الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Master

présenté par

ACHLAF Aymen

&

LAGREB Ali Hichem

Filière : Automatique

Spécialité : Automatique et Informatique Industrielle

Réalisation d'un thermomètre infrarouge automatisé à mémoire

Proposé par : Mr. Benselama Zoubir

Année Universitaire 2019-2020

Au terme de cette étude, nous tenons à présenter nos sincères remerciements au bon dieu de nous avoir accordé la connaissance de la science et de nous avoir aidé à réaliser ce modeste travail.

Un grand merci à notre encadreur MR. BENSLAMA ZOUBIR pour son soutien et sa disponibilité. Monsieur, le Président et les membres du jury, nous sommes très sensibles à l'honneur que vous nous faites en acceptant de juger ce travail. Nous adressons nos sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé nos réflexions et ont accepté de nous rencontrer et de répondre à nos questions durant nos recherches.

Enfin, Nous remercions nos familles et nos amis qui ont toujours été là pour nous. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide. À tous ces intervenants, nous présentons nos remerciements, notre respect et notre gratitude.

ملخص: العالم يمر بفترة وباء بسبب فيروس كوفيد 19 ، في هذه الأطروحة سوف نقدم التفاصيل المختلفة حول الفيروس ، وعموميات على بطاقات اردوينو ، وبشكل أكثر دقة بطاقة اردوينو UNO ، وسنقوم بتصنيع مستشعر درجة حرارة الأشعة تحت الحمراء مع امكانيات تخزين في شريحة ذاكرة.SD

كلمات المفاتيح: كوفيد 19 ; درجة حرارة الجسم ; اردوينو ; مستشعر حرارة ; وحدة الحفاظ على التوقيت RTC ; تخزين في شريحة .SD

Résumé : Le monde est en pleine pandémie à cause du virus COVID 19, dans ce mémoire nous allons explorer les différents détails sur le virus, des généralités sur les cartes Arduino, plus précisément la carte Arduino UNO, et nous allons réaliser un thermomètre de température infrarouge sans contact avec stockage SD.

Mots clés : COVID 19 ; Température corporelle ; Arduino ; Capteur de température infrarouge ; Thermomètre automatisé ; Module RTC ; Stockage SD.

Abstract : The world is in a global pandemic because of the COVID 19 virus, in this thesis we will explore the different details about the virus, generalities about Arduino cards, more precisely the Arduino UNO, and we will conceptualize a contactless infrared thermometer with SD logging capabalities.

Keywords : COVID 19 ; Body temperature ; Arduino ; Infrared temperature sensor ; Automated thermometer ; RTC Module ; SD Logging.

Listes des acronymes et abréviations

FDA	Food and Drug Administration
PCR	polymerase chain reaction
PID	controlleur proportional–integral–derivative
IDE	integrated development environment
USB	Universal Serial Bus
EEPROM	Electrically-erasable programmable read-only memory
PWM	Pulse Width Modulation
RAM	Random Access Memory
E/S	Entrée/Sortie
RX et TX	Receiving and Transmitting pins
TTL	Transistor-Transistor Logic
SPI	Interface Série Périphérique
LED	Light Emitting Diode
I2C	Inter Integrated Circuit
SMBus	System Management Bus
ADC	analog to digital converter
DSP	Digital Signal Processor
SCL	Serial Clock Line
SDA	Serial Data Line
RTC	Real Time Clock
ppm	*partie par million*
TWI	Two-Wire *Interface*
PC	Pesonal Computer
MISO	Master Input, Slave Output
MOSI	Master Output, Slave Input

LCD	Liquid Crystal Display
CNC	computer numerical control

Sommaire

Chapitre 2 : ARDUINO

Chapitre 3 : Réalisation

Liste des figures

Liste des tableaux

Introduction générale

Pour minimiser les risques de pandémie, nombreux pays ont eu recours au dépistage des symptômes comme première ligne de défense. Il existe actuellement plusieurs moyens de dépistage, on y trouve le scanner infrarouge, PCR, tests d'antigènes ... etc. Ces tests présentent une grande valeur dans la défense contre la propagation de la COVID-19 [1] Cependant, chacun de ces tests présente quelques inconvénients, certains tests ont un coût élevé, ils nécessitent un équipement et une longue durée pour les prises de décisions. Pour cela, la mesure de la température corporelle reste un élément essentiel vu leur efficacité et la rapidité et de prendre des décisions.

L'objectif de notre étude est de réaliser un thermomètre infrarouge capable de détecter la température automatiquement et puis enregistrer les données dans une carte SD à l'aide d'une carte de prototypage Arduino. Notre mémoire sera structuré comme suit

Le premier chapitre s'étalera sur des généralités sur la COVID-19 et les thermomètres.

Le deuxième chapitre traitera la carte de prototypage Arduino.

Le dernier chapitre sera consacré à la mise en œuvre de notre système ainsi que les résultats obtenus.

Chapitre 1 Généralités

1.1 Introduction

Le monde est en pleine pandémie à cause du virus COVID 19, dans ce chapitre nous allons explorer les différents symptômes du virus, les tests utilisés actuellement et le rôle important des thermomètres dans la détection du virus.

1.2 L'historique du COVID-19

A partir du 20 février 2020, le nouveau coronavirus de 2019 (désormais appelée SARS-CoV-2, qui cause la maladie COVID-19) a provoqué plus de 80 000 cas confirmés en Chine et s'est étendue à 213 autres pays [2]. Jusqu'à présent, la transmission locale est restée limitée en dehors de la Chine, mais à partir de cette période, de nouveaux foyers d'épidémie sont apparus sur plusieurs continents [3].

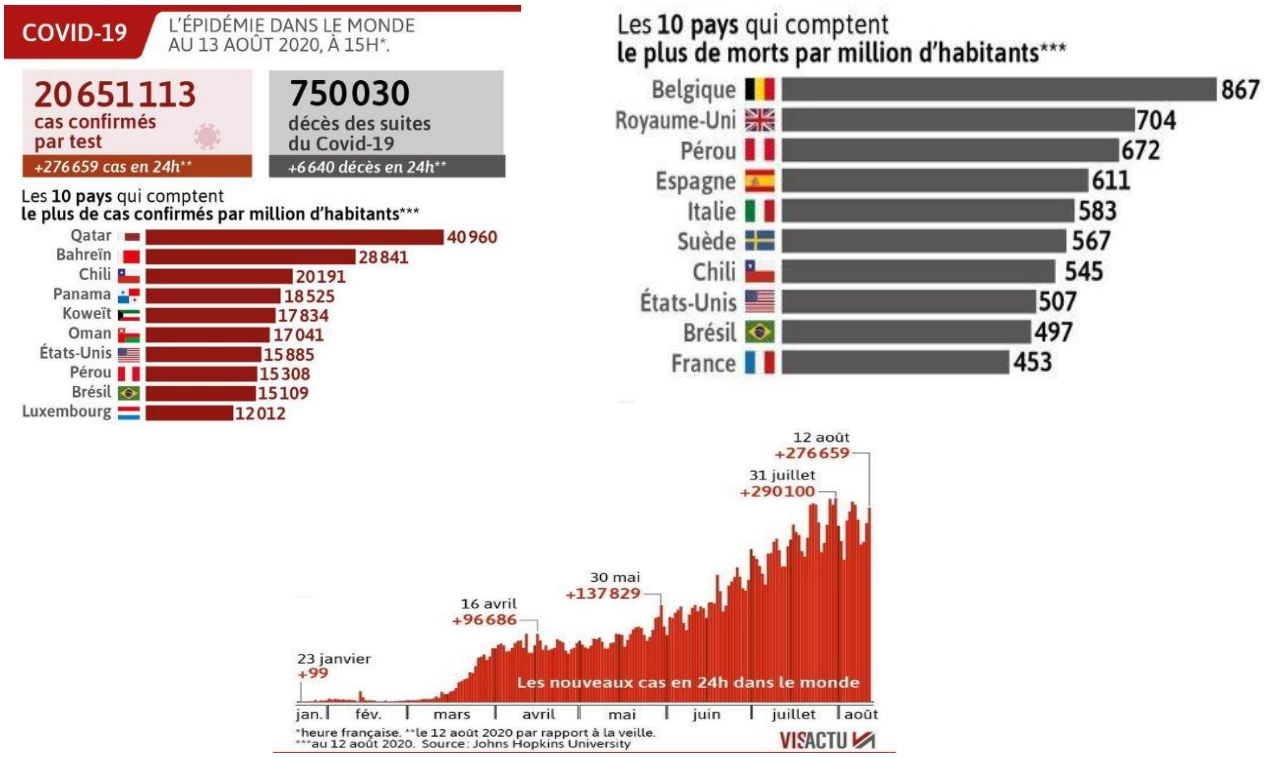

Figure 1.1 : cas de COVID19 dans le monde.

1.3 Les symptômes de la COVID-19

La COVID-19 affecte les individus de différentes manières. La plupart des personnes infectées développent une forme légère à modérée de la maladie et guérissent sans hospitalisation.

Symptômes les plus fréquents:

- Fièvre
- Toux sèche
- Fatigue

Symptômes moins fréquents:

- Courbatures
- Maux de gorge
- Diarrhée
- Conjonctivite
- Maux de tête
- Perte de l'odorat ou du goût
- Éruption cutanée, ou décoloration des doigts ou des orteils

Symptômes graves:

- Difficultés à respirer ou essoufflement
- Sensation d'oppression ou douleur au niveau de la poitrine
- Perte d'élocution ou de motricité

En moyenne, l'apparition des symptômes chez une personne infectée par le virus prend cinq à six jours. Cependant, ce délai peut s'étendre jusqu'à quatorze jours. [4]

1.4 Les types des tests pour dépister COVID-19

La FDA américaine a jusqu'à présent accordé une autorisation d'utilisation en urgence à plus de 200 tests différents destinés à détecter une infection actuelle ou passée due au SRAS-CoV-2, On cite quelques exemples :

1.4.1 Tests PCR

La majorité des tests COVID-19 sont des tests PCR. Ces tests détectent la maladie en recherchant des traces du matériel génétique du virus sur un échantillon le plus souvent prélevé par écouvillonnage du nez ou de la gorge.

La réalisation d'un test PCR et la lecture de ses résultats nécessitent un équipement et des produits chimiques spécifiques qui sont en nombre insuffisant, C'est en partie pour cette raison que les États-Unis ont accumulé un tel retard dans les tests, qui peut atteindre jusqu'à plusieurs semaines. [5]

1.4.2 Tests de salive

Les tests de salive de coronavirus sont un nouveau type de diagnostic par PCR pour la COVID-19. Les tests de salive dépendent de la technologie PCR standard et nécessitent un certain travail manuel pour les étapes du test.

Des études ont suggéré que jusqu'à 30 % des résultats du test de salive sont inexacts. [6]

1.4.3 Tests d'antigènes

Les tests d'antigènes peuvent donner des résultats en quelques minutes, mais la rapidité vient avec les compromis, ce test donne de nombreux résultats faussement positifs ou négatifs. [7]

1.4.4 Tests d'anticorps

Contrairement aux autres tests énumérés ici, les tests de détection des anticorps ne sont pas destinés à détecter une infection actuelle par le CoV-2 du SRAS. Ils recherchent plutôt les anticorps dans le sang, les faux résultats sont assez fréquents.

1.4.5 Test de température

Même si les tests de température ne détectent pas nécessairement la COVID-19, mais c'est un excellent outil pour détecter un cas potentiel pour des tests supplémentaires,

"Le dépistage d'entrée pour COVID-19 implique l'utilisation du balayage thermique et/ou le dépistage des symptômes", déclare Jeanine Pommier du Centre européen de prévention et de contrôle des maladies (ECDC). C'est pour cette raison des nombreux aéroports utilisent des tests thermiques pour détecter les passagers présentant des températures élevées. [8]

Les méthodes comprennent les scanners infrarouges corporels (qui mesurent la température de la peau en tant qu'indicateur de la température corporelle), les thermomètres infrarouges portatifs et les thermomètres à pistolet auriculaire. Ces deux derniers instruments ont été utilisés dans les aéroports d'Afrique de l'Ouest lors de la crise du virus Ebola de 2014.

1.5 Pourquoi la détection thermique

L'article "Le nouveau corona virus met le monde entier en alerte" a souligné la nécessité d'un programme à grande échelle pour dépister ou détecter les personnes susceptibles d'être infectées par le nouveau coronavirus COVID-19 [9], Ce qui nécessite l'utilisation d'un thermomètre infrarouge pour dépister la fièvre dans les hôpitaux, les cliniques de soins primaires et les bâtiments commerciaux.

Le cathétérisme des artères pulmonaires est la norme de référence pour mesurer la température corporelle centrale, mais il est invasif, nécessite des compétences et des équipements spécialisés et ne convient pas au dépistage de cohortes importantes. Au contraire, les substituts tels que les thermomètres rectaux et oraux sont moins invasifs et ont une corrélation modérée avec la température corporelle centrale, mais nécessitent un contact avec les fluides corporels et risquent donc d'être contaminés.

Le thermomètre infrarouge est couramment utilisé pour le dépistage de grandes cohortes car il est portable, ne nécessite pas de contact et ne cause pas de gêne à la personne évaluée.

1.6 Historique du thermomètre

Le premier instrument apportant une indication quantitative de la température de l'air fut réalisé **en 1608** par le médecin italien Sanctorius. Utilisé alors pour repérer la

température de l'air, ce thermomètre à air donna satisfaction jusqu'à la découverte de l'un de ses défauts majeurs : sa forte sensibilité à la pression atmosphérique. Le thermomètre scellé sortit alors des ateliers des savants florentins. Pendant que le thermoscope était en usage, durant quatre décennies, des essais de repérage du niveau de chaleur avec un liquide restèrent sans lendemain immédiat. Ce sont la description par Bartolomeo Telioux **en 1611** d'un appareil, dont la reproduction montre qu'il peut fonctionner comme un thermomètre, les essais de Sagredo **en 1615,** de Jean Rey **en 1631** et de Castelli quelques années plus tard. Cette activité a préparé le changement de l'instrument qui mesure la température de l'air. Elle a facilité l'apparition rapide du thermomètre. [10]

Vers 1700, Isaac Newton se consacra au problème de la chaleur. Il élabora une première échelle de température qualitative, consistant en une vingtaine de points de référence allant de « l'air froid en hiver » jusqu'aux « charbons ardents du feu de cuisine ». Cette façon de faire étant grossière et problématique, Newton en devint vite insatisfait. Au bout d'un certain temps, il définit « zéro degré de chaleur » comme correspondant à la neige fondante et « 33 degrés de chaleur » comme correspondant à l'eau bouillante.

En 1702, l'astronome Ole Christensen Rømer fabriqua, au Danemark, un thermomètre à alcool marquant l'eau bouillante à 60 degrés et la glace pilée à 7,5 degrés. En 1717, le savant allemand Gabriel Fahrenheit remplaça l'alcool par du mercure et donna au thermomètre sa forme définitive. Il proposa également la première échelle de températures à être adoptée assez largement, fixant à 32 °F la température de la glace fondante et à 96 °F la température normale du sang : 32 °F est alors le point de fusion de la glace et 212 °F est le point d'ébullition de l'eau sous pression atmosphérique normale.

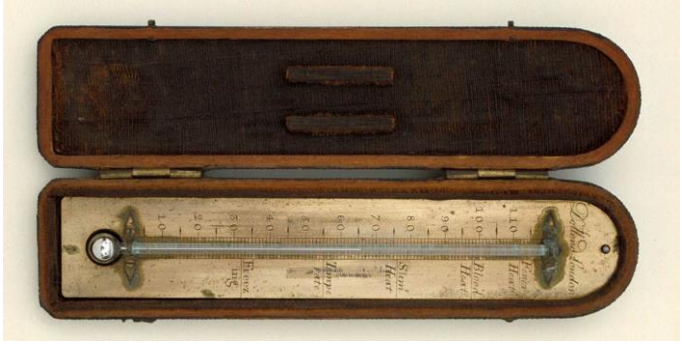

Figure 1.2 : Thermomètre de Fahrenheit de la fin du XVIIIe siècle. Musée Galilée, Florence.

En 1730, René-Antoine Ferchault de Réaumur, physicien et naturaliste français, construisit un thermomètre à « esprit de vin » (ancienne dénomination de l'éthanol), pour lequel il utilisa l'échelle 0-80, le zéro étant le point de congélation de l'eau, et le 80 est le point d'ébullition de l'alcool, que Réaumur tendait à confondre avec le point d'ébullition de l'eau.

En 1741 le physicien suédois Anders Celsius fit construire un thermomètre à mercure, gradué de sorte que 100° correspondît au point de congélation de l'eau, et 0° à son point d'ébullition, qui fut utilisé de 1742 à 1750 à l'observatoire d'Uppsala. L'échelle de Celsius était donc graduée en sens inverse de l'échelle centigrade que nous connaissons actuellement.

Cependant, à la même époque, le secrétaire perpétuel de l'académie des Beaux-Arts de Lyon, Jean-Pierre Christin, faisait construire par l'artisan lyonnais Pierre Casati un thermomètre à mercure à échelle centésimale ascendante, qu'il présenta le 19 mars 1743 à l'assemblée publique de cette académie. On attribua donc souvent à tort l'inversion de l'échelle mise au point par Celsius à Christin. [11]

Figure 1.3 : Thermomètre de Celsius.

1.7 Usages des thermomètres

1.7.1 Médical

Les thermomètres médicaux sont utilisés pour la mesure de la température corporelle. Pendant longtemps, Du fait de la faible corrélation entre la fièvre et les symptômes, les praticiens ont recouru longtemps à la prise de pouls pour apprécier la fièvre. En 1835

d'abord, Antoine Becquerel et Gilbert Breschet démontrèrent, à l'aide d'un thermocouple de fer et de cuivre, que la température d'un corps humain sain est constante à 37 °C. Cette découverte suscita l'intérêt pour l'utilisation médicale du thermomètre.

Les thermomètres à mercure, longtemps en usage, ont été progressivement retirés de la vente à cause de la toxicité de ce métal. Le thermomètre médical numérique a remplacé le thermomètre à mercure. Il contient des oxydes métalliques à résistance variable en fonction de la température (thermistance). Ce principe permet une mesure précise sur une gamme de température étroite, bien adaptée à l'usage médical.

Figure 1.4 : thermomètre numérique médical.

1.7.2 Alimentaire

Des thermomètres de cuisson permettent de vérifier et surveiller la température des aliments en cours de chauffe ou de cuisson. On les trouve notamment en pâtisserie, particulièrement pour le chocolat dont le contrôle de température est particulièrement important et précis.

Il existe aussi des diffèrent types de thermomètre alimentaire, on cite quelques exemples :

- **Thermomètre de confiseur** sert à mesurer de manière précise la température des sirops de sucre. Il est protégé par une cage métallique et gradué de 100 à 200 °C.
- **Thermomètre pour l'agroalimentaire** disposent de sondes comportant des capteurs à variation de résistance (CTN) ou de type PT100. Les étendues de mesure de ce type de matériel (-200 et +600 °C pour le type Pt100 et -50 à +150 °C pour le type CTN) permettent de couvrir l'ensemble des applications traditionnelles en agroalimentaire.

Figure 1.5 : Thermomètre agroalimentaire de type CTN.

1.7.3 Professionnel

Les thermomètres pour l'usage professionnel sont de très grande précision. Ils ont une très grande étendue de mesure et une grande rapidité d'acquisition. Ils peuvent afficher deux températures avec un calcul de ΔT. En fonction des besoins, différents capteurs sont utilisés. Le choix de la sonde adéquate dépend de différents critères.

1.8 Les types du thermomètre

1.8.1 Thermomètre à gaz

Le thermomètre à gaz est basé sur les variations de pression ou de volume d'un gaz en fonction de la température. Ce type de thermomètre utilise la loi d'Avogadro :

$$Pv = nRT$$

La première variante utilise un réservoir rempli de gaz et un tube ouvert dans lequel se trouve un bouchon mobile séparant le gaz du réservoir de l'air ambiant. La seconde variante de ce thermomètre garde le volume constant. Un réservoir contenant un gaz est connecté par un tube capillaire à un manomètre.

1.8.2 Thermomètre à cadran et aiguille

Le thermomètre bilame est constitué de deux lames de métaux ou d'alliages différents, souples, soudées ou collées l'une contre l'autre, dans le sens de la longueur. Ces deux plaques de métal soudées par laminage à froid, sont très souvent de l'invar et du nickel ayant un coefficient de dilatation différent. Leur dilatation étant différente, l'objet se déforme avec les variations de température. Cette déformation est lue sur un cadran via un mécanisme de micromètre.

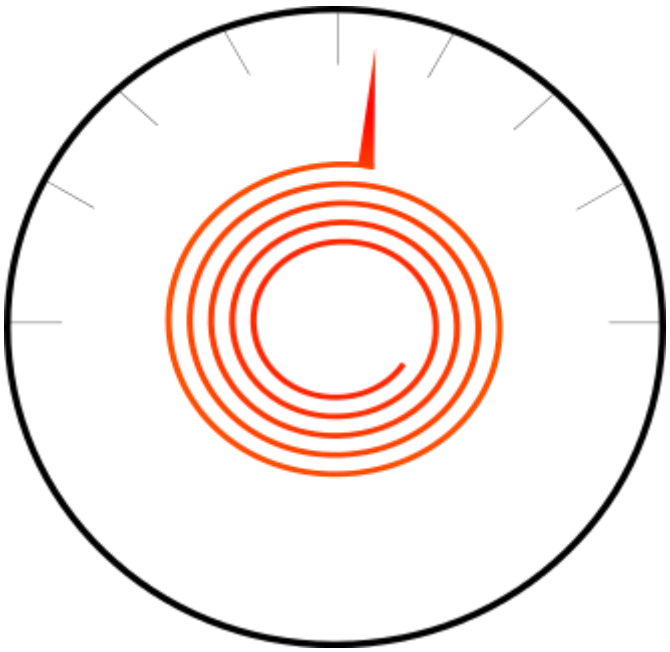

Figure 1.6 : Thermomètre à spirale.

1.8.3 Thermomètre à cristaux liquides

Les thermomètres à cristaux liquides utilisent des cristaux liquides qui changent de couleur selon la température. Ces thermomètres sont souvent utilisés pour les aquariums (modèles autocollants) ou dans le secteur médical (thermomètres frontaux), mais la mesure peut être inexacte.

1.8.4 Thermomètres à liquides

- **Thermomètre à alcool** : Le liquide organique du réservoir peut être de l'éthanol, du toluène, du kérosène ou de l'acétate de 3-méthylbutyle. Le liquide est coloré en rouge ou en bleu pour une meilleure lecture et peut se déplacer du réservoir vers un tube capillaire fermé hermétiquement et rempli d'azote. Un ménisque se forme à l'interface alcool-azote afin de pouvoir suivre l'expansion/contraction du liquide avec la variation de la température et ainsi pointer sur une échelle graduée la température.
- **Thermomètre à mercure** : Le thermomètre à mercure a été inventé par Daniel Gabriel Fahrenheit en 1724 Son fonctionnement repose sur du mercure contenu dans un tube de verre. Le volume du mercure, donc la longueur de la colonne dans le tube, est fonction de sa température. On peut lire cette dernière grâce à des marques inscrites le long du tube.
- **Thermomètre de Galilée** : Le thermomètre de Galilée est composé de flotteurs d'une densité moyenne proche du liquide dans lequel ils sont immergés. Lorsque le liquide du tube se dilate avec la température, il devient moins porteur, ce qui fait couler certains flotteurs. Plusieurs flotteurs lestés différemment peuvent montrer les températures différentes.

1.8.5 Thermomètre électronique

Les thermomètres électroniques sont très précis et performants. Ils permettent les mesures de température de l’air, des liquides, des matériaux, etc. La précision des thermomètres électroniques dépend cependant de leur fabrication et de l'usage auquel ils sont destinés.

Les types de sondes de température sont :

- Le thermocouple.
- Le capteur à résistance de platine.
- La thermistance.
- La diode.
- **Les thermomètres infrarouges** pour des mesures à distance ou sans contact. Ces derniers auraient été inventés par Charles R. Darling.

La précision des thermomètres électroniques dépend cependant de leur fabrication et de l'usage auquel ils sont destinés.

1.9 Le principe de fonctionnement de thermomètre infrarouge

La radiation infrarouge est une partie de la lumière solaire, et elle peut se diviser en la filtrant par un prisme. Cette radiation a de l'énergie. Au début du 20ème siècle, les scientifiques Planck, Stefan, Boltzmann, Wien et Kirchhoff ont défini l'activité du spectre électromagnétique et ont établi des équations pour décrire l'énergie de l'infrarouge. [12]

Le thermomètre infrarouge (IR) mesure la température par quantification de l'énergie radiative émise dans la bande spectrale de l'infrarouge. Tout objet au-dessus du zéro absolu (0 K) émet ces radiations. En connaissant la quantité d'énergie émise par un objet et son émissivité, sa température peut donc être déterminée. Cette méthode permet de mesurer la température à distance, contrairement aux autres types de thermomètres comme les thermocouples.

Le thermomètre infrarouge le plus basique est composé d'une lentille qui focalise l'énergie radiative infrarouge sur un détecteur qui la convertit en signal électrique. Après compensation, ce signal est converti à son tour en température. Cette méthode de mesure peut être très précise à condition cependant d'être bien calibrée, le rayonnement mesuré étant dépendant de nombreux paramètres : émissivité de l'objet, uniformité de la source, géométrie du dispositif. [13]

Chaque thermomètre infrarouge présente un rapport de distance au point de mesure (D:S) qui fournit le diamètre de la surface mesurée par rapport à la distance de la cible. Par exemple, si le rapport de distance au point de mesure est de 12:1, le thermomètre mesure un point d'environ 2,5 centimètres de diamètre lorsqu'il est à 30 cm de la cible. Ce thermomètre n'est donc pas par exemple adapté pour mesurer une surface de 5 cm à 1 m de distance car le thermomètre mesurera également la température hors de la zone ciblée. Selon les thermomètres infrarouges, les rapports de distance au point de mesure varient d'environ 1:1 pour les appareils entrés de gamme à environ 60:1 pour les modèles haut de gamme.

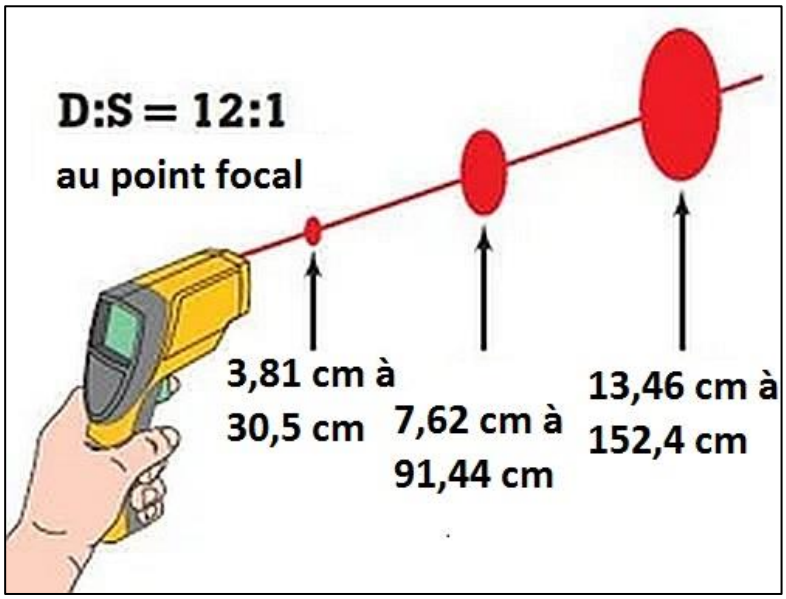

Figure 1.7 : le diamètre du point de mesure relevé à trois distances avec un thermomètre infrarouge.

Pour obtenir avec les techniques de mesure infrarouge, la température la plus proche possible de la valeur réelle, il convient cependant d'ajuster (quand l'instrument le permet) l'émissivité de l'appareil à celle du matériau ou de la surface dont on souhaite mesurer la température.

1.9.1 Le degré d'émissivité

Le degré d'émissivité est une mesure pour définir la capacité de matières d'absorber ou d'irradier l'énergie infrarouge. La valeur se trouve entre 0 et 1,0. Par exemple, un miroir possède un degré d'émissivité de 0,1. Cependant, les dénommés corps noirs possèdent un degré d'émissivité de 1,0.

Si vous régler le degré d'émissivité trop élevé, la température indiquée sera plus basse que la réelle, toujours lorsque la température de l'objet à mesurer est supérieure à la température ambiante. Vous vous avez réglé par exemple un facteur de 0,95, et le degré d'émissivité est de seulement 0,9, la température indiquée sera plus basse que la réelle. La peau humaine présente une émissivité variant de 0,97 à 23°C à 0,98 à 33°C. Elle présente donc une émissivité très proche de l'émissivité d'un corps noir.

1.9.2 Types de capteur

Le choix d'un capteur de température doit se faire en fonction du type d'application, en tenant compte des conditions ambiantes, des plages de température et de la précision de mesure souhaitée.

- **Le pyromètre monochromatique :** mesure l'énergie radiative à une longueur d'onde. On trouve parmi ces modèles des systèmes portables très basiques comme des instruments sophistiqués avec système de visée, fonction de mémorisation ou de PID. Des fibres optiques, des pointeurs laser, des systèmes de refroidissement, de protection ou de scanner peuvent compléter la configuration. On trouve également des instruments intégrant une caméra vidéo qui permet la visualisation de l'objet depuis une salle de contrôle. À chaque application correspond un système de mesure et il faudra particulièrement s'interroger sur le choix du détecteur, la gamme de température, les optiques, le temps de réponse et l'émissivité.
- **Le pyromètre bichromatique :** travaille à deux longueurs d'onde. L'état de surface peut modifier l'émissivité et donc la précision de la mesure. Cela est aussi vrai pour tous les obstacles présents sur le trajet optique. Une manière de s'affranchir en partie de ces inconvénients est de mesurer la température simultanément à deux longueurs d'onde et d'en extraire le rapport d'intensité.

1.10 Le problème de disponibilité

En raison de l'importance du thermomètre infrarouge, plusieurs pays l'utilisent comme première ligne de défense. Il est largement utilisé dans les aéroports, les centres commerciaux, les universités, les lieux de travail... etc.

La demande de pistolets à thermomètre a augmenté en flèche depuis le début de l'épidémie de coronavirus, ce qui fait que certains fabricants ont du mal à suivre. [14]

C'est pour cette raison il est important de trouver une solution à ce problème en modélisant et fabriquant un thermomètre infrarouge qu'on peut implémenter dans des lieux sensibles. On a opté pour une réalisation à base de carte Arduino.

1.11 Conclusion :

Après avoir vu les principaux tests utilisés pour la détection du COVID-19, nous nous sommes fixés comme tache, dans le cadre de notre PFE, la réalisation d'un thermomètre électronique pratique et convivial qui répond à la problématique du dépistage a tout point de vue économique et efficacité. Dans le chapitre qui suit nous allons développer l'outil principal qui réalisera notre thermomètre à savoir la carte de prototypage Arduino.

Chapitre 2 ARDUINO

2.1 Introduction

Aujourd'hui, l'intégration des systèmes numériques dans le domaine de l'électronique tend à remplacer les systèmes analogiques, dans le but de leur miniaturisation, formant ainsi des systèmes informatiques embarqués, toute en réduisant leur coût de fabrication. Il en résulte des systèmes plus complexes et performants pour un espace réduit. En fait depuis l'avènement de l'électronique, son avancée continuelle n'a cessé de progresser. Dans ce chapitre on va introduire un calculateur numérique qui n'est autre que la carte de prototypage Arduino qui servira comme outil d'acquisition et de traitement avec son outil de programmation.

2.2 Pourquoi la carte ARDUINO

Il y a de nombreuses cartes électroniques qui possèdent des plateformes basées sur des microcontrôleurs disponibles pour l'électronique programmée. Tous ces outils prennent en charge les détails de la programmation et les intègrent dans une présentation facile à utiliser. De la même façon, la carte de prototypage Arduino elle simplifie la façon de travailler avec le microcontrôleur tout en offrant aux utilisateurs plusieurs avantages dont :

- **Le prix (réduits):** les cartes ARDUINO sont relativement peu coûteuses comparativement aux autres plates-formes. La moins chère des versions du module ARDUINO peut être assemblée à la main.
- **Multi plateformes:** Le logiciel ARDUINO, écrit en JAVA, tourne sous les systèmes d'exploitation Windows, Macintosh et Linux. La plupart des systèmes à microcontrôleurs sont limités à Windows.

- **Un environnement de programmation clair et simple:** l'environnement de programmation Arduino (le logiciel Arduino IDE) est facile à utiliser pour les débutants, tout en étant assez flexible pour que les utilisateurs avancés puissent en tirer profit également.
- **Matériel Open source et extensible:** les cartes Arduino sont basées sur les Microcontrôleurs Atmel ATMEGA8, ATMEGA168, ATMEGA 328, les concepteurs des circuits expérimentés peuvent réaliser leur propre version des cartes Arduino, en les complétant et améliorant. Même les utilisateurs relativement inexpérimentés peuvent fabriquer une version sur plaque d'essai de la carte Arduino, dont le but est de comprendre comment elle fonctionne pour économiser le coût.

2.3 Introduction à la carte ARDUINO

La Carte Arduino est un circuit imprimé en matériel libre (plateforme de contrôle) dont les plans de la carte elle-même sont publiés en licence libre dont certains composants de la carte: comme le microcontrôleur et les composants complémentaires qui ne sont pas en licence libre. Un microcontrôleur programmé peut analyser et produire des signaux électriques de manière à effectuer des tâches très diverses. Arduino est utilisé dans beaucoup d'applications comme l'électronique industrielle et embarquée; le modélisme, la domotique mais aussi dans des domaines différents comme l'art contemporain et le pilotage d'un robot, commande des moteurs et faire des jeux de lumières, communiquer avec l'ordinateur, commander des appareils mobiles (modélisme). Chaque module d'Arduino possède un régulateur de tension +5 V et un oscillateur à quartez 16 MHz. Pour programmer cette carte, on utilise l'logiciel IDE Arduino.

2.3.1 Les gammes de la carte Arduino

Actuellement, il existe plus de 20 versions de carte Arduino, nous citons quelques un afin d'éclaircir l'évaluation de ce produit scientifique et académique

Cartes Arduino / Caractéristiques	UNO R3 (classique & CMS)	UNO R3 Ethernet (classique & POE)	Leonardo	Mega 2560	Mega ADK	DUE	Esplora	Mini	Nano
Microcontrôleur	ATmega328P	ATmega328P	ATmega32u4	ATmega2560	ATmega2560	AT91SAM3X8E	ATmega32u4	ATmega328P	ATmega328P
Cadencement Horloge	16 MHz	16 MHz	16 MHz	16 MHz	16 MHz	84 MHz	16 MHz	16 MHz	16 MHz
Tension d'entrée	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 12V	7 - 9V	7 - 9V
Tension de fonctionnement	5V	5V	5V	5V	5V	3,3V	5V	5V	5V
Entrée/Sortie Numérique	14/6	14/4	20/7	54/15	54/15	54/12	✗	14/6	14/6
Entrée-Sortie (PWM) Analogique	6/0	6/0	12/0	16/0	16/0	12/2 (DAC)	✗	8/0	8/0
Mémoire vive (Flash)	32 Ko	32 Ko	32 Ko	256 Ko	256 Ko	512 Ko	32 Ko	32 Ko	32 Ko
Mémoire vive (SRAM)	2 Ko	2 Ko	2,5 Ko	8 Ko	8 Ko	96 Ko	2,5 Ko	2 Ko	2 Ko
Mémoire morte (EEPROM)	1 Ko	1 Ko	1 Ko	4 Ko	4 Ko	✗	1 Ko	1 Ko	1 Ko
Interface USB	USB-B mâle	USB-B mâle	Micro-USB	USB-B mâle	USB-B mâle & USB-A pour Android	2 ports micro-USB (Native et programming)	Micro-USB	✗	Mini-USB
Port UART	1	1	1	4	4	4	✗	✗	1
Carte SD	✗	✓	✗	✗	✗	✗	✗	✗	✗
Ethernet	✗	✓	✗	✗	✗	✗	✗	✗	✗
Wi-Fi	✗	✗	✗	✗	✗	✗	✗	✗	✗
Dimensions	68x53mm	68x53mm	68x53mm	101x53mm	101x53mm	101x53mm	165x60mm	30x18mm	45x18mm

Tableau 2.1 : Caractéristiques des différentes cartes ARDUINO.

Nous avons choisi une carte Arduino UNO (carte Basique). L'intérêt principal de cette carte est de faciliter la mise en œuvre d'une commande qui sera détaillée par la suite.

L'Arduino fournit un environnement de développement s'appuyant sur des outils open source comme interface de programmation. L'injection du programme déjà converti par l'environnement sous forme d'un code «HEX» dans la mémoire du microcontrôleur se fait d'une façon très simple par la liaison USB. En outre, des bibliothèques de fonctions "clé en main" sont également fournies pour l'exploitation d'entrées-sorties. Cette carte est basée sur un microcontrôleur ATmega 328 et des composants complémentaires. La carte Arduino contient une mémoire morte EEPROM de 1 kilo octet. Elle est dotée de 14 entrées/sorties digitales (dont 6 peuvent être utilisées en tant que sortie PWM), 6 entrées analogiques et d'un quartz de 16 MHz, une connexion USB et possède un bouton de remise à zéro et une prise jack d'alimentation.

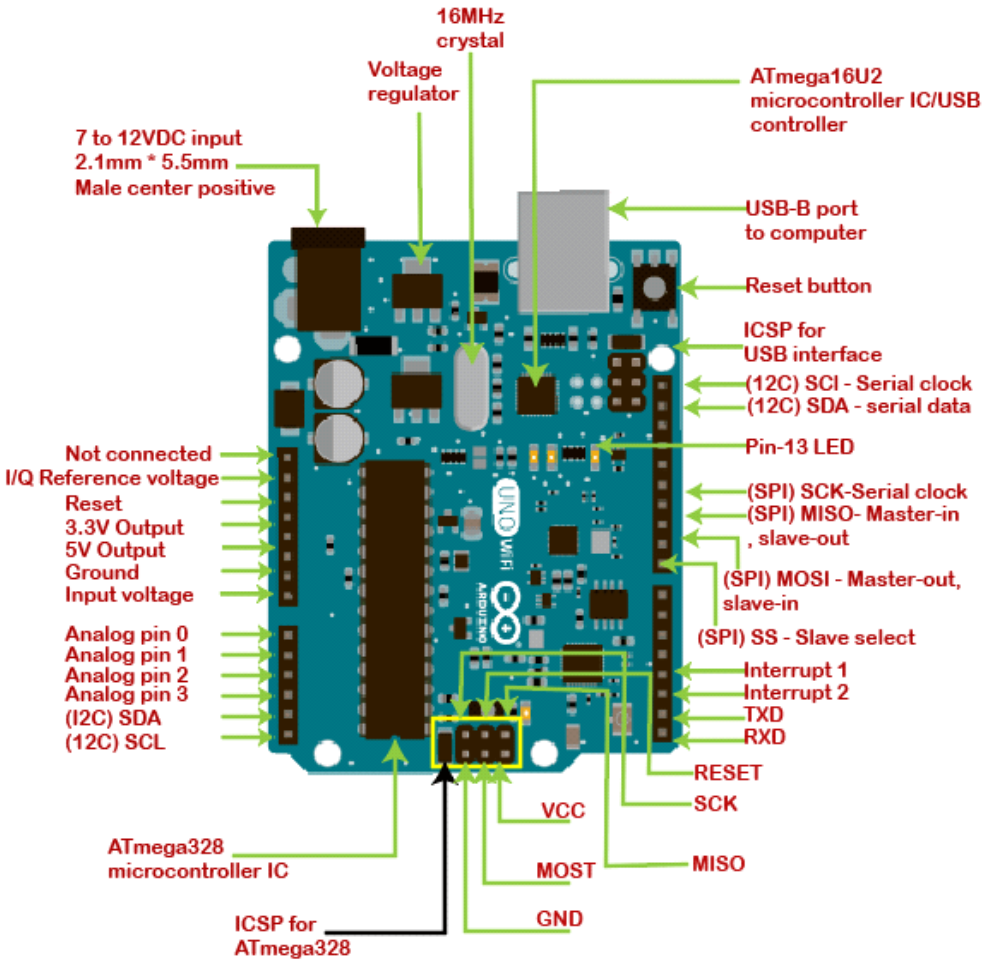

Figure 2.1 : Les spécifications de la carte ARDUINO UNO.

2.3.2 La constitution de la carte Arduino UNO

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

a Le Microcontrôleur ATMega328

Un microcontrôleur ATMega328 est un circuit intégré qui rassemble sur une puce plusieurs éléments complexes dans un espace réduit au temps des pionniers de l'électronique.

Aujourd'hui, avec l'avènement des circuits intégrés on n'a pas besoin de souder un grand nombre de composants encombrants; tels que des transistors; des résistances et des condensateurs car tout peut être logé dans un petit boîtier en plastique noir muni d'un certain nombre de broches dont la programmation peut être réalisée en langage C.

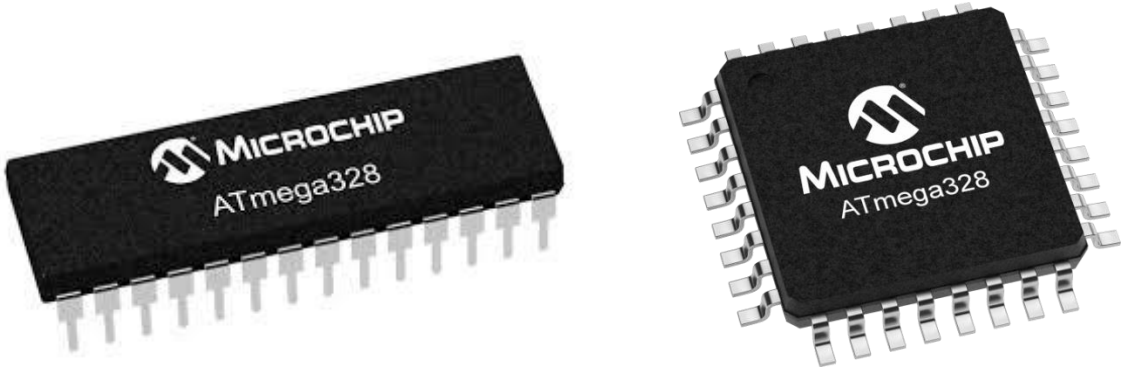

Figure 2.2 : ATmega238 classique et CMS.

Le microcontrôleur ATMega328 est constitué par un ensemble d'éléments qui ont chacun une fonction bien déterminée. Il est en fait constitué des mêmes éléments que sur la carte mère d'un ordinateur. Globalement, l'architecture interne de ce circuit programmable se compose essentiellement sur :

- **RAM** : c'est la mémoire dite "volatile", elle va contenir les variables du programme, et elle s'efface si on coupe l'alimentation du microcontrôleur. Sa capacité est 2 ko.
- **EEPROM** : C'est le disque dur du microcontrôleur. On y enregistre des infos qui ont besoin de survivre dans le temps, même si la carte doit être arrêtée. Cette mémoire ne s'efface pas lorsque l'on éteint le microcontrôleur ou lorsqu'on le reprogramme.
- **La mémoire Flash** : C'est celle qui contiendra le programme à exécuter. Cette mémoire est effaçable et réinscriptible mémoire programme de 32Ko (dont bootloader de 0.5 ko).

b Les E/S

Chacune des 14 broches numériques de la carte UNO (numérotées des 0 à 13) peut être utilisée soit comme une entrée numérique, soit comme une sortie numérique, en

utilisant les instructions pinMode(), digitalWrite() et digitalRead() du langage Arduino. Ces broches fonctionnent en 5V. Chaque broche peut fournir ou recevoir un maximum de 40mA d'intensité et dispose d'une résistance interne de "rappel au plus" (pull-up) (déconnectée par défaut) de 20-50 KOhms. Cette résistance interne s'active sur une broche en entrée à l'aide de l'instruction digitalWrite(broche, HIGH).

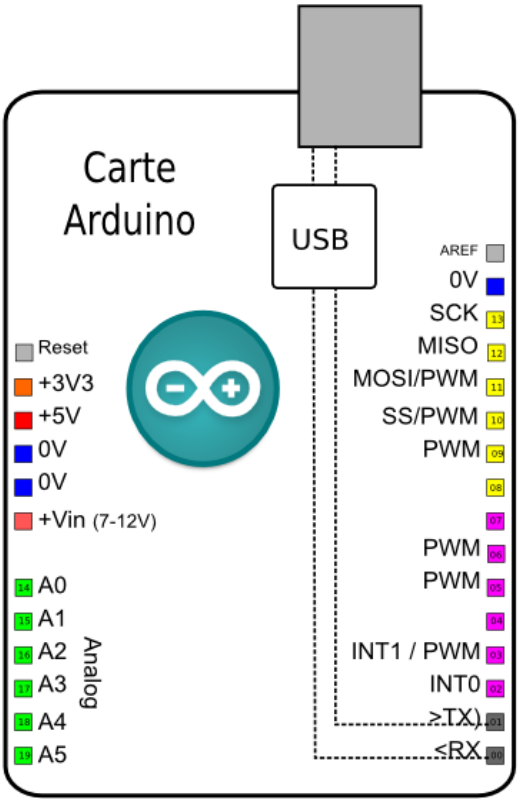

Figure 2.3 : Brochage de la carte Arduino Uno.

De plus, certaines broches ont des fonctions spécialisées :

- **Communication Serie**: Broches 0 (RX) et 1 (TX). Utilisées pour recevoir (RX) et transmettre (TX) les données séries de niveau TTL. Ces broches sont connectées aux broches correspondantes du circuit intégré ATmega8U2 programmé en convertisseur USB-vers-série de la carte, composant qui assure l'interface entre les niveaux TTL et le port USB de l'ordinateur.
- **Interruptions Externes:** Broches 2 et 3. Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur. Voir l'instruction attachInterrupt() pour plus de détails.

- **Impulsion PWM (largeur d'impulsion modulée):** Broches 3, 5, 6, 9, 10, et 11. Fournissent une impulsion PWM 8-bits à l'aide de l'instruction analogWrite().
- **SPI (Interface Série Périphérique):** Broches 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Ces broches supportent la communication SPI (Interface Série Périphérique) disponible avec la librairie pour communication SPI. Les broches SPI sont également connectées sur le connecteur ICSP qui est mécaniquement compatible avec les cartes Mega.
- **I2C:** Broches 4 (SDA) et 5 (SCL). Supportent les communications de protocole I2C (ou interface TWI (Two Wire Interface - Interface "2 fils"), disponible en utilisant la librairie Wire/I2C (ou TWI - Two-Wire interface - interface "2 fils") .
- **LED:** Broche 13. Il y a une LED incluse dans la carte connectée à la broche 13. Lorsque la broche est au niveau HAUT, la LED est allumée, lorsque la broche est au niveau BAS, la LED est éteinte.
- **Broches analogiques** : La carte Uno dispose de 6 entrées analogiques (numérotées de 0 à 5), chacune pouvant fournir une mesure d'une résolution de 10 bits (càd sur 1024 niveaux soit de 0 à 1023) à l'aide de la très utile fonction analogRead() du langage Arduino. Par défaut, ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023), mais il est possible de modifier la référence supérieure de la plage de mesure en utilisant la broche AREF et l'instruction analogReference() du langage Arduino.Les broches analogiques peuvent être utilisées en tant que broches numériques : elles sont numérotées en tant que broches numériques de 14 à 19.

c Alimentation

La carte Arduino Uno peut-être alimentée soit via la connexion USB (qui fournit 5V jusqu'à 500mA) ou à l'aide d'une alimentation externe. La source d'alimentation est sélectionnée automatiquement par la carte. L'alimentation externe (non-USB) peut être soit un adapteur secteur (pouvant fournir typiquement de 3V à 12V sous 500mA) ou des piles (ou des accus). L'adaptateur secteur peut être connecté en branchant une prise 2.1mm positif au centre dans le connecteur jack de la carte. Les fils en provenance d'un bloc de piles ou d'accus peuvent être insérés dans les connecteurs

des broches de la carte appelées Gnd (masse ou 0V) et Vin (Tension positive en entrée) du connecteur d'alimentation.

La plage idéale recommandée pour alimenter la carte Uno est entre 7V et 12V.

Les broches d'alimentation sont les suivantes :

- **VIN :** La tension d'entrée positive lorsque la carte Arduino est utilisée avec une source de tension externe (à distinguer du 5V de la connexion USB ou autre source 5V régulée). Vous pouvez alimenter la carte à l'aide de cette broche, ou, si l'alimentation est fournie par le jack d'alimentation, accéder à la tension d'alimentation sur cette broche.
- **5V :** La tension régulée utilisée pour faire fonctionner le microcontrôleur et les autres composants de la carte (pour info : les circuits électroniques numériques nécessitent une tension d'alimentation parfaitement stable dite "tension régulée" obtenue à l'aide d'un composant appelé un régulateur et qui est intégré à la carte Arduino). Le 5V régulé fourni par cette broche peut donc provenir soit de la tension d'alimentation VIN via le régulateur de la carte, ou bien de la connexion USB (qui fournit du 5V régulé) ou de tout autre source d'alimentation régulée.
- **3V3 :** Une alimentation de 3.3V fournie par le circuit intégré FTDI (circuit intégré faisant l'adaptation du signal entre le port USB de votre ordinateur et le port série de l'ATmega) de la carte est disponible : ceci est intéressant pour certains circuits externes nécessitant cette tension au lieu du 5V). L'intensité maximale disponible sur cette broche est de 50mA.
- **GND :** Broche de masse (ou 0V).

2.4 Software

2.4.1 Présentation d'IDE ARDUINO

Un IDE (environnement de développement) libre et gratuit est distribué sur le site internet d'Arduino. D'autres alternatives existent pour développer pour Arduino (extensions pour CodeBlocks, Visual Studio, Eclipse, XCode, etc…). L'interface de l'IDE

Arduino offre une interface minimale et épurée pour développer un programme sur les cartes Arduino. Il est doté d'un éditeur de code avec coloration syntaxique et d'une barre d'outils rapide. Ce sont les deux éléments les plus importants de l'interface. On retrouve aussi une barre de menus qui est utilisé pour accéder aux fonctions avancées de l'IDE. Enfin, une console affichant les résultats de la compilation du code source.

2.4.2 Langage de base utilisé

Le langage Arduino est inspiré de plusieurs langages. On retrouve notamment des similarités avec le C, le C++, le Java et le Processing. Le langage impose une structure particulière typique de l'informatique embarquée. Le code de la syntaxe est structuré par une ponctuation stricte :

- Toute ligne de code se termine par un point-virgule « ; ».
- Le contenu d'une fonction est délimité par des accolades « { » et « } ».
- Les paramètres d'une fonction sont contenus pas des parenthèses « (» et «) ».

Un programme Arduino doit se structurer en 4 parties :

a Les commentaires

La première partie du programme, doit impérativement débuter par des commentaires permettant de définir ce que doit faire le programme. Ceci permet de savoir en quoi consiste son fonctionnement, pourquoi pas sa date de création, sa version …etc.

Mais les commentaires ne se limitent pas à la première partie. Les commentaires doivent agrémenter le programme afin de le faire comprendre par une personne tierce, essayant de le lire. Les commentaires peuvent être en deux formes :

- Une ligne qui commence par "//" est considérée comme un commentaire.
- Un paragraphe qui commence par "/*" et qui se termine par "*/" est considéré comme un commentaire.

b Les variables

Une variable est un espace de stockage nommé qui permet de stocker une valeur utilisable par la suite dans la boucle d'un programme. Une variable peut aussi bien représenter des données lues ou envoyées sur un des ports analogiques ou numériques, une étape de calcul pour associer ou traiter des données, que le numéro 'physique' de ces entrées ou sorties sur la carte. Une "variable" n'est donc pas exclusivement un paramètre variant dans le programme.

La nature de la variable définit les limites de celle-ci, ainsi que la place nécessaire dans la mémoire de la carte Arduino. Ceci a de l'importance lorsque vous créez de gros programme impliquant beaucoup de variables qui prennent de la place en mémoire. Cette définition permet de limiter la place mémoire. De plus, il faut savoir que plus la variable prend de place, plus les opérations de calcul seront long et inversement. Il est donc nécessaire de définir les variables au plus juste en fonction de ce que le programme doit faire ; parmi les plus utilisées sont :

- **byte :** nombre de 0 à 255.
- **char :** définit un caractère alphanumérique.
- **int :** entier décimal allant de -32 768 à 32 767. Va de paire avec **unsigned int** qui lui ira de 0 à 65 535 (il augmente les positifs et élimine les négatifs).
- **string :** suite de caractères alphanumérique. Pas de limite, sauf celle de la mémoire.
- **long :** entier décimal allant de -2 147 483 648 à 2 147 483 648. Va de paire avec **unsigned long** qui lui ira de 0 à 4 294 967 295 (il augmente les positifs et élimine les négatifs).
- **array :** définition d'un tableau de variable.

c Configuration des entrées/sorties

La fonction setup permet dans la plupart du temps de configurer les différentes broches du microcontrôleur.

Dans cette partie on définit la nature des entrées/sorties matérielles utilisées (si telle ou telle broche est utilisée en entrée ou en sortie, de nature numérique (binaire : 0 ou

1) ou analogique. Cette phase doit être précédée par la commande "void setup() " suivie d'une Accolade "{" et se terminer d'une accolade "}".

Dans cette configuration on doit définir comment doivent se comporter les différentes broches de l'ARDUINO. Comme aussi définir les broches numériques, utilisant que du binaire, analogique servant soit aux convertisseurs ou à la PWM, ainsi qu'aux broches de communications.

- **Numérique :**

pinMode (Broche, état) ;

La première variable à mettre est le numéro de broche que l'on essaie de configurer.

La deuxième variable est l'état de configuration, soit " INPUT " (entrée, il s'agit de broche recevant des informations), soit " OUTPUT " (sortie, il s'agit de broche envoyant des informations).

- **Analogique :**

Pour la configuration de broche analogique, seul une sortie analogique doit être configurée. On doit simplement ajouter pinMode (numéro de broche, OUTPUT).

Pour les entrées analogiques, pas besoin de les définir, elles se mettront automatiquement en entrée lorsqu'on les sollicitera.

- **Communication :**

Il existe plusieurs moyens pour faire communiquer des systèmes électroniques entre eux et la carte arduino. Certaines broches pourront être utilisé pour des type de communication,Tel que liaison RS232, I2C ou autre. Ces broches ont des configurations particulières.

Serial.begin(X): définit la vitesse X (en bits par seconde) de transfert des données.

Cette fonction doit apparaitre dans le setup du programme. Il existe plusieurs vitesses de transmission des données : 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 250000.

serial.print(X) : permet d'envoyer l'information X.

serial.println(X) : même action que print mais avec un retour à la ligne à la fin du message.

d Programme principal

void loop( ) : cette instruction définit le début du programme principal, celui qui tournera en permanence sur la carte. La fonction loop permet justement de faire tourner en boucle le programme.

On commence et on termine par des accolades ' { ' et ' } '.

I. Lecture et écriture numérique et analogique

digitalRead (Broche) : cette instruction permet de lire la valeur d'une entrée. Comme cette valeur, pour être utilisable, doit être sauvegardée, nous allons utiliser une équation qui devra ressembler à ça :

variable = digitalRead (n° de broche)

digitalWrite(Broche, état) : cette instruction permet d'écrire sur une broche la valeur de l'état. Cet état sera soit HIGH (niveau haut : 1), soit LOW (niveau bas : 0).

analogRead(Broche) : cette instruction permet de lire une valeur analogique sur une des broches dédiées. Tout comme digitalRead, cette valeur doit être sauvegardée.

variable = analogRead(0); //sauvegarder la valeur du convertisseur lié à la broche A0 (entrée analogique 0) dans la variable.

Chaque conversion se fera par rapport à une tension par défaut de 5V maxi. La valeur convertit sera une image de la tension qui va évoluer entre 0 et 5V, par tranche de 4,88 mV.

Par exemple si la tension sur A0 est de 2,5 V, la valeur obtenue après l'instruction analogRead sera : 2,5/0,00488 = 512.

analogWrite(Broche,valeur) : cette instruction permet de créer une PWM : Cette fonction appelée PWM (Modulation à Largeur d'Impulsion), permet de générer un signal rectangulaire périodique de fréquence fixe (490 Hz, soit une période d'environ 2ms) mais de rapport cyclique variable (le temps au niveau haut évolue sans changer la période). Le rapport cyclique est réglé par la "valeur". Celle-ci peut être entre 0 et 255. Cette valeur divisée par 255, donnera le pourcentage de temps que prendra le niveau haut par rapport au niveau bas.

II. Gestion du temps

Il est souvent nécessaire d'effectuer des pauses dans le temps, de temporiser des programmes. Ceci est souvent utilisé dans la gestion de signaux numérique. La procédure avec l'ARDUINO est assez simple, il existe deux instructions permettant d'effectuer des pauses:

delay (ms): cette instruction permet d'effectuer des pauses d'une durée en milli seconde inscrite entre parenthèses.

delayMicroseconds(µs) : cette instruction permet d'effectuer des pauses d'une durée en micro seconde inscrite entre parenthèses.

III. Les structures de contrôle

Les structures de contrôle sont des blocs d'instructions qui s'exécutent en fonction du respect d'un certain nombre de conditions. Il existe quatre types de structure :

if...else : exécute un code si certaines conditions sont remplies et éventuellement exécutera un autre code avec sinon ;

while : exécute un code tant que certaines conditions sont remplies ;

for : exécute un code pour un certain nombre de fois ;

switch/case : fait un choix entre plusieurs codes parmi une liste de possibilités.

2.5 CONCLUSION

Dans ce chapitre, nous avons détaillé la carte de prototypage Arduino, cette derniere servira comme outil de base pour la réalisation de notre application, de l'acquisition, la mémorisation à l'affichage. Nous avons surtout insisté et développé les parties que nous avons utilisé, tels que les entrées analogique, la transmission de données, l'affichage avec la syntaxe nécessaire pour la programmation. Cette étape nous a permis d'entamer le chapitre qui suit à savoir l'étape de réalisation de notre application en toute quiétude.

Chapitre 3 REALISATION

3.1 Introduction

Après avoir vu en premier lieu les différents moyens de dépistage du COVID 19, ces derniers représentent chacun d'eux des avantages et inconvénient du point de vu cout et efficacité, nous nous fixe comme tache la réalisation d'un thermomètre électronique dont le synoptique est représenté comme suit.

3.2 Synoptique général

Le schéma synoptique général de notre réalisation est donné par la figure ci-dessous.

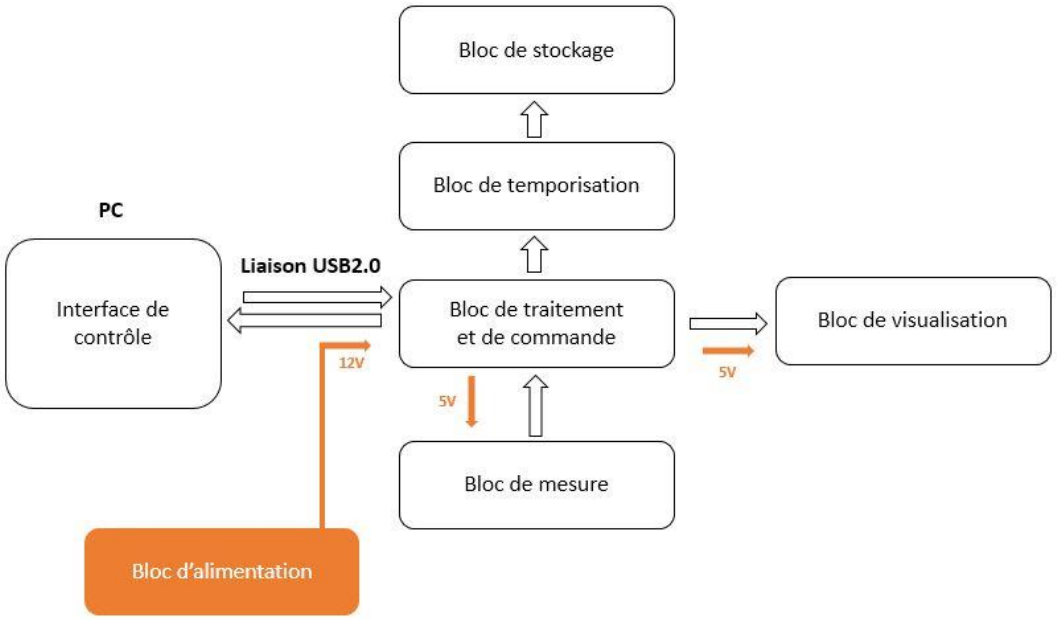

Figure 3.1 : Schéma synoptique général du projet.

3.2.1 Bloc de mesure

a Capteur de température infrarouge

I. **Définition**

Le capteur de température infrarouge numérique est un module de mesure de la température sans contact qui se base sur le capteur thermopile OTP-538U.

Figure 3.2 : Capteur de température du haut and et en bas.

La puce de détection de la thermopile sensible aux IR et la puce de conditionnement du signal sont toutes deux intégrées dans le même boîtier. Ce module communique avec Arduino en utilisant le SMBus, jusqu'à 127 capteurs peuvent être lus via 2 fils communs. Il est composé de 116 éléments de thermocouple en série sur une micro-membrane flottante, la surface noire du capteur est bonne pour absorber le rayonnement infrarouge thermique incident, ce qui pourrait déclencher une réponse en tension à la sortie.

Grâce à l'amplificateur à faible bruit du module, à l'ADC 16 bits et à la puissante unité DSP, il peut atteindre une précision élevée de 1℃ sur une large plage de température et une résolution de mesure élevée de 0,02℃.

Ce capteur fournit une tension analogique (0~1,1V) en fonction de la température cible.

II. **Spécifications**

	MIN	*Typique*	*MAX*	*Unité*
Voltage	2.6	3	3.4	V
Courant		1.4	1.5	mA
Plage de mesure	-10		100	C

Température de fonctionnement	-10	80	C
Dimensions	20x40x9.6		mm

Tableau 3.1 : Spécifications du capteur de température infrarouge.

III. **Les versions**

LA VERSION	MODIFICATIONS	DATE DE SORTIE
GROVE - CAPTEUR DE TEMPERATURE A INFRAROUGE V1.0	Produit initial	11 Décembre 2015
GROVE - CAPTEUR DE TEMPERATURE A INFRAROUGE V1.1	Optimiser la mise en forme	24 juillet 2016
GROVE - CAPTEUR DE TEMPERATURE A INFRAROUGE V1.2	Changer la puce d'alimentation pour rendre l'alimentation plus stable	10 février 2018

Tableau 3.2 : Les versions Capteur de température infrarouge.

IV. **Brochage du capteur**

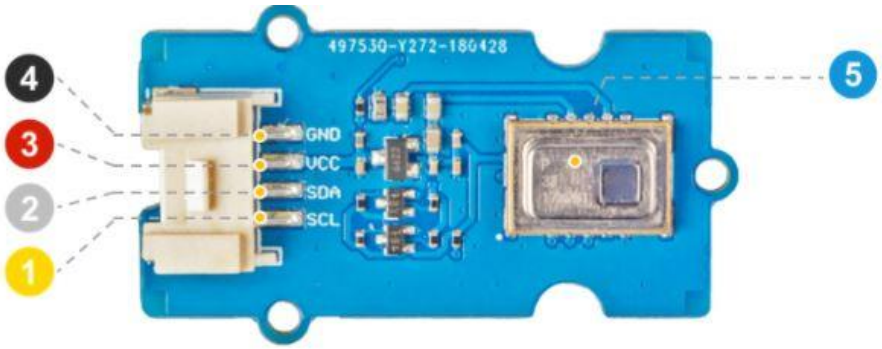

Figure 3.3 : Les pins du capteur.

1/ I2C SCL : ligne de données bidirectionnelle.

2/ I2C SDA : ligne d'horloge de synchronisation bidirectionnelle.

3/ VCC : on peut utiliser 5v ou 3.3v pour ce module.

4/ GND : on connecte ce module au système GND. 5/ Capteur thermopile OTP-538U

Ce module utilise le système des connecteurs Grove, c'est un système de prototypage de connecteurs modulaires et standardisés.

La figure ce dissous montre les dimensions standardisées du Grove 20X40 SMD Vertical.

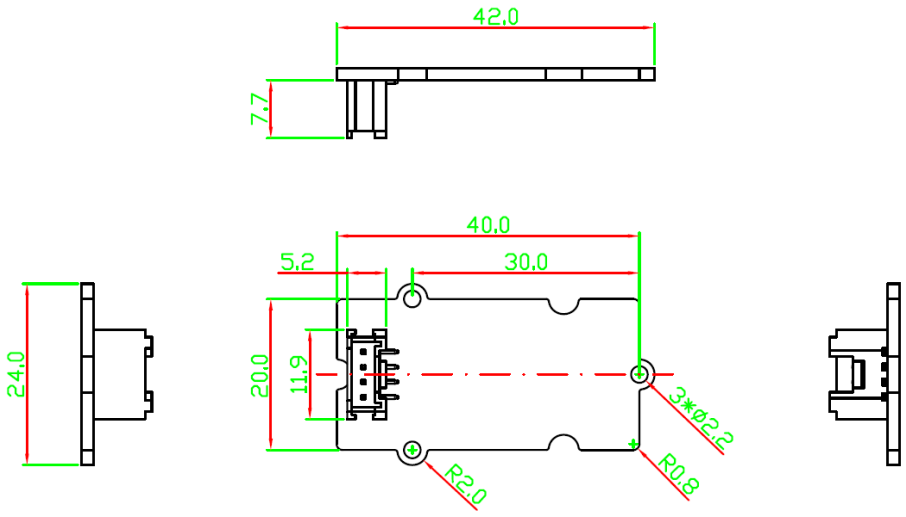

Figure 3.4 : Le dessin technique d'un capteur Grove 20X40 SMD vertical.

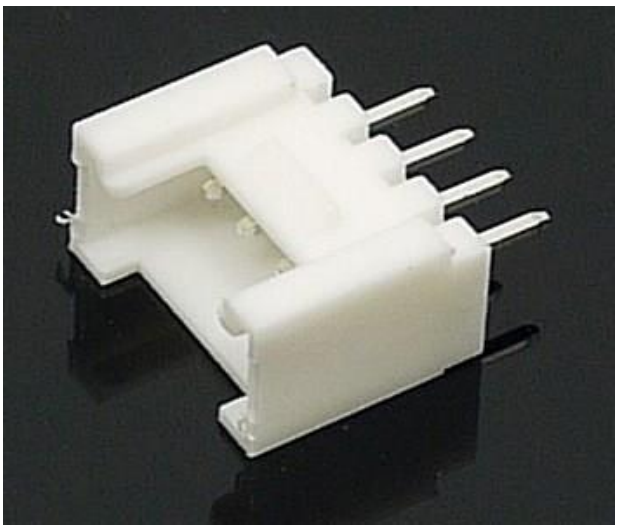

Figure 3.5 : Connecteur Grove.

V. Calibration du capteur

Le capteur de température infrarouge est réglé par défaut par le fabricant, mais chaque capteur est différent, c'est pourquoi il est important de le calibrer.

Le code donné en annexe 1 nous permet de mesurer la tension du capteur et de le calibrer ensuite.

Nous mettons le capteur dans un environnement normal pendant plus de 5 minutes, ce qui fait que la température du capteur est la même que la température ambiante. Puis on ouvre le « Serial Monitor » pour vérifier la tension que le capteur produit. Lorsque la température ambiante est égale à celle du capteur de température, la sortie du capteur infrarouge est de 0V. Nous devrions réguler la tension de référence qui se décale à 0,5V par le matériel. Comme indiqué ci-dessous, la tension du capteur est de 0,005V, il suffit de modifier la valeur offset_vol à 0,005 que nous obtenons du moniteur série dans le programme.

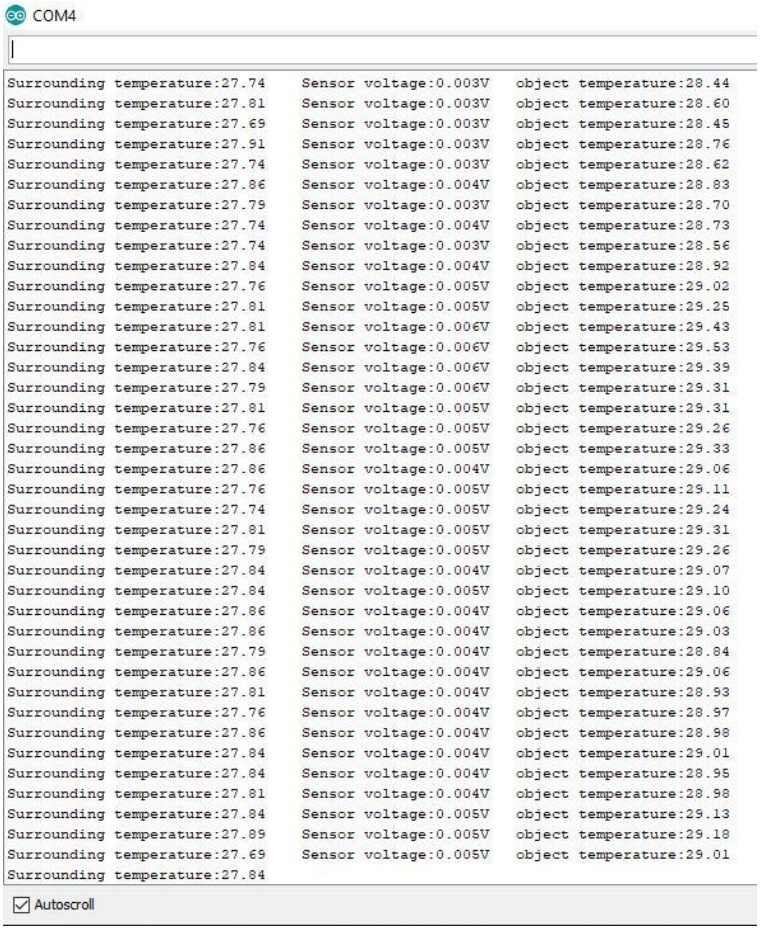

Figure 3.6 : Serial monitor du capteur infrarouge.

Après de nombreux tests, nous avons constaté que la meilleure distance de mesure est de 3~4 cm pour obtenir les résultats les plus précis.

COM4

```
Surrounding temperature:28.91    Sensor voltage:0.043V   object temperature:39.71
Surrounding temperature:28.83    Sensor voltage:0.040V   object temperature:38.85
Surrounding temperature:28.96    Sensor voltage:0.038V   object temperature:38.67
Surrounding temperature:28.93    Sensor voltage:0.037V   object temperature:38.19
Surrounding temperature:28.91    Sensor voltage:0.036V   object temperature:38.89
Surrounding temperature:28.86    Sensor voltage:0.035V   object temperature:38.57
Surrounding temperature:28.88    Sensor voltage:0.034V   object temperature:38.30
Surrounding temperature:28.91    Sensor voltage:0.033V   object temperature:37.89
Surrounding temperature:28.91    Sensor voltage:0.031V   object temperature:37.47
Surrounding temperature:28.96    Sensor voltage:0.031V   object temperature:37.53
Surrounding temperature:28.96    Sensor voltage:0.032V   object temperature:37.82
Surrounding temperature:28.98    Sensor voltage:0.032V   object temperature:37.85
Surrounding temperature:28.93    Sensor voltage:0.033V   object temperature:37.85
Surrounding temperature:28.93    Sensor voltage:0.032V   object temperature:37.77
Surrounding temperature:28.91    Sensor voltage:0.033V   object temperature:37.95
Surrounding temperature:28.93    Sensor voltage:0.033V   object temperature:37.85
Surrounding temperature:28.88    Sensor voltage:0.032V   object temperature:37.54
Surrounding temperature:29.01    Sensor voltage:0.032V   object temperature:37.67
Surrounding temperature:28.93    Sensor voltage:0.033V   object temperature:38.06
Surrounding temperature:28.98    Sensor voltage:0.031V   object temperature:37.55
Surrounding temperature:28.93    Sensor voltage:0.031V   object temperature:37.32
Surrounding temperature:29.04    Sensor voltage:0.030V   object temperature:37.28
Surrounding temperature:28.96    Sensor voltage:0.031V   object temperature:37.58
Surrounding temperature:28.98    Sensor voltage:0.033V   object temperature:37.91
Surrounding temperature:28.98    Sensor voltage:0.032V   object temperature:37.82
Surrounding temperature:29.06    Sensor voltage:0.032V   object temperature:37.89
Surrounding temperature:28.98    Sensor voltage:0.031V   object temperature:37.61
Surrounding temperature:28.96    Sensor voltage:0.031V   object temperature:37.53
Surrounding temperature:28.98    Sensor voltage:0.032V   object temperature:37.85
Surrounding temperature:28.88    Sensor voltage:0.032V   object temperature:37.74
Surrounding temperature:28.98    Sensor voltage:0.034V   object temperature:38.17
Surrounding temperature:29.04    Sensor voltage:0.032V   object temperature:37.84
Surrounding temperature:29.09    Sensor voltage:0.033V   object temperature:38.21
Surrounding temperature:28.98    Sensor voltage:0.033V   object temperature:37.99
Surrounding temperature:29.06    Sensor voltage:0.032V   object temperature:37.92
Surrounding temperature:29.06    Sensor voltage:0.031V   object temperature:37.57
Surrounding temperature:29.14    Sensor voltage:0.030V   object temperature:37.35
Surrounding temperature:29.04    Sensor voltage:0.030V   object temperature:37.25
Surrounding temperature:29.11    Sensor voltage:0.029V   object temperature:37.12
Surrounding temperature:29.04
```

Autoscroll

Figure 3.7 : Les tests de capteur infrarouge.

b Capteur infrarouge de proximité TCRT

I. Définitions

Le TCRT5000 est un capteur à réflexion qui contient un émetteur infrarouge et un phototransistor dans un emballage plombé qui bloque la lumière visible. La figure ci-dessous montre la LED infrarouge (de couleur bleue) et le phototransistor (de couleur noire).

Figure 3.8 : Capteur TCRT5000.

II. Caractéristiques

- Tension de service: 3.3VDC à 5VDC.
- Plage de fonctionnement: 0.2 mm à 25mm.
- Distance de fonctionnement maximale : 2,5 mm.
- Courant de sortie typique sous test : IC = 1 mA.
- Dimensions: 31mm x 14mm.

III. Principe *de fonctionnement*

Le TCRT5000 fonctionne en émettant une lumière infrarouge (950 nm) à partir de la LED et en enregistrant toute lumière réfléchie sur son phototransistor, ce qui modifie le flux de courant entre son émetteur et son collecteur en fonction du niveau de lumière qu'il reçoit.

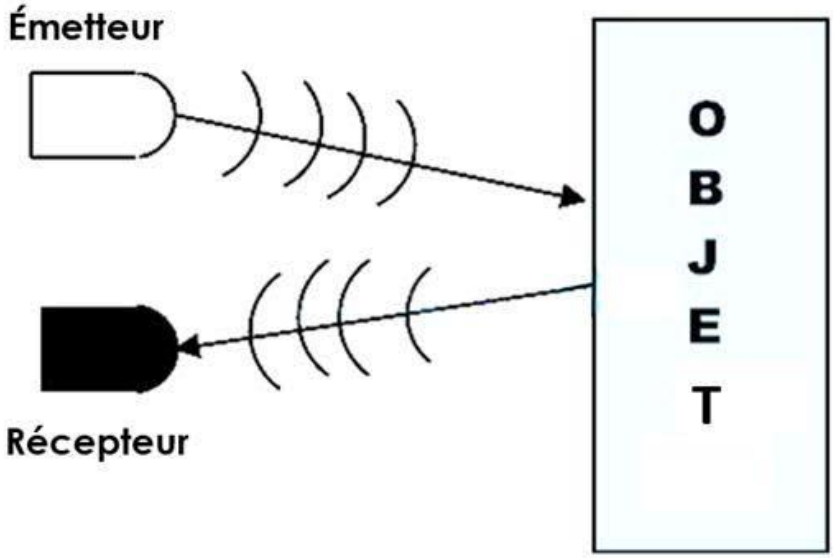

Figure 3.9 : Principe de fonctionnement.

Le phototransistor fonctionne comme interrupteur commandé par la lumière infrarouge qu'il reçoit :

Si la quantité de lumière est nulle ou inférieure à un seuil donné, le transistor est bloqué, tout se passe comme si l'interrupteur équivalent était ouvert, en revanche, si

la lumière que reçoit le phototransistor est supérieure au seuil, le phototransistor est passant, l'interrupteur équivalent est fermé.

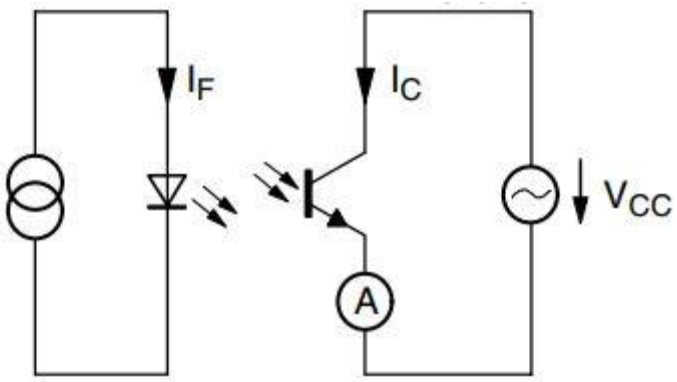

Figure 3.10 : Circuit TCRT5000.

Le Vcc augmente lorsque la distance entre l'objet et l'émetteur diminue.

3.2.2 Bloc de temporisation

a Module RTC DS3231

I. **Définition**

Le DS3231 est une horloge en temps réel I2C peu coûteuse et extrêmement précise avec un oscillateur à cristal intégré, L'appareil est alimenté par une batterie et maintient un chronométrage précis lorsque l'alimentation principale de l'appareil est interrompue. L'intégration du résonateur à cristal améliore la précision à long terme de l'appareil.

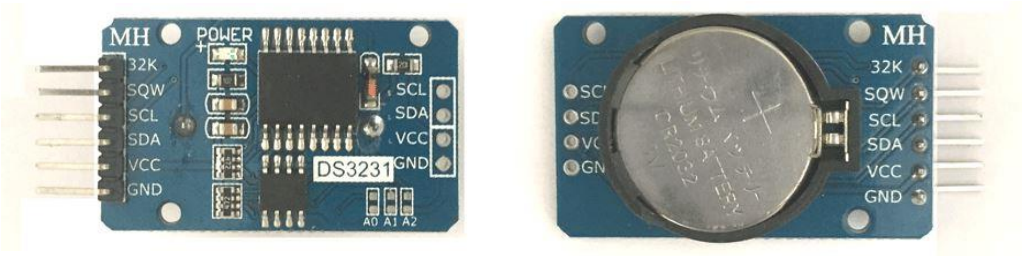

Figure 3.11 : RTC DS3231 du haut and et en bas.

Le RTC conserve les informations relatives aux secondes, aux minutes, aux heures, au jour, à la date, au mois et à l'année. La date à la fin du mois est automatiquement ajustée pour les mois de moins de 31 jours, y compris les corrections pour les années bissextiles. L'horloge fonctionne soit en format 24 heures, soit en format 12 heures avec un indicateur AM/PM. L'adresse et les données sont transférées en série par un bus bidirectionnel I2C.

II. Spécifications

- Tension d'alimentation de + 2,3 V à + 5,5 V.
- Haute vitesse (400 KHZ) du bus série I2C.
- Poids net: 2 g.
- Taille: 15 x 13 x 14 mm.
- Faible consommation d'énergie.
- Sortie 1 Hz et 32 768 kHz.

III. Brochage

Le DS3231 est un appareil à six bornes, dont l'utilisation de deux broches n'est pas obligatoire. Nous avons donc principalement quatre broches. Ces quatre broches se trouvent de l'autre côté du module qui porte le même nom.

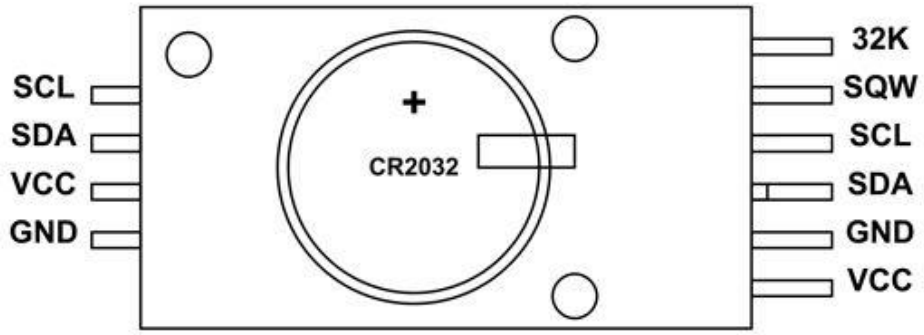

Figure 3.12 : Les pins du DS3231.

- **VCC :** Connecté au positif de la source d'énergie.
- **GND :** Connecté à GND.
- **SDA :** Serial Data pin (interface I2C).

- **SCL :** Serial Clock pin (interface I2C).
- **SQW :** Broche de sortie Square Wave**.**
- **32K :** Sortie de l'oscillateur 32K.

IV. Caractéristiques

- Précision : +2ppm à -2ppm pour 0ºC à +40ºC, +3,5ppm à -3,5ppm pour -40ºC à +85ºC.
- Capteur de température numérique avec une précision de ±3ºC.
- Sortie d'onde carrée programmable.
- Interface I2C 400Khz.
- Faible consommation d'énergie.
- Circuit de coupure automatique de la batterie.
- Pile de secours CR2032 avec une durée de vie de deux à trois ans.

V. Applications

- Bien qu'il existe de nombreux modules RTC sur le marché, le DS3231 est l'un des plus populaires en raison de sa précision. La puce maintient l'heure à jour avec plus de précision que la plupart des modules.
- Le MODULE RTC DS3231 consomme très peu d'énergie pour fonctionner. Ce module peut donc être utilisé sur des systèmes mobiles.
- MODULE DS3231 RTC capable de communiquer avec une interface TWI à grande vitesse.
- Le DS3231 peut également fonctionner sur des applications de recherche de format 24 heures et 12 heures dans les systèmes GPS. Avec deux réveils et un capteur de température à bord, l'utilisation du module DS3231 est encore plus prometteuse que celle des autres modules.

VI. Initialisation

Lors de la première utilisation ou si on change la pile, l’heure et la date ne seront pas réglées. La figure suivante montre l’état d’horloge avant l’initialisation.

Figure 3.13 : RTC3231 avant l'initialisation.

Dans le code suivant, il est possible de régler la date en entrant, dans le moniteur série, la date désirée sous la forme YYMMDDwHHMMSSx. Une fois la date entrée, il faut redémarrer la carte afin qu'elle soit prise en compte par le module. Le code est donné en annexe 2.

On donne la date et l'heure exacte dans l'input de « Serial monitor » et on clique sur « SEND ».

Figure 3.14 : Serial monitor input.

Après le redémarrage de la carte Arduino, On constate que l'horloge est bien initialisée.

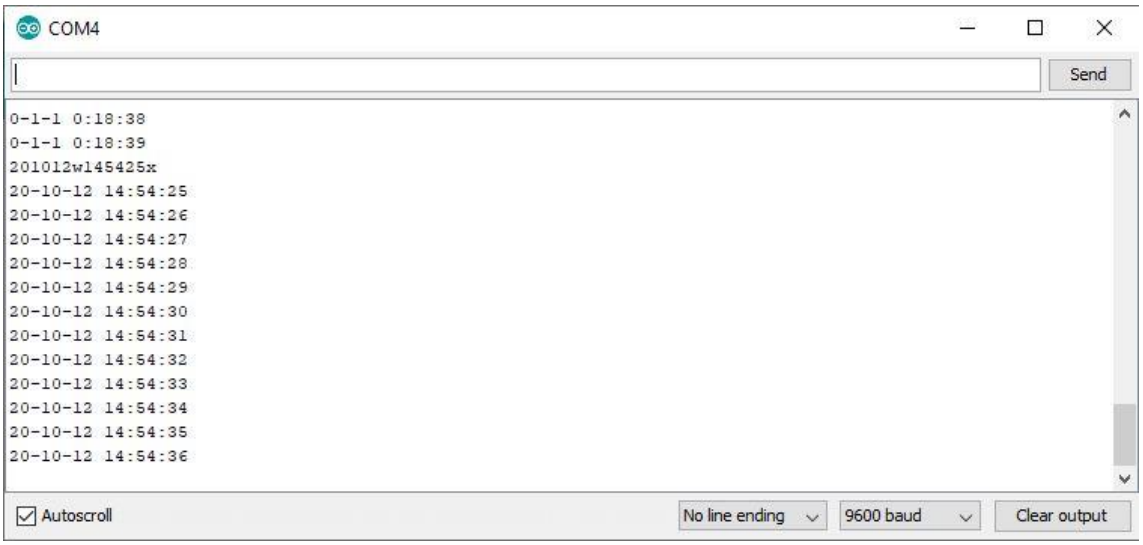

Figure 3.15 : état de RTC3231 après l'initialisation.

3.2.3 Bloc de stockage

a Module Carte SD

I. **Définition**

Le stockage des données est l'une des parties les plus importantes de chaque projet. Il existe plusieurs façons de stocker les données en fonction de leur type et de leur taille. Les cartes SD et micro SD sont l'un des dispositifs de stockage les plus pratiques.

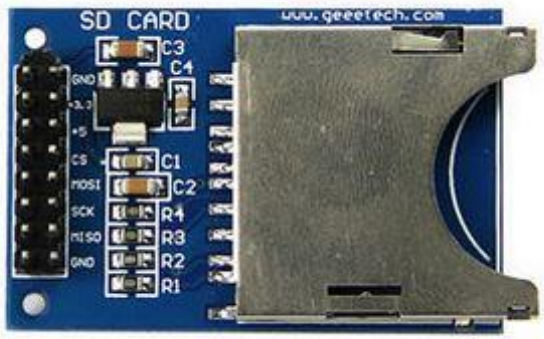

Figure 3.16 : carte SD module.

Les modules de cartes SD et micro SD vous permettent de communiquer avec la carte mémoire et d'écrire ou de lire les informations qu'elle contient. Le module s'interface dans le protocole SPI.

II. **Caractéristiques**

	MIN	TYPIQUE	MAX	UNITÉ
TENSION D'ALIMENTATION VCC	4.5	5	5.5	V
COURANT	0.2	80	200	mA
POTENTIEL ÉLECTRIQUE DE L'INTERFACE		3.3 ou 5		V

TYPE DE CARTE SUPPORTÉ	Micro carte SD (<=2go)	Micro carte SDHC (<=32go)	
DIMENSIONS	42x24x12		mm
POIDS	5		g

Tableau 3.3 : Les Caractéristiques de la carte sd.

III. **Brochage de la carte**

Le module de carte SD présente 6 broches pour permettre d'établir la connexion. 2 connexions pour l'alimentation et 4 pour établir la liaison SPI comme montré dans la figure ci-dessous :

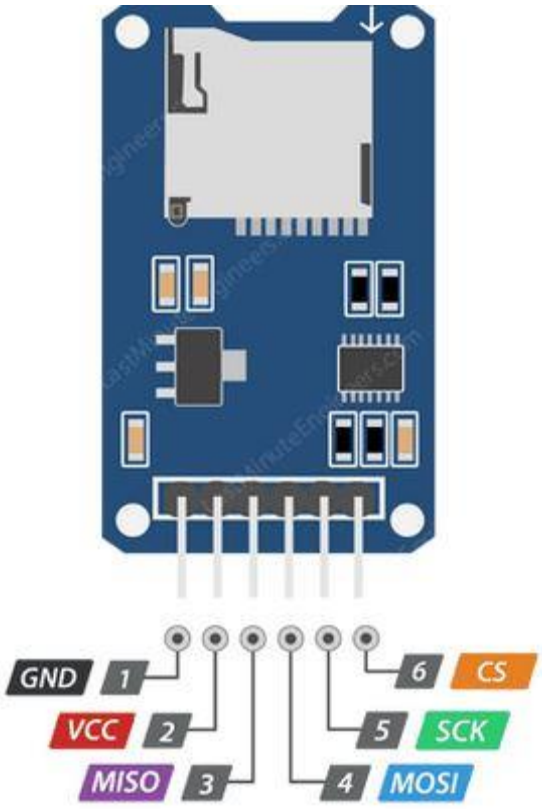

Figure 3.17 : Les pins du la carte.

- **GND** : la masse du module.
- **VCC :** 5V ou 3.3V pour l'alimentation du module.

- **MISO :** broche de transmission équivalent à la borne Tx d'un port série. Sortie du module.
- **MOSI :** broche de réception équivalent à la borne Rx d'un port série. Entrée du module.
- **SCK :** horloge permettant de synchroniser la communication.
- **CS :** (ChipSelect) pour activer la communication.

IV. Disposition du module

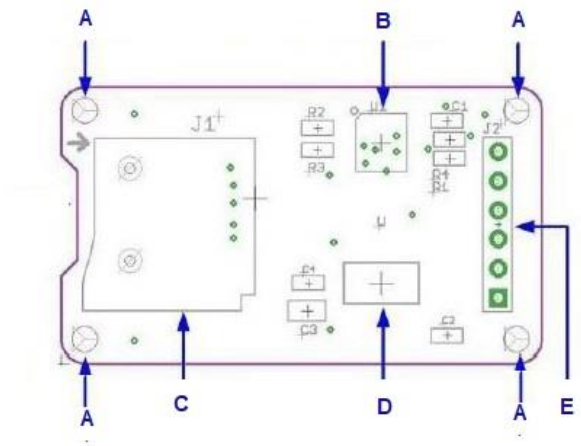

Figure 3.18 : Schéma structurelle du module.

ETIQUETTE	FONCTION
A	trou de positionnement
B	Circuit de conversion de niveau
C	support de carte micro SD
D	Circuit régulateur de tension 3.3v
E	L'interface de contrôle

Tableau 3.4 : les composants de la carte sd.

V. Préparation de la carte SD

La première étape de l'utilisation du module de carte SD avec Arduino consiste à formater la carte SD en FAT16 ou FAT32.

Pour formater la carte SD, on l'insère dans notre ordinateur. Allez dans Mon ordinateur et cliquez sur la carte SD avec le bouton droit de la souris. Sélectionnez Format comme indiqué dans la figure ci-dessous.

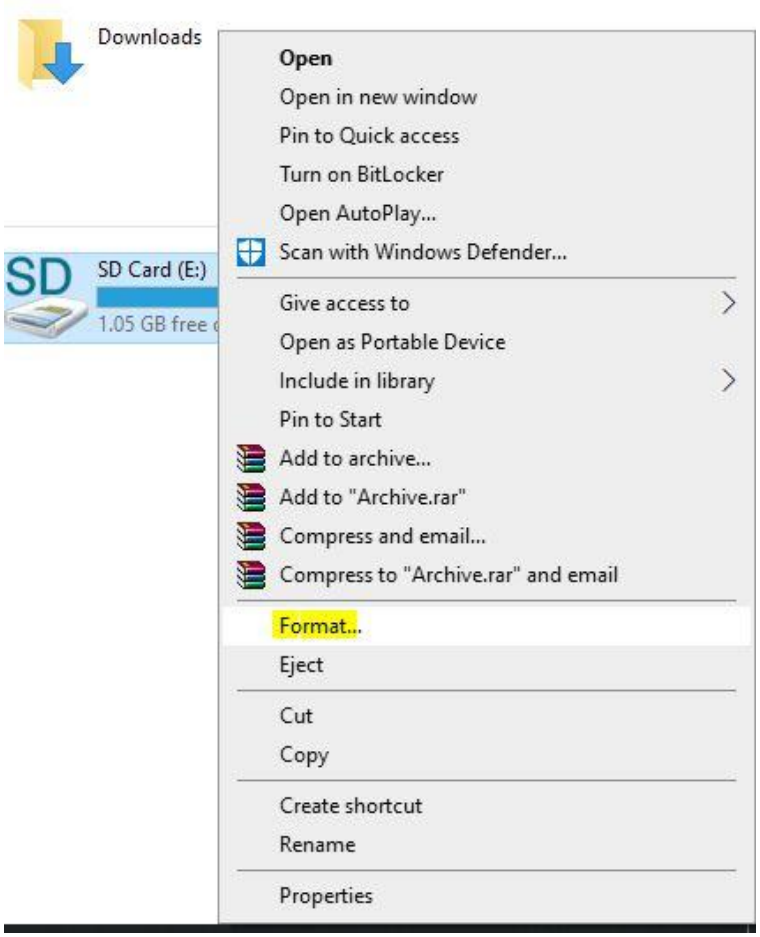

Figure 3.19 : menu de formatage.

Une nouvelle fenêtre s'ouvre. On sélectionne FAT32, on appuie sur Démarrer pour initialiser le processus de formatage et on suit les instructions à l'écran.

Figure 3.20 : initialisation de formatage.

Après le formatage de la carte SD, la fenêtre suivante apparaît

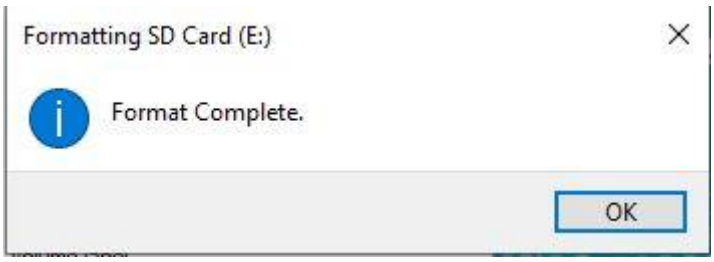

Figure 3.21 : formatage de la carte SD.

VI. Tester le module de la carte SD

On insère la carte SD formatée dans le module de la carte SD, on câble le module de la carte SD à l'Arduino en respectant les connexions pin, puis on connecte l'arduino au PC à l'aide du câble USB.

Pour nous assurer que tout est correctement câblé et que la carte SD fonctionne correctement, nous téléchargeons le code suivant sur notre carte Arduino. Tout en nous assurons que nous avons la bonne carte et le bon port COM sélectionné. Le code est donné en annexe 3.

Nous ouvrons le Serial Monitor à une vitesse de 9600 bauds et nous devrions voir les informations de notre carte SD.

Si tout fonctionne correctement, nous verrons le message suivant sur le moniteur série.

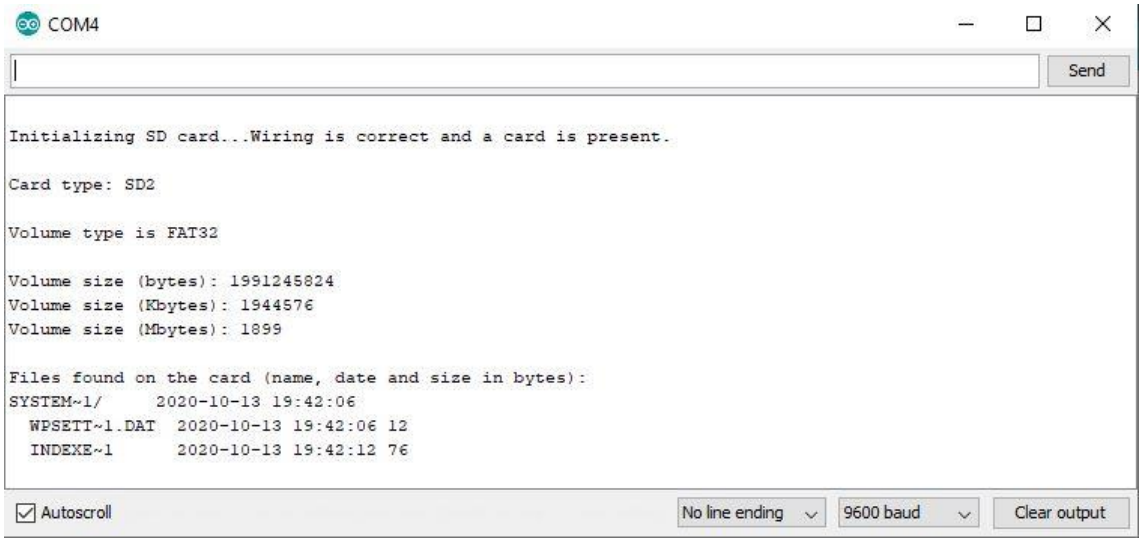

Figure 3.22 : texte sd card.

3.2.4 Bloc de Visualisation

a Ecran LCD 20x4

I. Définition

LCD signifie "Liquid Crystal Display" et se traduit, en français, par "Écran à Cristaux Liquides", Ces écrans sont utilisés dans plein d'appareils électroniques disposant d'afficheur : les montres, le tableau de bord de votre voiture, les calculatrices, etc. Cette utilisation intensive est due à leur faible consommation et coût.

Figure 3.23 : Afficheur LCD ADDICORE 20x4.

Cette technologie est bien maîtrisée et donc le coût de production est assez bas. Dans les années à venir, ils vont avoir tendance à être remplacés par les écrans à affichage LED qui sont pour le moment trop chers. Il existe deux types d'ecran LCD : les premiers sont monochromes (une seule couleur) tandis que les seconds sont colorés (rouge, vert et bleu).

Nous utiliserons uniquement le premier type dans notre projet pour des raisons de simplicité et de coût.

Il existe aussi plusieurs tailles d'écran, nous utiliserons l'écran LCD 20×4 car il s'agit essentiellement d'une version plus grande (nombre accru de lignes et de colonnes) de l'écran LCD 16×2, ce qui le rend parfait pour afficher une grande quantité de texte sans défilement. Chacune des colonnes a une résolution de 5×8 pixels, ce qui lui assure une visibilité à une distance importante.

II. Principe de fonctionnement

Comme son nom l'indique, un écran LCD possède des cristaux liquides. Si on regarde de très près un écran LCD on peut voir une grille de carré. Ces carrés sont appelés des pixels (Élément d'image). Lorsqu'aucun courant ne le traverse, ses molécules sont orientées dans un sens (0°). En revanche lorsqu'un courant le traverse, ses molécules vont se tourner dans la même direction (90°).

Figure 3.24 : Composition d'un écran LCD.

En effet, entre les cristaux liquides et la source lumineuse se trouve un filtre polariseur de lumière. Ce filtre va orienter la lumière dans une direction précise. Entre nos yeux et les cristaux se trouve un autre écran polariseur, qui est perpendiculaire au premier. Ainsi, il faut que les cristaux liquides soient dans la bonne direction pour que la lumière passe de bout en bout et revienne à nos yeux. Enfin, vient le rétro-éclairage (fait avec des LED) qui nous permettra de lire l'écran même en pleine nuit.

III. **Commande du LCD**

Pour pouvoir afficher des caractères sur l'écran il nous faudrait activer individuellement chaque pixel de l'écran. Un caractère est représenté par un bloc de 75 pixels. Ce qui fait qu'un écran de 16 colonnes et 2 lignes représente un total de 16275 = 1120 pixels, ce qui serait difficile, mais le décodeur de caractères va nous simplifier la tâche.

IV. Le décodeur de caractères

Ce composant va servir à décoder un ensemble "simple" de bits pour afficher un caractère à une position précise ou exécuter des commandes comme déplacer le curseur par exemple. Ce composant est fabriqué principalement par Hitachi et se nomme le **HC44780**. Il sert de décodeur de caractères. Ainsi, plutôt que de devoir multiplier les signaux pour commander les pixels un à un, il nous suffira d'envoyer des octets de commandes.

V. Brochage du LCD

Un LCD ADDICORE 20 colonnes 4 lignes possède 16 broches, leur rôle sera expliqué dans le tableau ci-dessous:

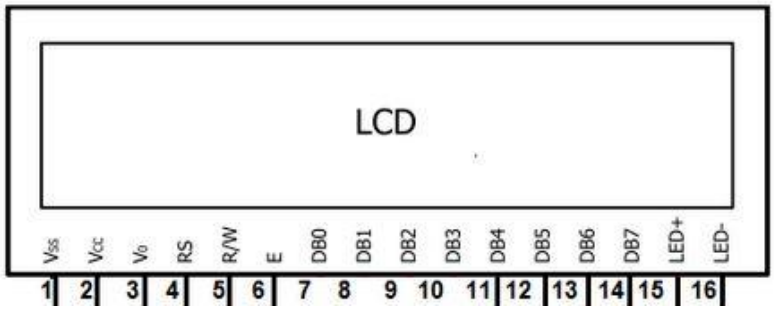

Figure 3.25 : les broches du LCD.

C	NOM	RÔLE
1	VSS	Masse
2	VDD	5V
3	Vo	Réglage du contraste
4	RS	Sélection du registre (commande ou donnée
5	R/W	Lecture ou écriture
6	E	Entrée de validation
7 - 14	Do – D7	Bits de données
15	A	Anode du rétroéclairage
16	K	Cathode du rétroéclairage

Tableau 3.5 : Liste des broches du LCD et leur rôle.

Ce composant possède tout le système de traitement pour afficher les caractères. Il contient dans sa mémoire le schéma d'allumage des pixels pour afficher chacun d'entre eux. Voici la table des caractères affichables :

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Figure 3.26 : La table ASCII du LCD.

3.2.5 Les LEDs et Buzzer

Nous avons également opté pour un système de notification par LED et Buzzer afin d'assurer une meilleure compréhension à l'utilisateur en facilitant l'interface.

a Les LEDs

Le terme "LED" signifie Light Emitting Diode, Diode électroluminescente en francais. Une LED est une diode qui émet de la lumière. Comme toute diode elle comporte une anode (borne +) qui est la patte la plus longue et une cathode (borne -) qui est la patte la plus courte.

Pour qu'une LED s'allume il faut obligatoirement relier sa patte - à la borne - de l'alimentation et sa patte + à la borne + de l'alimentation. Si la LED est branchée à l'envers elle ne s'allumera pas (le courant ne la traversera pas).

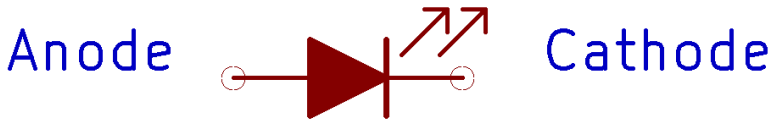

Figure 3.27 : Symbole de la LED.

b La résistance de protection d'une LED

Chaque LED accepte une certaine plage de tension d'alimentation. Il faudra donc placer avant la LED une résistance pour "consommer" le surplus de courant.

$$R = (U – Ul) / I$$

R : Valeur de la résistance en Ohm (Ω).

U : Tension de l'alimentation délivrée par l'Arduino.

Ul : Tension d'alimentation de la led (V).

I : Intensité de la led (A).

c Buzzer

Le buzzer est un composant constitué essentiellement d'une lamelle réagissant à l'effet piézoélectrique. La piézoélectricité est la propriété que possèdent certains minéraux de se déformer lorsqu'ils sont soumis à un champ électrique. Ce phénomène est réversible ; si nous déformons ce minéral, il produit de l'énergie électrique.
Dans l'univers Arduino, le buzzer est principalement utilisé pour émettre un son.

Figure 3.28 : Le Buzzer.

3.2.6 Bloc d'alimentation

L'alimentation indépendante de la carte arduino est effectuée par un adaptateur AC/DC, le modèle utilisé est le HJ-120100E.

Figure 3.29 : Adaptateur HJ-120100E.

Cet adaptateur peut être décrit comme un chargeur, il convertie un courant alternatif en un courant continu à faible tension il contient :

Un transformateur abaisseur : converti la tension alternative du secteur 220 v en une tension alternatif inferieur de 12 v et un courant de 1 A.

Un redresseur double alternance : aussi connue sous le nom de pont de graetz il fournit en sortie une tension alternative redressée.

Un condensateur de filtrage : le filtrage transforme la tension redressée en une tension aussi constante que possible, le composant chargé du filtrage est un condensateur ou plusieurs en parallèles, plus la capacité du condensateur est élevée plus le filtrage est mieux.

Un circuit stabilisateur ou régulateur : malgré utilisation d'un filtrage, la tension obtenue n'est pas pratiquement continue c'est pour cela qu'il est nécessaire d'ajouter un circuit intégré nommé régulateur de tension qui permet d'obtenir une tension continue.

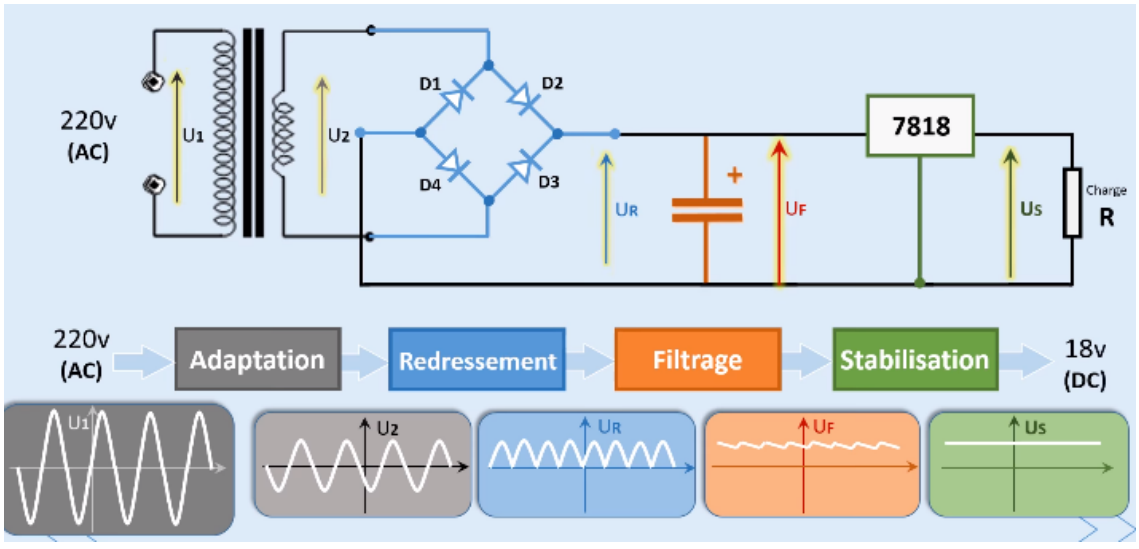

Figure 3.30 : Schéma d'alimentation stabilisé.

3.2.7 Bloc de traitement et de commande :

Le bloc de traitement et de commande, représente le cœur de notre travail il s'articulera autour de la carte de prototypage Arduino Uno, ce dernier a été détaillé précédemment dans le chapitre 2.

a Programme principal :

Le code complet de notre projet est donné en annexe. Nous expliquons ici certaines parties importantes du code. Dans ce programme, nous allons utiliser les bibliothèques suivantes :

« Wire.h » : Cette bibliothèque nous permet de communiquer avec les appareils I2C / TWI.

« DS3231.h » : Bibliothèque Arduino pour l'horloge en temps réel DS3231 (RTC)

« LiquidCrystal.h » : cette bibliothèque permet à une carte Arduino de contrôler des écrans à cristaux liquides (LCD) basés sur le chipset Hitachi HD44780 (ou un chipset compatible), que l'on trouve sur la plupart des LCD en mode texte. La bibliothèque fonctionne en mode 4 ou 8 bits.

« SPI.h » : Cette bibliothèque nous permet de communiquer avec les appareils SPI, avec l'Arduino comme appareil maître.

« SD.h » : La bibliothèque SD permet de lire et d'écrire sur des cartes SD. La bibliothèque supporte les systèmes de fichiers FAT16 et FAT32 sur les cartes SD standard et les cartes SDHC.

b Explication :

Nous commençons le programme en incluant les bibliothèques mentionnées précédemment et en définissant les noms de pin.

```
#define Buzz 7

#define Green 6

#define Red 5

#include <Wire.h>

#include "DS3231.h"

#include <LiquidCrystal.h>

#include <SPI.h>

#include <SD.h>
```

Nous déclarons ensuite nos variables, nous avons beaucoup de variables dans notre programme, mais les plus importantes sont celles que nous devons ajuster en fonction de notre demande.

```
float temp_calibration=0;     // ce paramètre a été utilisé pour calibrer la température

float offset_vol=0.05 ; //ce paramètre a été utilisé pour définir la tension de niveau moyen,

int Range_sensitivity = 200 ; //Diminuer cette valeur pour augmenter la portée du capteur TCRT5000
```

Nous avons alors la fonction qui permet d'ouvrir la communication SPI avec la carte SD et de créer un fichier à l'intérieur de la carte SD appelé "Temp_Log.text". Ensuite, nous allons également ouvrir ce fichier et écrire l'en-tête du journal. Ici, nous enregistrons la date, l'heure et la température et nous donnons un nom à l'en-tête en accord avec

```
void Initialize_SDcard()
{
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    return;
  }
  File dataFile = SD.open("Temp_Log.txt", FILE_WRITE);
  if (dataFile) {
    dataFile.println("Date,Time,Temperature");
    dataFile.close();   }
}
```

celui-ci. Cette fonction ne sera appelée qu'une seule fois lors de l'exécution de la fonction de configuration.

Ensuite, nous avons la fonction qui sert à enregistrer les valeurs réelles dans le fichier texte que nous venons de créer. Cette fonction sera appelée à chaque fois qu'une nouvelle lecture sera effectuée. Le format de l'enregistrement sera séparé par des virgules comme "Date,Heure,Température", un exemple d'enregistrement sera quelque chose comme "16.10.2020,10:45:17,35.6". Il peut être un peu difficile à lire, mais ce format est appelé CSV "valeurs séparées par des virgules" et peut être facilement ouvert et compris par un excel.

```
void Write_SDcard()
{
 File dataFile = SD.open("Temp_Log.txt", FILE_WRITE);
 if (dataFile) {
  dataFile.print(rtc.getDateStr());
  dataFile.print(",");
  dataFile.print(rtc.getTimeStr());
  dataFile.print(",");
  dataFile.print(temperature);
  dataFile.println();
  dataFile.close();
 }
 else
 Serial.println("OOPS!! SD card writing failed");
```

Les fonctions que nous avons ensuite sont des fonctions liées au capteur de température infrarouge, les fonctions sont utilisées pour calculer la température ambiante et la température de l'objet (la main)

```
float binSearch(long x) // cette fonction est utilisée pour mesurer la température ambiante

float arraysearch(float x,float y) // x est la température ambiante, y est la température de l'objet

float measureSurTemp() // cette fonction est utilisée pour convertir les valeurs de volts en degrés ceclius

float measureObjectTemp() // cette fonction est utilisée pour mesurer la température de l'objet, elle a besoin de la température ambiante mentionnée précédemment pour fonctionner correctement
```

Ensuite, nous avons la fonction "void setup" où nous initialisons l'appareil. Nous affichons juste un message d'introduction sur l'écran LCD et nous initialisons également la carte SD prête à l'emploi. Nous définissons également les broches des LED et du buzzer comme sorties

```
void setup() {
  Serial.begin(9600);
    rtc.begin();
  lcd.begin(16,2);
  lcd.print("Temp. Scanner");
  lcd.setCursor(0,1);
  lcd.print("Saad Dahleb Blida");
  pinMode(4,OUTPUT);
  pinMode(Buzz,OUTPUT);
  pinMode(Red,OUTPUT);
  pinMode(Green,OUTPUT);
  digitalWrite(Buzz,LOW);
  digitalWrite(Red,LOW);
  digitalWrite(Green,LOW);
  Initialize_SDcard(); }
```

Ensuite, à l'intérieur de la fonction Void Loop, nous allumons la broche numérique 2 qui est connectée à la LED de l'émetteur IR et nous effectuons la lecture analogique en A3 à laquelle la LED du récepteur IR est connectée. Nous répétons ensuite cette opération avec la LED IR éteinte. Cela nous aide à mesurer la valeur du bruit et du bruit+signal du capteur IR. Il nous suffit ensuite de soustraire la valeur du bruit de la valeur du bruit + signal pour obtenir la valeur du signal.

```
digitalWrite(4,HIGH);
 delayMicroseconds(500);
 Noise_P_Signal=analogRead(A3);
 digitalWrite(4,LOW);
 delayMicroseconds(500);
 Noise=analogRead(A3);
 Signal = Noise - Noise_P_Signal;
```

La valeur du signal nous indiquera à quel point la personne est proche du capteur IR TCRT5000 sans l'influence de la lumière du soleil autour de la personne. Ensuite, en comparant cette valeur de signal et la valeur de bruit, nous déclencherons notre thermomètre pour lire la valeur de température et la stocker également dans la carte SD. Si la température est normale, la LED verte s'allumera et si la température est élevée, la LED rouge s'allumera.

```
if (Signal>Range_sensitivity && Noise >500)
{
  digitalWrite(Buzz,HIGH);
 if (trigger == true)
  Serial.println("start");
  digitalWrite(4,LOW);
 measureSurTemp();
  temperature = (measureObjectTemp()) ;
  Serial.println(temperature,1);
  delay(150);

digitalWrite(Buzz,LOW);
 lcd.clear();
 lcd.setCursor(0,0);
 lcd.print("Temp:  ");
 lcd.print(temperature,1);
 lcd.setCursor(0,1);
 lcd.print("Saved to SD card");
 Write_SDcard();
```

3.2.8 Interface de control :

Pour créer une interface facile à lire sur le PC, nous avons opté pour l'interface graphique de MATLAB. Les figures suivantes montrent les démarches suivies :

Pour créer GUI Matlab → App → GUIDE.

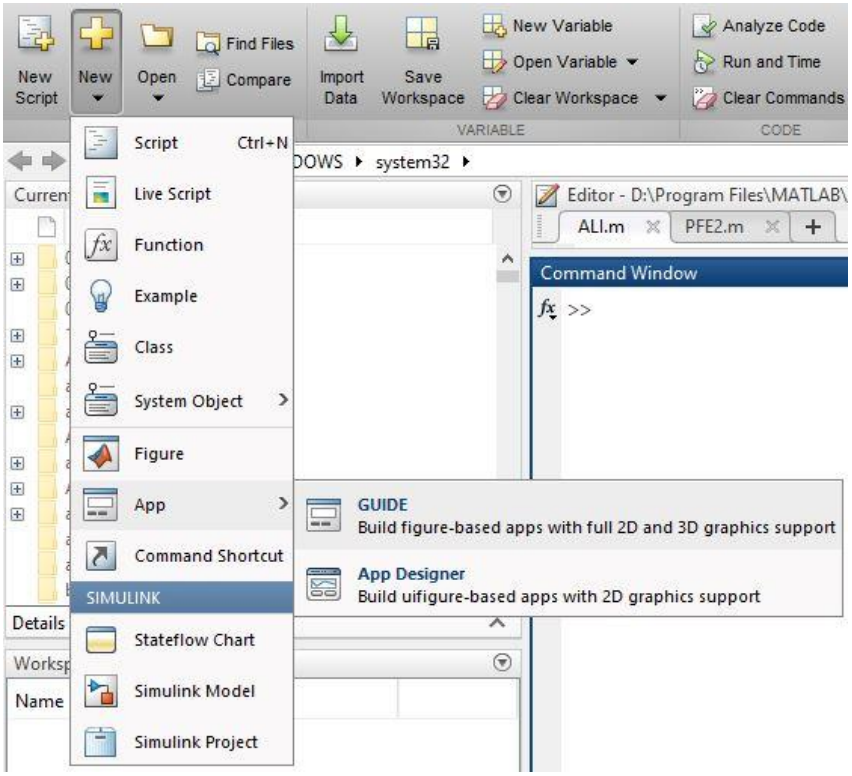

Figure 3.31 : Barre d'outil MATLAB.

A partir de la fenêtre qui apparaît → Blank GUI → OK.

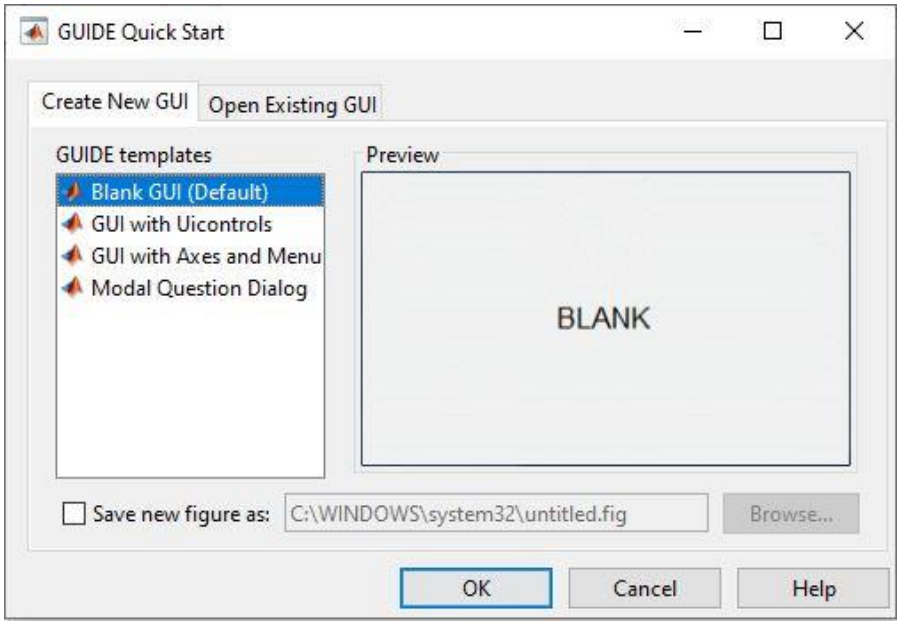

Figure 3.32 : Fenêtre GUIDE.

Une nouvelle fenêtre présente une interface graphique, nous pouvons créer l'interface que nous voulons à partir de la boîte à outils de gauche.

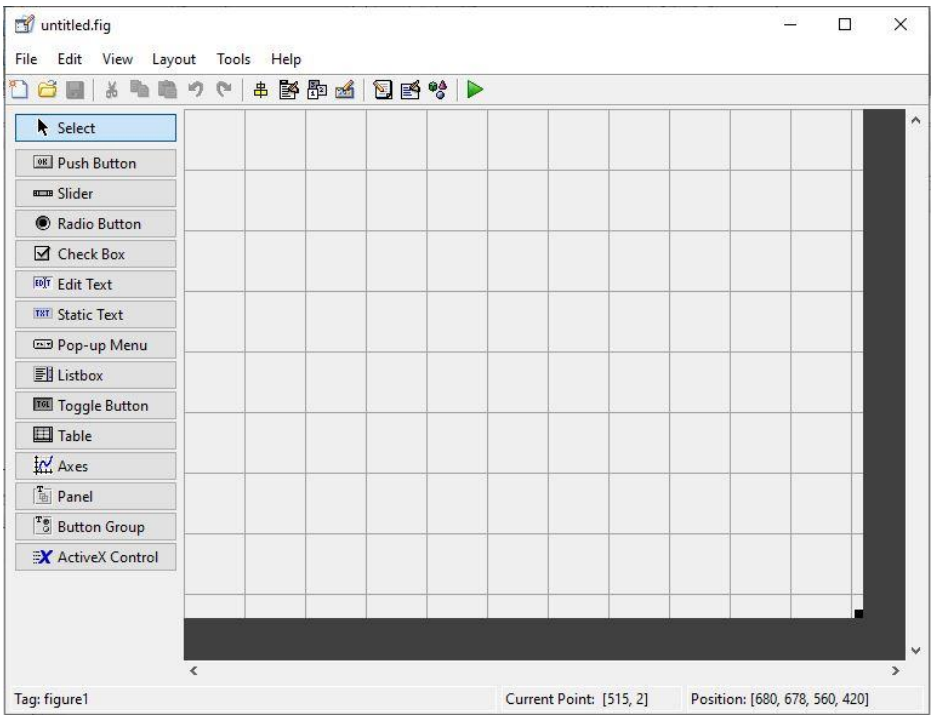

Figure 3.33 : Interface graphique vide.

Après avoir créé notre interface, on l'enregistre pour pouvoir la programmer.

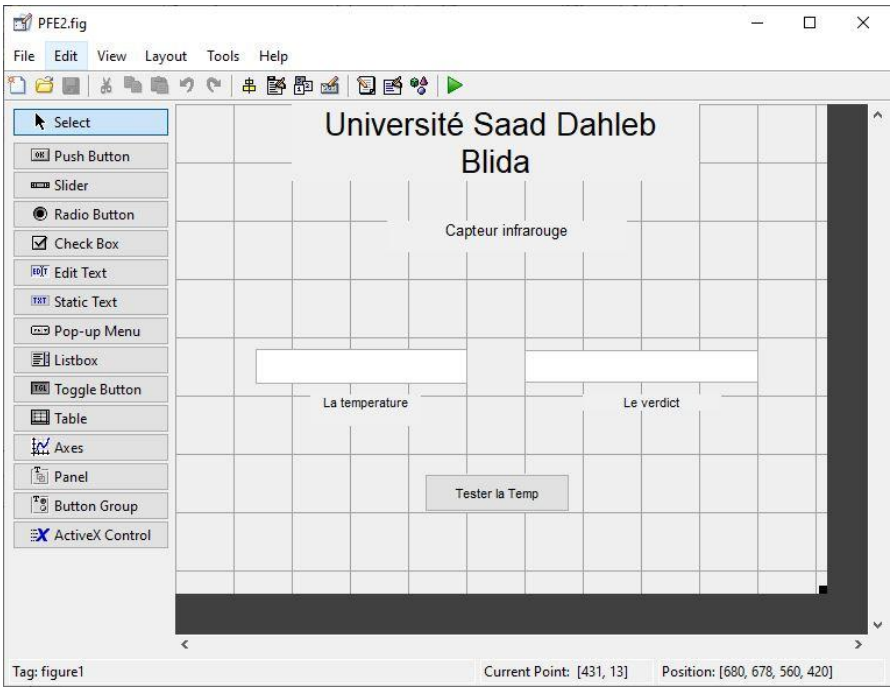

Figure 3.34 : GUI finale.

Notre interface fonctionne en cliquant sur le bouton "Tester la Temp. ", il faut donc le programmer par la suite : Click droit sur le bouton → View callbacks → callback.

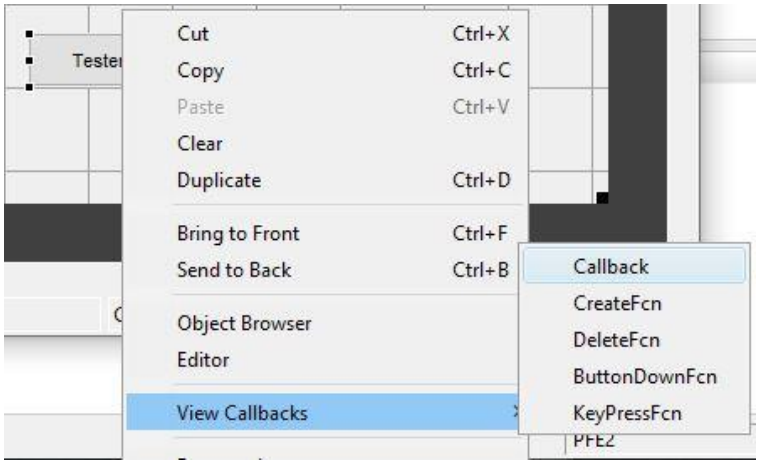

Figure 3.35 : Menu Callback.

Nous écrivons notre programme sous la fonction de bouton poussoir comme indiqué dans la figure suivante nous écrivons notre programme sous la fonction de bouton poussoir comme indiqué dans la figure suivante.

```
123   function pushbutton1_Callback(hObject, eventdata, handles)
124   % hObject    handle to pushbutton1 (see GCBO)
125   % eventdata  reserved - to be defined in a future version of MATLAB
126   % handles    structure with handles and user data (see GUIDATA)
127 -   delete(instrfindall);
128 -   s = serial('COM4');
129 -   fopen(s);
130 -   temp = str2num(fscanf(s))
131 -   set(handles.Temp,'String',temp);
132 -   if temp<38
133 -       set(handles.message,'String','normal');
134 -   else
135 -   set(handles.message,'String','cas suspect');
136 -   end
```

Figure 3.36 : Programme GUI.

Explication du programme :

`delete(instrfindall);` cette commande nous permettre de déconnecter et supprimer tous les objets « instruments », c'est-à-dire déconnecter la liaison sérial crée précédemment pour garantir une connexion plus fiable à chaque fois on clique sur le bouton.

`s = serial('COM4');` On initialise une connexion série à partir du câble USB inséré sur le port COM4 du PC.

`fopen(s);` on lance la communication en série pour lire les valeurs.

`temp = str2num(fscanf(s))` on lit la valeur envoyée par notre arduino et on la convertit en un chiffre et on le stock dans la case mémoire « temp ».

`set(handles.Temp,'String',temp);` on écrit la valeur dans la zone de texte avec taggée « Temp ».

```
if temp<38
    set(handles.message,'String','normal');
else
set(handles.message,'String','cas suspect');
end
```

Si la temperature est inferieure a 38, le verdict est « normal ». si non, si la temperature est supérieure, c'est un cas suspect.

L'application est démontrée dans les deux figures suivantes :

Figure 3.37 : démonstration d'un cas normal.

Figure 3.38 : démonstration d'un cas suspect.

3.3 Schéma général :

Après avoir décrit chaque élément le schéma global du projet est présenté sur les figures suivantes :

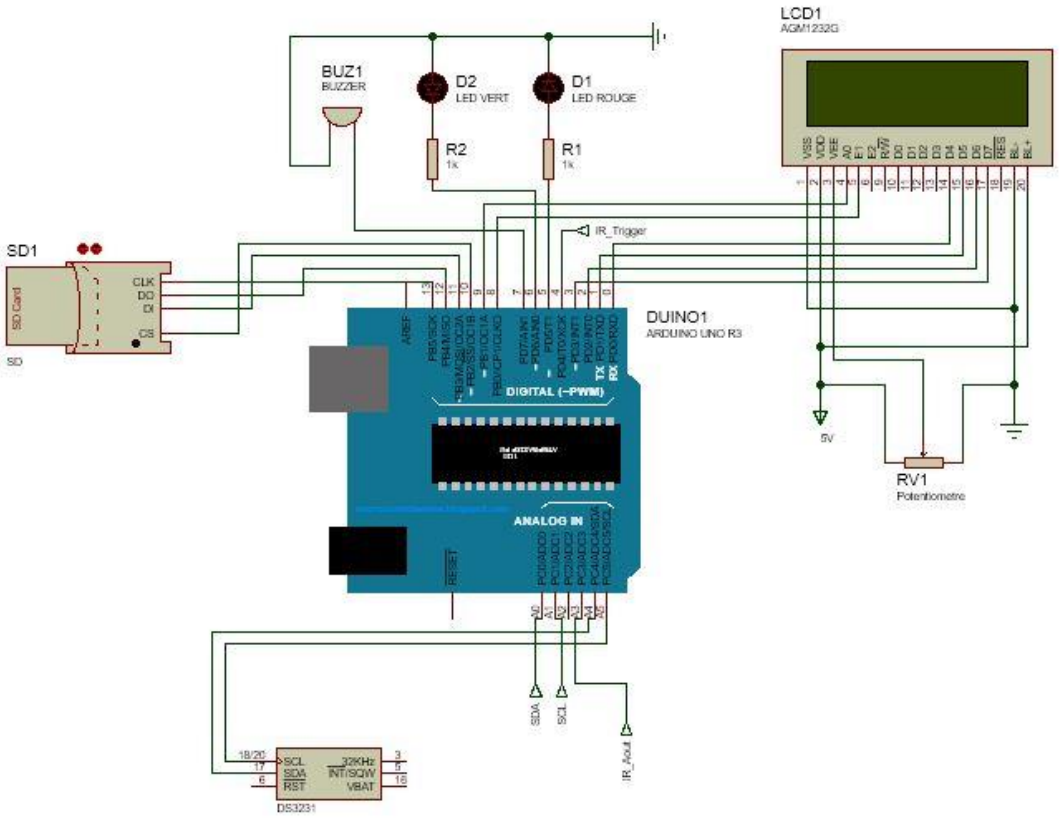

Figure 3.39 : Schéma général partie 1.

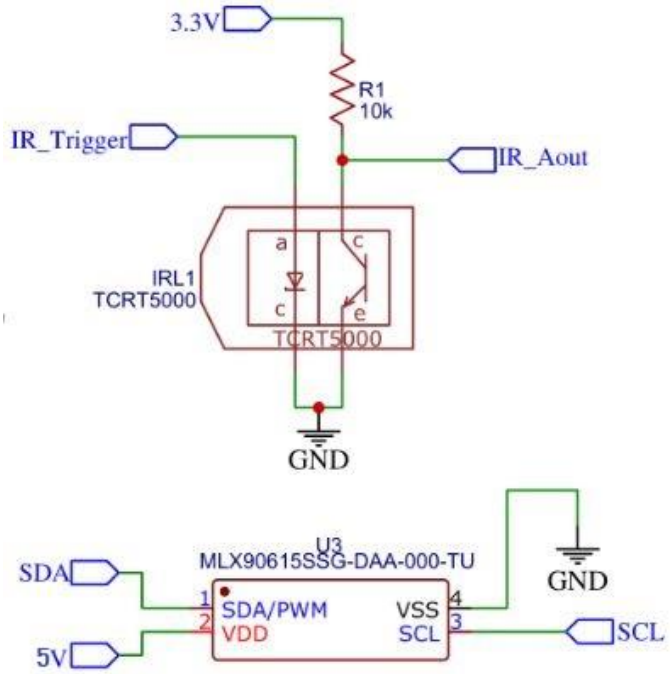

Figure 3.40 : Schéma général partie 2.

3.4 Réalisation :

Après avoir vu l'aspect théorique de notre projet, nous avons réalisé notre circuit sur une plaque d'essai comme montré dans la figure suivante :

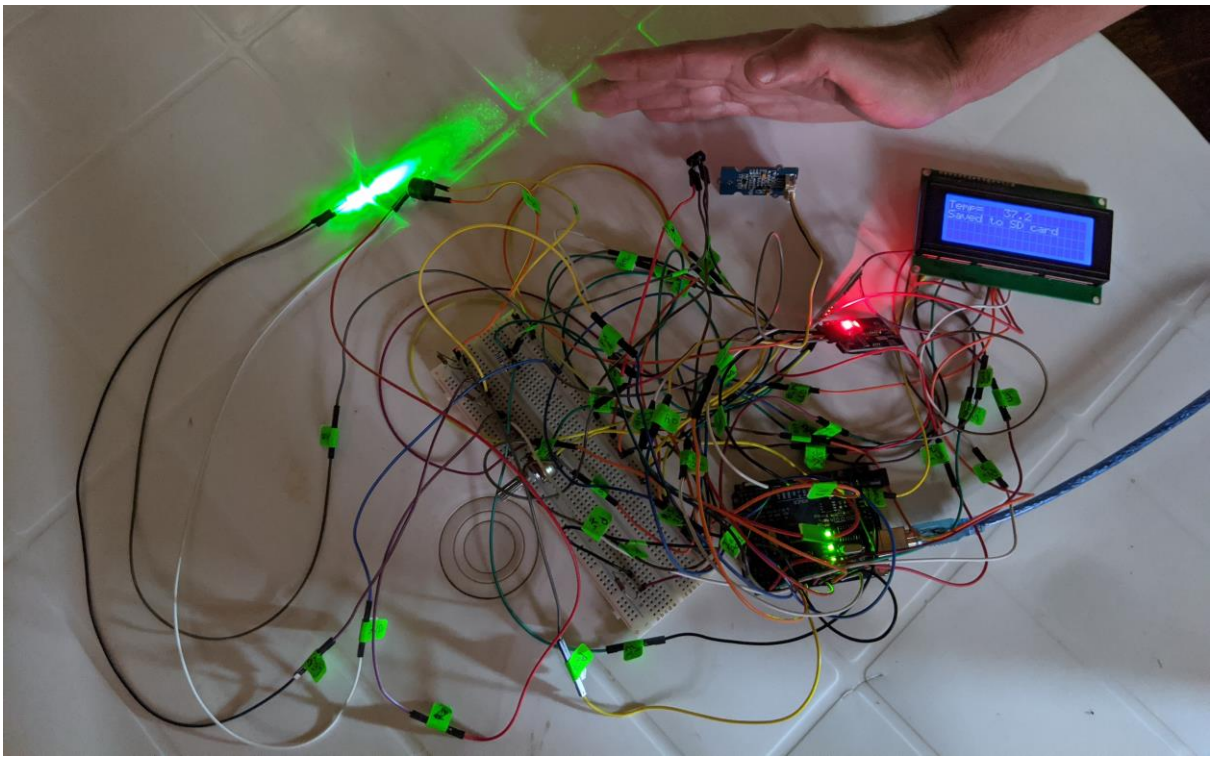

Figure 3.41 : Vue de haut du prototype.

Le brochage des pins utilisé est démontré par la suite :

Ecran LCD : E → pin 8 ; RS → pin 9 ; → DB4 → pin 5 ; DB5 → pin6 ; DB6 → pin 2 ;

DB7 → pin 3.

Module Carte SD: CS → pin 10 ; SCK → pin 13 ; MOSI → pin 11 ; MISO → pin 12.

Module RTC: SDA → A4 ; SCL → A5.

TCRT 5000: IR_Trigger → pin 4 ; IR_Aout → A3.

IR Température: OBJ → A1 ; SUR → A0.

LED Rouge → pin A2 ; **LED Verte** → Pin 7 ; **Buzzer** → pin 1 ;

Pour rendre notre projet plus pratique et professionnel, nous avons créé une boîte en aluminium qui permet de conserver tous nos composants à l'intérieur, nous avons utilisé un découpeur CNC pour couper l'interface de nos capteurs et de nos éléments de visualisation.

Les dimensions de notre boîte sont indiquées dans la figure suivante :

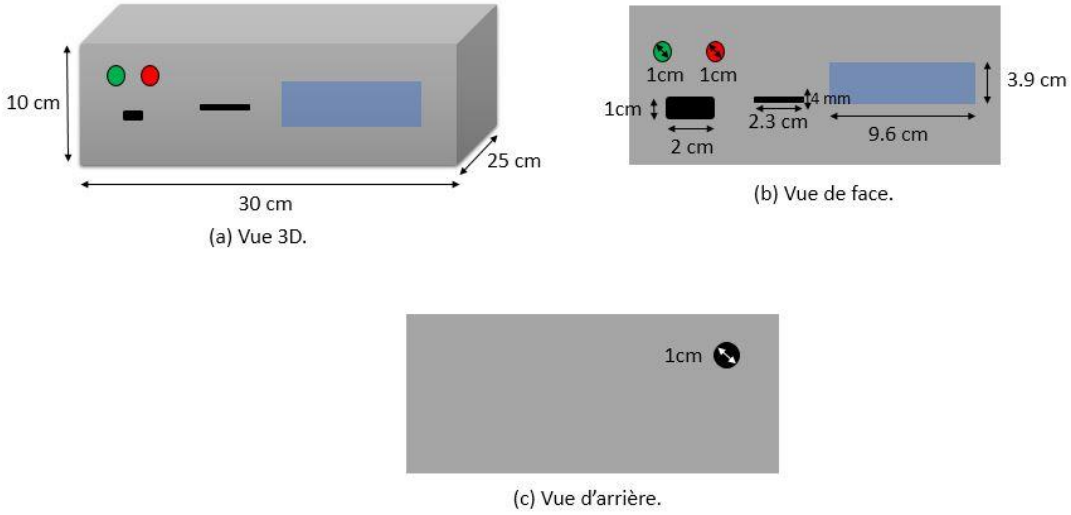

Figure 3.42 : Les dimensions de la boîte.

Après le découpage et le modelage, notre boîte ressemble à ce qui suit :

Figure 3.43 : Le boitier.

3.5 Tests et discussions :

Après de nombreux tests, nous avons pu calibrer correctement nos capteurs, et nous avons pu tester la fonctionnalité de notre prototype, nous avons pu enregistrer nos mesures de température dans « Serial monitor » de logiciel Arduino IDE comme montré dans la figure suivante:

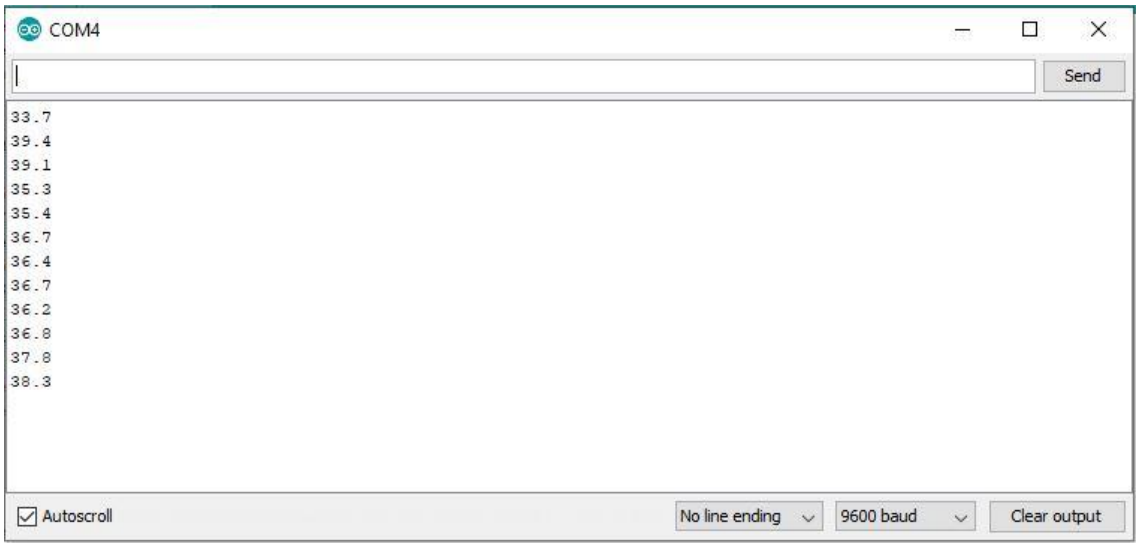

Figure 3.44 : Les tests dans serial monitor.

Les conditions ont été remplies, lorsque la température saisie dépasse 38 degrés Celsius, la LED rouge s'allume et le buzzer sonne, si la température est normale la LED verte s'allume. À la fin du test, nous avons pris la carte SD pour voir si l'enregistrement des données se fait correctement. Les résultats sont présentés sur la figure suivante :

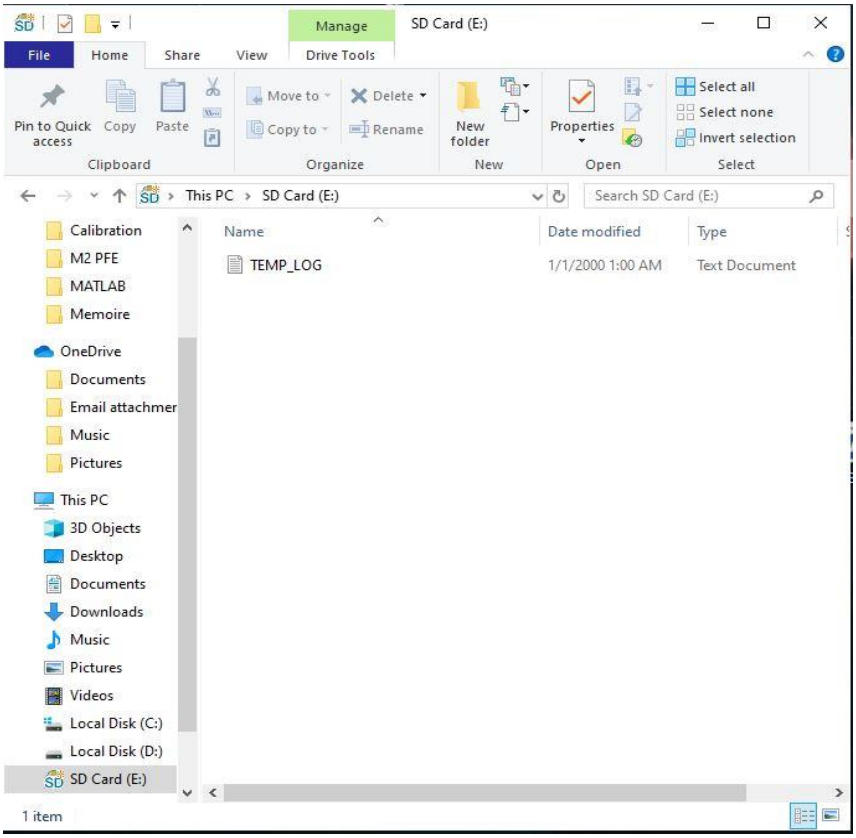

Figure 3.45 : La carte SD après la saisie des températures.

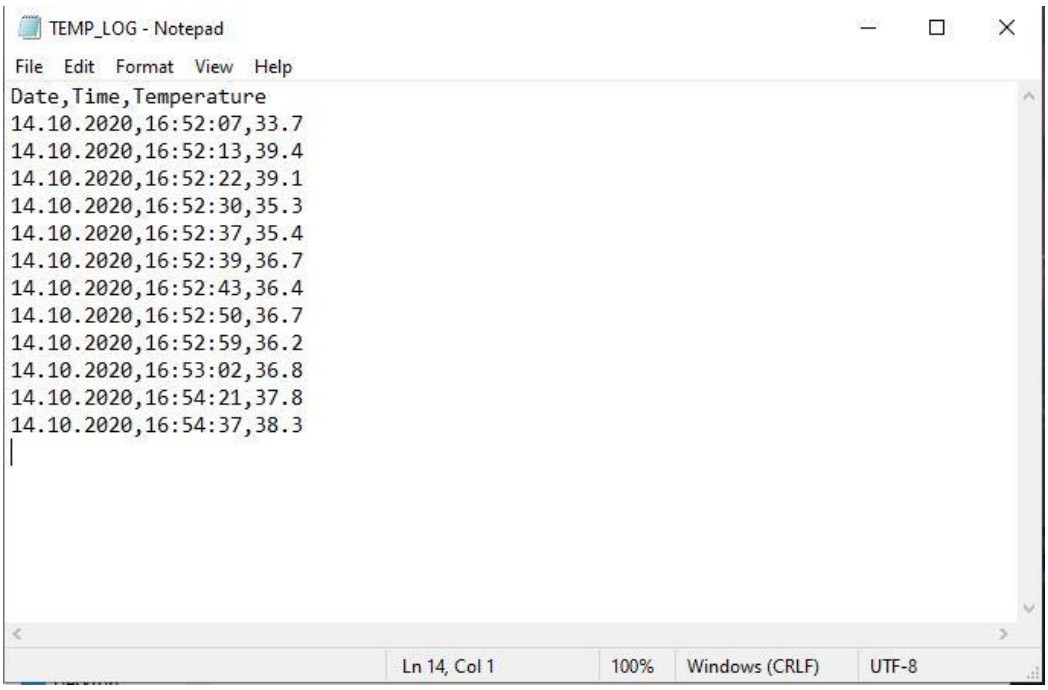

Figure 3.46 : Le contenu du fichier crée.

On constate que les températures ont été correctement enregistrées, et que l'heure et la date sont également enregistrées avec leur test de température respectif

Pour une meilleure visualisation et compréhension de nos données, nous pouvons utiliser Excel pour mieux les organiser.

Dans Excel → Data → From Text

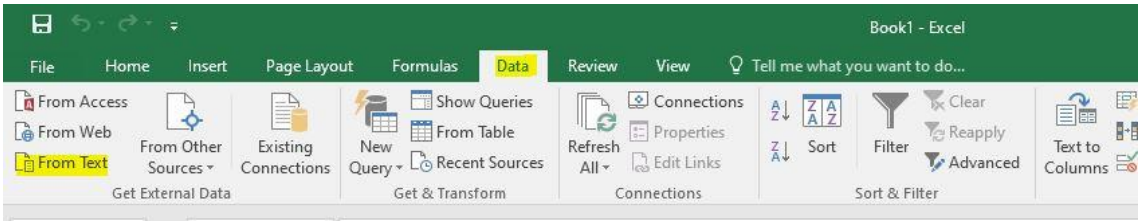

Figure 3.47 : La barre d'outil d'excel.

Dans la fenêtre qui s'ouvre, nous cherchons notre fichier qui est enregistré dans notre PC.

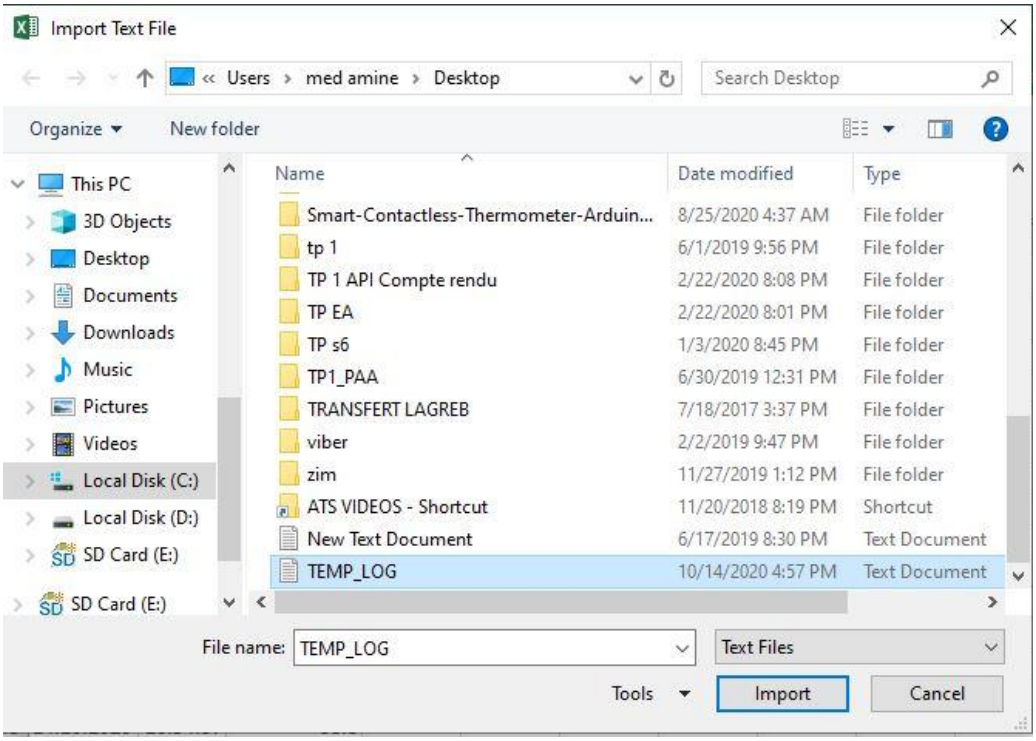

Figure 3.48 : Le répertoire de notre fichier TXT.

Dans la première étape de l'assistant d'importation, on sélectionne "Délimité" → on sélectionne que nos données ont des en-têtes (on coche la case) puis on clique sur suivant.

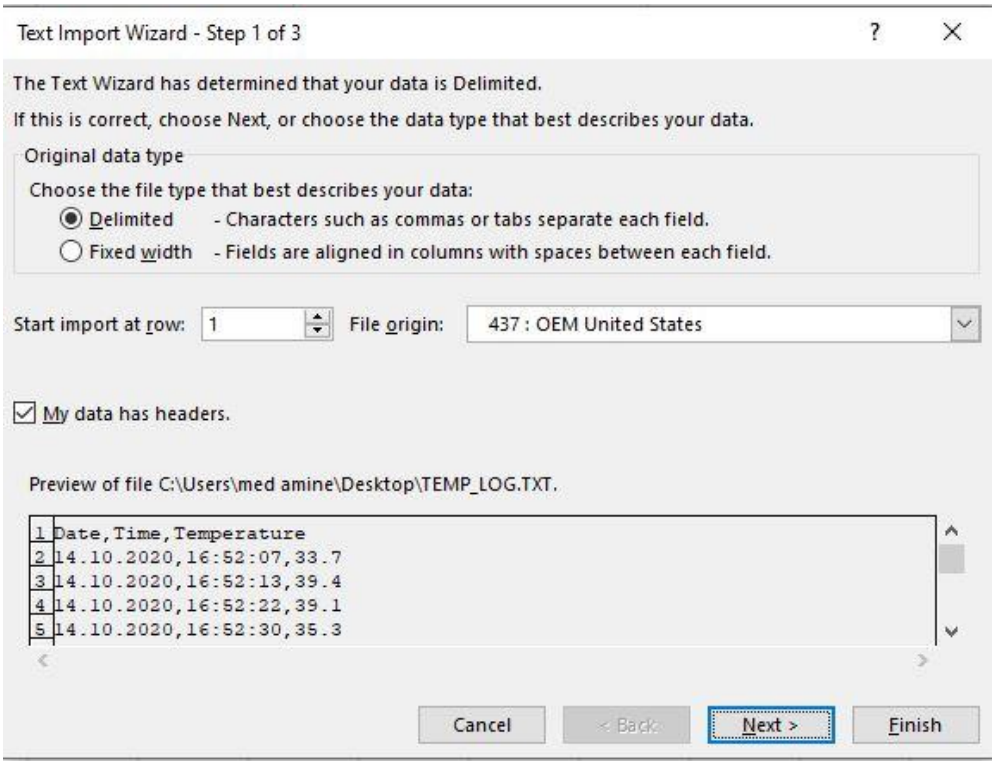

Figure 3.49 : La première étape de l'importation des données.

Dans la deuxième étape, nous sélectionnons la virgule comme délimiteur, nous pouvons voir que le texte change comme indiquer dans les figures suivantes et on clique sur finish:

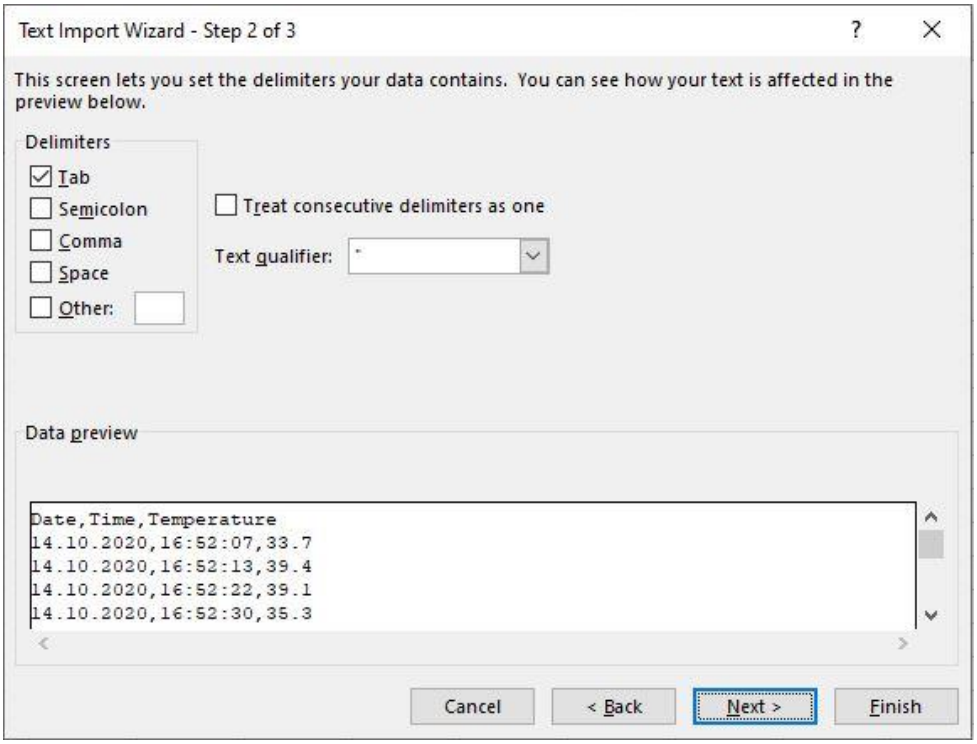

Figure 3.50 : Les données avant choisir « comma » comme délimiteur.

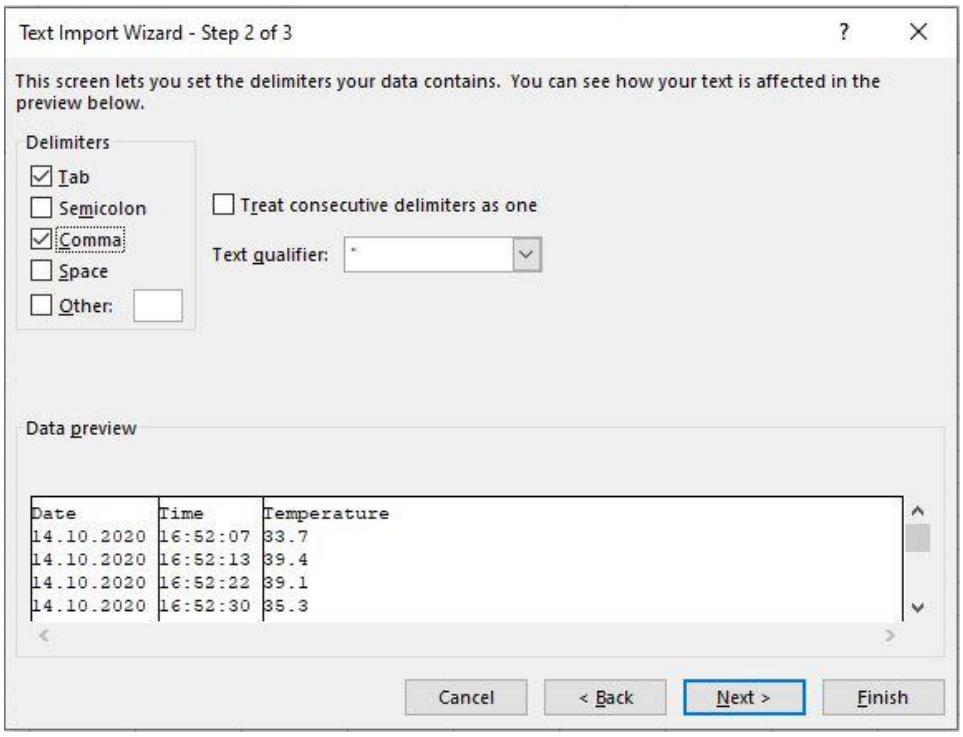

Figure 3.51 : Les données après choisir « comma » comme délimiteur.

Lorsqu'on a terminé, on peut constater que nos données sont présentées d'une manière plus pratique et plus facile à comprendre.

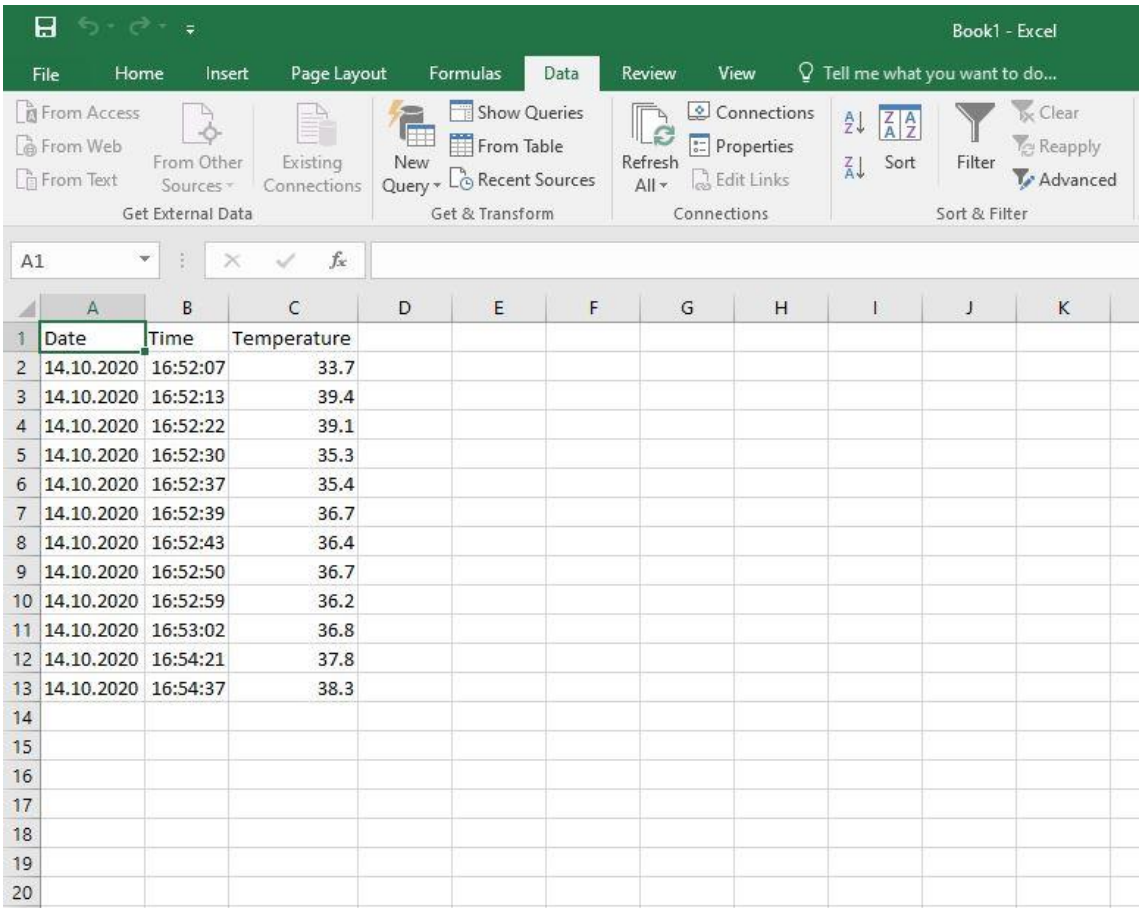

Figure 3.52 : les données après l'organisation.

3.6 Conclusion :

Dans ce chapitre, nous avons défini les différentes composantes que nous avons utilisées dans notre projet, nous avons pu mettre en lumière leurs différentes caractéristiques et leurs différents processus d'initialisation. Nous avons également donné le schéma général et la réalisation. Nous avons terminé le chapitre en effectuant différents tests et calibrages.

Conclusion générale

Dans ce mémoire, nous avons mis en lumière la pandémie mondiale causée par le COVID-19 et présenté les différents détails qui l'entourent. Nous avons également exploré les différentes possibilités de lutte contre ce virus, et comment le thermomètre thermique joue un rôle important dans cette lutte. Nous avons également pu définir la problématique de la faible disponibilité de ces thermomètres.

Après avoir défini les composantes importantes de nos projets, nous avons pu commencer à réaliser un thermomètre infrarouge basé sur la carte de prototypage Arduino, nous avons également ajouté plus de fonctionnalités en ajoutant des capacités de stockage tout en maintenant un point de vue économique et efficacité.

Annexes

Annexe 1 :

```
#include <math.h>

#define SUR_TEMP_PIN A0

#define OBJ_TEMP_PIN A1

float temp_calibration=0

//float objt_calibration=0.000;

float temperature_range=10;

float offset_vol=0.014;

float tempValue = 0;

float objtValue= 0;

float current_temp=0;

float temp=0;

float temp1=0;

float temp2=0;

unsigned int temp3=0;

const float reference_vol=0.500;

unsigned char clear_num=0;//when use lcd to display

float R=0;

float voltage=0;
```

```
long res[100]={


318300,302903,288329,274533,261471,249100,237381,226276,215750,2
05768,
196300,187316,178788,170691,163002,155700,148766,142183,135936,1
30012,
124400,119038,113928,109059,104420,100000,95788,91775,87950,8430
5,
80830,77517,74357,71342,68466,65720,63098,60595,58202,55916,
53730,51645,49652,47746,45924,44180,42511,40912,39380,37910,
36500,35155,33866,32631,31446,30311,29222,28177,27175,26213,
25290,24403,23554,22738,21955,21202,20479,19783,19115,18472,
17260,16688,16138,15608,15098,14608,14135,13680,13242,12819,
12412,12020,11642,11278,10926,10587,10260,9945,9641,9347,
9063,8789,8525,8270,8023,7785,7555,7333,7118,6911};

float obj [13][12]={

/*0*/          { 0,-0.274,-0.58,-0.922,-1.301,-1.721,-2.183,-2.691,-3.247,-
3.854,-4.516,-5.236},

/*1*/          { 0.271,0,-0.303,-0.642,-1.018,-1.434,-1.894,-2.398,-2.951,-
3.556,-4.215,-4.931},

/*2*/          { 0.567,0.3,0,-0.335,-0.708,-1.121,-1.577,-2.078,-2.628,-
3.229,-3.884,-4.597},

/*3*/          { 0.891,0.628,0.331,0,-0.369,-0.778,-1.23,-1.728,-2.274,-
2.871,-3.523,-4.232},

/*4*/          { 1.244,0.985,0.692,0.365,0,-0.405,-0.853,-1.347,-1.889,-
2.482,-3.13,-3.835},

/*5*/          { 1.628,1.372,1.084,0.761,0.401,0,-0.444,-0.933,-1.47,-2.059,-
2.702,-3.403},

/*6*/          { 2.043,1.792,1.509,1.191,0.835,0.439,0,-0.484,-1.017,-1.601,-
2.24,-2.936},

/*7*/          { 2.491,2.246,1.968,1.655,1.304,0.913,0.479,0,-0.528,-1.107,-
1.74,-2.431},

/*8*/          { 2.975,2.735,2.462,2.155,1.809,1.424,0.996,0.522,0,-0.573,-
1.201,-1.887},

/*9*/          { 3.495,3.261,2.994,2.692,2.353,1.974,1.552,1.084,0.568,0,-
0.622,-1.301},

/*10*/          {
4.053,3.825,3.565,3.27,2.937,2.564,2.148,1.687,1.177,0.616,0,-0.673},
```

```
void setup()
{       Serial.begin(9600);
        analogReference(INTERNAL);
}
void loop()
{
        measureSurTemp();
        measureObjectTemp();
}

float binSearch(long x)
{
        int low,mid,high;
        low=0;
        //mid=0;
        high=100;
        while (low<=high)
        {
                mid=(low+high)/2;
                if(x<res[mid])
                        low= mid+1;
                else (x>res[mid])
                        high=mid-1;
        }
        return mid;
}
```

```
float arraysearch(float x,float y) {
        int i=0;
        float tem_coefficient=100;
        i=(x/10)+1;
        voltage=(float)y/tem_coefficient;
        for(temp3=0;temp3<13;temp3++)
        {
                if((voltage>obj[temp3][i])&&(voltage<obj[temp3+1][i]))

                {       return temp3;           }       }   }
float measureSurTemp()
{       unsigned char i=0;
        float current_temp1=0;
        int signal=0;
        tempValue=0;
        for(i=0;i<10;i++)
        {       tempValue+= analogRead(SUR_TEMP_PIN);
                delay(10);                  }
        tempValue=tempValue/10;
        temp = tempValue*1.1/1023;
        R=2000000*temp/(2.50-temp);
        signal=binSearch(R);
        current_temp=signal-1+temp_calibration+(res[signal-1]-
R)/(res[signal-1]-res[signal]);
        Serial.print("Surrounding temperature:");
        Serial.print(current_temp);
        return current_temp;  }
```

```
float measureObjectTemp()
{       unsigned char i=0;
        unsigned char j=0;
        float sur_temp=0;
        unsigned int array_temp=0;
        float temp1,temp2;
        float final_temp=0;
        objtValue=0;
        for(i=0;i<10;i++)
        {       objtValue+= analogRead(OBJ_TEMP_PIN);
                delay(10);     }
        objtValue=objtValue/10;
        temp1=objtValue*1.1/1023;
        sur_temp=temp1-(reference_vol+offset_vol);
        Serial.print("\t Sensor voltage:");
        Serial.print(sur_temp,3);
        Serial.print("V");
        array_temp=arraysearch(current_temp,sur_temp*1000);
        temp2=current_temp;
        temp1=(temperature_range*voltage)/(obj[array_temp+1][(int)(te
mp2/10)+1]-obj[array_temp][(int)(temp2/10)+1]);
        final_temp=temp2+temp1;
        if((final_temp>100)||(final_temp<=-10))
                {       Serial.println ("\t out of range!");        }
        else
                {       Serial.print("\t object temperature:");
                        Serial.println(final_temp,2);              }     }
```

Annexe 2 :

```
#include <Wire.h>

#include <DS3231.h>

byte Year ;

byte Month ;

byte Date ;

byte DoW ;

byte Hour ;

byte Minute ;

byte Second ;

bool Century  = false;

bool h12 ;

bool PM ;

DS3231 Clock ;

void setup(){

Serial.begin(9600);

Serial.println(F("Initialize System"));

Wire.begin();

}

void loop(){

setDate();

readRTC();

}
```

```
void readRTC( ){
Serial.print(Clock.getYear(), DEC);
      Serial.print("-");
      Serial.print(Clock.getMonth(Century), DEC);
      Serial.print("-");
      Serial.print(Clock.getDate(), DEC);
      Serial.print(" ");
      Serial.print(Clock.getHour(h12, PM), DEC); //24-hr
      Serial.print(":");
      Serial.print(Clock.getMinute(), DEC);
      Serial.print(":");
      Serial.println(Clock.getSecond(), DEC);
      delay(1000);
    }
void setDate( ){
if (Serial.available()) {
   GetDateStuff(Year, Month, Date, DoW, Hour, Minute, Second);
   Clock.setClockMode(false);
   Clock.setSecond(Second);
   Clock.setMinute(Minute);
   Clock.setHour(Hour);
   Clock.setDate(Date);
   Clock.setMonth(Month);
   Clock.setYear(Year);
   Clock.setDoW(DoW);
   }  }
```

```
void GetDateStuff(byte& Year,byte& Month,byte& Day,byte& DoW,byte&
Hour,byte& Minute,byte& Second){/* function GetDateStuff */
 boolean GotString = false;
 char InChar;
 byte Temp1, Temp2;
 char InString[20];
 byte j=0;
 while (!GotString) {
  if (Serial.available()) {
    InChar = Serial.read();
    InString[j] = InChar;
    j += 1;
    if (InChar == 'x') {
      GotString = true;      }      }      }
 Serial.println(InString);
  Temp1 = (byte)InString[0] -48;
 Temp2 = (byte)InString[1] -48;
 Year = Temp1*10 + Temp2;
 Temp1 = (byte)InString[2] -48;
 Temp2 = (byte)InString[3] -48;
 Month = Temp1*10 + Temp2;
  Temp1 = (byte)InString[4] -48;
 Temp2 = (byte)InString[5] -48;
 Day = Temp1*10 + Temp2;
 DoW = (byte)InString[6] - 48;
  Temp1 = (byte)InString[7] -48;
 Temp2 = (byte)InString[8] -48;
 Hour = Temp1*10 + Temp2;
```

```
  Temp1 = (byte)InString[9] -48;
  Temp2 = (byte)InString[10] -48;
  Minute = Temp1*10 + Temp2;
  Temp1 = (byte)InString[11] -48;
  Temp2 = (byte)InString[12] -48;
  Second = Temp1*10 + Temp2;
}
```

Annexe 3 :

```
#include <SPI.h>
#include <SD.h>
Sd2Card card;
SdVolume volume;
SdFile root;
const int chipSelect = 10;
void setup() {
Serial.begin(9600);
  while (!Serial) {    ;   }
Serial.print("\nInitializing SD card...");
if (!card.init(SPI_HALF_SPEED, chipSelect)) {
    Serial.println("initialization failed. Things to check:");
    Serial.println("* is a card inserted?");
    Serial.println("* is your wiring correct?");
    Serial.println("* did you change the chipSelect pin to match your shield
or module?");
    return;  }
else  {     Serial.println("Wiring is correct and a card is present.");   }
```

```
Serial.print("\nCard type: ");
 switch (card.type()) {
  case SD_CARD_TYPE_SD1:
   Serial.println("SD1");
   break;
  case SD_CARD_TYPE_SD2:
   Serial.println("SD2");
   break;
  case SD_CARD_TYPE_SDHC:
   Serial.println("SDHC");
   break;
  default:
   Serial.println("Unknown");
 }
 if (!volume.init(card)) {
  Serial.println("Could not find FAT16/FAT32 partition.\nMake sure you've
formatted the card");
  return;
 }
 uint32_t volumesize;
 Serial.print("\nVolume type is FAT");
 Serial.println(volume.fatType(), DEC);
 Serial.println();
 volumesize = volume.blocksPerCluster();
 volumesize *= volume.clusterCount();
 volumesize *= 512;
 Serial.print("Volume size (bytes): ");
 Serial.println(volumesize);
 Serial.print("Volume size (Kbytes): ");
```

```
  volumesize /= 1024;
  Serial.println(volumesize);
  Serial.print("Volume size (Mbytes): ");
  volumesize /= 1024;
  Serial.println(volumesize);
  Serial.println("\nFiles found on the card (name, date and size in bytes): ");
  root.openRoot(volume);
  root.ls(LS_R | LS_DATE | LS_SIZE);
}
void loop(void) {
}
```

Annexe 4 :

```
#define SUR_TEMP_PIN A0
#define OBJ_TEMP_PIN A1
float temp_calibration=0;
float temperature_range=10;
float offset_vol=0.014;
float tempValue = 0;
float objtValue= 0;
float current_temp=0;
float temp=0;
float temp1=0;
float temp2=0;
unsigned int temp3=0;
const float reference_vol=0.500;
unsigned char clear_num=0;
float R=0;
float voltage=0;
long res[100]={
                318300,302903,288329,274533,261471,249100,237381,226276,215750,205768,
                196300,187316,178788,170691,163002,155700,148766,142183,135936,130012,
                124400,119038,113928,109059,104420,100000,95788,91775,87950,84305,
                80830,77517,74357,71342,68466,65720,63098,60595,58202,55916,
                53730,51645,49652,47746,45924,44180,42511,40912,39380,37910,
                36500,35155,33866,32631,31446,30311,29222,28177,27175,26213,
                25290,24403,23554,22738,21955,21202,20479,19783,19115,18472,
                17260,16688,16138,15608,15098,14608,14135,13680,13242,12819,
                12412,12020,11642,11278,10926,10587,10260,9945,9641,9347,
                9063,8789,8525,8270,8023,7785,7555,7333,7118,6911};


float obj [4][12]={

                { 0.567,0.3,0,-0.335,-0.708,-1.121,-1.577,-2.078,-2.628,-3.229,-3.884,-4.597},
/*3*/           { 0.891,0.628,0.331,0,-0.369,-0.778,-1.23,-1.728,-2.274,-2.871,-3.523,-4.232},
/*4*/           { 1.244,0.985,0.692,0.365,0,-0.405,-0.853,-1.347,-1.889,-2.482,-3.13,-3.835},
/*5*/           { 1.628,1.372,1.084,0.761,0.401,0,-0.444,-0.933,-1.47,-2.059,-2.702,-3.403},

};


int Range_sensitivity = 200;
#define Buzz 1
#define Green 7
#define Red A2
#include <Wire.h>
#include "DS3231.h"
#include <LiquidCrystal.h>
#include <SPI.h>
#include <SD.h>
DS3231  rtc(SDA, SCL);
LiquidCrystal lcd(9, 8, 5, 6, 2, 3);
const int chipSelect = 10;
int Noise;
int Signal;
int Noise_P_Signal;
boolean trigger = true;
float temperature;
float pvs_temperature;
```

```
void Initialize_SDcard()
{

  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    return;
  }

  File dataFile = SD.open("Temp_Log.txt", FILE_WRITE);

  if (dataFile) {
    dataFile.println("Date,Time,Temperature");
    dataFile.close();
  }
}
void Write_SDcard()
{

  File dataFile = SD.open("Temp_Log.txt", FILE_WRITE);

  if (dataFile) {
    dataFile.print(rtc.getDateStr());
    dataFile.print(",");
    dataFile.print(rtc.getTimeStr());
    dataFile.print(",");
    dataFile.print(temperature);
    dataFile.println();
    dataFile.close();
  }
  else
  Serial.println("OOPS!! SD card writing failed");  }
```

```
float binSearch(long x)
{
  int low,mid,high;
  low=0;
  //mid=0;
  high=100;
  while (low<=high)
  {
    mid=(low+high)/2;
    if(x<res[mid])
      low= mid+1;
    else//(x>res[mid])
      high=mid-1;
  }
  return mid;
}

float arraysearch(float x,float y)
{
  int i=0;
  float tem_coefficient=100;
  i=(x/10)+1;
  voltage=(float)y/tem_coefficient;
  for(temp3=0;temp3<13;temp3++)
  {
    if((voltage>obj[temp3][i])&&(voltage<obj[temp3+1][i]))
    {
      return temp3;
    }
  }
}
```

```
float measureSurTemp()
{
  unsigned char i=0;
  float current_temp1=0;
  int signal=0;
  tempValue=0;
  tempValue+= analogRead(SUR_TEMP_PIN);
  temp = tempValue*5/1023;
  R=2000000*temp/(2.50-temp);
  signal=binSearch(R);
  current_temp=signal-1+temp_calibration+(res[signal-1]-R)/(res[signal-1]-res[signal]);

  return current_temp;    }

float measureObjectTemp()
{
  unsigned char i=0;
  unsigned char j=0;
  float sur_temp=0;
  unsigned int array_temp=0;
  float temp1,temp2;
  float final_temp=0;
  objtValue=0;
  objtValue+= analogRead(OBJ_TEMP_PIN);
  temp1=objtValue*5/1023;//+objt_calibration;
  sur_temp=temp1-(reference_vol+offset_vol);
  array_temp=arraysearch(current_temp,sur_temp*1000);
  temp2=current_temp;
  temp1=(temperature_range*voltage)/(obj[array_temp+1][(int)(temp2/10)+1]-obj[array_temp][(int)(temp2/10)+1]);
  final_temp=temp2+temp1;
    return final_temp;      }
```

```
void setup() {
  Serial.begin(9600);
  rtc.begin();
  lcd.begin(20, 4);
  lcd.print("Capteur Temp.");
  lcd.setCursor(0,1);
  lcd.print("ACHLAF Aymen");
  lcd.setCursor(0,2);
  lcd.print("LAGREB Ali Hichem");
  pinMode(4,OUTPUT);
  pinMode(Buzz,OUTPUT);
  pinMode(Red,OUTPUT);
  pinMode(Green,OUTPUT);
  digitalWrite(Buzz,LOW);
  digitalWrite(Red,LOW);
  digitalWrite(Green,LOW);
  Initialize_SDcard();
}
void loop() {
 lcd.setCursor(0,0);
 lcd.print("Time:  ");
 lcd.print(rtc.getTimeStr());
 lcd.setCursor(0,1);
 lcd.print("Date: ");
 lcd.print(rtc.getDateStr());
 lcd.setCursor(0,2);
 lcd.print("Saad Dahleb Blida ");
  digitalWrite(4,HIGH);
  delayMicroseconds(500);
  Noise_P_Signal=analogRead(A3);
  digitalWrite(4,LOW);
```

```
  delayMicroseconds(500);
  Noise=analogRead(A3);
  if (Signal>Range_sensitivity && Noise >500)
{
   digitalWrite(Buzz,HIGH);
  if (trigger == true)
   Serial.println("start");
   digitalWrite(4,LOW);
   for (int i=1; i<=5; i++)
   {
   temperature = (measureObjectTemp()+23 );
   Serial.println(temperature,1);
   delay(150);
   }
digitalWrite(Buzz,LOW);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temp:  ");
  lcd.print(temperature,1);
  lcd.setCursor(0,1);
  lcd.print("Saved to SD card");
 Write_SDcard();
  if (temperature>38)
  {
  digitalWrite(Red,HIGH);
  digitalWrite(Buzz,HIGH);
  delay (5000);
  }
  else
  {
  digitalWrite(Green,HIGH);
```

```
,
else
{
  delay(100);
  trigger = true;
  digitalWrite(13,LOW);
  Serial.println("position_error");
}
digitalWrite(Red,LOW);
digitalWrite(Buzz,LOW);
digitalWrite(Green,LOW);
}
```

Bibliographie

[1] https://www.theguardian.com/world/2020/feb/04/coronavirus-quarantine-precautions-around-the-world (Consulté le 20 Septembre 2020)

[2] https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200219-sitrep-30-covid-19.pdf?sfvrsn=3346b04f_2 (Consulté le 20 Septembre 2020)

[3] https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200226-sitrep-37-covid-19.pdf?sfvrsn=2146841e_2 (Consulté le 20 Septembre 2020)

[4] https://www.who.int/fr/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/q-a-coronaviruses#:~:text=symptomes (Consulté le 21 Septembre 2020)

[5] https://time.com/5878732/covid-19-testing-delays/ (Consulté le 21 Septembre 2020)

[6] https://www.nejm.org/doi/full/10.1056/NEJMp2015897 (Consulté le 21 Septembre 2020)

[7] https://www.sciencemag.org/news/2020/05/coronavirus-antigen-tests-quick-and-cheap-too-often-wrong (Consulté le 21 Septembre 2020)

[8] https://www.20minutes.fr/paris/2779099-20200514-coronavirus-cameras-thermiques-installees-aeroport-roissy-detecter-covid-19 (Consulté le 21 Septembre 2020)

[9] « Novel coronavirus is putting the whole world on alert», Journal of Hospital Infection, Volume 104 , issue 3, Mars 2020 , 252-253.

[10] M Beaurepaire - La Météorologie, 1995 - documents.irevues.inist.fr

[11] O Rodríguez-Musso - Museum International (Edition Francaise), 1990 - Wiley Online Library

[12] https://www.pce-instruments.com/f/french/media/thermometre-infrarouge-mesure-temperature-sans-contact.pdf

[13] https://fr.wikipedia.org/wiki/Thermom%C3%A8tre_infrarouge (Consulté le 19 Septembre 2020)

[14] https://www.airport-technology.com/features/coronavirus-screening-at-airports/ (Consulté le 21 Septembre 2020)

[15] PSCAL MASSON « Electronique avec Arduino », École Polytechnique Universitaire de Nice Sophia-Antipolis, Edition 2015-2016-V32

[16] ERIC BARTMANN « LE GRAND LIVRE D'ARDUINO » EYROLLES 2015

[17] https://www.arduino.cc/

[18] https://files.seeedstudio.com/wiki/Grove-Infrared_Temperature_Sensor/res/OTP-538Udatasheet.zip (Capteur infrarouge DataSheet)

[19] https://www.vishay.com/docs/83760/tcrt5000.pdf (TCRT5000 DataSheet)

[20] https://datasheets.maximintegrated.com/en/ds/DS3231.pdf (DS3231 DataSheet)

[21] http://datalogger.pbworks.com/w/file/fetch/89507207/Datalogger%20-%20SD%20Memory%20Reader%20Datasheet.pdf (Module Carte SD DataSheet)

[22] http://www.systronix.com/access/Systronix_20x4_lcd_brief_data.pdf (LCD 20x4 DataSheet)