Lycée Felix Le Dantec - Lannion

Connexion sans fil
Transmettre ou recevoir un signal à distance

TSTI2D SIN
TP15 - 4h

NOM :	CLASSE :
PRÉNOM :	

Condition :	• travail en binôme ; durée 4 heures
Matériel :	• un ordinateur sous Ubuntu ou Windows avec les logiciels Processing et Arduino • carte ESP8266 WiFi, module Bluetooth HC-06, modules XBee
Documents :	• le sujet du TP

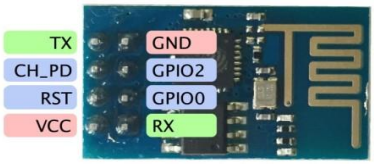
L'objectif du TP est de voir la mise en œuvre d'une transmission sans fil entre un PC et un système. Vous allez tester différents types de connexion sans fil : WiFi, Bluetooth et XBee.

1. Transmission par liaison WiFi

1.1 La carte ESP8266

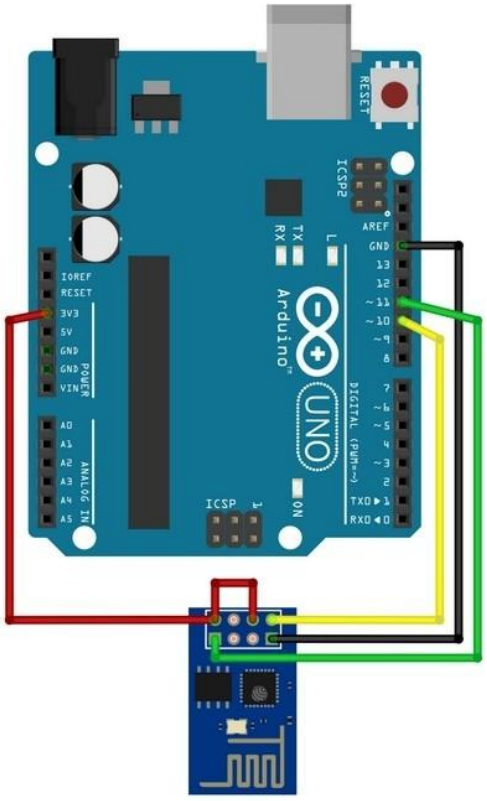
On utilise le module ESP8266 esp01 comme carte Wifi. Il communique via un port série logiciel avec une carte Arduino (voir schéma ci-contre).
Les signaux sont :

TX:envoie des données
RX : reception des données



Pour relier les deux cartes :

TX_esp vers Rx Arduino
RX_esp vers TX_Arduino



1.2 Test de connexion

- ✔ Décompressez le fichier WiFi_esp8266.zip trouvé sur le réseau dans le dossier arduino/libraries/ de votre dossier personnel. Il s'agit de la bibliothèque permettant d'utiliser l'ESP8266 en WiFi.
- ✔ Connectez un ESP8266 WiFi sur un Arduino que vous connectez à votre PC.

La section STI2D SIN dispose d'un point d'accès WiFi installé en C002. Le nom du réseau, encore appelé **ssid** pour Service Set IDentifier, est *STI2D_SIN* et le mot de passe *sti2dsin*. Vous allez tenter de connecter votre Arduino à ce réseau à l'aide de l'ESP8266 WiFi.

point d'accès WiFi

ssid : STI2D_SIN
pass : sti2dsin

On donne ci-dessous un programme de test de connexion à ce réseau STI2D_SIN :

```
/*
 Test de connexion de l'ESP8266 WiFi à un réseau WiFi
 Affiche dans le moniteur série si la connexion est établie
 ainsi que la puissance de réception du signal WiFi
 janvier 2018
 */

#include "WiFiEsp.h"
#include "SoftwareSerial.h"

SoftwareSerial Serial1(10, 11); // RX, TX

char ssid[] = "STI2D_SIN";      //  your network SSID (name)
char pass[] = "sti2dsin";  // your network password
int status = WL_IDLE_STATUS;     // the Wifi radio's status

void setup() {
   Serial.begin(115200);  // initialisation du port série vers moniteur série pour debugage
   Serial1.begin(9600);      //  initialisation du port série vers module ESP
   WiFi.init(&Serial1);      // initialisation module ESP
}

void loop() {
  do
  {
    Serial.print("Tentative de connexion au reseau SSID : ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);   // tentative de connexion au réseau WPA/WPA2
    delay(7000);    // attente de 7 secondes pour la connexion
  }
  while ( status != WL_CONNECTED);  // boucle tant que pas connecté
  Serial.println("Connexion reussie");
  Serial.print("Signal: ");
  Serial.print(WiFi.RSSI());  // affichage de la puissance de la réception en dBm
  Serial.println(" dBm");
  Serial.println(); //Passer une ligne
  Serial.println();
  WiFi.disconnect();    // déconnexion
  delay(3000);    // attente de 3 secondes pour la déconnexion
}
```

Liaison série ESP vers Arduino

paramètres de notre réseau

tentative de connexion

affichage de la puissance de réception du signal WiFi

- ✔ Copiez-collez ce programme dans l'éditeur Arduino. Programmez l'Arduino et déplacez-vous à l'intérieur puis à l'extérieur de la salle (peut-être à l'aide d'un PC portable de la section SIN). Complétez le tableau ci-dessous pour différentes mesures. Quelle est la portée de notre réseau WiFi dans le bâtiment C ?

Distance du point d'accès (C002) en mètres								
Puissance en dBm								

Maintenant que vous avez effectué les tests de connexion, vous allez récupérez les informations sur la connexion de votre Arduino.

- ✔ Compléter le fichier précédent entre les lignes « `Serial.println(" dBm"); » et « Serial.println(); »`

```
  Serial.println(" dBm");

       // Affiche l'adresse IP
  IPAddress ip= WiFi.localIP();
  ip = WiFi.localIP();        // Deux fois pour avoir un affichage correct de la puissance et de l'IP.
  Serial.print("IP Address: ");
  Serial.println(ip);

    //  Affiche l'adresse MAC
  byte mac[6];
  WiFi.macAddress(mac);
  char buf[20];
  sprintf(buf, "%02X:%02X:%02X:%02X:%02X:%02X", mac[5], mac[4], mac[3], mac[2], mac[1], mac[0]);
  Serial.print("MAC address: ");
  Serial.println(buf);

  Serial.println(); //Passer une ligne
```

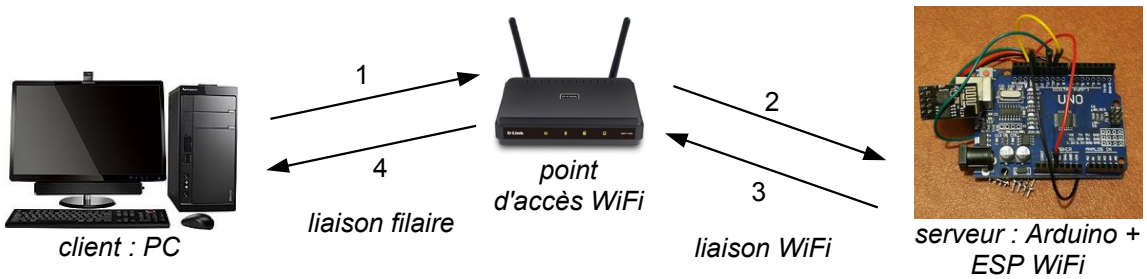
✔ Programmez l'Arduino et ouvrez le moniteur série, réglez le sur 115200 baud. Vérifiez la bonne connexion de votre ESP WiFi au réseau STI2D_SIN. Relevez l'adresse l'adresse MAC de votre shield ainsi que l'adresse IP qui lui a été attribuée par le serveur DHCP.

adresse MAC du shield WiFi						adresse IP du shield WiFi			
						192	168	11	

✔ Faîtes un ping à partir de la console de votre PC vers l'adresse IP du shield WiFi. Vérifiez la connexion.

1.3 Créer un serveur Web avec le shield WiFi

L'Arduino, et son espWiFi, est un serveur : il envoie des données, ici une page web à un PC client qui se connecte en utilisant un navigateur web.



✔ Ouvrez le programme Examples>WiFiEsp>WiFiWebServer. Modifiez ce programme en notant le ssid, le mot de passe de notre réseau ainsi que les bornes du port série logiciel.

✔ Programmez l'Arduino et ouvrez le moniteur série. Tapez, dans un navigateur web, l'adresse IP de l'ESP WiFi et vérifiez que vous recevez bien une page web affichant les valeurs lues sur les entrées analogiques de l'Arduino. Relier A0 à la masse pour obtenir l'affichage « A0:0 ».

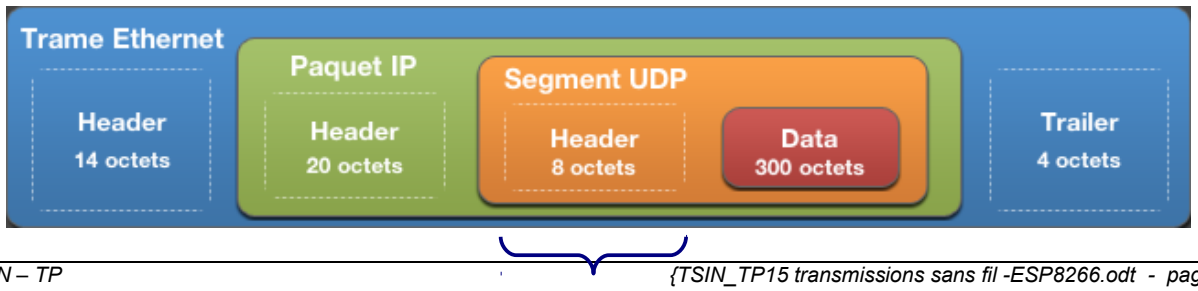
page web générée par l'Arduino

Ce programme nous montre comment créer aisément un serveur web à partir d'un Arduino connecté en WiFi à un réseau.

1.4 Communiquer des données, vers un shield WiFi, à partir d'une application Processing

L'objectif est de commander 6 leds connectées à l'Arduino à partir d'une application développée sur Processing.

Pour transmettre les données de l'application vers la maquette nous utiliserons un protocole simple mais non sécurisé, l'**UDP** pour *User Datagram Protocol***.** Il s'agit d'un protocole simple, sans connexion, permettant de transférer des données sans alourdir la communication par du trafic de contrôle. UDP rajoute seulement 8 octets d'information en tête de chaque segment de données à transporter dont principalement le port UDP source (2 octets) et le port UDP de destination (2 octets) chargés d'identifier, sur chaque hôte, les applications qui communiquent. UDP est très utilisé pour les applications comme les jeux en ligne, le streaming, etc.
Le **segment UDP** est encapsulé dans un **paquet IP** transporté par une **trame Ethernet**. Il comporte une en-tête suivi des données proprement dites à transporter.



L'en-tête d'un datagramme UDP la suivante :

contenu	taille	détail
Port Source	2 octets	port de l'émetteur
Port Destination	2 octets	port du récepteur
Longueur	2 octets	longueur totale (exprimée en octets) du segment UDP (en-tête et données)
Somme de contrôle	2 octets	permet de s'assurer de l'intégrité du paquet reçu

Les leds sont connectées sur les broches numériques de l'Arduino : 2, 3, 4, 5, 6, et 7.

- ✔ Câblez sur une plaque labdec 6 leds sur ces broches avec une résistance de 330Ω en série.
- ✔ On vous donne ci-dessous le programme de l'application Processing et celui de l'Arduino. Retrouvez dans ces programmes les éléments suivants :
  - si caractères reçus ;
  - envoi d'un tableau de 6 états logiques précédé d'un # ;
  - ouverture d'un port pour une communication utilisant le protocole UDP ;
  - lecture des octets reçus.

Application Processing

```
/*
 Envoi d'un tableau de 6 entiers afin de commander 6 leds
 utilisation du protocole UDP
 Ch. Le Bris
 janvier 2015
 */

import hypermedia.net.*;

UDP udp;
String ip="192.168.11.XX";
int port=8888;
int tab[]= { 0, 0, 0, 0, 0, 0 };

void setup() {
  udp=new UDP(this, 6000);
  udp.log(true);  // affichage de messages dans la console
  size(600, 400);
  background(255);
  smooth();
}

void draw() {
  background(255);
  text("COMMANDE DES SPOTS", 220, 20);
  for (int i=5;i>=0;i--) {
    fill(0);
    text("LED"+i, 85+80*(5-i), 55);
    if (tab[i]==1)   fill(#00FF00);
    else   fill(#FF0000);
    ellipse(100+80*(5-i), 100, 60, 60);
  }
  fill(0);
  text("OFF", 40, 230);
  text("ON", 40, 265);
  for (int i=5;i>=0;i--) {
    text("I"+i, 95+80*(5-i), 180);
    noFill();
    rect(80+80*(5-i), 200, 40, 80);
    fill(0);
    if (tab[i]==0)
      rect(80+80*(5-i), 200, 40, 40);
    else
      rect(80+80*(5-i), 240, 40, 40);
  }
}

void mousePressed()
{
  for (int i=5;i>=0;i--) {
       if       ((mouseX>(80+80*(5-i)))&&(mouseX<(120+80*(5-
i)))&&(mouseY>200)&&(mouseY<280)) tab[i]=1-tab[i];
  }
            udp.send("#"+str(tab[0])+str(tab[1])+str(tab[2])
+str(tab[3])+str(tab[4])+str(tab[5]), ip, port);
}
```

@IP de votre shield WiFi

Arduino

```
/*
 Réception d'un tableau de 6 entiers afin de commander 6
leds
 utilisation du protocole UDP
 janvier 2018
 */

#include "SoftwareSerial.h"
#include "WiFiEsp.h"
#include <WiFiEspUdp.h>

#define LED0 2

SoftwareSerial Serial1(10, 11); // RX, TX

char ssid[] = "STI2D_SIN"; //  your network SSID (name)
char pass[] = "sti2dsin";  // your network password

int status = WL_IDLE_STATUS;     // the Wifi radio's status

unsigned int localPort = 8888;      // local port to listen
on
 packetBuffer[255]; //buffer to hold incoming packet

WiFiEspUDP Udp;

void setup() {
    Serial.begin(115200);   // initialisation du port série
vers moniteur série pour debugage
    Serial1.begin(9600);   //   initialisation du port série
vers module ESP
   WiFi.init(&Serial1); // initialisation module ESP
   delay(3000);
  // déclaration en sortie des broches où sont cablées les
leds
  pinMode(LED0,OUTPUT);

  // attempt to connect to Wifi network:
  while ( status != WL_CONNECTED) {
    Serial.print("Connexion au reseau : ");
    Serial.println(ssid);
    status = WiFi.begin(ssid,pass);
    // wait 5 seconds for connection:
    delay(5000);
  }
  Serial.println("Connecte au WiFi");
  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("Adresse IP : ");
  Serial.println(ip);

  Serial.println("\nPret a recevoir des donnees...");
  Udp.begin(localPort);
}

void loop() {
  int packetSize = Udp.parsePacket();
  if (packetSize)
  {
    Serial.print("Recu ");
    Serial.print(packetSize);
    Serial.print(" octets de ");
```

```
    IPAddress remoteIp = Udp.remoteIP();
    Serial.print(remoteIp);
    Serial.print(":");
    Serial.println(Udp.
    // read the packet
    int len = Udp.re
    if (len > 0) packetB
    Serial.println("Contenu:");
    Serial.println(packetBuffer);

    // envoi des états logiques sur les leds
    Udp.read(packetBuffer,255);
     if (packetBuffer[0]==0x23)   {   //La trame envoyée par
Processing commence par # = 0x23 en ASCII
        digitalWrite(LED0,packetBuffer[1]-48);      //ASCII
vers nombre
    }
  }
}
```

- ✔ Quel est le numéro de port utilisé par Processing ?

- ✔ Même question pour la maquette Arduino ?

- ✔ Copiez ces programmes et faîtes le test... seule la led D2 doit être commandée par l'application sous Processing.
- ✔ Modifiez alors les programmes afin de pouvoir commander toutes les leds. Utiliser le tableau suivant et des boucles : int Led[] ={2,3,4,5,6,7}; //tableau brochage des leds=> led0 en 2 …
  Testez et faîtes contrôler par le professeur.

Validation prof :

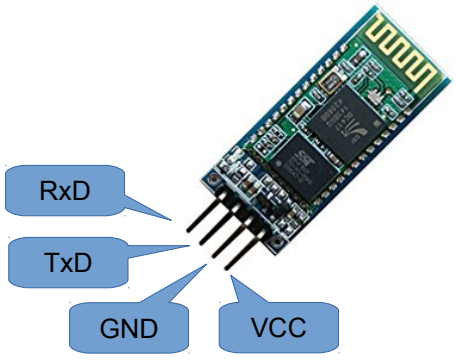
- ✔ Si vous avez du temps, remplacez le montage à leds par un câblage vers les spots du fond de la salle C001. Vous pouvez également transférez votre application Processing sur une tablette ou votre smartphone. Testez et faîtes contrôler par le professeur.

Validation prof :

2. Transmission par liaison Bluetooth

2.1 Les modules Bluetooth HC-06

Pour la communication Bluetooth, on utilise le module HC-06. Il est constitué de 4 broches, deux pour son alimentation et deux pour la communication série (RxD et TxD).



Pour permettre la communication entre le module et l'Arduino, tout en conservant la communication entre l'Arduino et le PC, nous utiliserons les broches 10 et 11 de l'Arduino :

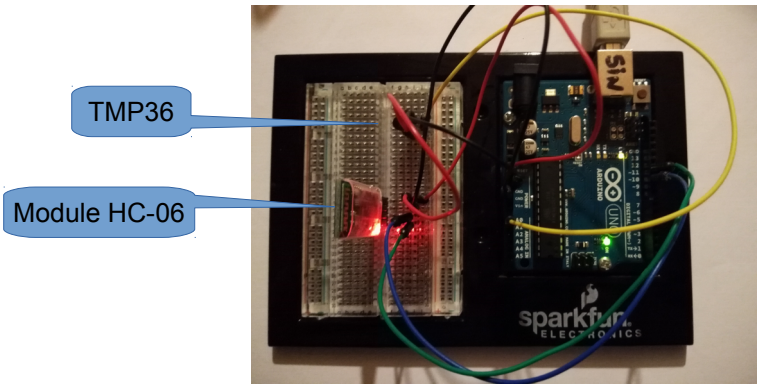
RxD (HC-06) câblé à D11
TxD (HC-06) câblé à D10

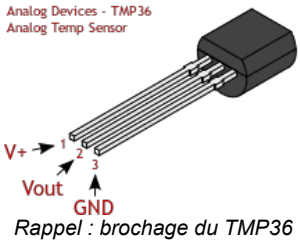
2.2 Lecture d'une température par l'Arduino et envoi vers une application sur smartphone

Vous allez tester la connexion entre un l'Arduino associé au module HC-06 et un smartphone disposant du

Bluetooth. L'objectif est d'afficher sur l'écran du téléphone la température de la salle.
Attention : le module Bluetooth doit être appairé à votre téléphone pour pouvoir communiquer avec lui, son code PIN est 1234.

- ✔ Relier un capteur TMP36 sur l'entrée analogique A0. Connecter également le module Bluetooth HC-06 comme décrit ci-dessus.





Rappel : brochage du TMP36

✔ On vous donne les programmes Arduino et Processing (incomplet). Complétez ce programme et testez.

Validation prof :

Arduino

```
/*
 tempBT.ino
 Envoi de la température via un module Bluetooth HC-06
 Pour la config du module HC-06, voir http://leroyd.com/articles/module_bluetooth.php
 Ou encore https://knowledge.parcours-performance.com/arduino-bluetooth-hc-05-hc-06/
 Christophe Le Bris
 janvier 2017
 */

#include <SoftwareSerial.h>

int temp;
SoftwareSerial hc06(10,11);  // RX, TX à croiser vers module hc-06

void setup() {
  // initialize serial:
  Serial.begin(9600);
  hc06.begin(9600);
}

void loop() {
  temp=map(analogRead(A0),0,1023,-50,450);

  Serial.println(temp);
  hc06.print(temp);
  hc06.print('\n'); // ajout d'un saut de ligne
  delay(2000);
}
```

Application Android sous Processing

```
/**
 * receiveTempBT.pde
 * communication avec un module HC-06 sur Arduino
 * module HC-06_1 : 98:D3:31:20:2F:A3
 * module HC-06_2 : 98:D3:31:20:2F:9A
 * source : https://polaridad.es/processing-android-bluetooth-ketai-libreria/
 * Christophe Le Bris
 * janvier 2017
 */

//required for BT enabling on startup
import android.os.Bundle;      // onCreate Bundle
import android.content.Intent; // onActivityResult Intent
import ketai.net.*;
import ketai.net.bluetooth.*;  // KetaiBluetooth

KetaiBluetooth bluetooth;
String MAC_moduleBT="98:D3:31:20:2F:9A";  // @MAC du module Bluetooth connecté à l'Arduino
String temp="";
String messageRecu="";

void onCreate(Bundle instance)
{
  super.onCreate(instance);
  bluetooth=new KetaiBluetooth(this);
}

void onActivityResult(int codigo_solicitud, int codigo_resultado, Intent datos)
{
  bluetooth.onActivityResult(codigo_solicitud, codigo_resultado, datos);
}

void setup()
{
  size(600, 600);
  noStroke();
  bluetooth.start();
  bluetooth.connectDevice(MAC_moduleBT);  // connexion au module Bluetooth
  println("connecté");
}

void draw()
{


}

void onBluetoothDataEvent(String emetteur, byte[] data)
{
  if (emetteur==MAC_moduleBT)
  {
    messageRecu+=new String(data);
    if (data[data.length-1]==10)    // détection du saut de ligne, code ASCII 0x10
```

@ MAC du module Bluetooth dont vous disposez

À compléter, affichage de la variable temp sur l'écran du smartphone

```
    {
      temp=messageRecu;
      messageRecu="";
    }
  }
}
```

La technologie Bluetooth utilise les ondes radios (bande de fréquence des 2.4 GHz) pour communiquer, si bien que les périphériques ne doivent pas nécessairement être en liaison visuelle pour communiquer.

- ✔ Déplacez-vous à l'intérieur puis à l'extérieur de la salle. Quelle est la portée de votre réseau Bluetooth ?

2.3 Envoi de données du smartphone vers l'Arduino par la liaison Bluetooth

L'objectif est de tester la communication du smartphone vers la carte Arduino.
Vous vous aiderez du site web ketai.org pour trouver la fonction permettant d'envoyer un message à partir du smartphone.

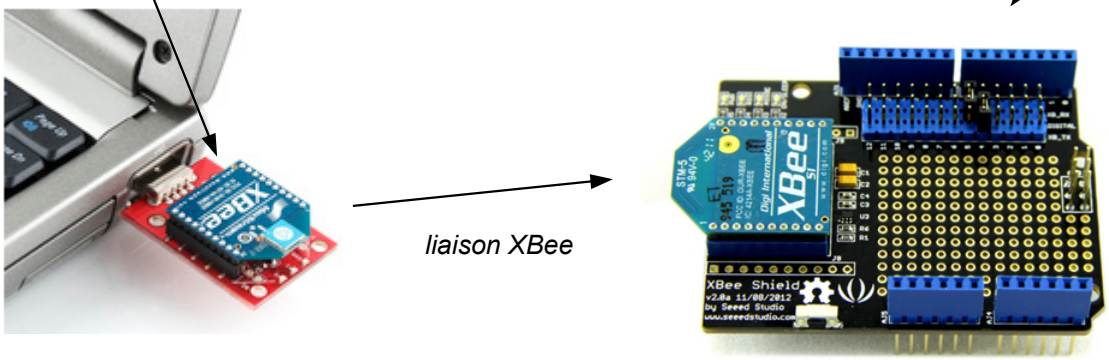
- ✔ Modifiez les programmes précédents et le câblage afin d'allumer une led câblée sur la broche 8 à partir de l'application sur le smartphone. Testez et faîtes contrôler par le professeur.

Validation prof :

3. Transmission par liaison XBee

3.1 Les modules XBee

Les modules XBee utilisent la technologie ZigBee, un protocole de communication qui s'appuie sur le travail du groupe IEEE 802.15.4.
Nous disposons d'un dongle USB compatible PC sur lequel on insère un module XBee, puis d'un shield pour Arduino sur lequel on place un deuxième module XBee.



Il ne reste plus qu'à établir le dialogue entre ces 2 modules.

3.2 Test de connexion

L'objectif est de tester la connexion entre le dongle XBee et le shield XBee.

- ✔ Insérez le dongle USB équipé d'un module XBee sur un des ports USB d'un premier PC, si possible un PC portable. Connectez à un autre PC l'Arduino avec son shield XBee équipé lui aussi d'un module Xbee.

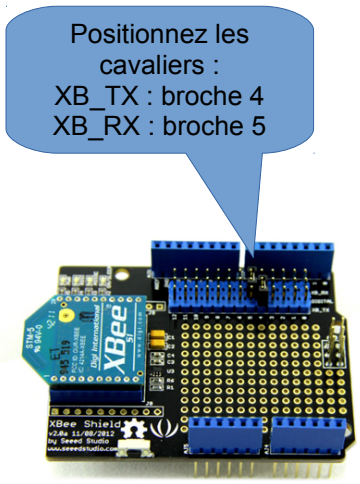
- ✔ Sur le PC connecté à l'Arduino, écrivez le programme ci-dessous qui renvoie en continue les caractères reçus par le module XBee. Le module XBee va utiliser les broches 4 (XB_TX) et 5 (XB_RX) pour communiquer avec l'Arduino. Positionnez les cavaliers afin de configurer des broches. Programmez l'Arduino.

```
/*
  testConnexion.ino
  lit et renvoie les caractères reçus
  Le module XBee utilise les broches 4 et 5 de l'Arduino
  laissant libre la liaison série classique pour la programmation
  Ch. Le Bris
  janvier 2015
*/

#include <SoftwareSerial.h>
// XBee's DOUT (TX) is connected to pin 4 (Arduino's Software RX)
// XBee's DIN (RX) is connected to pin 5 (Arduino's Software TX)
SoftwareSerial XBee(4,5); // RX, TX

void setup()
{
  XBee.begin(9600);
}

void loop()
{
  if (XBee.available())
    XBee.write(XBee.read());
}
```



✔ Sur le PC équipé du dongle, ouvrez Processing. On vous donne le code source d'un programme qui écrit sur la liaison série USB0, celle du dongle XBee, le code ASCII de la touche appuyée et lit les caractères reçus sur cette liaison série. Copiez-collez ce code et exécutez-le. Tapez des caractères, et vérifiez que la carte Arduino et son shield vous renvoie les caractères envoyés.

```
/**
 * Serial Duplex
 * by Tom Igoe.
 *
 * Sends a byte out the serial port when you type a key
 * listens for bytes received, and displays their value.
 * This is just a quick application for testing serial data
 * in both directions.
 */


import processing.serial.*;
int val;
Serial myPort;      // The serial port
int whichKey = -1;  // Variable to hold keystoke values
int inByte = -1;    // Incoming serial data

void setup() {
  size(400, 300);
  // create a font with the third font available to the system:
  PFont myFont = createFont(PFont.list()[2], 14);
  textFont(myFont);

  // List all the available serial ports:
  printArray(Serial.list());

  // I know that the first port in the serial list on my mac
  // is always my  FTDI adaptor, so I open Serial.list()[0].
  // In Windows, this usually opens COM1.
  // Open whatever port is the one you're using.
  String portName = Serial.list()[32];
  myPort = new Serial(this, portName, 9600);
}

void draw() {
  background(0);
  text("Last Received: " + inByte, 10, 130);
  text("Last Sent: " + whichKey, 10, 100);
   if ( myPort.available() > 0) {  // If data is available,
    val = myPort.read();
```

```
    background(255);            // Set background to white
          // If the serial value is 0,
   fill(0);                  // set fill to black
  text("temp"+val,10,100);// read it and store it in val
  }
}

void serialEvent(Serial myPort) {
  inByte = myPort.read();
}

void keyPressed() {
  // Send the keystroke out:
  myPort.write(key);
  whichKey = key;
}
```

réception d'un caractère sur la liaison série

envoie du code ASCII de la touche appuyée

✔ Déplacez-vous à l'intérieur puis à l'extérieur de la salle. Quelle est la portée de votre réseau XBee ?

3.3 Commande d'une led en XBee

L'objectif est de commander une led connectée sur la broche 8 de la carte Arduino (équipée du shield XBee) à partir de l'application précédente sous Processing

✔ Modifiez le programme testConnexion.ino afin d'allumer la led 8 si le caractère reçu est 'H' et de l'éteindre si le caractère reçu est 'L'. Testez ce programme à partir de l'application sous Processing. Faîtes contrôler par le professeur.

Validation prof :

3.4 Affichage de la température

L'objectif est d'afficher dans l'application Processing la mesure de température effectuée à distance par la carte Arduino et son shield XBee.

✔ Connectez un capteur de température TMP36 sur une broche analogique de l'Arduino.

✔ Modifiez les programmes Arduino et Processing afin d'afficher dans la fenêtre de l'application la température relevée par l'Arduino. Testez et faîtes contrôler par le professeur.

Validation prof :