4 on NodeMCU (GPIO2)

void setup() {
  // Initialize status LED
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH); // LED off initially (NodeMCU LED is active LOW)

  // Initialize serial communication with Arduino
  Serial.begin(115200);

  // Connect to WiFi
  setupWifi();

  // Setup MQTT client
  mqtt.setServer(mqtt_server, mqtt_port);
  mqtt.setCallback(mqttCallback);

  // Set buffer size for PubSubClient (NodeMCU has more memory)
  mqtt.setBufferSize(512);
}

void setupWifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to WiFi: ");
  Serial.println(ssid);

  // Disconnect if previously connected
  WiFi.disconnect();
  delay(100);

  // Set WiFi mode
  WiFi.mode(WIFI_STA);

  // Begin connection
  WiFi.begin(ssid, password);

  // Wait for connection with timeout
  int attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 20) {
    digitalWrite(LED_PIN, LOW);  // LED on during connection
    delay(250);
```

```
    digitalWrite(LED_PIN, HIGH); // LED off
    delay(250);
    Serial.print(".");
    attempts++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    digitalWrite(LED_PIN, LOW); // LED on when connected
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println("");
    Serial.println("WiFi connection failed. Running in offline mode.");
  }
}

void reconnectMqtt() {
  // Attempt to reconnect to MQTT broker
  if (WiFi.status() != WL_CONNECTED) {
    return; // Skip if WiFi is not connected
  }

  int attempts = 0;
  while (!mqtt.connected() && attempts < 3) {
    Serial.print("Connecting to MQTT broker...");
    if (mqtt.connect(client_id, username, NULL)) {
      Serial.println("connected");

      // Subscribe to command topics
      mqtt.subscribe(pump_command_topic);
      mqtt.subscribe(valve_command_topic);
      mqtt.subscribe(auto_command_topic);
      mqtt.subscribe(update_request_topic);

      // Publish initial status
      publishSensorData();
    } else {
      Serial.print("failed, rc=");
      Serial.print(mqtt.state());
      Serial.println(" retrying...");

      // Flash LED to indicate connection failure
      for (int i = 0; i < 3; i++) {
        digitalWrite(LED_PIN, LOW);
        delay(100);
        digitalWrite(LED_PIN, HIGH);
        delay(100);
      }

      delay(2000);
      attempts++;
    }
  }
}

void mqttCallback(char* topic, byte* payload, unsigned int length) {
  // Convert payload to string
  String message;
  for (int i = 0; i < length; i++) {
    message += (char)payload[i];
  }

  Serial.print("MQTT message [");
```

```
  Serial.print(topic);
  Serial.print("]: ");
  Serial.println(message);

  // Flash LED to indicate message received
  digitalWrite(LED_PIN, HIGH);
  delay(50);
  digitalWrite(LED_PIN, LOW);
  delay(50);
  digitalWrite(LED_PIN, HIGH);
  delay(50);
  digitalWrite(LED_PIN, LOW);

  // Handle pump commands
  if (String(topic) == pump_command_topic) {
    if (message == "ON") {
      Serial.println("P1"); // Send command to Arduino
    } else if (message == "OFF") {
      Serial.println("P0"); // Send command to Arduino
    } else if (message == "AUTO") {
      Serial.println("A1"); // Send command to Arduino
    }
  }

  // Handle valve commands
  if (String(topic) == valve_command_topic) {
    if (message == "OPEN") {
      Serial.println("V1"); // Send command to Arduino
    } else if (message == "CLOSE") {
      Serial.println("V0"); // Send command to Arduino
    }
  }

  // Handle auto mode commands
  if (String(topic) == auto_command_topic) {
    if (message == "ON") {
      Serial.println("A1"); // Send command to Arduino
    } else if (message == "OFF") {
      Serial.println("A0"); // Send command to Arduino
    }
  }

  // Handle update requests
  if (String(topic) == update_request_topic) {
    Serial.println("S"); // Request status from Arduino
  }
}

void readFromArduino() {
  while (Serial.available()) {
    char c = Serial.read();
    if (c == '\n') {
      parseData(serialData);
      serialData = "";
    } else {
      serialData += c;
    }
  }
}

void parseData(String data) {
  // Expected format: T:23.5,H:45.2,L:78.3,Q:0.25,P:1,V:0,K:0,A:1

  int tIndex = data.indexOf("T:");
  int hIndex = data.indexOf(",H:");
```

```
  int lIndex = data.indexOf(",L:");
  int qIndex = data.indexOf(",Q:");
  int pIndex = data.indexOf(",P:");
  int vIndex = data.indexOf(",V:");
  int kIndex = data.indexOf(",K:");
  int aIndex = data.indexOf(",A:");

  if (tIndex != -1 && hIndex != -1 && lIndex != -1 && qIndex != -1 &&
      pIndex != -1 && vIndex != -1 && kIndex != -1 && aIndex != -1) {

    temperature = data.substring(tIndex + 2, hIndex).toFloat();
    humidity = data.substring(hIndex + 3, lIndex).toFloat();
    level = data.substring(lIndex + 3, qIndex).toFloat();
    flow = data.substring(qIndex + 3, pIndex).toFloat();
    pumpRunning = (data.substring(pIndex + 3, vIndex) == "1");
    valveOpen = (data.substring(vIndex + 3, kIndex) == "1");
    leakDetected = (data.substring(kIndex + 3, aIndex) == "1");
    autoMode = (data.substring(aIndex + 3) == "1");

    // If connected to MQTT, publish the data
    if (mqtt.connected()) {
      publishSensorData();
    }
  }
}

void publishSensorData() {
  if (!mqtt.connected()) {
    return;
  }

  char buffer[10];

  // Publish level
  dtostrf(level, 4, 1, buffer);
  mqtt.publish(level_topic, buffer);

  // Publish flow
  dtostrf(flow, 4, 2, buffer);
  mqtt.publish(flow_topic, buffer);

  // Publish temperature
  dtostrf(temperature, 4, 1, buffer);
  mqtt.publish(temperature_topic, buffer);

  // Publish humidity
  dtostrf(humidity, 4, 1, buffer);
  mqtt.publish(humidity_topic, buffer);

  // Publish status
  mqtt.publish(pump_status_topic, pumpRunning ? "ON" : "OFF");
  mqtt.publish(valve_status_topic, valveOpen ? "OPENED" : "CLOSED");
  mqtt.publish(leak_status_topic, leakDetected ? "DETECTED" : "NORMAL");
  mqtt.publish(auto_mode_topic, autoMode ? "ON" : "OFF");

  // Blink LED to indicate data published
  digitalWrite(LED_PIN, HIGH);
  delay(50);
  digitalWrite(LED_PIN, LOW);
}

void loop() {
  unsigned long currentTime = millis();

  // Handle WiFi reconnection if needed
```

```
  if (WiFi.status() != WL_CONNECTED) {
    digitalWrite(LED_PIN, HIGH); // LED off when disconnected
    setupWifi();
  }

  // Handle MQTT connection
  if (WiFi.status() == WL_CONNECTED && !mqtt.connected()) {
    reconnectMqtt();
  }

  if (mqtt.connected()) {
    mqtt.loop();
  }

  // Read data from Arduino
  readFromArduino();

  // Publish data periodically if connected
  if (currentTime - lastMqttPublish >= mqttPublishInterval && mqtt.connected()) {
    lastMqttPublish = currentTime;
    Serial.println("S"); // Request fresh status from Arduino
  }

  // Prevent watchdog timer issues
  yield();
}
```