PROJET – PEIP2

Année scolaire 2020-2021

Un réveil intelligent avec Arduino

Etudiants : MARTINS Naimé
PROT Raphaelle

Encadrants : MASSON Pascal

Ecole Polytechnique Universitaire de Nice Sophia-Antipolis, Département électronique
1645 route des Lucioles, Parc de Sophia Antipolis, 06410 BIOT

SOMMAIRE

I Introduction

Nous avons décidé de concevoir un réveil intelligent grâce à une carte Arduino. Afin d'offrir un réveil plus agréable et naturel, celui-ci simule l'aube en projetant une lumière. Le réveil possède deux boutons, un permettant d'activer le mode nuit (éteindre l'écran du réveil) et un permettant à l'utilisateur d'arrêter la sonnerie à son réveil.
Le réveil est contrôlable depuis le téléphone grâce à une application. L'utilisateur peut y choisir la musique de la sonnerie parmi une liste enregistrée, l'heure du réveil, l'activation ou non du simulateur d'aube.
Le réveil est également constitué d'un écran sur lequel peuvent être affichés la température intérieure, la météorologie de la journée et les rendez-vous de l'utilisateur si celui si le désire.

II L'horloge du réveil

Le réveil doit pouvoir sonner au bon jour et à la bonne heure choisis par l'utilisateur. Pour cela il faut que l'Arduino ait « une notion de temps » afin qu'elle compare le jour et l'heure actuels aux jour et l'heure choisis. Une fois correspondance établie le son sera activé (et de même pour la lumière). Cette « horloge interne » est donc essentiel dans le projet.

i. Divers systèmes

- Les modules temps réel (RTC, Real Time Clock)

Un module d'horloge en temps réel est généralement équipé d'un oscillateur à cristaux de quartz qui permet de mesurer le temps, grâce à une fréquence de 32,768kHz. Cette fréquence, représentant 2^{15} cycles par seconde, permet au module de garder le timing.
Cette fréquence assez basse permet de ne pas consommer trop d'énergie mais reste au-dessus de la portée de l'audition humaine.
Lorsque l'Arduino est connecté au téléphone (via le Bluetooth), le téléphone lui transmet l'heure, le jour et la semaine actuels. Ces données sont stockées dans la mémoire EEPROM (Electrically-Erasable Progammable Read-Only Memory) de l'Arduino. Le module prend ces informations pour régler son heure et est ensuite autonome. Lorsque le téléphone se déconnecte, le module RTC continue de compter le temps.
Il possède également une pile permettant de garder en mémoire cette mesure et de continuer à compter si l'alimentation principale venait à être éteinte.
Le module utilise la communication I^2C et les librairies Wire ou Rtc.

- Les réglages manuels

Il est possible de ne pas utiliser d'autres modules. Pour cela il faut donner la possibilité à l'utilisateur de régler l'heure chaque fois qu'il allume son réveil pour s'en servir. On peut ensuite demander à l'Arduino d'utiliser millis() pour compter le temps jusqu'à la sonnerie. Millis() se réinitialise à 0 au bout de 49 jours (et 70 heures pour micro()).Mais dans ce cas l'horloge n'est pas entièrement indépendante.

Une autre façon est d'initialiser l'heure à chaque fois que le réveil est allumé, puis d'utiliser des delay(1000) et réactualiser l'affichage de l'heure en ajoutant une seconde en plus. Ensuite il faut comparer l'heure affichée à l'heure demandée par l'utilisateur.
Un des désavantages à noter est le fait que cette méthode encombre le système principal et peut empêcher ou ralentir d'autres tâches.

- Récupérer l'heure depuis internet grâce aux modules tels que l'ESP8266 ou un Ethernet Shield

Le concept est le suivant : communiquer avec un serveur NTP pour récupérer par le wifi l'heure en continue. Il faudra alors initialiser l'« offset time » qui correspond à notre heure de décalage par rapport à Greenwich en secondes. Par exemple : utc+x = x*60*60, donc offset time = x*60*60 (utc= coordinated universal time).
Le serveur renvoie une chaine de 48 bits et est convertie en format Unix Time qui est lisible pour l'utilisateur. Ce protocole fonctionnera sans erreur jusqu'en 2038 puis il y aura un changement d'époque qui entraînera des modifications dans le programme.
Cette méthode utilise la bibliothèque NTP et requière une connexion constante au wifi.

- GPS (Global Positioning System)

Le GPS est un système de navigation basé sur au moins 24 satellites. Il fonctionne en continue. Le GPS reçoit des signaux des satellites et en les décodant il récupère les données de positions.
Grâce à des Shields GPS basés sur des modules tels que le module Vincotech A1080-B qui utilise les protocoles NMEA et SIRF III. Le module est connecté à la carte Arduino ainsi qu'à une antenne GPS. Il permet de transmettre les données de position via un port série mais également l'heure et le jour. C'est donc une autre méthode possible mais à privilégier si un module de GPS est déjà utilisé dans le montage.

Etant donné que nous souhaitons créer un réveil autonome il est préférable de ne pas imposer à l'utilisateur de devoir régler l'heure à chaque fois qu'il allume son réveil. De plus il serait plus agréable que le réveil fonctionne indépendamment d'internet au moins pour sa fonction première qui est de sonner à une heure donnée. Il parait donc préférable de choisir un module RTC pour régler l'horloge de l'Arduino.
Il existe plusieurs modules RTC pour Arduino.

ii. Les modules RTC

- Le DS3231

Le composant est une horloge en temps réel qui contient du quartz.
Mais le problème avec ces cristaux est que la température extérieure peut affecter leur fréquence d'oscillation. Ce changement de fréquence peut être négligeable, mais il s'additionne sûrement.
Pour éviter de telles légères dérives dans le cristal, DS3231 est réglé grâce à un oscillateur de cristal compensé en température (TCXO). Cet oscillateur utilise une diode à capacité variable pour corriger la fréquence de l'oscillateur en fonction de la température. Il n'est donc pas impacté par les variations de températures externes. (voir *Fig.1*)
Protocole : I²C
Données fournies : secondes, minutes, heures, jour, date, mois, année
Le module est capable de différencier les mois de 30 et 31 jours ainsi que les années bissextiles (jusqu'à 2100).
Alimentation : 3.3 à 5V
Format possible : 24h ou 12h avec am/pm
Précision = +/- 2 minutes par an

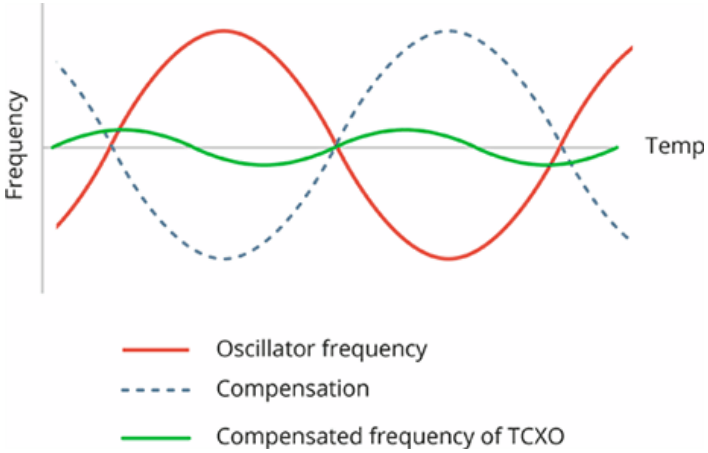


Figure 1 : Représentation de la compensation du changement de température par l'oscilloscope

- Le DS1307

La grande différence entre le DS1307 et le DS3231 est que le premier est moins précis au cours du temps.
Le DS1307 ne possède pas d'oscillateur de cristal compensé en température et donc est affecté par la température externe (qui fait très facilement varier la fréquence d'oscillation).
De ce fait, l'horloge se dérègle d'environ 5minutes par mois.
Mis à part la précision, les caractéristiques sont les même entre les deux modules.

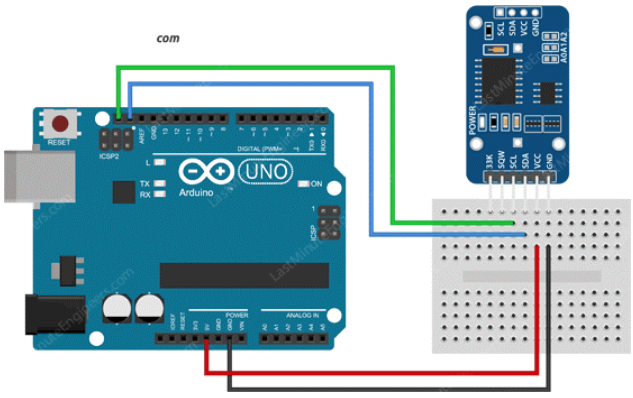
- Le DS3232

Le module DS3232 est une version évoluée du module DS3231. Il possède les mêmes caractéristiques que le DS3231 mais à cela s'ajoute une batterie supplémentaire et un taux d'échantillonnage du capteur de température réglable. Il possède une SRAM (static random access memory) qui est un type de mémoire vive n'ayant pas besoin de rafraîchir régulièrement son contenu. Elle est donc moins énergivore et plus rapide (mais plus onéreuse). Il est donc plus précis et plus performant.

- Le PCF8563

Données fournies : année, mois, jour, jour de la semaine, heures, minutes, secondes
plage d'alimentation : 1 à 5.5V
0.25µA pour 3V à 25°C
Il possède des propriétés permettant de régler des alarmes et des timers.
Le PCF8563 n'a pas de RAM. La RAM (Random Access Memory) peut être comparée à notre mémoire à court terme. Elle permet un fonctionnement plus fluide et stocke temporairement les données. Il est donc moins performant.

Nous allons donc choisir le module DS3231 puisqu'il est plus fiable, possède une mémoire vive et est suffisamment évolué pour le travail qui lui est demandé.



Montage du module

Grâce à ce module le programme va pouvoir comparer la date et l'heure actuelles avec la date et l'heure entrée par l'utilisateur dans l'application. Pour cela il va falloir convertir la donnée de l'application en une donnée comparable c'est-à-dire de même format comme celui-ci par exemple : MMDDhhmm.

iii. Fonctionnement du protocole I²C

Un protocole I²C permet d'établir une communication entre l'Arduino et les différents périphériques de la carte, ou des périphériques externes (capteurs de température, d'humidité, accéléromètres ou gyroscopes). En effet, I²C, Inter Integrated Circuit , est un protocole de communication crée à la base par Philips en 1982 pour standardiser l'échange de données entre différents circuits intégrés d'un même système. Ce protocole est basé sur un bus de communication sériel. Sériel signifie qu'un seul câble est utilisé pour le transfert de données.
En pratique, le bus I²C est constitué de deux câbles, un donc pour les données, nommé SDA (Serial Data) et l'autre déterminant la fréquence de la communication, nommé SCL (Serial Clock).
Ainsi, tous les périphériques sont connectés sur le même bus. Tout d'abord, il n'y a qu'un maître (le microcontrôleur de la carte Arduino), et une multitude d'esclaves (les périphériques), chacun identifiés par une adresse unique. Seul le maître peut initier une communication.
Le maître envoie l'adresse du périphérique dont il désire recevoir les données. L'esclave envoie un premier signal de confirmation pour signifier qu'il a bien reçu la demande. Puis le maître envoie l'adresse d'un registre interne du périphérique. Un deuxième signal de confirmation est envoyé par le périphérique. Enfin c'est le périphérique qui émet cette fois le message, en transférant la valeur du registre qui a été sollicité. Il termine avec une dernière confirmation, après quoi le maître envoie un signal spécifique pour mettre fin à la communication.

III Sonnerie et musiques

Lorsque l'heure actuelle correspond à l'heure choisie par l'utilisateur pour se réveiller, le réveil doit sonner. L'utilisateur peut au préalable choisir la musique qu'il désire entendre depuis son téléphone. Le sujet de cette partie concerne le moyen de propagation du son dans la pièce et son stockage.

i. Choix et transmission de la sonnerie

Afin de récupérer la musique choisie pour réveiller l'utilisateur, il existe plusieurs solutions :

- Grâce à l'application, l'utilisateur pourra choisir la sonnerie du réveil parmi une liste de musique. Le téléphone transmettrait alors ce fichier à la carte Arduino qui devra le stocker dans sa mémoire. Néanmoins, il y a transmission de données inutiles surtout si la musique choisie ne change pas.
- Toujours, l'utilisateur peut choisir la sonnerie du réveil parmi une liste de musique. Ce téléphone transmettra alors à la carte Arduino seulement le nom de la musique choisie. La carte Arduino récupèrera cette musique dans sa carte sd (où les musiques seront toutes stockées au format waw). Ainsi, la mémoire de l'Arduino sera moins rapidement saturée. (voir XI iii)

ii. Choix du haut-parleur

Afin de diffuser la musique, il faut un haut-parleur. Plusieurs solutions s'offrent à nous :

- Utiliser une prise de jack branchée à l'Arduino et à une enceinte. Le son sera forcément suffisant et réglable.
- Utiliser un haut-parleur pour Arduino qui diffusera le son. Néanmoins, la puissance maximale en sortie d'un tel haut-parleur est 3 watts soit 0.6 A pour 8 Ω. Le son ne sera pas très fort mais sans doute suffisant pour une sonnerie de réveil… Si le son n'est pas suffisant, on peut rajouter un amplificateur ou encore installer le haut-parleur de façons différentes :

1. Méthode de l'enceinte clause

Les ondes que produit le haut-parleur ''devant'' et ''derrière'' sont opposées et donc certaines s'annulent.

Les ondes intérieures sont enfermées et donc on entend que les ondes sonores extérieures (voir *Fig.1*) Le son augmentera puisque les ondes produites par le haut-parleur ne s'annuleront pas.

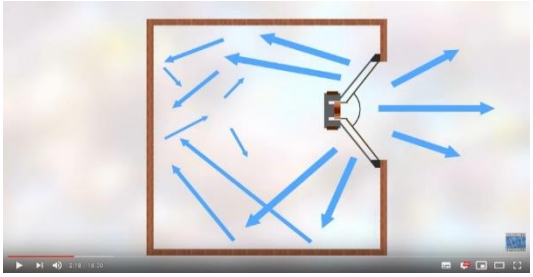


Figure 1: schéma explicatif de l'enceinte claus

2. Enceinte basse-réflexe qui possède un évent

Le son sort de l'évent avec un certain retard ce qui permet d'additionner l'onde sonore du haut- parleur avec celle en sortie de l'évent. (voir *Fig.2*) Les amplitudes s'additionnent et donc le son augmente. Cette technique permet de ne pas consommer plus d'énergie.

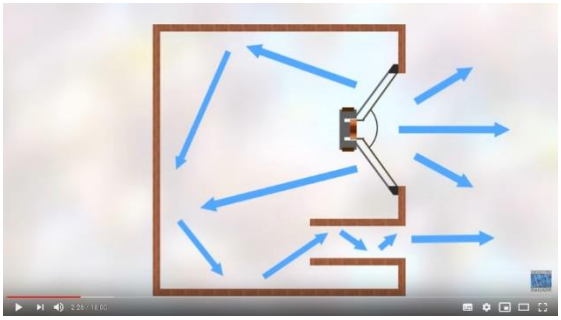


Figure 2 : schéma explicatif de l'enceinte basse-réflexe

Si l'on veut faire varier le volume de la musique, il suffit d'ajouter un simple potentiomètre. De plus, le son n'étant pas notre principale préoccupation dans le cadre de notre projet, il n'est pas nécessaire d'ajouter des filtres passe bas ou passe haut, différents hauts parleurs…

iii. Montage de l'Arduino

On suppose donc que l'on branche un haut-parleur de 8 Ω directement à l'une des sorties de l'Arduino. La sortie de l'Arduino sera traversée par un courant de 625 mA ce qui n'est pas conseillé. Le branchement risque donc de griller le microcontrôleur. Le mieux est que le courant ne soit pas supérieur à 20 mA.
Nous pouvons donc ajouter une résistance en série avec le haut-parleur de façon que le courant soit de 20 mA. Nous avons donc besoin d'une résistance de 5/0.02 soit 250 Ω. Le son sera surement très faible et donc insuffisant pour réveiller quelqu'un !
La solution la plus simple et moins risquée serait donc d'ajouter un transistor (voir III.iv).
Finalement, le faible courant qui circule dans la sortie numéro 8 de l'Arduino et dans la base du transistor contrôle le courant qui circule dans le haut-parleur à travers le trajet collecteur-émetteur du transistor.
Notre calcul plus haut a montré qu'un haut-parleur de 8 Ω soumis à une tension de 5 volts sera traversé par un courant de 600 mA, ce qui semble un peu trop élevé. Bien sûr le transistor absorbera une petite fraction de ces 5 volts. Il est quand même plus sage d'ajouter une résistance :
SI l'on choisit une résistance de 100 Ω, la puissance dissipée par la résistance sera inférieure à 0.25 watt. En effet, la plupart des résistances et potentiomètres supportent 1/4 de W.
Nous obtenons le montage ci-contre (voir *Fig.3*).

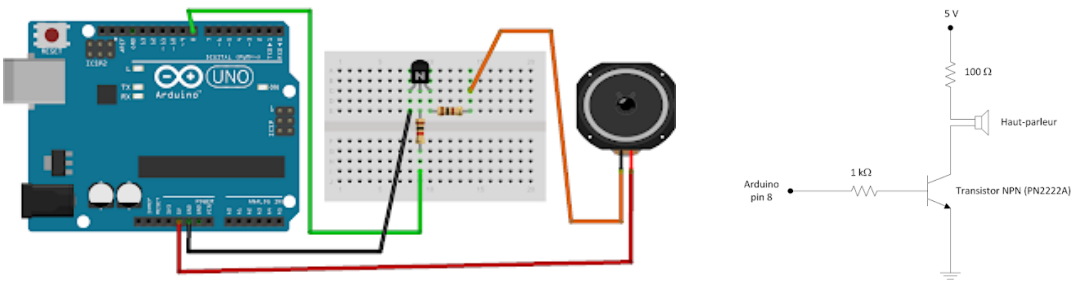
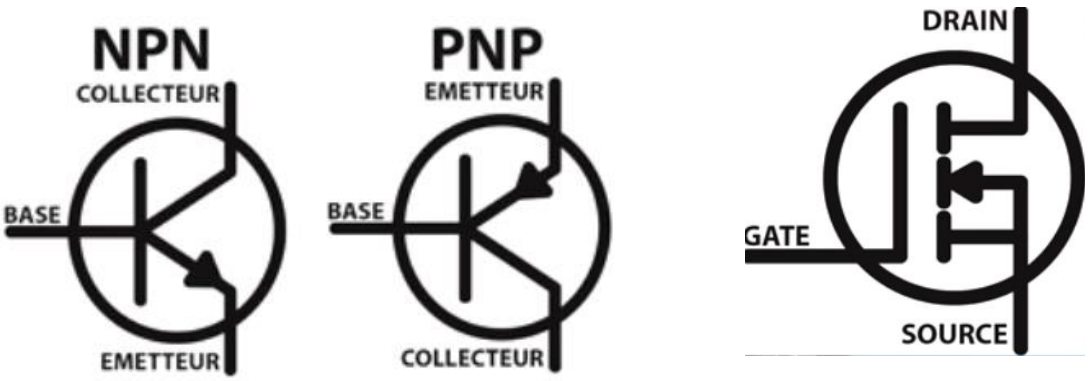


Figure 3 : Montage du haut-parleur avec la carte Arduino

iv. Choix du transistor

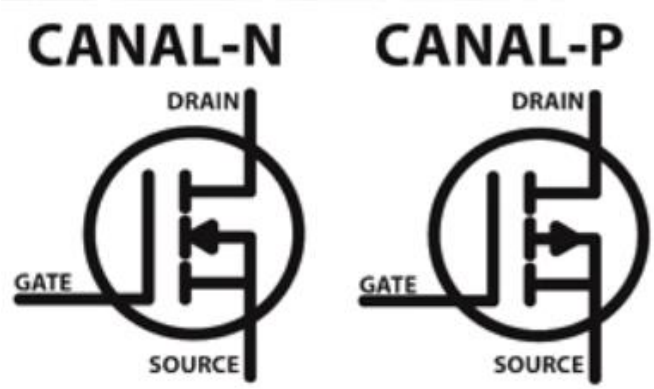
Définitions transistors et MOSFET

Un transistor est un dispositif semi-conducteur à trois électrodes actives. Il peut jouer deux fonctions : celle d'amplifier ou celle d'interrupteur.
La base du transistor bipolaire joue une fonction de commande. Si elle reçoit un courant elle va alors permettre le passage d'un courant plus important dans un sens précis entre les deux autres électrodes.
Il existe deux grandes catégories de transistors que l'on définit par les transistors bipolaires ou les transistors MOSFET.
La principale différence entre les deux est que le transistor bipolaire est contrôlé par le courant alors que le transistor MOSFET est contrôlé par la tension.
De plus, les transistors bipolaires se répartissent en deux catégories : les transistors PNP et les transistors NPN. Le symbole PNP signifie Positif, Négatif, Positif. Le symbole NPN veut dire Négatif, Positif, Négatif. Lorsqu'un transistor bipolaire est représenté sur un schéma électrique, parfois on peut lire les indications C, B et E. La première lettre désigne le collecteur (C), la deuxième la base (B) et la troisième l'émetteur (E). On utilise les lettres EBC sur les feuilles de données des composants pour indiquer la fonction de chacune des trois électrodes. La base est toujours la broche qui contrôle le transistor. Lorsqu'on lui envoie le bon courant, on commande le transistor. De plus, l'intensité appliquée sur la base amplifie l'intensité de la sortie. Pour les transistors NPN, le courant doit circuler du collecteur vers l'émetteur alors que pour les transistors PNP, le courant circule de l'émetteur vers le collecteur (lorsqu'il est branché sur une alimentation positive). Il y a aussi le courant de la base qui circule vers l'émetteur, mais le courant appliqué sur le collecteur ne se rend jamais à la base.



Transistor bipolaire Transistor MOSFET

Pour ce qui est des transistors MOSFET, les deux sous catégories sont les transistors de canal P et les transistors de canal N. De la même façon, le canal-P a été conçu pour les tensions négatives et le canal-N pour contrôler les tensions positives.



Choix du transistor

Le courant étant positif, nous choisissons un transistor NPN. Le transistor NPN doit pouvoir supporter un courant de 600 mA à travers le trajet-collecteur-émetteur.

IV Simulateur d'aube

L'une des options que nous proposons à l'utilisateur est la possibilité de se faire réveiller plus naturellement par une lumière progressive (imitant le soleil à l'aube).

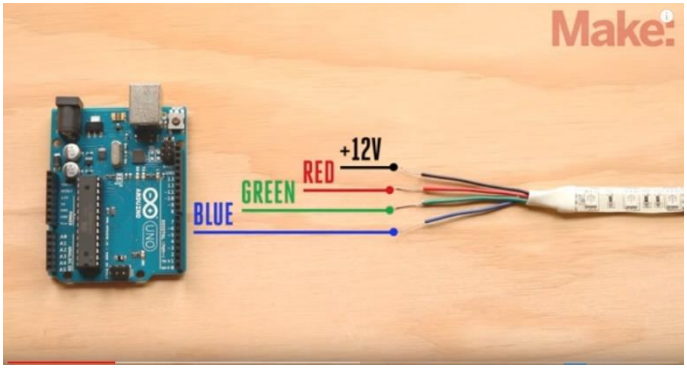
La simulation de l'aube peut être produite de différentes manières.

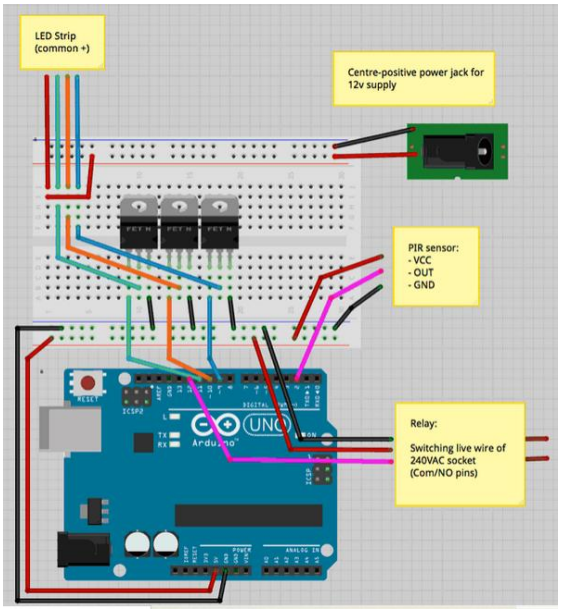
Solution 1 : Lampe de chevet

Il est possible de connecter l'Arduino à une lampe de chevet. Ainsi, l'Arduino ajustera la lumière de cette lampe. Cette solution ne correspond pas au résultat que nous voulons produire.

Solution 2 : Bande de LED RGB

Il existe des bandes de LED RGB ou même d'une seule couleur (par exemple, blanc chaud). Nous pouvons contrôler la couleur tout simplement en faisant varier le rouge, le vert et le bleu. Par exemple pour obtenir du jaune, la valeur RGB sera : 255,255,0.
Pour faire varier l'intensité lumineuse du ruban LED, il suffit de connecter les MOSFETS, qui enverront le courant aux LED, aux sorties PWM de l'Arduino. En effet, les MOSFETS recevront un signal d'amplitude modulable de l'Arduino puis le transmettront aux LEDS.





Ainsi, étant donné les nombreuses LEDS de ces bandes lumineuses, l'intensité lumineuse sera suffisante pour reproduire l'aube.
Néanmoins, ce type de bande lumineuse nécessite donc une alimentation à 12 V. De plus, une bande lumineuse est plus encombrante qu'une LED.
L'utilisation de ruban LEDS paraît plus convenable pour couvrir une grande zone bien que l'on puisse couper le ruban pour obtenir une taille plus petite.

Solution 3 : LED puissante de couleur chaude

Afin de simuler l'aube, nous pouvons aussi utiliser une LED puissante de couleur chaude. Grâce à un MOFSET branché à une sortie PWM de l'Arduino, nous pourrons contrôler l'intensité de la LED. Le MOFSET permettra de fournir une tension considérable et suffisante pour que la LED est une intensité lumineuse comparable à la lumière du soleil observée sur terre.
La température de la couleur du soleil en plein jour est de 6500K contre 3400K au maximum durant le lever du soleil. Notre LED doit donc avoir une température de couleur entre 3000K et 3500K.

Le transistor PNP sur le schéma correspond en réalité au MOFSET. Le gate est branché à la PWM, le drain à la LED et la source à la masse (voir *Fig.1*).

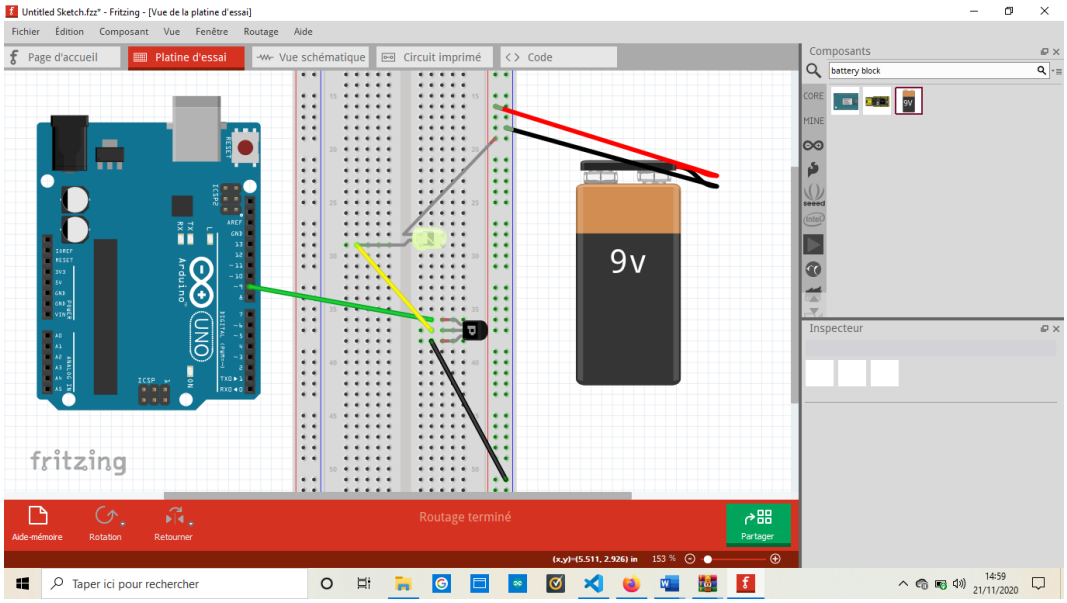


Figure 1 : Montage du circuit

Ce type de LED est moins cher qu'une bande lumineuse tout en convenant parfaitement à notre projet. Afin que le projet ne soit pas trop encombrant, il convient de choisir une seule LED puissante qu'on logera à l'intérieur du réveil. De plus, une LED de couleur 3000K a besoin d'une alimentation de 9V à 12V donc inférieure à celle dont les bandes lumineuses ont besoin.
Nous choisirons donc une LED puissante de couleurs chaude.

V Bouton d'arrêt et de mise en veille

Il est plus agréable pour l'utilisateur de pouvoir arrêter son réveil en appuyant sur un simple bouton plutôt qu'en arrêtant son alarme depuis son téléphone, ainsi que de pouvoir mettre son réveil en mode veille.
Nous allons pour cela utiliser des boutons poussoirs.

Un bouton poussoir est un interrupteur monostable : il n'est stable que dans la position « relâché ». Il peut être ouvert ou fermé en fonctionne si l'on appuie ou pas.

Nous voulons intégrer deux boutons poussoirs au réveil :

i. Activation du mode nuit

Le premier permet d'activer le mode nuit c'est-à-dire qui éteint l'écran du réveil

Par défaut, l'écran sera allumé. Lorsque l'utilisateur appuiera sur le bouton poussoir, les pixels de l'écran seront éteints et resteront éteints quand l'utilisateur relâchera le bouton. Lorsqu'il appuiera de nouveau, l'écran sera allumé et restera allumé quand l'utilisateur relâchera le bouton.
Grâce au code, nous pourrons programmer cela très facilement.
De plus, quand le réveil sonne l'écran s'allumera.

ii. Un qui éteint la sonnerie du réveil

Le second bouton poussoir permet d'arrêter la sonnerie.
De la même façon, quand on appuiera sur le bouton et si le réveil sonne, la sonnerie du réveil s'arrêtera. En revanche, si on rappuie alors qu'il n'y a aucune sonnerie, rien ne changera.

Il existe des boutons poussoirs double pour commander deux choses différentes avec le même bouton. Dans notre cas, ce n'est pas utile puisque nous ne voulons pas forcément placer les deux boutons au même endroit. De plus, au réveil, il pourrait avoir confusion entre les deux commandes.
Nous choisissons donc deux boutons poussoirs.

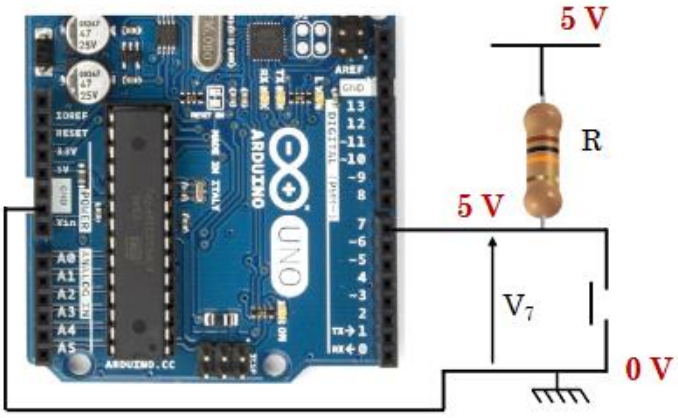


Figure 1 : Montage avec l'arduino
Avec R=10 kΩ

VI Communication entre le réveil et le téléphone

Le but est que notre réveil soit réglé par l'utilisateur via son téléphone. Pour cela il faut que la carte Arduino au cœur du réveil puisse communiquer avec le téléphone. Il faut donc les relier, mais un fil USB serait trop contraignant. En effet il empêcherait de mettre la carte Arduino dans une boite et limiterait la distance de séparation entre la carte et le téléphone.
Il existe plusieurs types de communications sans fil pour contourner ce problème.

i. Différents moyens de communication

- Wifi

Initialement conçu pour se débarrasser de câble Ethernet pour pouvoir fabriquer des ordinateurs portables, il est également utilisé pour développer des objets connectés, notamment avec son module ESP8266. Il permet de couvrir une habitation. Il est idéal pour réaliser des objets qui doivent interagir avec un service en ligne (par exemple publier des résultats). Ce n'est pas notre cas ici, nous ne retiendrons donc pas cette possibilité.

- Bluetooth

Le Bluetooth est une technologie de communication à courte distance (une dizaine de mètres). Il est plus réactif que le Wifi et est donc privilégié dans la réalisation d'objets télécommandés depuis des interfaces comme le téléphone. Il est avant tout destiné à un usage grand public où un seul utilisateur pourra se connecter à un appareil (donc idéal pour une télécommande).
Les modules HC-05 et HC-06 dont les plus courants.

- Xbee et Zigbee

Très utilisé dans le monde industriel, sur le même principe du bluetooth, ils permettent de créer un réseau d'objets. Le Zigbee n'est pas conçu pour faire transiter un grand nombre de données (250 kbps maxi) mais il le fait en consommant peu et de manière sécurisée et fiable.
Son coût est plus conséquent que celui du EPS8266.

Quelques caractéristiques supplémentaires :
- consommation 3,3V pour 50mA
- chiffrement de 128-bits
- 65 000 nœuds maximum
- topologies de réseaux possibles : maillé, point à point, point à multipoint

- Ondes radios sans licence

Les solutions vues précédemment sont des technologies de transmission de données par onde radio propriétaires. Elles sont développées par un regroupement d'entreprises.
Il existe aussi des modules radio génériques qui n'utilisent aucun protocole de communication propriétaire.
Il est possible de générer soi-même la communication entre deux appareils en se servant de la bibliothèque VirtualWire par exemple (ou les bibliothèques Mirf, Radio…).
Pour générer un réseau d'objets connectés et y transmettre des données, la bibliothèque MySensors est plus adaptée. Par exemple dans le cas d'un projet de domotique, développer ses propres capteurs ou actionneurs. Cette librairie peut utiliser le module nRF24L01 qui utilise une bande de fréquence 2,4GHz, mais il en existe bien d'autres de fréquences variantes.

- LoRaWAN ou Sigfox

Ces deux technologies de communication par onde radio se développement fortement de nos jours. Elles proposent un système de communication numérique à longue portée (plusieurs kilomètres) utilisant la bande de fréquence 868,1 MHz.
Sigfox fournit une solution complète (protocole et antennes) et s'adresse avant tout à l'industrie et aux infrastructures (ville intelligente). Contrairement LoraWAN (Long Range Wide-area network) qui est un projet ouvert (chacun peut acheter des modules et développer son réseau d'objets connectés), il faut prendre un abonnement auprès de la société Sigfox pour pouvoir communiquer avec ses objets.

ii. Notre choix

Pour choisir la technologie de communication la plus adéquate avec l'utilisation que nous allons en faire nous devons nous baser sur les critères suivants : la portée souhaitée, la configuration, le fonctionnement, le budget, encombrement disponible et notre niveau.
Pour une portée courte, dans une habitation, sur secteur et avec un budget, un temps et des compétences limités il parait préférable de choisir le Bluetooth.

iii. Renseignements approfondis sur le Bluetooth

Le Bluetooth est donc un protocole de communication sans fil ayant vu le jour à la fin des années 1990 mais n'ayant réellement percé qu'à partir des années 2000. Il utilise la bande de fréquence 2,4 GHZ. C'est une communication bidirectionnelle c'est-à-dire que les deux modules peuvent communiquer ensemble en même temps. Le comportement utilisé est « maître/esclave ». Un esclave pourra parler avec un seul maître, mais un maître pourra dialoguer avec plusieurs esclaves.
Le fonctionnement se passe en plusieurs étapes :
1. Le maître se met en mode « reconnaissable »
2. L'esclave trouve le maître et demande à s'y connecter
3. Le maître accepte la connexion
4. Les périphériques sont alors apparaiés (ou associés)
5. La communication peut s'effectuer et dépend alors du type de composants associés

iv. Présentation des modules HC-05 et HC-06

Le module HC-05 est un module maitre (il peut proposer à d'autres appareils de s'apparier à lui) tant dis que le module HC-06 est un module esclave. Il ne peut pas se connecter à d'autres appareils tout seul. Dans le cadre du projet il nous faudra donc utiliser un module HC-05.
Le module HC-05 est en fait un montage d'un module Bluetooth sur un petit PCB (circuit imprimé). Les broches sont connectées à l'alimentation et la masse (VCC, GND) et Rx/Tx pour la communication. Il y a aussi une broche « Key » qui sert à envoyer les commandes de configurations du module mais qui n'est pas branché lors de la communication. La dernière broche nommée « led » permet de brancher une LED pour obtenir un signal sur l'état du module. (voir *Fig.2*)

Le module fonctionne via une voie série. La seule que l'on trouve sur l'Arduino est généralement utilisé pour le debugging. Il faut donc en émuler une autre. Il est possible de coder la nouvelle voie soi-même ou d'utiliser la bibliothèque SoftwareSerial.

Dès la première utilisation il faut modifier les paramètres internes du module. Pour être configuré le module utilise un ensemble de commandes qu'on appelle « commandes AT ». Afin de les modifier, le module doit être placé en mode « commande » (en passant la broche key en état HIGH ou appuyer sur le bouton poussoir s'il y en a un, voir *Fig.1*). Toutes les informations envoyées au module seront alors interprétées pour faire de la configuration et rien ne sera envoyé en bluetooth à l'autre appareil.
Il faut donc commencer par configurer le module pour pouvoir l'utiliser ensuite, c'est-à-dire vérifier que l'AT est actif, afficher la version du firmware, configurer les paramètres de communication, modifier le rôle du module (esclave ou maitre).
Une fois le programme téléversé, il faut aller dans les réglages du téléphone pour l'associer au périphérique HC-05 de l'Arduino (le mot de passe par défaut étant 1234).
Si la LED clignote rapidement cela signifie qu'elle cherche à s'appairer. Si le démarrage est correctement effectué, la LED clignote lentement tous les deux secondes environ.

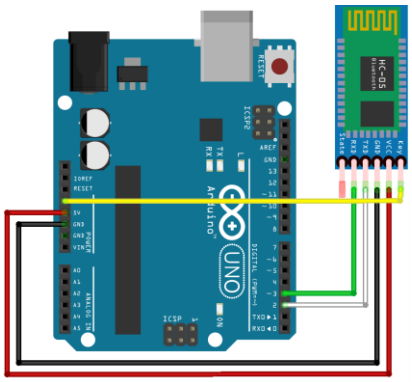


Figure 1 : mode de commande

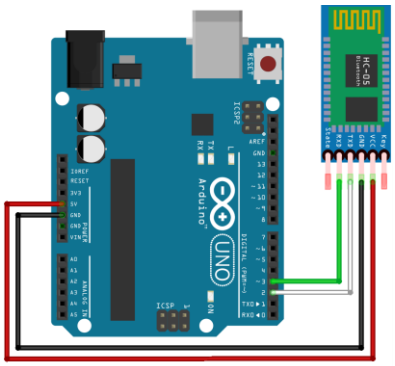


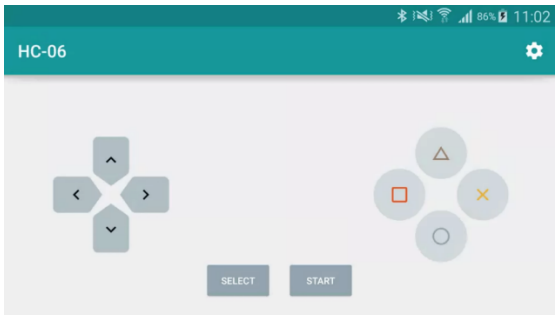
Figure 2 : mode de communication

VII Contrôle depuis le téléphone

Nous voulons donner la possibilité à l'utilisateur de personnaliser son réveil. En plus de pouvoir régler l'heure à laquelle le réveil doit sonner, il doit donc avoir le choix parmi plusieurs options pour que l'objet lui soit le plus agréable et adapté possible. Par exemple, veut-il la lumière progressive ou non ? Est-il intéressé par la météo à son réveil ? Veut-il avoir accès à son emploi du temps de la journée ? Pour cela nous devons créer une application afin de fournir une interface lui permettant d'envoyer ces informations à la carte Arduino par Bluetooth (mode de communication choisi au préalable).

Il existe différentes applications pour contrôler le réveil depuis le téléphone. Les critères de sélections sont principalement l'esthétique, la simplicité et l'adaptation à notre projet.

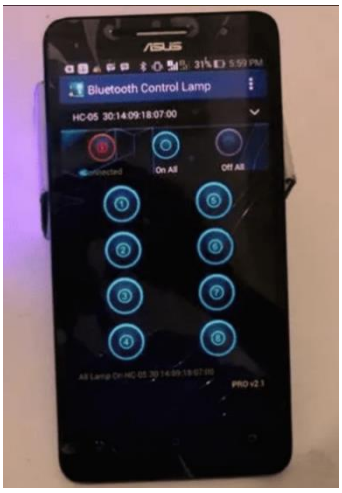
- Arduino Bluetooth Controller



Cette interface permet d'effectuer 4 types de commandes différents.

- Le mode de contrôle ; il présente une interface analogue au contrôle de jeu. L'utilisateur peut appuyer sur l'un des boutons de l'écran et enverra alors une commande correspondante à l'Arduino.
- Le mode switch ; il ne présente qu'un seul bouton permettant de contrôler un commutateur connecté à distance.
- Le mode graduateur ; il peut envoyer des valeurs changeantes à votre Arduino, utile pour contrôler des facteurs tels que la luminosité et la vitesse.
- Le mode terminal ; permet d'envoyer des commandes personnalisées. Il faut écrire le code sur l'Arduino pour décoder ces commandes.

- Bluetooth Controller 8 Lamp

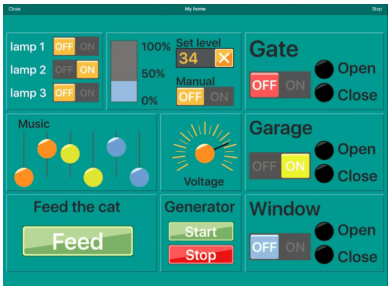


Cette application permet de contrôler 8 canaux. Son interface est composée de 8 boutons, un pour chaque canal. Deux boutons supplémentaires permettent de basculer tous les canaux à la fois (« all on » et « all of »).

- Bluetooth Voice Control for Arduino

L'Arduino a une mémoire trop petite et une capacité de traitement trop faible pour faire de la reconnaissance vocale. L'application permet donc de faire cette reconnaissance vocale et ensuite d'envoyer à l'Arduino les données sous forme de chaine de caractères.

- RemoteXY : Arduino Control



Depuis le site web de remotexy, il est possible de créer sa propre interface graphique. Elle peut être composée de boutons, de switcher, de sliders ou de joysticks par exemple. Elle permet aussi une connexion au cloud, ce qui rend l'application accessible de n'importe quel appareil.

- Virtuino

Grâce à cette application il est possible de contrôler plusieurs Arduino à la fois. Elle permet de créer une interface visuelle pour les LED, les commutateurs, les graphiques, les instruments analogiques, les compteurs … Il faut rajouter du code pour que l'Arduino puisse décoder les données de l'interface et communiquer avec.

- Blynk

Cette application fonctionne de manière similaire aux précédentes. Elle ne présente pas de particularités.

- AppInventor 2

L'application permet une mise en page graphique incluant des images. La programmation est fortement inspirée du langage C mais se fait graphiquement en utilisant des schémas de boites, correspondant à du code.
Une grande variété de codes est mise à disposition et il est donc possible de personnaliser son application.

Nous ne voulons contrôler que notre Arduino et n'avons pas besoin de la reconnaissance vocale ni de représentations de graphes. Nous allons donc choisir AppInventor 2 car cela nous permettra de designer l'application selon nos choix. De plus la partie code de l'application se fait au préalable sur l'ordinateur sur deux pages de traitement, une pour le graphisme et une pour le code en block (qui fait le lien entre l'action de l'utilisateur sur l'application et l'Arduino). Une fois programmée, un code de référence est attribué. Il faut alors télécharger l'application AppInventor2 sur le téléphone Android et rentrer le code. L'interface est alors créée de manière automatique en reprenant les éléments codés sur l'ordinateur. Une fois que le téléphone et l'Arduino sont associés en Bluetooth dans les réglages du téléphone, il faut aller sur l'application et connecter via l'application le téléphone à l'Arduino (toujours en Bluetooth).
Dès lors chaque action sur l'interface de l'utilisateur enverra une donnée à l'Arduino, par le changement de variables ou de code correspondant.

VIII Connexion internet

Notre réveil doit avoir un accès à certaines données que l'on trouve sur internet telles que la météo ou les rendez-vous de l'utilisateur via son compte google. En effet l'utilisateur a la possibilité de les afficher sur son réveil.

i. Notion d'API

Une API (Application Programming Interface) fournit une liste d'opérations que les développeurs d'application peuvent utiliser. Nous pouvons alors faire une requête à une API qui nous fournira une réponse adaptée. L'API nous donne d'abord une clé et grâce à celle-ci, il est possible de faire des demandes plus ou moins précises selon les opérations que l'API propose.
Récupérer des informations provenant d'internet n'est pas toujours simples. Nous supposons que l'utilisateur du réveil possède un accès au wifi. Deux possibilités s'offrent à nous :

- L'application récupère des données via une API puis les transmet à l'Arduino via Bluetooth.

Inconvénients de cette méthode :
1.Il y a encore des transferts de données via Bluetooth entre le téléphone et l'Arduino.
2.Le téléphone doit être obligatoirement allumé et connecté à Internet pour pouvoir récupérer les données (par exemple météorologiques qui nous intéressent).

- Nous pouvons aussi directement connecter l'Arduino via un Ethernet Shielder à internet et utiliser une API.

Inconvénient de cette méthode :
L'Arduino doit être branchée à un module wifi ou Ethernet Shielder.

ii. Choix d'une API pour le calendrier google

Une des options du réveil est de pouvoir afficher les rendez-vous du jour de l'utilisateur sur le réveil. Pour cela, nous avons créé un nouveau compte google entièrement dédié au projet. L'API que nous utiliserons est celle développée par GOOGLE : Google calendar api.

iii. Module wifi ou Ethernet Shield

Pour la connexion à internet deux choix s'offrent à nous : utiliser un Ethernet Shield W5100 ou un module wifi.

- Ethernet Shield

L'Ethernet Shield W5100 est un des seuls compatible avec l'Arduino uno. Nous pouvons y mettre une carte micro-sd et elle s'emboite directement sur la carte Arduino uno. Néanmoins, pour le connecter à un internet, il faut un cable RJ45 branché à la box… Il ne sera donc pas possible de le tester à Polytech ! Nous ne retenons pas cette possibilité.

- Module wifi

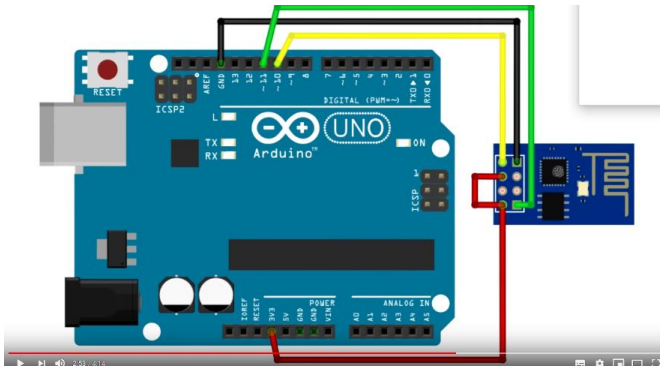
Il existe deux principaux modules wifi : ESP32 et ESP8266.
Ils sont tous deux des modules WIFI peu chers. L'ESP32 est le successeur de l'ESP8266.

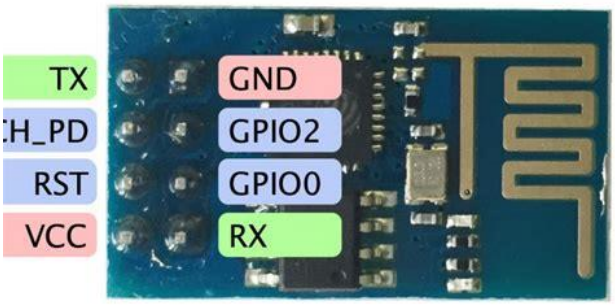
L'ESP8266 offre un processeur 32 bit cadencé à 80 Mhz alors que l'ESP32 propose un double cœur 32 bit cadencé entre 160 et 240 Mhz. Les specs des deux puces sont proches mais naturellement l'ESP32 est légèrement supérieur avec son cœur en plus.
L'ESP32 offre aussi en prime une connectivité Bluetooth. Il pourra donc remplacer le module Bluetooth.
En moyenne un ESP32 coûte 10 euros et l'ESP8266 coûte 7 euros. Or, un module Bluetooth HC-05 coûte en moyenne 9 euros. Il est donc plus rentable de choisir un ESP32 (10 euros) plutôt qu'un ESP8266 et un HC-05 (total de 16 euros). De plus, avoir qu'un seul module libérera de la place dans le réveil et consommera sans doute moins d'énergie.
De plus, il existe différents modèles d'ESP32.

		Processeur Xtensa	Bluetooth	Core	Broches	Mémoire Flash (MB)	PSRAM (MB)	Deep Sleep*
ESP32-S2		LX7 jusqu'à 240MHz	Non	x1	42			20 µA
	WROOM	ESP32-S2				4,8,16	–	
	WROVER	ESP32-S2				4,8,16	2	
	Mini	ESP32-S2FH4						
ESP32		LX6 80 ~ 240MHz	Oui	x2	38			5 µA
	VROOM					4,8,16	–	
	WROVER					4,8,16	8	
ESP32-SOLO-1		LX6 @ 160MHz		x1	38	4	–	5 µA

Celui qui nous avantage est donc le ESP32 (qui possède le Bluetooth) VROOM. En effet, le PSRAM est une mémoire qui s'effacera dès que le module sera éteint. Il permet d'augmenter la mémoire flash. La mémoire flash, quant à elle, garde les données même si le module est éteint.

Après de conséquentes recherches, nous nous sommes finalement rendu compte qu'il était impossible de brancher un ESP32 avec une carte Arduino. En effet, l'ESP32 peut remplacer l'Arduino avec des fonctionnalités en plus (wifi+bluetooth). La carte Arduino est plutôt utilisée par des débutants car elle est plus facile d'utilisation que l'ESP32.
Nous choisissons donc un ESP8266.Grâce au code, l'ESP8266 se connectera au réseau wifi de notre choix. Le montage est le suivant :





IX Météorologie

1. Température intérieure

L'une des options possibles de notre réveil est d'afficher la température dans la maison de l'utilisateur. Pour cela il faut récupérer la température ambiante à l'aide d'un capteur. Il existe une grande variété de capteurs ayant des caractéristiques différentes qu'il va donc falloir analyser pour choisir celui qui remplira au mieux sa tache dans le cadre de notre projet.
De nos jours, les capteurs ne sont plus conçus à base de mercure, de bandes bimétalliques, ni de thermistances. A la place, on se base sur le fait que lorsque la température augmente, la tension aux bornes du capteur augmente proportionnellement. Il est alors simple de créer un signal analogique proportionnel à la température.
Les capteurs n'ayant pas de parties mobiles, sont précis et ne nécessitent pas de calibrage. Ils fonctionnent dans de nombreuses conditions environnementales, sont peu couteux et simples d'utilisation.

1.i. Divers capteurs

Voici les 9 capteurs les plus souvent utilisés :

- Le DHT11

Ce capteur permet de mesurer la température ainsi que l'humidité. Il contient une puce qui convertit le signal analogique pour renvoyer un signal numérique correspondant à des valeurs de température et d'humidité.
Protocole utilisé : One-Wire
Alimentation : 3 à 5.5V
Plage de température : 0 à 50°C avec une précision de +/- 2°C
Plage d'humidité : 20 à 90% avec une précision de +/-5%
Période d'échantillonnage : 1 seconde
Librairie Arduino : Adafruit DHT Library

- Le DHT22

Ce capteur est assez similaire au précédent en termes de fonctionnement. Il s'en démarque par des données plus précises et des plages de mesure plus vastes. Il est également un peu plus coûteux.
Protocole utilisé : One-Wire
Alimentation : 3 à 6V
Plage de température : -40 à 80°C avec une précision de +/-0.5°C
Plage d'humidité : 0 à 100% avec une précision de +/-2%
Période d'échantillonnage : 2 secondes
Librairie Arduino : Adafruit DHT Library

- Le LM35DZ / Le LM335 / Le LM34

Ce capteur ne mesure que la température et est calibré en Celsius. Les valeurs analogiques sont proportionnelles à l'échelle de Celsius, soit 10mV pour 1°C.
De la même façon le LM335 est calibré en Kelvin et le LM34 en Fahrenheit.

Pour le LM35DZ :
Protocole utilisé : pas de protocole, on récupère la donnée sur une sortie analogique
Alimentation : 4 à 30V
Plage de température : -55 à 150°C avec une précision de +/-0.5°C (à 25°C)

- Le BMP180

Ce capteur est tout d'abord conçu pour mesurer la pression atmosphérique mais il mesure également la température. Il est recommandé pour les projets qui touchent à la météorologie.
Protocole utilisé : I²C
Alimentation : 1.8 à 3.6V pour la puce et 3.3 à 5V pour le module
Plage de température : 0 à 65°C avec une précision de +/-0.5°C (à25°C)
Librairie Arduino : Adafruit BME085

- Le TMP36

Ce capteur mesurant uniquement la température renvoie une valeur analogique proportionnelle à la valeur de température ambiante. Il est similaire au LM35.
Protocole utilisé : pas de protocole, on récupère la donnée sur une sortie analogique
Alimentation : 2.7 à 5.5V
Plage de température : -40 à 125°C avec une précision de +/-1°C (à 25°C)

- Le LM75

Protocole utilisé : I²C
Alimentation : 3 à 5.5V
Plage de température : -55 à 125°C avec une précision de +/-2°C
Librairie Arduino : I²C Library for LM75

- Le BME280

Ce capteur est tout d'abord un capteur de baromètre mais mesure aussi la température et l'humidité.
Protocole utilisé : I²C or SPI
Alimentation : 1.7 à 3.6V pour la puce

Plage de température : -40 à 85°C avec une précision de +/-0.5°C (à 25°C)
Librairie Arduino : Adafruit BME280 library

- Le DS18B20

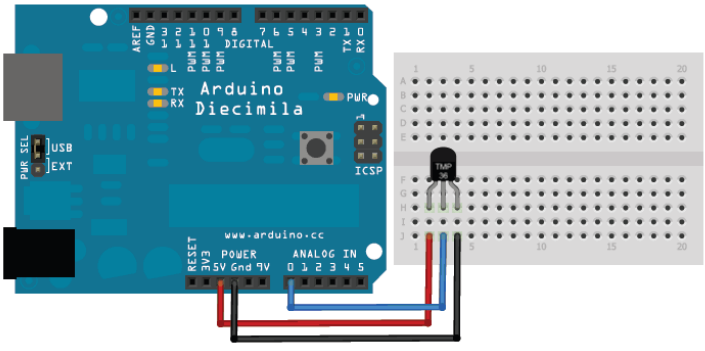
L'avantage principal de ce capteur est le suivant : chaque capteur DS18B20 a un code série codé sur 64 bits unique. Ainsi il est possible d'en utiliser plusieurs avec le même data wire (fil de données), c'est-à-dire récupérer plusieurs données par la même sortie.
Protocole utilisé : One-wire
Alimentation : 3 à 5.5V pour la puce
Plage de température : -55 à 125°C avec une précision de +/-0.5°C (de -10 à 85°C)
Librairie Arduino : Dallas Temperature ou One-wire

- Le Waterproof DS18BM20

Ce capteur est une version étanche du DS18BM20. Les câbles sont protégés pour du PVC. Il est donc à privilégier dans le cas de mesure de liquide ou si le capteur est exposé à de l'eau.
Protocole utilisé : One-wire
Alimentation : 3 à 5.5V pour la puce
Plage de température : -55 à 125°C avec une précision de +/-0.5°C (de -10 à 85°C)
Librairie Arduino : Dallas Temperature ou One-wire

Pour notre projet, un capteur ne mesurant que la température suffit. On exclut donc ceux qui mesure également l'humidité et/ ou la pression. Il sera intégré au réveil donc il n'y aura pas de contact avec de l'eau possible. De plus, nous ne voulons la température ambiante que d'un endroit dans la maison donc un seul capteur suffira. Enfin une plage de températures de -40 à 125°C est tout à fait correcte pour la température d'une maison.
Le capteur TMP36 est donc celui qui répondra le mieux à nos exigences.

Photographie du capteur TMP36



Montage du capteur

1.ii Traitement de la mesure

Pour récupérer la mesure de température, il faut convertir la valeur analogique en tension comprise entre 0 et 5 puis convertir cette nouvelle valeur en °C grâce au rapport suivant : 10mV = 1 °C.
Autrement dit :
tension = analogRead(A0)*5000/1023
//permet de convertir une valeur analogique en tension en mV
temp= (tension -500)/10
//le 500mV correspond à la valeur de l'offset

A noter : pour avoir une valeur plus précise il est possible d'utiliser le 3.3V pour alimenter (dans ce cas on multipliera par 3000 et non 5000)
La température sera ensuite affichée sur l'écran (*cf. Ecran*)

2. La météo extérieure

Nous voulons que sur le réveil s'affiche également la température minimale et maximale de la journée ainsi que la météo globale (ensoleillé, nuageux, pluvieux). Pour cela nous allons utiliser une API.

Choix d'une API pour les données météorologiques

Afin de choisir une API adaptée à notre réveil, il faut connaitre les données exactes que nous voulons recevoir ainsi que le nombre d'appels (c'est-à-dire de demandes) que nous allons faire par jour.
Dans le but de ne pas produire de dépenses inutiles, nous choisirons des API gratuites.
Pour récupérer les données météorologiques qui seront affichées sur le réveil, nous utiliserons : openweathermap.
Celle-ci peut par exemple renvoyer les données suivantes :


```
{
  "coord": {
    "lon": -122.08,
    "lat": 37.39
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 282.55,
    "feels_like": 281.86,
    "temp_min": 280.37,
    "temp_max": 284.26,
    "pressure": 1023,
    "humidity": 100
  },
  "visibility": 16093,
  "wind": {
    "speed": 1.5,
    "deg": 350
  },
  "clouds": {
    "all": 1
```

```
},
"dt": 1560350645,
"sys": {
  "type": 1,
  "id": 5122,
  "message": 0.0139,
  "country": "US",
  "sunrise": 1560343627,
  "sunset": 1560396563
},
"timezone": -25200,
"id": 420006353,
"name": "Mountain View",
"cod": 200
}
```


Grâce à des changements dans le code, nous pouvons récupérer seulement les données qui nous sont utiles.
De plus, grâce à cette API, nous pouvons faire un 1 appel par seconde ce qui sera bien plus que suffisant.
Il existe aussi un site Temboo, très pratique mais payant. Nous ne la retiendrons donc pas.

X Ecran d'affichage

Notre réveil doit être composé d'un écran afin d'y afficher l'heure, et les diverses informations (température, météo, rendez-vous…).

i. Divers types d'écrans

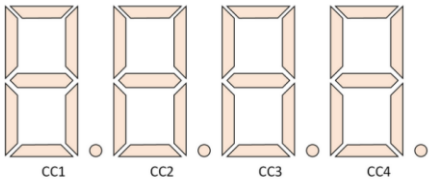
Il existe différents types d'écrans compatibles avec l'Arduino :

- Les écrans LCD

Il existe plusieurs tailles d'écran LCD. Ces écrans possèdent un décodeur de caractères intégré qui permet d'afficher ce qui sera envoyé par l'Arduino et le garder en mémoire. Les instructions sont données en mode parallèle.
Librairie : LiquidCrystal

- Les afficheurs 7 segments

Un afficheur à 7 segments permet de représenter principalement les chiffres de 0 à 9, ainsi que quelques lettres de l'alphabet, moyennant une utilisation mixe des majuscules et des minuscules.
Il est composé de 8 LEDs indépendantes, les sept premières (*a* à *g*) permettant d'allumer un segment spécifique et la dernière (*dp*) un point décimal.
L'afficheur 4 digits combine 4 afficheurs 7 segments disposés côte-à-côte.



- Les afficheurs TFT

TFT pour "Thin-film transistor" est une technologie utilisée pour certains écrans LCD. La technologie TFT permet de contrôler chaque pixel d'une dalle LCD avec un ou plusieurs transistors.
Les transistors TFT sont constitués de plusieurs couches minces (semi-conducteur, couches métalliques...), ils jouent le rôle d'interrupteur dans le circuit de commande des pixels d'un écran LCD. Ainsi, nous pouvons afficher sur l'écran des contenus de différentes couleurs.

- Les afficheurs OLED

Contrairement à ce que leur nom pourrait laisser penser, les afficheurs OLED (Organic Light Emitting Diode) ne ressemblent pas aux afficheurs à LED.
En effet, les afficheurs à LED sont en effet constitués de quelques LED pour illuminer de "larges" segments, alors que les afficheurs OLED sont composés d'une matrice de points aptes à représenter des lettres ou des graphiques !
Les afficheurs OLED sont constitués de matériaux semi-conducteurs basés sur le carbone et l'hydrogène qui s'éclairent quand ils sont soumis à une source de tension électrique, comme une LED standard. Néanmoins, en plus, un écran OLED ne nécessite pas de rétro-éclairage comme c'est le cas pour les écrans LCD. Il offre un contraste plus élevé et une bonne lisibilité sur un large angle de vision, car la lumière émise est plus diffuse.
Le temps de réponse des écrans OLED (c'est-à-dire temps mis par l'afficheur pour activer ses pixels) est inférieur à la milliseconde, c'est plus rapide que le temps de réponse des écrans LCD et à priori adapté à la vidéo.
Leur durée de vie est pour la plupart d'un peu plus de 10000 heures, durée de vie estimée d'un peu plus d'un an.
De plus, nous avons la possibilité d'éteindre les pixels pour passer l'écran en mode ''veille'' la nuit grâce au code.

Tableau comparatif :

	Ecran LCD	Ecran OLED	Ecran TFT	Ecran 7 segments
Avantages	-Ecran LCD rétroéclairé -écriture lisible -Ecran de taille moyenne -affichage monochrome -possibilité de régler le contraste	-Ecran compact -écriture très lisible même avec des angles différents -affichage monochrome -format carré ou rectangle -affichage d'images --2 branchements avec I²C sans compter la masse et l'alimentation	-Ecran tft couleurs -possibilité d'être tactile -affichage de textes, graphiques, images	-pilotable avec 2 fils -affichage de chiffres et lettres -possibilité de régler la luminosité
Inconvénients	-7 broches à utiliser sur	-la librairie Arduino pour la	-code lourd dû aux couleurs	-pas de possibilité

	l'Arduino sans compter la masse et l'alimentation	faire fonctionner prend beaucoup de place -durée de vie ''courte''	-il restera peu de broches sur l'Arduino	d'afficher de dessins…

ii. Notre choix

Nous ne retenons par l'écran tft :

Dans notre cas, nous n'avons pas besoin de couleurs. Afin de ne pas alourdir le code, il n'est pas judicieux d'utiliser un tel afficheur. De plus, le branchement nécessite énormément de sorties Arduino ne laissant très peu de places aux autres composants…
Nous ne retenons pas l'écran 7 segments :
Nous voulons afficher des images (nuages, soleil…) et plusieurs données.

Nous ne retenons pas l'écran LCD :

Certes nous en possédons déjà un mais il faut ''condamner'' énormément de broches sur l'Arduino juste pour l'écran. Ce n'est pas envisageable avec tous les composants que l'on doit brancher (module Bluetooth, module de température).
Il est possible de rajouter un module I²C réduisant ainsi le nombre broches de l'Arduino utilisées. Néanmoins, pour brancher un tel I²C, notre écran doit posséder « des trous » là où actuellement il y a déjà broches. Il serait inconvenant de racheter un écran LCD juste pour brancher un I²C sachant qu'il existe des écrans plus performants tel que les OLED.

Finalement, l'écran de type OLED I²C sera le plus adapté à notre projet.

iii. Les écrans OLED

Afin de déterminer le modèle exact, nous devons connaître les différents types :

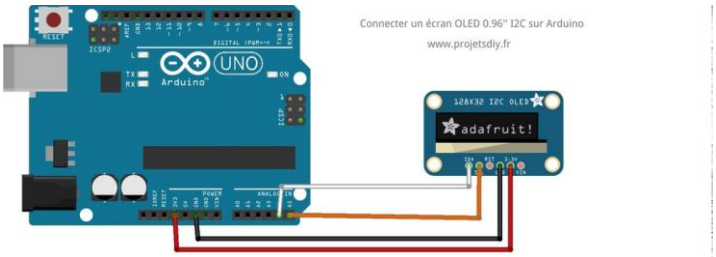
Ecran OLED à matrice passive (PMOLED, PM = Passive Matrix)

Un écran PMOLED nécessite une tension de commande plus élevée pour disposer d'une luminosité comparable à celle offerte par les écrans AMOLED (matrice active). Un des inconvénients imposés par cette technologie est la résolution réduite des écrans, car plus on augmente le nombre de lignes (ou de colonnes) et plus la tension de commande doit être élevée. C'est pourquoi, ces écrans sont de tailles réduites.

Ecran OLED à matrice active (AMOLED, AM = Active Matrix)

Un écran AMOLED comporte, en plus des pixels, des condensateurs destinés à "garder en mémoire" les états d'activation de chaque ligne. La tension requise pour activer les pixels est de ce fait moindre, et cette simple caractéristique suffit pour augmenter la durée de vie de l'écran. De plus, l'écran AMOLED consomme moins d'énergie et présente une vitesse de rafraichissement accrue. Les écrans AMOLED sont ainsi de tailles plus conséquentes.
S'il est possible nous privilégierons donc le type AMOLED.

Il existe beaucoup d'écrans OLED sur le marché et de tailles différentes (par exemple 128x64, 128x96, 160x128, etc.).



Montage de l'écran

iv. Affichage

Sur cet écran il sera donc question d'afficher l'heure actuelle récupérée par le module RTC, les informations telles que la météo ou les rendez-vous récupérés grâce à l'Ethernet Shield, ainsi que la température indiquée par le capteur TMP36.
Nous avons choisi de simplifier certaines données. Pour cela nous allons au préalable coder pixel par pixel un soleil, un nuage et une goutte de pluie, qui sera affiché selon l'information récupérée sur internet.
Enfin sur l'écran sera affiché l'heure du réveil choisie par l'utilisateur et récupérée depuis l'application.

XI Stockage

i. Mémoire de l'Arduino
Les 3 types de mémoires au sein d'un Arduino Uno ATmega328 sont :

- La mémoire FLASH

Cette mémoire sert à stocker les programmes à exécuter, c'est une mémoire qui perdure après arrêt de l'alimentation. L'Arduino uno en est doté de 32ko.

- La mémoire SRAM (Static Read Access Memory)

Cette mémoire sert à stocker des données temporaires (les variables…). C'est une mémoire volatile. L'Arduino en possède 2 ko.

- La mémoire EEPROM (Electrically-Erasable Programmable Read-Only Memory)

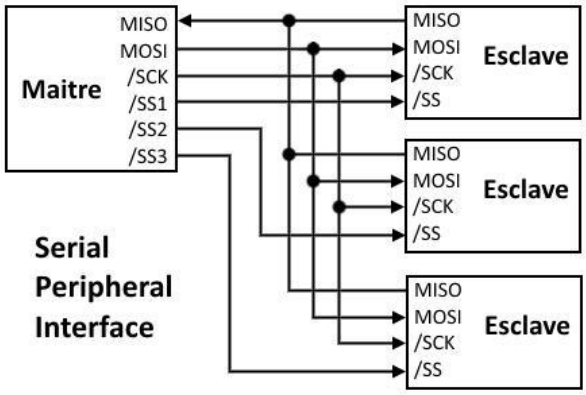
Cette mémoire permet le stockage par le programme de données persistantes. Lente, l'atmega 328 en possède 1 kilooctet. Cette mémoire permet de stocker des données devant être conserver de manière pérenne (après arrêt de l'alimentation).

Arduino	Processor	Flash	SRAM	EEPROM
UNO, Uno Ethernet, Menta, Boarduino	Atmega328	32K	2K	1K

Un simple programme qui allume une LED avec un bouton poussoir et mémorise l'état du bouton poussoir fait déjà 1Ko.
Or nous devons stocker tous les programmes sur l'Arduino ainsi que des musiques et dessins. Il est donc préférable, au risque de la saturer rapidement, d'utiliser une carte sd que nous mettrons sur l'Ethernet Shield.

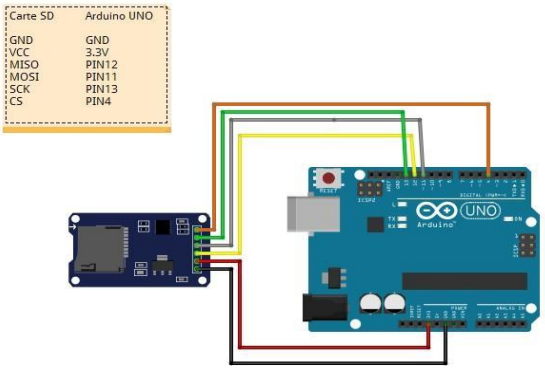
ii. Méthode de communication SPI

Une liaison SPI (Serial Peripheral Interface) est une liaison série synchrone créée par Motorola et qui fonctionne en full duplex (les deux circuits peuvent communiquer en même temps sur le même bus) . Comme pour la liaison I²C, la communication est réalisée selon un schéma maître-esclaves, où le maître s'occupe totalement de la communication. Plusieurs esclaves peuvent être reliés au même bus et la sélection du destinataire se fait par une ligne appelée Slave Select (SS).
Un bus SPI est composé de quatre fils : deux fils de données (MOSI et MISO), un fil d'horloge (SCK) et un fil d'adressage par périphérique (/CS ou /EN).
Les trois premiers fils sont communs à tous les périphériques d'un même bus SPI. Le quatrième fil (/CS) est unique par périphérique et permet au maître de choisir avec quel périphérique il souhaite communiquer.
Il est possible d'avoir plusieurs périphériques SPI sur un même bus. Le maître communique avec un et unique périphérique SPI à la fois en sélectionnant le périphérique de son choix via sa broche /CS ("Chip Select"). Par convention, mettre la broche /CS d'un périphérique SPI à LOW signifie qu'on souhaite communiquer avec celui-ci. Mettre la broche /CS à HIGH signifie que le périphérique doit libérer le bus et rester silencieux.



iii.Choix du shield sd

Il existe dans le commerce de nombreuses cartes filles compatibles Arduino disposant d'un emplacement pour carte SD ou micro SD. On appelle couramment ces cartes filles des "shields". IL y a 3 grands shields SD : le shield Arduino Ethernet, le shield SD de Adafruit, le shield SD de Sparkfun. Puisque le shield Arduino Ethernet ne peut être qu'utilisé, dans notre projet, pour la carte sd et qu'il a un cout considérable, nous choisissons un shield de Adafruit ou Sparkfun.



iV. Choix de la carte sd

Toutes les cartes SD ne sont pas compatibles avec les cartes Arduino. Cela est dû au fait que les cartes Arduino communiquent avec la carte SD via un bus SPI. La communication en mode SPI n'est pas la méthode classique de communication avec une carte SD.

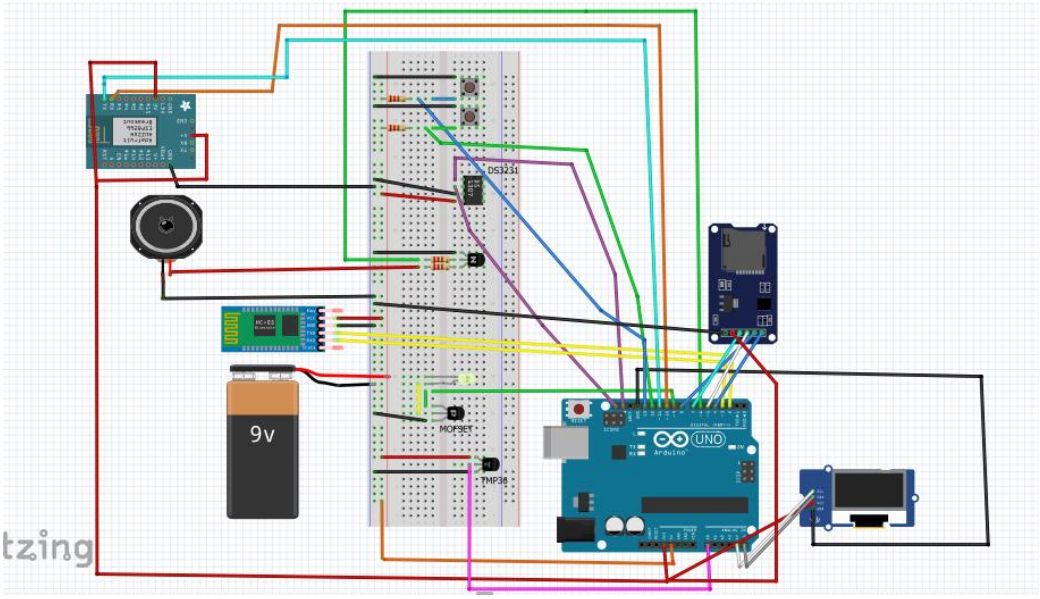
Par exemple, les cartes SDXC (Extrem Capacity) ne sont pas compatibles car leur espace de stockage va de 32GO à 1To…

Il faut choisir une carte micro-sd (pour qu'elle est la taille compatible avec l'Ethernet Shield) de type SDHC (High Capacity) pour qu'elle soit compatible avec l'Arduino. De plus, celle-ci doit avoir une capacité faible (2,4,8 Go) et classe de vitesse lente (2,4,5,8 ou 10).

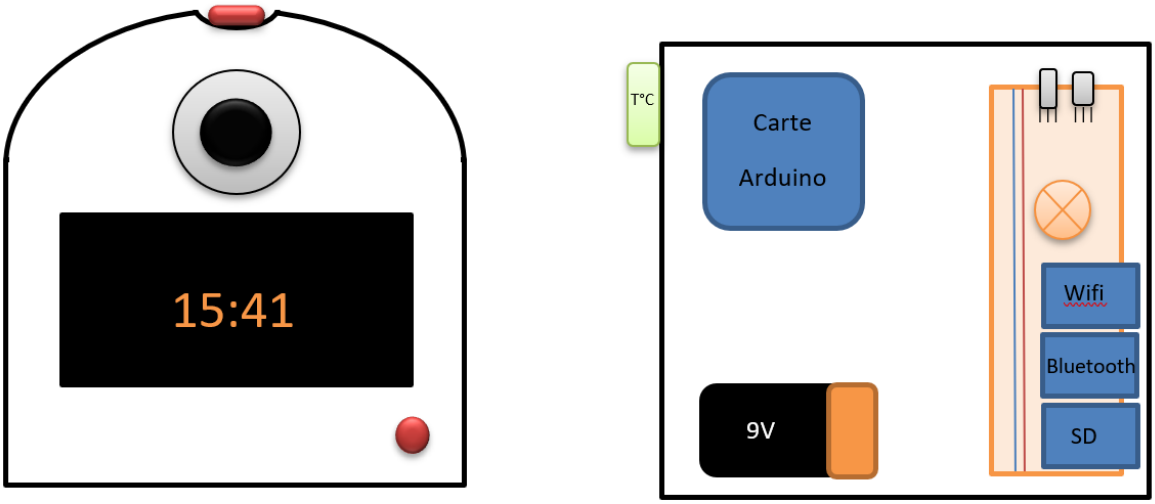
L'indice de vitesse mesure la vitesse de transfert maximale pour l'écriture et la lecture des images depuis et sur une carte mémoire exprimé en méga-octets par seconde. Tandis que, la classe de vitesse est la vitesse minimale de la carte Sd (par exemple, pour la classe 4, la vitesse minimale est de 4Mo/s).

Nous avons donc besoin d'une carte micro-SDHC de classe 4 et de capacité 8Go.

XII Modélisation



Modélisation du circuit électronique final



Aspect du projet

XIII Matériels nécessaires

Voici la liste du matériel qui va être nécessaire à la conception de notre réveil.

- Un module rtc DS3231 Module RTC pour Arduino module d'horloge en temps réel avec AT24C02 EEPROM et Interface I2C de haute précision | AliExpress
- Un module Bluetooth HC-05 HC-05 USB Bluetooth Modul Button Master & Slave 6 Pin Arduino | eBay
- Un capteur de température TMP36 TMP36 GZ Temperatur Sensor / Digital Thermometer / Temperature ARDUINO #A768 | eBay (voir les lots qui sont beaucoup moins onéreux si plusieurs en ont besoin)
- Un transistor MOFSET https://fr.farnell.com/nte-electronics/nte2395/transistor-mosfet-n/dp/1611414?ost=transistor+mofset+nte2395&cfm=true
- Un afficheur OLED I²C https://fr.rs-online.com/web/p/afficheurs-oled/1250232/
- Un haut-parleur 8 Ω et 3W https://fr.farnell.com/visaton/2915/speaker-k-50-wp-8-ohms/dp/1683894?st=haut%20parleur%208ohm%203%20watt
- Un transistor NPN https://fr.rs-online.com/web/p/transistors-bipolaires-bjt/1663204/
- Un ESP8266 -01 https://fr.aliexpress.com/item/4000505567851.html?spm=a2g0o.productlist.0.0.458b7d965w9lZm&algo_pvid=f416f7ff-ddc2-4638-81f3-702925319847&algo_expid=f416f7ff-ddc2-4638-81f3-702925319847-16&btsid=2100bb5116064198150602419e0ed5&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_
- Led 3W, 3000K, 300Ma https://www.amazon.co.uk/dp/B00M3TF3OY/ref=pe_3187911_189395841_TE_3p_dp_1
- Shield sd https://www.ebay.fr/itm/Micro-SD-Storage-Board-Mciro-SD-TF-Card-Memory-Shield-Module-SPI-For-Arduino/264146434611?hash=item3d805aba33:g:q7oAAOSwEcdcQXKf

- Carte micro SDHC : N'ayant pas trouvé de carte microSDHC de classe 4 sur les sites recomandées, voici une carte SDHC 8GO, classe 10. https://fr.rs-online.com/web/p/cartes-sd/7582574/
- Une pile 9V ou 12V (en fonction de ce qui est déjà disponible) sinon : https://www.ebay.fr/itm/1-Pile-Industrial-9V-alcaline-6LR61-PP3-MN1604-6LF22/162112006392?hash=item25bea0e8f8:g:Ty0AAOSwm~daW4GG
- 2 Résistances de 10kΩ et 2 boutons poussoirs que nous avons déjà
- Résistance de 1kΩ https://www.ebay.fr/itm/10-a-100pcs-resistance-1-4w-0-25w-au-choix-1-a-100k-ohms-carbon-resistor/174064664631?hash=item2887100037:g:9pEAAOSwPyBZtE8P
- Résistance de 100Ω https://www.ebay.fr/itm/10-a-100pcs-resistance-1-4w-0-25w-au-choix-1-a-100k-ohms-carbon-resistor/174064664631?hash=item2887100037:g:9pEAAOSwPyBZtE8P