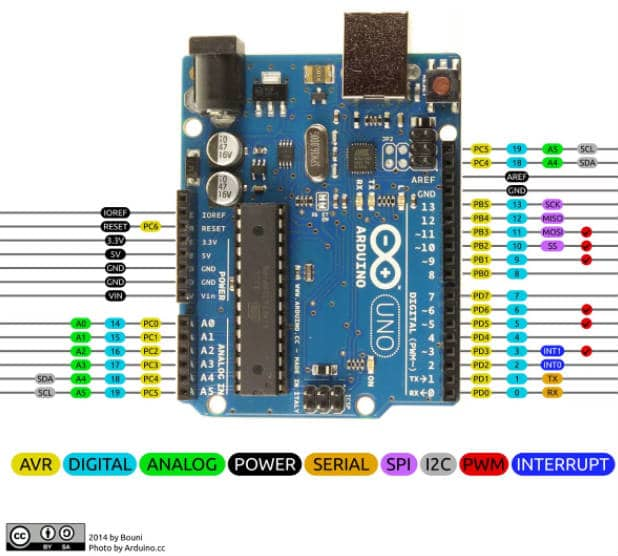
Rapport Arduino 1:

Présentation de la carte Arduino UNO et son microcontrôleur

La carte Arduino UNO:

L'**Arduino Uno** est l'une des cartes de développement compactes les plus demandées du marché. Tenant confortablement dans la paume de la main (68 x 53 mm), la carte est basée sur le **microprocesseur amovible ATMEL ATmega328P**, qui est doté d'une **résolution 8 bits** et de son **propre chargeur d'amorçage** (bootloader) USB pour la reprogrammation. Avec **32 Ko de mémoire flash**, **2 Ko de mémoire SRAM** et **1 Ko de mémoire EEPROM**, ce petit processeur peut prendre en charge la plupart des tâches que vous pouvez lui confier.  
  
La carte elle-même comporte un cristal de quartz assurant une c**adence d'horloge** de **16 MHz**, un raccordement USB, une prise d'alimentation et une en-tête ISCP. L'Arduino Uno est dotée de six broches variables et de **20 broches d'E/S**, avec la possibilité de faire fonctionner six des sorties numériques en tant que sorties [PWD](https://en.wikipedia.org/wiki/Pulse-width_modulation). Reportez-vous au graphique ci-dessous pour un mappage détaillé des broches susmentionnées.

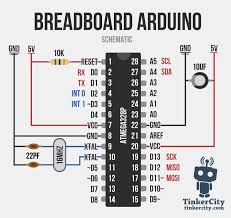


L'accessibilité universelle de la carte Arduino Uno s'explique en partie par le fait que Arduino dispose d'un environnement de développement intégré (IDE) L'IDE open source utilise Java pour la création de commandes et de programmes. Du point de vue de l'utilisateur, l'IDE d'Arduino comporte plusieurs fonctionnalités très utiles pour écrire du code à la fois propre et performant, par exemple les fonctions copier/coller du texte et de mise en évidence de la syntaxe. Comme il s'agit d'un environnement open source, l'IDE d'Arduino possède une bibliothèque communautaire associée, qui permet aux nouveaux utilisateurs de trouver des infrastructures de programmes plus adaptées à leurs besoins et qui laisse les codeurs chevronnés publier leurs projets et leurs idées pour en expliquer la syntaxe à ceux qui ne connaissent pas le langage.

Les **shields Arduino** constituent un autre aspect formidable de la carte Arduino Uno. Les shields sont des modules matériels complémentaires qui assurent l'interface directe avec les en-têtes de broche d'Uno et qui peuvent renforcer leurs capacités. Un shield se compose d'une carte de circuit imprimé intégrant des composants et des raccordements pour faciliter ses fonctionnalités spécifiques et d'un logiciel pour communiquer ses capacités à la carte Uno. Les shields peuvent également s'empiler les uns sur les autres pour ajouter un grand nombre de fonctionnalités à une carte Arduino Uno.

Certains shields Arduino sont très demandés, par exemple le [shield Wi-fi](https://www.arduino.cc/en/Guide/ArduinoWiFiShield), qui permet à la carte Uno d'utiliser la connectivité sans fil et le [shield moteur](https://www.youtube.com/watch?v=6qINU4rLZpQ), qui permet à l'Uno de contrôler les moteurs à courant continu ou pas-à-pas. Grâce aux shields, tout le monde peut aisément intégrer des fonctionnalités clés dans un système Arduino Uno avec un minimum d'effort pour le codage et encore moins pour le bricolage. Grâce à la vaste gamme de shields Arduino, il est possible de simplifier et d'améliorer les performances et la mise en œuvre de quasiment tous les projets.

Microcontrôleur Arduino UNO (Mikon ou ATMega328P):

[Fonctionnement et Programmation](http://perso-laris.univ-angers.fr/~cottenceau/ArduinoCottenceau2016.pdf)

Pour comprendre le fonctionnement de la carte et comment coder le microcontrôleur cliquez sur le lien ci-dessus en bleu.

Installation de l’IDE Arduino pour Programmer la carte

Pour l’installation de l’IDE suivre le lien <https://www.arduino.cc/en/Main/Software>

### Le code minimal

Avec Arduino, nous devons utiliser un *code minimal* lorsque l’on crée un programme. Ce code permet de diviser le programme que nous allons créer en deux grosses parties.

// fonction d'initialisation de la carte

**void setup()**  
 **{**  
 // contenu de l'initialisation  
 **}**  
  
// fonction principale, elle se répète (s’exécute) à l'infini  
 **void loop()  
 {**  
 // contenu de votre programme  
 **}**

Le code minimal d’un programme Arduino

Vous avez donc devant vous le code minimal qu’il faut insérer dans votre programme. Mais que peut-il bien signifier pour quelqu’un qui n’a jamais programmé ?

#### **La fonction setup**

Dans ce code se trouvent deux fonctions. Les fonctions sont en fait *des portions de code*.

// fonction d'initialisation de la carte  
void setup()  
{  
 // contenu de l'initialisation  
 // on écrit le code à l'intérieur  
}

Zoom sur la fonction setup

Cette fonction **setup()** est appelée *une seule fois* lorsque le programme commence. C’est pourquoi c’est dans cette fonction que l’on va écrire le code qui n’a besoin d’être exécuté une seule fois. On appelle cette fonction : **fonction d’initialisation**. On y retrouvera la mise en place des différentes sorties et quelques autres réglages. C’est un peu le check-up de démarrage. Imaginez un pilote d’avion dans sa cabine qui fait l’inventaire *patte 2 en sortie, état haut ? - OK - timer 3 à 15 millisecondes ? - OK …*

Une fois que l’on a initialisé le programme il faut ensuite créer son "cœur", autrement dit le programme en lui même.

#### **La fonction loop:**

La fonction principale, elle se répète (s’exécute) à l'infini  
void loop()  
{  
 // contenu de votre programme  
}

Zoom sur la fonction principale

C’est donc dans cette fonction **loop()** où l’on va écrire le contenu du programme. Il faut savoir que cette fonction est appelée en permanence, c’est-à-dire qu’elle est exécutée une fois, puis lorsque son exécution est terminée, on la ré-exécute et encore et encore. On parle de **boucle infinie**.

A titre informatif, on n’est pas obligé d’écrire quelque chose dans ces deux fonctions. En revanche, il est **obligatoire** de les écrire, même si elles ne contiennent aucun code !

**Résumé**:

La carte arduino Uno est une carte à puce de 14 pattes de chaque côté d’ou 28 pattes au total.

Les broches Tx et Rx sont réservées pour la communication entre la carte avec votre ordinateur.

Les broches en entrées 2 -> 13 traitent des informations en entrées numériques.

Les broches A0 -> A5 traitent des informations en entrées analogiques.

Et tous les broches D0 -> D13 sont des sorties numériques.

Pour l’etude de quelques codes de bases suivre le lien suivant <https://zestedesavoir.com/tutoriels/686/arduino-premiers-pas-en-informatique-embarquee/743_gestion-des-entrees-sorties/3420_notre-premier-programme/>

### Shield wifi et la carte arduino:

• Le [shield Wifi](http://arduino.cc/en/Main/ArduinoWiFiShield) est un adaptateur sans-fil (client) pilotable depuis la carte Arduino au travers de l'interface SPI grâce à la librairie logicielle [Wifi](http://arduino.cc/en/Reference/WiFi).

Du point de vue des services Ethernet, Internet et web, il peut fonctionner en tant que client (qui va chercher les infos sur un serveur) ou en tant que serveur (qui fournit les infos que les clients lui demandent).

Mais pour la liaison sans-fil, il doit nécessairement se connecter à un réseau Wifi déjà existant (box, Yún ou autre).

• La carte [Arduino Yún](http://arduino.cc/en/Main/ArduinoBoardYun) est en quelque sorte la combinaison d'un Arduino standard de type Leonardo avec un routeur Ethernet/Wifi tournant sous Linux (Linino).

Ici, c'est le système Linux qui gère les communications, et la partie Arduino standard peut communiquer avec lui au travers de l'interface série (pins #0 et #1) en utilisant la librairie logicielle [Bridge](http://arduino.cc/en/Reference/YunBridgeLibrary).

Le système Linux prend en charge les services Ethernet, Internet et web.

Concernant la liaison sans-fil, la carte de communication fonctionne par défaut en point d'accès Wifi (en d'autres terme, c'est elle qui crée son propre réseau), mais elle peut être re-configurée pour fonctionner dans d'autres modes (client, ad-hoc, ...).

En utilisant une carte Arduino Yún, il est ainsi possible de partager des données avec des clients Wifi, tel qu'un Arduino avec un shield Wifi, un smartphone, un PC, une autre Yún reconfigurée en client, etc. ..

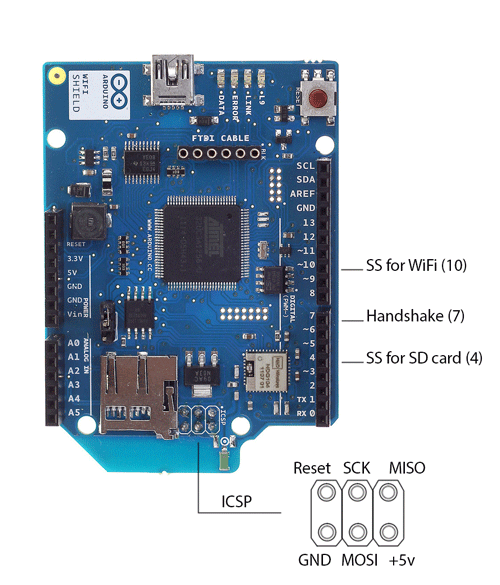
Dans ce projet nous allons utiliser la carte Arduino UNO de ce faits nous ne pourra pas utiliser la carte Arduino Yun ainsi nous allons opter pour une carte shield wifi.

### Programmation et câblage d’une carte shield wifi:

1. Connecter le shield :

Pour utiliser le shield, montez-le sur une carte Arduino Uno. Pour télécharger des croquis sur le tableau, connectez-le à votre ordinateur avec un câble USB comme vous le feriez normalement. Une fois l'esquisse téléchargée, vous pouvez déconnecter la carte de votre ordinateur et l'alimenter avec une source d'alimentation externe.

La **broche numérique 7** sert de broche de liaison entre le blindage WiFi et l’Arduino et ne doit pas être utilisée.



1. Paramétrage réseau:

Le shield se connectera aux réseaux ouverts, ainsi qu’à ceux utilisant le cryptage WEP et WPA2 Personal. Le shield ne se connectera pas aux réseaux utilisant le cryptage WPA2 Enterprise.

Le SSID (nom du réseau) doit être diffusé pour que le shield puisse se connecter.

Selon la configuration de votre routeur sans fil, vous avez besoin d'informations différentes.

* Pour un réseau ouvert (non chiffré), vous avez besoin du SSID.
* Pour les réseaux utilisant le cryptage WPA / WPA2 Personal, vous avez besoin du SSID et du mot de passe.
* Les mots de passe réseau WEP sont des chaînes hexadécimales appelées clés. Un réseau WEP peut avoir 4 clés différentes; une valeur "Index clé" est attribuée à chaque clé. Pour les réseaux cryptés WEP, vous avez besoin du SSID, de la clé et du numéro de clé.

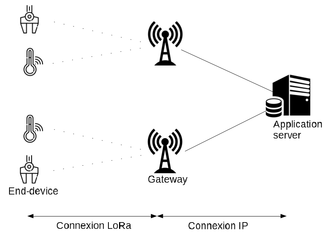
2.1 Réseau ouvert:

Le programme ci-dessous vous montre comment établir une connexion avec un réseau ouvert appelé "votre réseau".



2.2 Réseau utilisation le cryptage WPA/WPA2:

L'exemple ci-dessous montre comment se connecter à un réseau crypté WPA/WPA2 Personal nommé "yourNetwork" avec le mot de passe "12345678".





Pour plus d’information pour les connexion WEP ou pour la connexion d’un shield wifi avec une autre carte arduino de version ultérieure veuillez suivre ce lien <https://www.arduino.cc/en/Guide/ArduinoWiFiShield> .

### **LoraWan:** Connexion Wifi

**LoRaWAN** est un protocole de [télécommunication](https://fr.wikipedia.org/wiki/T%C3%A9l%C3%A9communication) permettant la communication à bas débit, par [radio](https://fr.wikipedia.org/wiki/Onde_radio), d'objets à faible consommation électrique communiquant selon la technologie **LoRa** et connectés à l'[Internet](https://fr.wikipedia.org/wiki/Internet) via des [passerelles](https://fr.wikipedia.org/wiki/Passerelle_(informatique)), participant ainsi à l'[Internet des objets](https://fr.wikipedia.org/wiki/Internet_des_objets). Ce protocole est utilisé dans le cadre des [villes intelligentes](https://fr.wikipedia.org/wiki/Ville_intelligente), le monitoring industriel ou encore l'[agriculture](https://fr.wikipedia.org/wiki/Agriculture). ( definition wikipedia )

#### Architecture LoraWan:

[savoir plus +](https://fr.wikipedia.org/wiki/LoRaWAN#Architecture)

### Quelques Programmes LoraWAN:

LoraWan Activation en direct:

#### La description

Tentative de connexion au réseau LoRaWAN à l' aide de l'activation en direct (OTAA)

#### Syntaxe

modem.joinOTAA (appEui, appKey)

modem.joinOTAA (appEui, appKey, devEui)

#### Paramètres

* - appEui : identifiant d'application 64 bits, EUI-64 (unique)
* - appKey : clé d'application 128 bits
* - devEui : identificateur de périphérique final 64 bits, EUI-64 (unique)

*/\**

*First Configuration*

*This sketch demonstrates the usage of MKR WAN 1300 LoRa module.*

*This example code is in the public domain.*

*\*/*

#include <MKRWAN.h>

LoRaModem modem;

*// Uncomment if using the Murata chip as a module*

*// LoRaModem modem(Serial1);*

String appEui;

String appKey;

String devAddr;

String nwkSKey;

String appSKey;

void **setup**() {

*// put your setup code here, to run once:*

Serial.begin(115200);

while (!Serial);

Serial.println("Welcome to MKRWAN1300 first configuration sketch");

Serial.println("Register to your favourite LoRa network and we are ready to go!");

*// change this to your regional band (eg. US915, AS923, ...)*

if (!modem.begin(EU868)) {

Serial.println("Failed to start module");

while (1) {}

};

Serial.print("Your module version is: ");

Serial.println(modem.version());

Serial.print("Your device EUI is: ");

Serial.println(modem.deviceEUI());

int mode = 0;

while (mode != 1 && mode != 2) {

Serial.println("Are you connecting via OTAA (1) or ABP (2)?");

while (!Serial.available());

mode = Serial.readStringUntil('**\n**').toInt();

}

int connected;

if (mode == 1) {

Serial.println("Enter your APP EUI");

while (!Serial.available());

appEui = Serial.readStringUntil('**\n**');

Serial.println("Enter your APP KEY");

while (!Serial.available());

appKey = Serial.readStringUntil('**\n**');

appKey.trim();

appEui.trim();

connected = modem.joinOTAA(appEui, appKey);

} else if (mode == 2) {

Serial.println("Enter your Device Address");

while (!Serial.available());

devAddr = Serial.readStringUntil('**\n**');

Serial.println("Enter your NWS KEY");

while (!Serial.available());

nwkSKey = Serial.readStringUntil('**\n**');

Serial.println("Enter your APP SKEY");

while (!Serial.available());

appSKey = Serial.readStringUntil('**\n**');

devAddr.trim();

nwkSKey.trim();

appSKey.trim();

connected = modem.joinABP(devAddr, nwkSKey, appSKey);

}

if (!connected) {

Serial.println("Something went wrong; are you indoor? Move near a window and retry");

while (1) {}

}

delay(5000);

int err;

modem.setPort(3);

modem.beginPacket();

modem.print("HeLoRA world!");

err = modem.endPacket(true);

if (err > 0) {

Serial.println("Message sent correctly!");

} else {

Serial.println("Error sending message :(");

}

}

void **loop**() {

while (modem.available()) {

Serial.write(modem.read());

}

modem.poll();

}

LoraWan envoie de paquet:

#### La description

Termine le processus d'envoi d'un paquet LoRaWAN démarré par [beginPacket](https://www.arduino.cc/en/Reference/MKRWANBeginPacket) .

#### Syntaxe

modem.endPacket ();

#### Revenir

Renvoie un int: 1 si le paquet a été envoyé avec succès, 0 s'il y avait une erreur

*/ \**

*Lora Envoyer et recevoir*

*Ce croquis montre comment envoyer et recevoir des données avec le module MKR WAN 1300 LoRa.*

*Cet exemple de code est dans le domaine public.*

*\* /*

#include <MKRWAN.h>

modem LoRaModem ;

*// Ne commentez pas l'utilisation de la puce Murata en tant que module*

*// Modem LoRaModem (Serial1);*

#include "arduino\_secrets.h"

*// Veuillez saisir vos données sensibles dans l'onglet Secret ou arduino\_secrets.h*

String appEui = SECRET\_APP\_EUI ;

String appKey = SECRET\_APP\_KEY ;

void **setup** ( ) {

*// mettez votre code de configuration ici, pour que vous puissiez l'exécuter une fois:*

Serial. commencer ( 115200 ) ;

while ( ! série ) ;

*// change ceci en votre bande régionale (US915, AS923, ...)*

si ( ! modem. begin ( EU868 ) ) {

Serial . println ( "Impossible de démarrer le module" ) ;

tandis que ( 1 ) { }

} ;

Série . print ( "La version de votre module est:" ) ;

Série . println (modem. version ( ) ) ;

Série . print ( "Votre EUI de périphérique est:" ) ;

Série . println ( modem. deviceEUI ( ) ) ;

int connecté = modem. joinOTAA ( appEui , appKey ) ;

si ( ! connecté ) {

Série . println ( "Quelque chose ne va pas; êtes-vous à l'intérieur? Rapprochez-vous d'une fenêtre et réessayez" ) ;

while ( 1 ) { }

}

*// Définissez l'intervalle d'interrogation sur 60 secondes.*

modem. minPollInterval ( 60 ) ;

*// REMARQUE: indépendamment de ce paramètre, le modem*

*// ne permettra pas d'envoyer plus d'un message toutes les 2 minutes,*

*// ceci est imposé par le microprogramme et ne peut pas être modifié.*

}

void **loop** ( ) {

Série . println ( ) ;

Série . println ( "Entrez un message à envoyer au réseau" ) ;

Série . println ("(assurez-vous que" NL "en fin de ligne est activé)" ) ;

while ( ! Série . disponible ( ) ) ;

String msg = Série . readStringUntil ( ' **\ n** ' ) ;

Série . println ( ) ;

Série . print ( "Envoi:" + msg + "-" ) ;

pour ( unsigned int i = 0 ; i< msg. longueur ( ) ; i ++ ) {

Série . print ( msg [ i ] >> 4 , HEX ) ;

Série . print ( msg [ i ] & 0xF , HEX ) ;

Série . print ( "" ) ;

}

Série . println ( ) ;

int err ;

modem.beginPacket ( ) ;

modem. imprimer ( msg ) ;

err = modem. endPacket ( true ) ;

si ( err > 0 ) {

Série . println ( "Message envoyé correctement!" ) ;

} else {

Série . println ( "Erreur lors de l'envoi du message :(" ) ;

Serial . println ("(vous pouvez envoyer un nombre limité de messages par minute, en fonction de la force du signal" ) ;

Série . println ( "cela peut varier de 1 message toutes les deux secondes à 1 message toutes les minutes)" ) ;

}

délai ( 1000 ) ;

if ( ! modem. available ( ) ) {

Serial . println ( "Aucun message de liaison descendante reçu à ce moment." ) ;

retour ;

}

String rcv ;

rcv. réserve ( 64) ;

while ( modem. available ( ) ) {

rcv + = ( char ) modem. read ( ) ;

}

Série . print ( "Received:" + rcv + "-" ) ;

for ( unsigned int i = 0 ; i < rcv. length ( ) ; i ++ ) {

Série . impression( rcv [ i ] >> 4 , HEX ) ;

Série . print ( rcv [ i ] & 0xF , HEX ) ;

Série . print ( "" ) ;

}

Série . println ( ) ;

}

[Savoir plus +](https://www.arduino.cc/en/Reference/MKRWANParsePacket)

### Programmation et câblage sensors températures:

<https://create.arduino.cc/projecthub/TheGadgetBoy/ds18b20-digital-temperature-sensor-and-arduino-9cc806>

Lien important c’est quoi le rasberry : suivre ce lien

<https://www.grafikart.fr/blog/raspberry-pi-utilisation>

un box domotique

<http://www.magdiblog.fr/domotique/raspberrypi-razberry-rfxtrx433usb-domoticz-le-combo-parfait/>

sonde temperature:

<https://www.lesimprimantes3d.fr/forum/topic/9198-r%C3%A9alisation-dune-sonde-de-temp%C3%A9rature-sans-fil-433-mhz-sur-base-arduino/>

andreas spiess

<https://letmeknow.fr/blog/2015/10/27/tutomodulelora/> communications entre Lora et Arduino

NB:

* Une **carte de concentration LoRa** pour recevoir les paquets LoRaWAN. La carte la plus couramment utilisée est la [carte iMST iC880a](https://wireless-solutions.de/products/radiomodules/ic880a.html) .
* Une **antenne** , pour amplifier le signal.
* Un **ordinateur** , pour traiter les paquets LoRaWAN entrants et sortants, et pour les échanger dans les deux sens avec la carte du concentrateur. De nombreux utilisateurs utilisent un [BeagleBone](http://beagleboard.org/bone) ou un [Raspberry Pi](https://www.raspberrypi.org/) .