

Bibliographie

BERGER Alexis
LOUIZ Malek

MIROIRDUINO

3 Décembre 2020

Projet PeiP2 - Électronique

Vue d'ensemble

Qui ne s'est jamais surpris à chanter devant son miroir ? Vous trouvez qu'un miroir n'est pas très utile ou manque de technologie ? Nos maisons tendent à être de plus en plus connectées. De nombreux projets de domotique avec Raspberry, Arduino ou Jeedom voient le jour.

Le "Miroirduino" est un projet ayant pour but de créer un smart-mirror à l'aide d'une carte Arduino. Ce miroir connecté possèdera plusieurs fonctionnalités : il affichera l'heure, la date, la météo et la température intérieure/extérieure, il sera aussi capable de jouer et gérer de la musique. De nombreux capteurs seront disposés sur le miroir pour assurer ses fonctionnalités.

Sommaire

1. **Ecran** : Matrice LED pour afficher les informations (heure, météo, musique...).
2. **Données** : Récupérer à l'aide de capteurs, extraire via des sites web, les informations à afficher sur l'écran
 - a. Température
 - b. Météo
 - c. Boutons tactiles
3. **Interactions** : Commander les fonctionnalités du miroir à l'aide de mouvements.
 - a. Capteur de proximité
 - b. Capteur de mouvement
4. **Musique** : Permettre à l'utilisateur d'écouter et de gérer sa musique à travers le miroir.
5. **Problématique** : Gérer diverses problèmes potentiels et savoir les résoudre
6. **Design/Esthétique** : Créer le design du miroir.
7. **Liste du matériel**

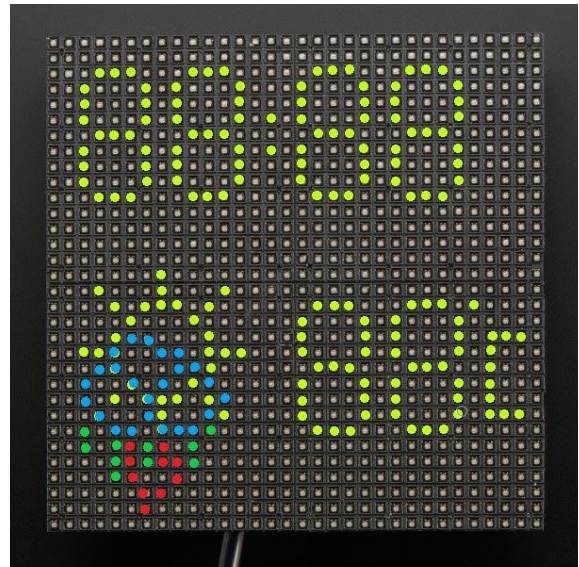
1/Écran

1.1/Matrice LED

Concernant l'affichage de l'heure et de la température, plusieurs options s'offrent à nous : utiliser un écran classique, utiliser une matrice de LED ou encore utiliser un afficheur LCD comme nous possédons déjà.

Etant donnée la puissance de calcul de la carte Arduino, nous pouvons oublier la solution d'utiliser un écran classique. Nombreux projets de smart miroir utilisent Raspberry et permet de gérer l'affichage des données via un écran ou moniteur via HDMI. Utiliser un afficheur comme celui du TD, ne permettra pas un affichage optimal. L'écran est bien trop petit par rapport à la taille du miroir.

L'une des solutions, qui sera la plus adaptée, est d'utiliser un écran de matrice de LED. Il en existe de plusieurs tailles (16x16, 32x16, 32x32, 64x32, 64x64). Ceci est une matrice de taille 32x32. On peut voir que la taille est suffisante pour afficher l'ensemble des données. De plus, la taille de ce genre de matériel est d'environ 16cm x 16cm ce qui semble plutôt adéquate pour la taille du miroir



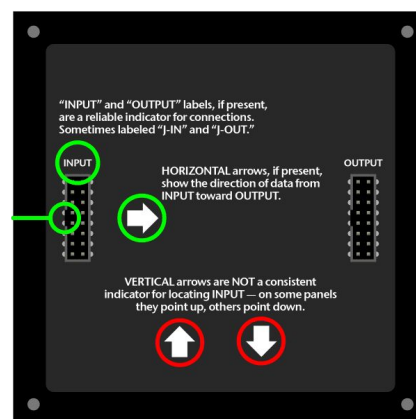
Il existe une multitude de fabricants pour des matrices LED.

Dans la liste des matrices LED, on retient les produits Adafruit qui semblent facile d'utilisation. Ces produits utilisent une librairie commune à de nombreuses matrices LED, la librairie **GFX** ou **PxMatrix** facilitant l'écriture de texte sur la matrice. Les panneaux sont alimentés par du **5V**, il n'est donc pas nécessaire d'utiliser un adaptateur pour l'alimentation. A noter que pour connecter la matrice à l'Arduino, il faut utiliser 12 ports de la carte.

Deux connexions sont possibles pour les matrices LED :

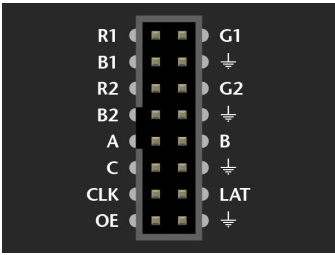
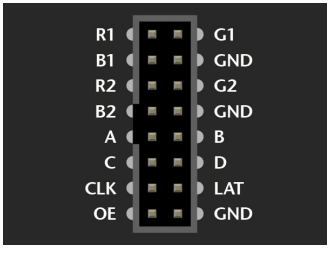
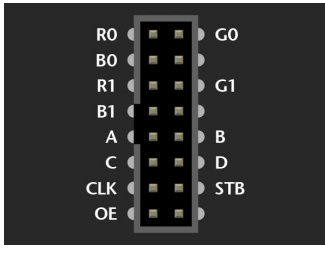
- jumper wire : relier directement les sorties de l'écran aux ports de la carte.
- shield : relier l'écran et l'arduino par l'intermédiaire d'un RGB Shield. C'est un moyen plus rapide et moins complexe de relier la matrice à la carte Arduino. Cependant, le shield d'Adafruit n'est pas compatible avec les Arduino Mega mais qu'avec les Uno et n'est utile que pour des matrices faisant la taille de la carte Arduino. Cette option semble alors compromise.

En général, voici comment se présente l'arrière d'un panneau LED. Les matrices LED sont reliables entre elles, ce qui explique la présence de deux ports de connexion. Comme expliqué plus haut, une matrice de taille 32x32 nous suffit largement pour afficher les informations. Nous devrions alors utiliser qu'un seul panneau LED. Le port INPUT sur le schéma ci-contre est l'entrée par laquelle l'écran reçoit l'information. C'est par ce port que l'on connecte la carte Arduino. Si un deuxième écran doit être relié, on relie alors le port OUTPUT du premier écran au port INPUT du second.

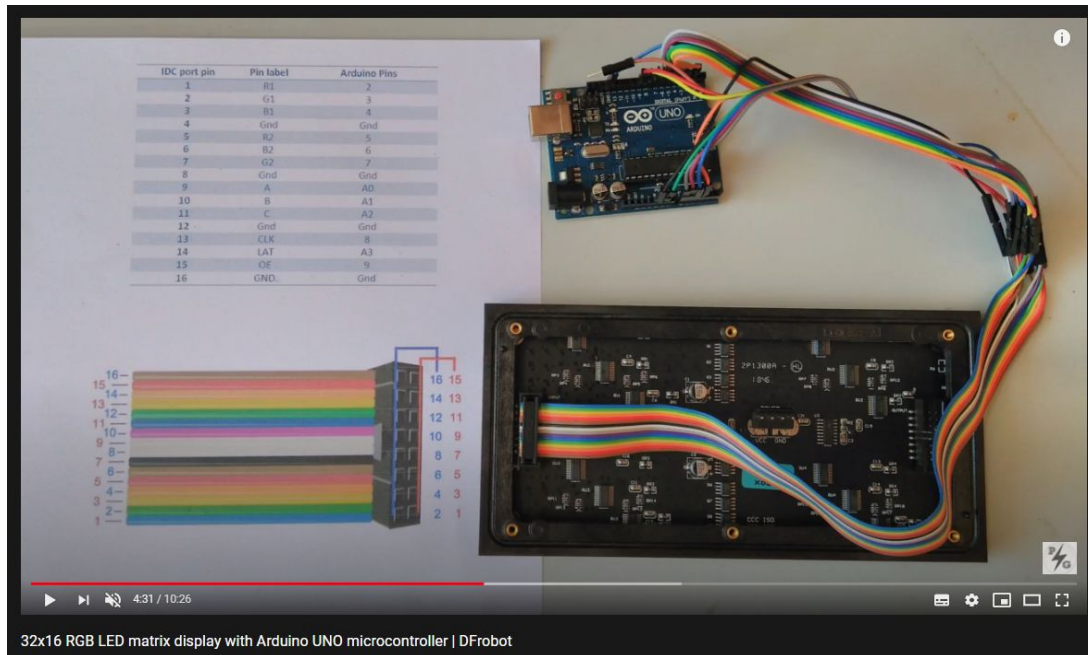


<https://learn.adafruit.com/32x16-32x32-rgb-led-matrix>

En ce qui concerne le port INPUT de l'écran, il se compose de 14 broches. Le port INPUT peut légèrement varier d'une matrice à l'autre en fonction de sa taille.

Taille	32x16	32x32 et 64x32	
Schéma			

Broche	Fonction	Ports UNO	Ports MEGA
R1	Données RVB supérieurs Envoie les données sur la première moitié (moitié haute) de la matrice	2/3/4	24/25/26
G1			
B1			
B2	Données RVB inférieurs Envoie les données sur la deuxième moitié de la matrice	5/6/7	27/28/29
R2			
G2			
A	Ligne de sélection Sélectionne les données de l'écran qui sont actuellement allumées (certaines matrices ont une broche D; quand il n'y a pas on la relie à la masse)	A0/A1/A2 (A3 si D)	
B			
C			
CLK	Clock - Signale de l'arrivée de chaque bit de données	8	11
OE	Output Enable - Éteint les LED lorsque l'on passe d'une donnée à l'autre	9	
LAT	LATCH - Permet de mettre fin à l'envoi de données	10	



<https://www.youtube.com/watch?v=1U4DILN2p44>

Exemple de branchement d'un panneau 32x16 sur une Arduino UNO

1.2/Affichage des données :

Trois options s'offrent à nous. Nous choisirons l'une de ces options en fonction de ceux que l'on veut afficher.

- Si les données à afficher ne sont pas nombreuses, on peut faire un affichage fixe des informations à l'écran avec une mise à jour à un intervalle de temps choisi pour chaque données
- Pour éviter de saturer l'écran et permettre un affichage clair et lisible, on peut procéder à un défilement des informations selon un laps de temps. Toutes les 30 secondes, l'heure laisse place à la météo et la température externe et inversement les 30 prochaines secondes.
- Mettre un place un système (bouton ou mouvement) permettant de faire défiler les informations

Pour afficher les informations sur l'écran, nous aurons besoin de la librairie GFX ou PxMatrix, généralement utilisées pour des écrans matrice. Voici quelques fonctions utiles :

PxMatrix	
Nom	Fonction
display(X,Y,P_LAT,P_OE,P_A,P_B,P_C)	Initialiser les ports de la carte Arduino utilisé pour l'écran
begin(X)	Définir le dispositif d'affichage
setCursor(X,Y)	Mettre le curseur à la position (X,Y)
color565(R,G,B)	Définir sa propre couleur via les niveaux de Rouge, de Bleu et de Vert
clearDisplay()	Réinitialiser l'écran
print("X")	Afficher du texte sur la matrice
setBrightness()	Configurer la luminosité de l'écran
GFX	
Nom	Fonction
drawPixel(X,Y,color)	Dessiner un pixel aux coordonnées (X,Y) et de couleur Color
drawChar(X,Y,String,Color,Size)	Ecrire un texte String de couleur Color et de taille Size aux coordonnées (X,Y)
fillscreen(color)	Initialiser l'écran avec la couleur color

<https://platformio.org/lib/show/5437/PxMatrix%20LED%20MATRIX%20library>

https://wiki.mchobby.be/index.php?title=Tutoriel_Librairie_Adafruit_GFX

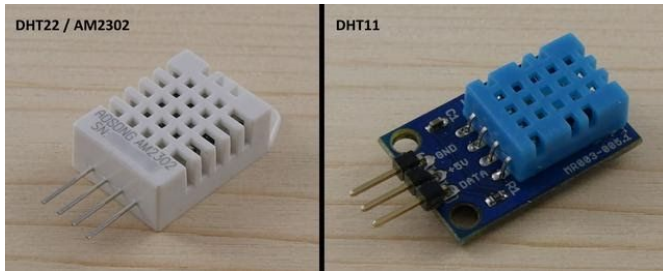
Remarque : Un adaptateur secteur est nécessaire pour alimenter l'écran. Il faudra penser à installer une alimentation adaptée à l'écran pour un fonctionnement optimal : par exemple, l'écran Adafruit du site ci-dessus consomme un courant de **5V** mais de **4A** quand l'écran est entièrement allumé. A titre comparatif, la carte Arduino a besoin d'une intensité de 50mA.

<https://learn.adafruit.com/32x16-32x32-rgb-led-matrix/powering>

2/Données

2.1/Température :

L'une des premières informations que nous souhaitons afficher sur le miroir est la température. Il existe de nombreux capteurs de température et d'humidité sur le marché. Les plus courants



sont les capteurs **DHT11**, **DHT22** et **LM35**. Le DHT22 est plus précis mais plus coûteux que le DHT11. Le LM35, précis et polyvalent, possède de nombreuses fonctionnalités que nous n'utiliserons pas. La température du DHT11 est assez approximative mais semble suffisante. Nous

nous pencherons sur les modules DHT. Généralement, le module possède 4 broches mais il est souvent soudé sur un shield à 3 broches. Niveau branchement, la broche **5V** et GND servent à alimenter le module et la broche DATA est reliée à un port de l'Arduino assurant la transmission des informations. Ces modules utilisent la librairie "**DHT**" associée à la librairie "Adafruit_Sensor".

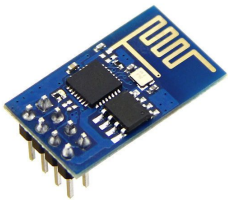
DHT (combinée avec la librairie Adafruit_Sensor)	
Nom	Fonction
dht(PIN,TYPE,16)	Définir la sortie utilisée, le type du module et taux de rafraîchissement
begin()	Lancer la prise de données
readHumidity()	Lire l'humidité en %
readTemperature()	Lire la température en Celsius (TRUE : pour Farenheit)

<https://projetsdiy.fr/mesure-humidite-temperature-capteur-dht11-dht22-arduino-raspberry/>

2.2/Heure et météo :

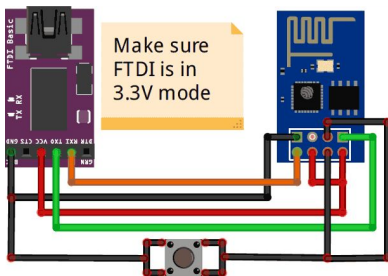
L'heure et la météo sont aussi des données que l'on souhaite intégrer au Miroirduino. Le miroir serait capable d'afficher l'heure, la météo et la température extérieure ainsi que la température dans la pièce.

Pour cette fonctionnalité, nous allons user d'internet en récupérant les relevés météorologiques publiés. Nous aurons alors besoin d'un module Wi-Fi. Deux solutions se présentent à nous : les modules ESP8266 très fréquemment utilisés dans les projets Arduino ou un Ethernet Shield offrant plus de fonctionnalités mais aussi très onéreux. Le ESP8266 semble être un excellent compromis pour informer l'utilisateur de la météo. Il en existe deux types : le module ESP8266 nodeMCU autonome, le ESP8266-01 comme ci-contre.

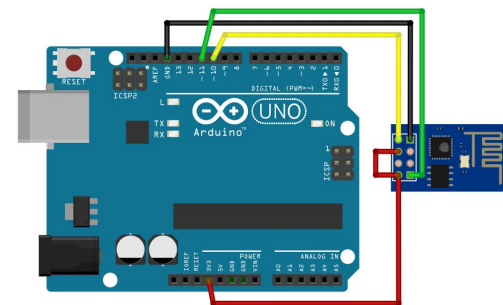


Comment intégrer des données particulières d'un site web dans notre carte Arduino et cela en temps réel ? Le module ESP8266 va nous permettre d'envoyer à la carte Arduino le contenu d'une page internet lettre par lettre. Ainsi il nous faudra choisir les caractères spécifiques qui nous intéressent.

Ce module série possède un microcontrôleur capable de se connecter sur Internet. Généralement utilisé pour communiquer entre l'Arduino et un autre appareil, son utilisation dans ce miroir connecté sera plus passive. En effet, c'est l'IDE de l'Arduino qui nous permettra d'extraire les informations et de les afficher. Le programme permettra d'utiliser une API pour lire les fichiers JSON grâce à des requêtes HTTP.



Niveau branchement, le module s'alimente en 3,3V seulement. Certains modules ESP8266 sont plus facilement programmables via un programmeur USB. Le module FTDI USB to TTL Serial Adapter Module (3.3V) permet de communiquer avec le module WiFi sans Arduino.



Pour relier le module au réseau internet, nous aurons besoin d'utiliser les commandes manuelles **AT** de l'Arduino. Ensuite, le module devra être capable d'envoyer une requête HTTP à un site météorologique. L'un des plus utilisés et des plus simples est le site openweathermap.org . Le site est capable de générer des API en Json. L'Arduino devra donc être capable de déchiffrer le Json pour extraire les données utiles.

Commandes AT	
Nom	Fonction
AT+RST	Relancer le module
AT+CWMODE = X	Configurer le mode WiFi (1:Sta / 2:AP /3:both)
AT+CWLAP	Listes les points d'accès
AT+CWJAP = "Nom" , "MdP"	Connecter au réseau

Pour cela, il nous faudra utiliser la librairie "**ArduinoJson**". Elle a pour but de convertir la requête HTTP et d'extraire les données. Afin d'assurer la transmission et la réception, le module ESP8266 nécessite la librairie "**ESP8266WiFi**".

ArduinoJson	
Nom	Fonction
jsonBuffer(capacity)	Création du buffer
jsonBuffer.parseObject(client)	Création du tableau contenant la réponse à notre requête

Assistant ArduinoJson pour nous aider : <https://arduinojson.org/v5/assistant/>

Remarque : si l'on configure le module WiFi via l'Arduino, il faut penser à désactiver la liaison série du module pour pouvoir téléverser et communiquer avec. Avant de procéder au câblage, il ne faut pas oublier de charger le programme Blink sur la carte, disponible via l'IDE Arduino (Onglet "Fichier" -> "Exemple" -> "1.Basics"). Pour communiquer avec l'ESP8266 avec les commandes AT, avec le moniteur série, il faut configurer la liaison série en Both NL & CR (Newline and Carriage Return) avec un baud rate de 115200. Afin d'appliquer ces changements, il faut penser à reset la carte avec le bouton situé sur la carte.

Remarque 2 : Lors de la configuration réseau du module, il est nécessaire que le port TX et RX du module doivent être branchés sur les ports 0 et 1 de la carte Arduino. Cependant, certains modules sont configurés en liaison croisée selon les fabricants, en cas d'absence de réponses, il faudra inverser les branchements du port 0 et 1. On pourra par la suite brancher le module sur d'autres ports et restaurer la connexion réseau via un programme d'initialisation automatique pour que l'Arduino puisse fonctionner normalement.

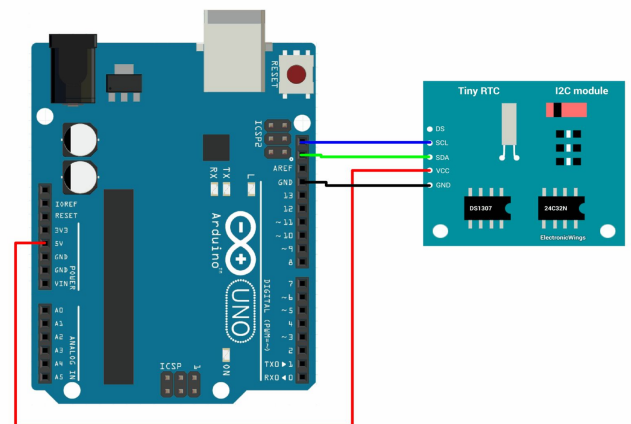
Exemple d'extraction des données météorologiques avec un ESP8266-01 :

<http://weldybox.com/comment-utiliser-lapi-openweather-map-avec-lesp8266/>

<https://openclassrooms.com/forum/sujet/extraction-de-donnees-d-une-page-web-arduino>

En ce qui concerne l'heure, il existe une autre possibilité. Au lieu d'extraire l'heure exacte via un site web grâce au module ESP8266, il existe des modules RTC (Real Time Clock). Le module DS1307 est une horloge numérique autonome qui est capable de donner l'heure quand on le lui demande. Ce module fonctionne en **5V**. Le module **RTC DS1307** est constitué de 5 broches : 5V, GND, SDA, SCL et SQW. SDA et SCL sont les deux broches assurant la transmission des données entre le module et les ports analogiques de la carte Arduino. Pour récupérer l'heure, il nous faudra utiliser la librairie RTCLib et plus précisément la librairie **DS1307**.

DS1307	
Nom	Fonction
year()	Récupérer l'année
month()	Récupérer le mois
day()	Récupérer le jour
hour()	Récupérer l'heure
minute()	Récupérer les minutes



3/Interaction

3.1/Glisement de la main pour passer les musiques

Plutôt que d'utiliser un simple bouton pour gérer le passage d'une musique à la suivante, il suffirait de passer sa main de gauche à droite à quelques centimètres du haut de cadre pour écouter le morceau suivant.

Pour cela, nous utiliserons des capteurs HC-SR04, capteurs de distances utilisés en TD. Ces capteurs permettent de calculer une distance, mais nous pouvons les modifier pour qu'il devienne une sorte de capteur de proximité. Il suffit de définir une valeur seuil au-delà de laquelle l'appareil de calculer la distance et de renvoyer TRUE lorsqu'il rencontre un obstacle (en l'occurrence ici notre main) en dessous de cette valeur.



En plaçant deux capteurs à quelques centimètres l'un de l'autre, on serait capable de détecter un mouvement du capteur 1 au capteur 2 et inversement.

Le module **HC-SR04**, un capteur de distance par ultrasons, est suffisant pour cette fonctionnalité. Le capteur est capable de détecter une main à quelques centimètres. Le module possède 4 broches : 2 broches d'alimentation et 2 broches de communication. **Echo** donne l'ordre de créer les impulsions sonores, **Trigger** renvoie le temps parcouru par le son. Il existe une librairie utile, la librairie **"NewPing"** :

NewPing	
Nom	Fonction
sonar(trig,echo,X)	Initialise les ports trigger, echo et détermine la distance max X calculée.
sonar.ping_cm()	Renvoie la distance en centimètres.

Cependant, la difficulté va être de créer un algorithme capable de détecter ce mouvement.

Lorsque l'un des capteurs détecte un obstacle, il lance un timer. Si le deuxième capteur capte un obstacle avant la fin du timer, on en déduit que le mouvement part du capteur 1 pour arriver au capteur 2.

Le défi est donc de trouver ou créer une fonction de Arduino permettant de lancer un chrono ou timer sans arrêter l'ensemble du programme. On peut exclure la fonction "**delay**" et ses variantes du brainstorming.

millis() est une fonction de base d'Arduino permettant de retourner le nombre de millisecondes passées depuis le lancement du programme. Ainsi à l'aide d'une double boucle conditionnelle, on peut déterminer le passage de la main selon une direction (gauche ou droite) en fonction des deux capteurs de distance.

3.2/Des boutons tactiles pour mettre sur pause et modifier le volume

Afin de mettre sur pause et de gérer le volume, nous avons pensé à du tactile étant donné la surface vitrée que nous offre le miroir. Il existe des capteurs tactiles capables de détecter un doigt malgré une couche par dessus. Ces capteurs seront alors placés sur le plexiglass.

Il existe de nombreux capteurs capacitifs : des communicateurs ON/OFF lumineux, des capteurs simples, etc... Le module TTP223B est un capteur tout simple assez puissant pour détecter sous une fine couche. On peut toujours se demander si le capteur est assez puissant pour détecter malgré le plexiglas, mais comme dirait le proverbe "qui ne tente rien, n'a rien". Donc à tester !



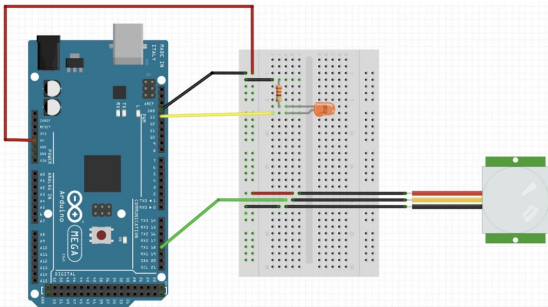
Niveau électronique, 3 broches : VCC et GND pour l'alimentation, SIG pour la transmission d'informations à l'Arduino. Aucune librairie n'est nécessaire pour exploiter les données transmises par le module.

<https://www.youtube.com/watch?v=WdfA57TUyII>

3.3/Capteur de mouvement

Pour des soucis d'économie et de consommation, la présence d'un capteur de mouvement ferait usage d'interrupteur On/Off. On déterminera par la suite une valeur temporelle seuil pour laquelle le miroir éteindrait l'affichage et les capteurs de distance. Seule la musique, si elle n'est pas mise sur pause, continuerait à fonctionner.

HC-SR501 est le module qu'il nous faudra pour permettre cette fonctionnalité.



Pour les branchements, aucune grande difficulté n'est à prévoir. Les broches aux deux extrémités servent à l'alimentation et la broche du milieu renvoie les données à la carte Arduino. Sur le schéma ci-contre, le module est branché sur le port **18** de la carte.

Le module est alimenté par un courant de **5V** discontinu et de **20mA**. Il a un angle de détection de **110°**. Lorsqu'il détecte une personne ou un animal, il passe de l'état bas (LOW) à l'état (HIGH). On retrouve deux potentiomètres sur le module : celui de gauche permettant de modifier le délai de changement d'état et celui de droite permettant de gérer la portée du dispositif (min : **3m** / max : **7m**)

<https://euro-makers.com/projet/99/tutoriel-arduino-capteur-pir-passive-infrared-sensor>

Remarque : Aucune librairie n'est nécessaire pour son utilisation.

4/Musique

Qui ne s'est jamais surpris à chanter sous la douche ou devant son miroir ? C'est pour cela que nous avons pensé à ajouter une fonctionnalité pour écouter de la musique via le miroir.

Le projet pourrait être difficile à réaliser. Pour pallier cela, nous avons pensé à plusieurs alternatives.

Notre première volonté serait d'installer des haut-parleurs avec un module MP3 pour permettre la lecture de musique au travers du miroir. En cas d'échec, nous pourrions tout de même ajouter une fonctionnalité permettant de gérer sa musique (mettre en pause, reprendre, passer à la musique suivante) en envoyant une requête via le module WiFi à notre téléphone pour qu'il exécute ses fonctionnalités.

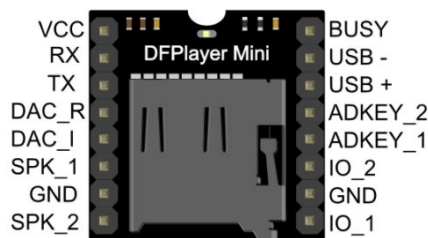
Pour jouer de la musique, nous aurons besoin d'un module MP3. Le module **DFPlayer Mini MP3 player** semble être l'un des plus fiables et des plus faciles d'utilisation. Une carte **micro-SD** et un haut-parleur sont aussi nécessaires. Les commandes pour changer de musique et mettre sur pause sont gérées par les capteurs de distance comme expliqué précédemment. Une roulette/potentiomètre ou deux boutons permettent de baisser le volume.

Documentation pour le DFPlayer Mini MP3 player :

https://wiki.dfrobot.com/DFPlayer_Mini_SKU_DFR0299

Branchements :

Pin Map

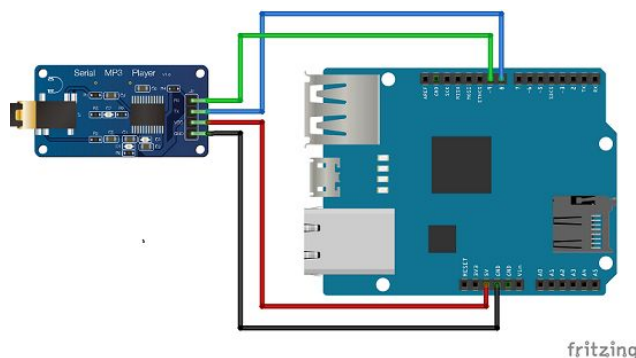


Pin	Description	Note
VCC	Input Voltage	DC3.2~5.0V;Type: DC4.2V
RX	UART serial input	
TX	UART serial output	
DAC_R	Audio output right channel	Drive earphone and amplifier
DAC_L	Audio output left channel	Drive earphone and amplifier
SPK2	Speaker-	Drive speaker less than 3W
GND	Ground	Power GND
SPK1	Speaker+	Drive speaker less than 3W
IO1	Trigger port 1	Short press to play previous (long press to decrease volume)
GND	Ground	Power GND
IO2	Trigger port 2	Short press to play next (long press to increase volume)
ADKEY1	AD Port 1	Trigger play first segment
ADKEY2	AD Port 2	Trigger play fifth segment
USB+	USB+ DP	USB Port
USB-	USB- DM	USB Port
BUSY	Playing Status	Low means playing High means no

La librairie associée à ce module est la librairie “**DFRobotDFPlayerMini**”.

DFRobotDFPlayerMini	
Nom	Fonction
play(N);	Jouer la N-ième musique.
volume(X); volumeUp() volumeDown()	Mets le volume à X. X doit être compris entre 0 et 30.
previous() next()	Passer à la musique précédente/suivante.
outputDevice(SD)	Emplacement des fichiers.
readVolume()	Lire le volume.

Un autre module est disponible sur le marché. Le module **Serial Music MP3 Player** est une alternative au module de DFRobot. Composé de 4 broches et d'un port jack, il possède aussi un compartiment micro-SD pour permettre la lecture de musique via Arduino. Le module est alimenté par un courant compris entre 3,2 et 5,2V pour une intensité de **200mA**.



Concernant les branchements, les deux broches à gauche servent à l'alimentation. La broche RX est reliée au port 14 de l'Arduino Mega et la broche TX au port 15 de la carte. Ces deux broches permettent de transmettre les commandes au module et de recevoir les résultats.

<https://www.youtube.com/watch?v=HSLKefx1VK4>

La librairie associée au module Serial MP3 Player est la librairie “**SerialMP3Player**” :

SerialMP3Player	
Nom	Fonction
SerialMP3Player mp3(RX,TX);	Initialiser les ports de la carte Arduino
begin()	Commencer la communication entre le module et l’Arduino
sendCommand(CM D_SEL_DEV,0,2)	Sélectionner la carte SD
play()	Lecture
pause()	Mettre sur pause
playNext()	Musique suivante
volUp() / volDown()	Monter/baisser le volume
setVol(X)	Définir le volume au niveau X
sleep()	Eteindre le module
wakeup()	Allumer le module

<https://github.com/salvadorrueda/SerialMP3Player/blob/master/examples/BasicCommands/BasicCommands.ino>

Remarque : Afin de diffuser le son sur des **haut-parleurs**, il sera nécessaire de brancher un **amplificateur** à la sortie jack du module, et cela quel que soit le module choisi. Une autre solution serait de relier avec un câble jack/jack le module directement à une enceinte en dehors du cadre. Nous préférons tout de même la première solution.

5/Problématique

5.1/Gestion des ports d'entrée/sortie de la carte

L'une des très certaines problématiques auxquelles nous devons faire face est celle de la gestion des ports d'entrée/sortie de la carte Arduino. En effet, notre projet comporte de nombreux modules (environ une dizaine). La carte Arduino Uno ne possède que 14 ports de connexion. Il sera donc préférable aussi d'un point de vue puissance de calcul et espace de stockage de concevoir le projet à l'aide d'une carte Arduino Mega.

	Arduino Uno	Arduino Mega 2560	Arduino Micro
			
Price Points	\$19.99-\$23.00	\$36.61 - \$39.00	\$19.80 - \$24.38
Dimension	2.7 in x 2.1 in	4 in x 2.1 in	0.7 in x 1.9 in
Processor	Atmega328P	ATmega2560	ATmega32U4
Clock Speed	16MHz	16MHz	16MHz
Flash Memory (kB)	32	256	32
EEPROM (kB)	1	4	1
SRAM (kB)	2	8	2.5
Voltage Level	5V	5V	5V
Digital I/O Pins	14	54	20
Digital I/O with PWM Pins	6	15	7
Analog Pins	6	16	12
USB Connectivity	Standard A/B USB	Standard A/B USB	Micro-USB
Shield Compatibility	Yes	Yes	No
Ethernet/Wi-Fi/Bluetooth	No (a Shield/module can enable it)	No (a Shield/module can enable it)	No

<https://www.arrow.com/fr-fr/research-and-events/articles/arduino-uno-vs-mega-vs-micro>

5.2/Gestion de l'espace stockage de la carte Arduino

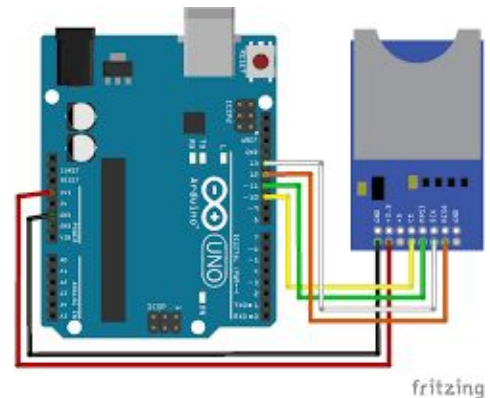
Notre programme en vue des différentes fonctionnalités qu'il possède risque d'être assez gourmand en termes de stockage. Etudions alors les différentes mémoires d'Arduino.

Arduino possède 3 types de mémoire :

- Mémoire **FLASH** : sert à stocker les programmes à exécuter. C'est le "disque dur" de la carte Arduino // Capacité pour Arduino Mega : 256 Ko.
- Mémoire **SRAM** : sert à stocker les données temporaires (variables statique, globale ou temporaires, appels de fonction, interruptions) // Capacité : 8 Ko.
- Mémoire **EEPROM** : sert à stocker des données persistantes // Capacité : 4Ko.

Solution au manque de mémoire :

- Optimiser la taille des variables (ne pas utiliser float si on travaille qu'avec des entiers)
- Utiliser une partie de la mémoire flash pour des données statiques. Cela se fait via l'instruction `PROGMEM` (librairie `avr/pgmspace.h`)
- Ajouter de la mémoire de stockage grâce à un module SD. En effet, l'ultime recours est d'ajouter un module de stockage SD contenant des fonctions appelées dans la boucle du programme afin d'alléger la mémoire FLASH et SRAM.



<https://quai-lab.com/arduino-ses-memoires/>

6/Design

6.1/Cadre

Notre objectif est de créer un miroir connecté possédant toutes les fonctionnalités décrites ci-dessus. Dans cette partie, nous allons réfléchir à la conception du cadre et à la mise en place des composants pour rendre le miroir fonctionnel et ergonomique.

En ce qui concerne l'aspect extérieur, nous avons choisi de dissocier le cadre qui donnera l'allure générale miroir et la partie "arrière" qui contiendra les composants.

Pour le **cadre**, nous avons le choix entre le bois et le métal : niveau esthétique, le bois donne un côté chaleureux tandis que le métal rentre dans un style contemporain. Niveau pratique, le bois se travaille beaucoup plus facilement que le métal et si le bois ne nous convient pas, nous avons toujours la possibilité de le repeindre.

Pour le **caisson** (partie arrière qui contiendra les composants), le PVC nous semble être le meilleur compromis. Léger, facile à travailler et économique, le plastique est surtout perméable aux radios fréquences.

Il ne faut pas oublier qu'il faut laisser des espaces et faire des trous pour certains composants.

- Deux trous à l'arrière pour placer les haut-parleurs
- Deux encoches au sommet du cadre pour les capteurs de distances
- Deux trous sur les côtés pour alimenter les bandes LED
- Un pour l'alimentation générale

Pour la vitre, il faut choisir un matériel réfléchissant mais pas totalement opaque pour permettre l'affichage des données par la matrice LED. L'une des solutions les plus efficaces et les moins coûteuses est de placer une plaque de **plexiglas** entre le cadre et le caisson recouvert d'un côté par un film semi-réfléchissant ou film **sans tain**. Le verre étant trop coûteux, lourd et fragile, le plexiglass fera l'affaire. Note à nous-même qu'il faudra bien appliquer le film sur la plaque pour que le rendu soit optimal et que nous ne nous effrayions pas avec notre reflet.

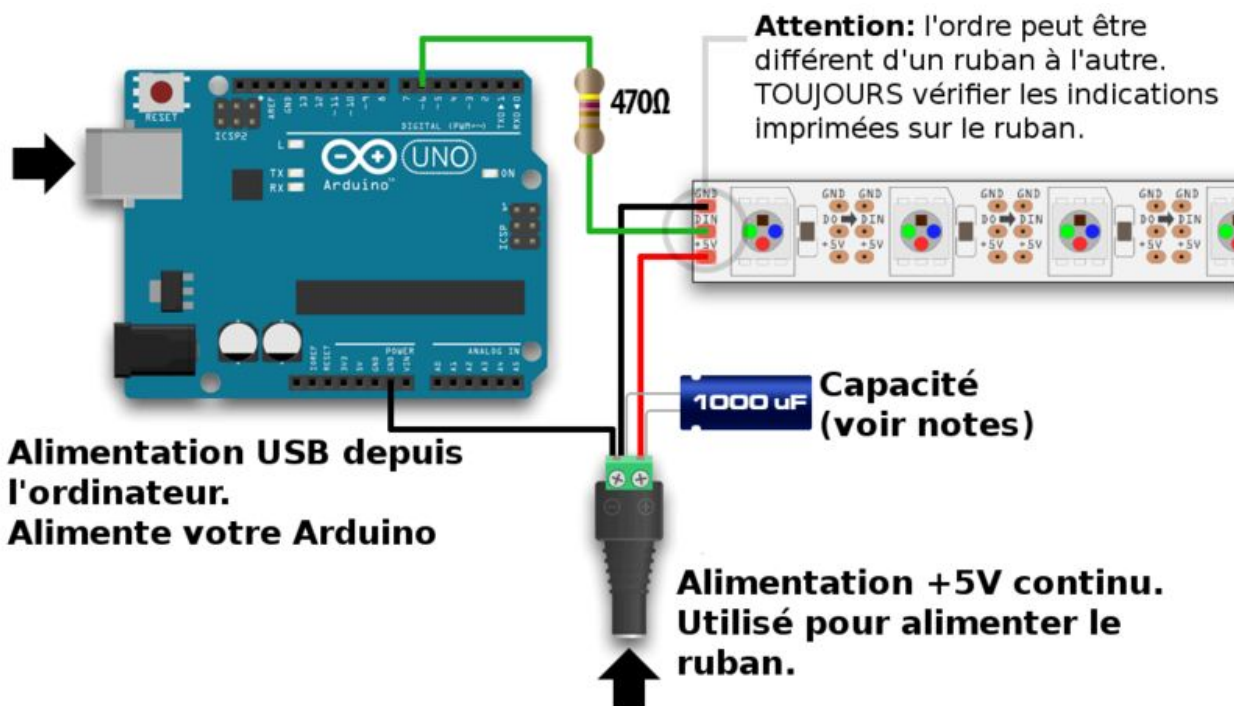
6.2/Bandes LED



Pour l'aspect pratique mais surtout esthétique, nous pensions ajouter deux bandes LED programmables sur les bords gauche et droite du miroir. Généralement, les bandes LED programmables possèdent trois broches : **5V**, GND et DATA. Chaque bande sera reliée par un fil avec l'Arduino. La librairie "**Neopixel**" est la librairie la plus courante pour les bandes LED.

<http://www.fablabredon.org/wordpress/2019/01/02/ruban-de-led-le-retour/>

Afin de donner un côté visuel "high-tech" de l'extérieur, les deux bandes LED réagiront en fonction de vos actions sur le miroir. L'avantage qu'elles soient programmables est que nous pouvons créer des animations lumineuses. Par exemple, lorsque la musique est mise sur pause, les lumières deviennent rouges, ou lorsque l'on passe à la musique suivante, la bande de droite clignote.



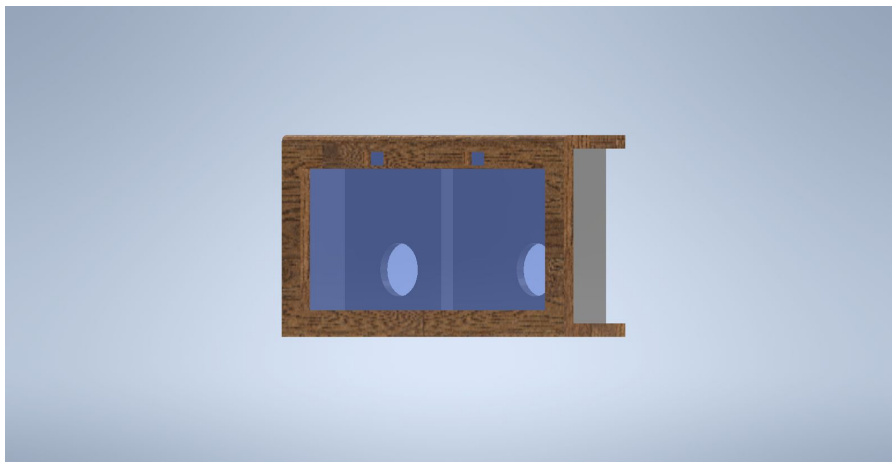
Ces animations seront permises par la librairie **“Neopixel”** développée par la société Adafruit. Voici un exemple de fonctions qui peuvent nous servir :

Neopixel	
Nom	Fonction
setPixelColor(n, r, g, b) setPixelColor(n,color)	n : numéro de pixel r,g,b : niveau de luminosité des couleurs (entre 0 et 255)
show()	Met à jour le ruban LED
setBrightness(X)	Modifie la luminosité

https://wiki.mchobby.be/index.php?title=NeoPixel-UserGuide-Arduino&mobileaction=toggle_view_desktop#

6.3/Croquis

A l’aide du logiciel **“Autodesk Inventor”** que nous avons utilisé en PeiP1, nous avons réalisé le design du miroir en prenant compte les contraintes des différents composants.



Conception 3D et croquis disponibles sur : <http://alexis-berger.online/gallerie.html>

Module	Fonction	Qté
Matrice LED 32x32	Ecran pour afficher les informations	1
Arduino Mega	Carte Arduino programmable	1
HC-SR04	Capteur de distance // Passer d'une musique à une autre	2
HC-SR501	Capteur de mouvement // Allumer l'écran et d'autres fonctions	1
Tactil sensor	Capteur tactile // Mettre sur pause et touches Volume	3
DFPlayer Mini	Module son // Diffuser de la musique	1
Amplificateur	Amplifier le son pour haut-parleurs	1
Haut-parleur	Diffuser le son	2
Carte SD	Carte pour stocker la musique	1
DHT11	Capteur de température et d'humidité	1
ESP8266	Module Wi-Fi // Récupérer les informations de météo et transmettre d'autres informations	1
Ruban LED	Éclairer et animer le miroir	2
Adaptateur secteur	Alimenter l'écran avec l'intensité nécessaire	1
Cadre	Cadre en bois	1
Boîtier	Boîtier en PVC pour contenir les composants	1
Plexiglass	Plaque de plexiglass transparent ou sans teint	1

Conclusion

Le Miroirduino sera donc un smart-mirror programmé via un **Arduino Méga**. Composé d'un cadre en bois, il aura la possibilité d'afficher plusieurs données à travers une plaque de plexiglas sans tain à l'aide d'une **matrice LED** de taille 32x32. Le plexiglas sans tain permettra de refléter l'image comme un miroir et de laisser passer la lumière de l'écran.

Grâce au capteur de température **DHT11**, du module WiFi **ESP8266** et du module **RTC**, le miroir affichera l'heure, la météo extérieure de la journée, la date, la température intérieure et extérieure. Un bouton tactile (**Capacitive sensor**) placé sous la vitre permettra de passer d'une donnée à l'autre.

Trois autres boutons tactiles seront placés pour gérer la musique. En effet, le miroir possédera un module MP3 (**DFPlayer Mini Player** ou **Serial MP3 Player**) relié à deux haut-parleurs avec amplificateurs afin de diffuser la musique. Ces trois boutons permettront de modifier le volume (+ et -) et de mettre sur pause ou de reprendre la lecture. Deux capteurs de distances **HC SR-04**, placés sur le haut du cadre en bois, passeront d'une musique à une autre en balayant de gauche à droite (ou de droite à gauche) devant le miroir.

De plus, deux **bandes LED** programmables seront placées sur les côtés afin d'éclairer et de créer des animations lors d'une interaction avec le miroir.

Enfin, dans un souci d'économie d'énergie, un capteur de mouvement **HC SR-501** sera capable de détecter les mouvements. En cas d'inactivité prolongée, l'écran du miroir s'éteindra.