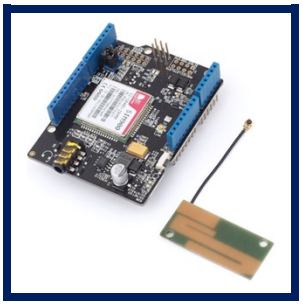
STI2D – Enseignement de spécialité SIN

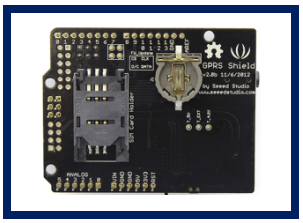
MODULE GSM/GPRS SEEDSTUDIO

1 – LE MODULE GSM/GPRS

1.1 – Présentation

Le module **GMS/GPRS** de chez SeedStudio est une carte d'interface compatible Arduino. Elle permet d'envoyer et recevoir des SMS, des données ou des communications vocales depuis le réseau mobile. Le module est basé sur le circuit SIM900 de la société SIMCOM. Il est contrôlé via les commandes AT depuis une carte Arduino.





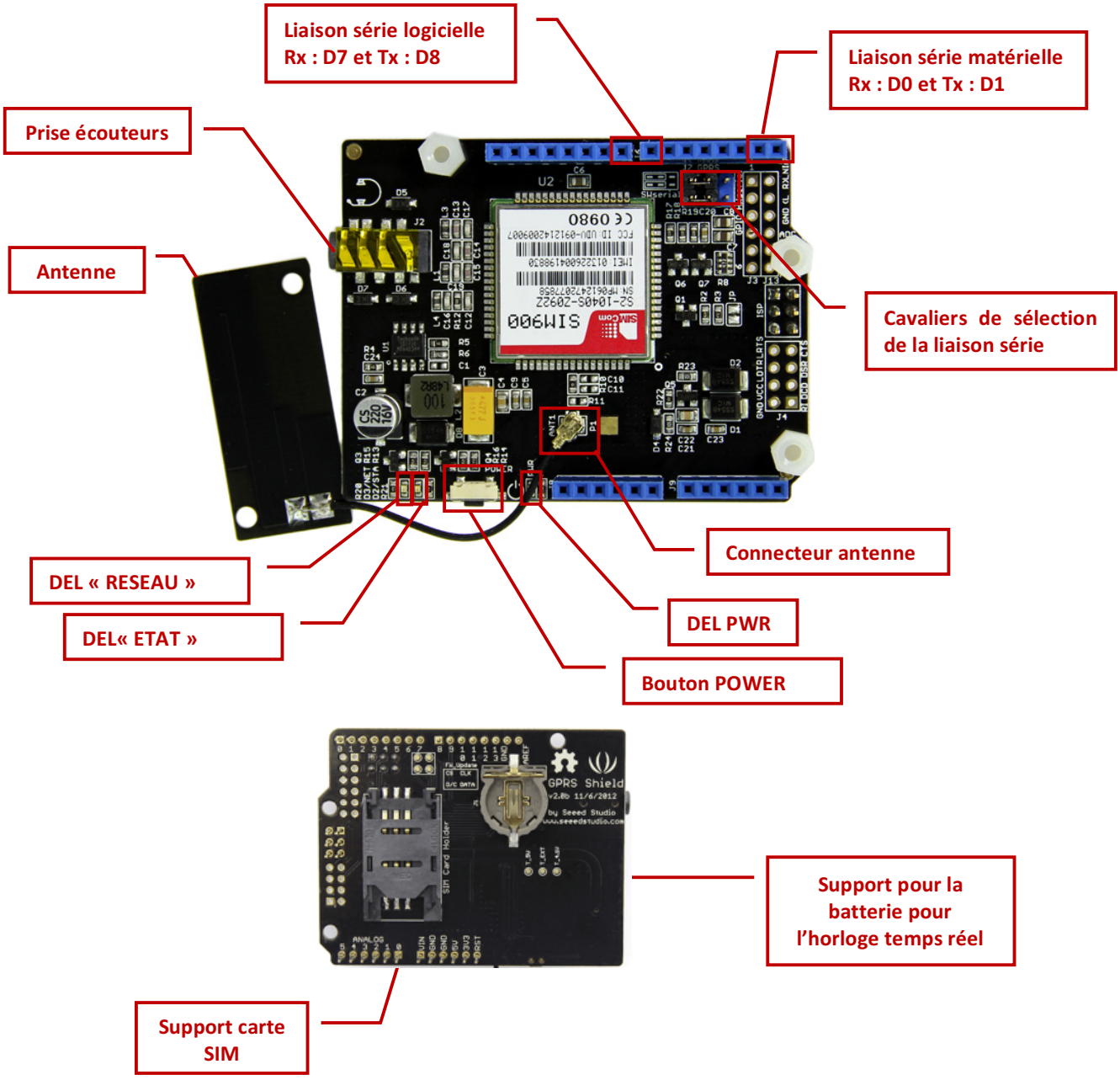
Le module est livré avec une antenne patch déportée. Un connecteur au dos de la platine est prévu pour recevoir une carte SIM ainsi qu'une pile Lihtium CR1220.

La communication entre le module et une carte Arduino est réalisée par la **liaison série asynchrone** : UART ou une liaison série logicielle.

1.2 – Caractéristiques principales

Module Quad-band	850/900/1800/1900 MHz
Protocoles supportés	TCP/UDP
Tension d'alimentation	5 V par la broche 5V 6,5 V à 12 V par la broche Vin
Consommation	1,5 mA en veille 400 mA max
Puissance	Classe 4 (bandes 850/ 900 MHz) : 2 W Classe 1 (bandes 1800/ 1900 MHz) : 1 W
Température de fonctionnement	– 40 °C à + 85 °C
Dimensions	68.58 x 53.34mm

1.3 – Description du module



1.4 – Liaison série

Le choix des broches permettant la communication entre le shield GPRS et la carte Arduino via la liaison série est réalisé via deux cavaliers.

	
Liaison série logicielle : **Rx = D7** et **Tx = D8**	**Liaison série matérielle** : **Rx = D0** et **Tx = D1**

La liaison série doit être réglée avec une **vitesse de 19200 bits/s**.

1.5 – Mise sous tension

La mise sous tension du shield est réalisée de manière matérielle via le **bouton** « **POWER** ». Cette mise sous tension peut être réalisée également de manière logicielle en appliquant un **niveau logique haut sur la broche D9**.

1.6 – DELs

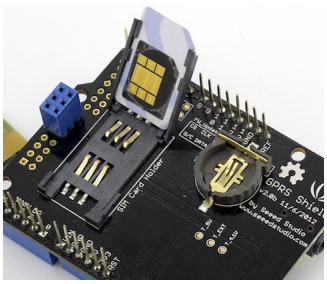
Les indications données par les 3 DELs sont :

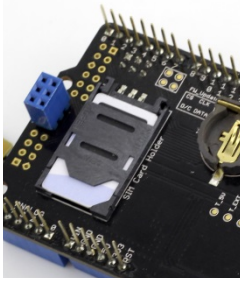
DEL « PWR » (verte)	Eteinte	Le shield est hors tension
	Allumée	Le shield est sous tension
DEL « ETAT » (Rouge)	Eteinte	Le module SIM900 est hors tension
	Allumée	Le module SIM900 est sous tension
DEL « RESEAU » (Verte)	Eteinte	Le module SIM900 en veille
	Allumée : 64 ms Eteinte : 800 ms	Le module ne trouve pas de réseau
	Allumée : 64 ms Eteinte : 3000 ms	Le module a trouvé un réseau
	Allumée : 64 ms Eteinte : 300 ms	Communication GPRS

2 – UTILISATION DU MODULE GSM/GPRS

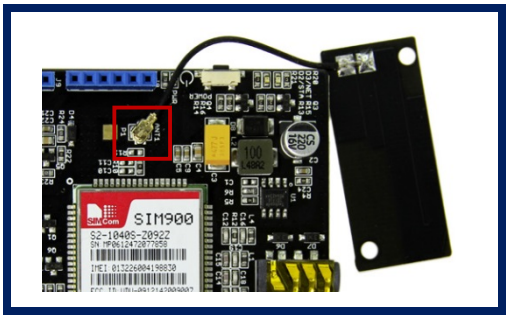
- Placer les cavaliers afin de configurer la liaison série.

- Insérer une carte SIM active dans le support prévu à cet effet.

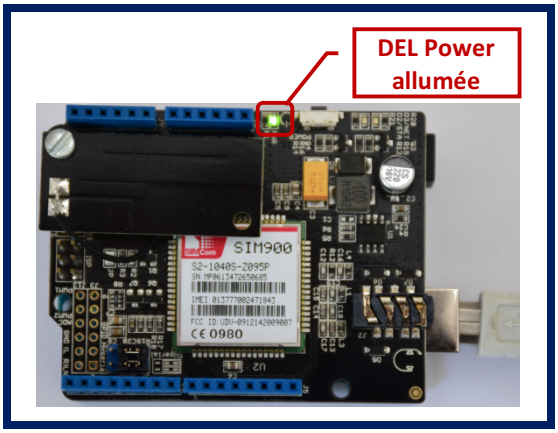




- Vérifier que l'antenne est correctement connectée.



- Connecter le shield sur une carte Arduino ou une carte équivalente. Alimenter la carte Arduino. La DEL « POWER » doit s'allumer.



3 – COMMANDES AT SUR ARDUINO

3.1 – Présentation

Les commandes AT sont un jeu de commandes textuelles permettant de gérer la plupart des modems ou des modules GSM. Ces commandes commencent toujours par les lettres « **AT** » et se terminent obligatoirement par un **retour chariot**.

3.2 – Configuration de la liaison série logicielle

L'Atmega328 possède une **interface de communication série** UART accessible, grâce aux **broches numér**iques **0** (**Rx**) et 1 (**Tx**). La bibliothèque « **SoftwareSerial** » a été développée pour permettre la communication série sur d'autres broches numériques de l'Arduino de manière logicielle. Il est possible de gérer plusieurs ports séries logiciels avec des vitesses allant jusqu'à **115 200 bps** cependant un seul peut recevoir des données à la fois.

Inclure la bibliothèque « SoftwareSerial »

Pour inclure la librairie « SerialSoftware » dans un programme, on ajoutera au début du programme la ligne suivante :

```
#include <SoftwareSerial.h>
```

Créer une liaison série logicielle

```
SoftwareSerial Nom_Liaison(Broche_Rx, Broche_Tx)
```

Exemple : Création d'une liaison série logicielle pour la liaison Bluetooth sur les broches 2 (Rx) et 3 (Tx) :

```
SoftwareSerial MyGSM(7, 8)
```

Fonction « begin »

```
Nom_Liaison.begin(Vitesse) ;
```

Cette fonction qui doit être **appelée au moins une fois**, généralement dans la fonction setup(), permet de **définir la vitesse** utilisée par la liaison série.

La valeur prise par la variable « **Vitesse** » doit être une des **vitesses définies par la norme RS232**.

```
MyGSM.begin(115200);
```

3.3 – Entrer le code PIN

Code PIN

Pour entrer le code il faut envoyer la commande AT suivante :

```
AT+CPIN = "XXXX"
```

Si le code PIN est correct, le module GSM répondra à cette commande par « **+CPIN: READY** ».

```
MyGSM.println("AT+CPIN = \"1234\"");        // Code PIN carte SIM
```

3.4 – Envoyer un message texte (SMS)

Mode Texte

Le shield GPRS peut envoyer des SMS selon deux modes : le **mode texte** et le mode PDU (binaire). Pour envoyer un message lisible, il faut sélectionner le mode texte en envoyant la commande AT :

```
AT+CMGF = 1
```

Le shield répondra à cette commande par « **OK** ».

```
MyGSM.println("AT+CMGF = 1");        // Mode Texte
```

Numéro destinataire SMS

Il faut ensuite indiquer le numéro du téléphone du destinataire du SMS. Pour cela il faut utiliser la commande AT ci-dessous. Le numéro doit être indiqué au **format E.123**.

```
AT+CMGS = "+XXXXXXXXXXX"
```

```
MyGSM.println("AT+CMGS = \"+3368825263411\"");  // Numéro destinataire
```

Envoi texte message

Il suffit ensuite d'envoyer le texte du message. La dernière ligne du message doit comporter un retour chariot. Le message se termine par la **séquence d'échappement CTRL-Z** (caractère 26).

```
MyGSM.println("Comment Allez-vous ?");      // Texte SMS
MyGSM.print((char) 26);                     // CTRL-Z
```

3.5 – Recevoir un message texte (SMS)

Lorsque le GPRS Shield reçoit un SMS, le module SIM900 envoi spontanément un message similaire à "**+CMTI: "SM",21**". « **+CMTI: »** servira de détecteur pour identifier la réception du message. Le « 21 » indiqué dans l'exemple ci-dessus identifie la position "mémoire" où le message est stocké. Il s'agit d'une information importante.

La carte SIM dispose de 25 emplacements pour stocker les SMS en cours de réception. Une fois le message traité, l'emplacement mémoire devrait être libéré à l'aide de la commande AT adéquate. S'il n'y a plus de notifications « **+CMTI** » c'est probablement parce que la mémoire est saturée

Lecture du contenu d'un SMS

Pour lire le contenu d'un SMS, il faut utiliser la commande suivante avec « XX » représentant la " la position mémoire où est stocké le message.

```
AT+CMGR = "XX"
```

Le module SIM900 produira le résultat suivant :

```
+CMGR: "REC UNREAD","+33658964125","","12/04/17,14:00:39+08"
Message
```

Il est alors possible d'identifier le statut du message (REC UNREAD), le numéro de l'émetteur (**+33658964125**), la date l'heure (**12/04/17 à 14:00:39)** et le contenu du message.

```
MyGSM.println("AT+CMGR = \"19\"");          // Lecture message 19
```

Effacer un SMS

Pour effacer un SMS, il faut utiliser la commande suivante avec « XX » représentant la " la position mémoire où est stocké le message.

```
AT+CMGD = "XX"
```

```
MyGSM.println("AT+CMGD = \"19\"");          // Effacer message 19
```

4 – MISE EN OEUVRE DU SHIELD GSM AVEC UNE CARTE ARDUINO

4.1 – Emission d'un SMS

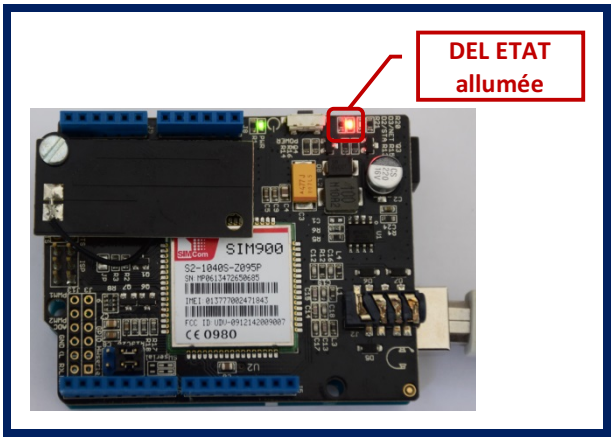
Le programme suivant permet d'émettre un SMS lorsqu'un caractère quelconque est entré dans le moniteur série

```
#include <SoftwareSerial.h>

SoftwareSerial MyGSM(7, 8);      // Communication avec le Shield GSM via la liaison série logicielle
byte SMS = 0;                    // Indicateur SMS envoyé
void setup()
{
  MyGSM.begin(19200);            // Vitesse de communication liaison série logicielle
  delay(1000);                   // Attendre une seconde que le modem retourne "OK"
  Serial.begin(19200);           // Vitesse de communication liaison série UART
  Serial.println("Test Shield GSM");

}

void loop()
{
  if (Serial.available() && SMS == 0) {            // Si une donnée est entrée sur le moniteur série et si pas de SMS envoyé
    Serial.read();
    MyGSM.println("AT+CMGF=1");                    // Passer en mode Texte
    delay(100);
    MyGSM.println("AT+CMGS=\"+33648459480\"");     // Numéro destinateur
    delay(100);
    MyGSM.println("Test Envoi SMS");               // Message à envoyer
    MyGSM.print((char) 26);                        // Indicateur de fin de message (CTLR Z)
    delay(100);
    SMS = 1;                                      // SMS envoyé
  }
   else  if(MyGSM.available()){                   // Si une donnée récue par la liason série logicielle (GSM)
       Serial.write(MyGSM.read());
   }
}
```

- Editer dans l'IDLE Arduino, le programme ci-dessus.

- Presser le bouton « POWER » pendant 2s. La DEL « ETAT » doit s'allumer et la DEL « RESEAU » doit clignoter si un réseau mobile a été trouvé.

- Presser le bouton « POWER » pendant 2s. La DEL « ETAT » doit s'allumer et la DEL « RESEAU » doit clignoter si un réseau mobile a été trouvé.

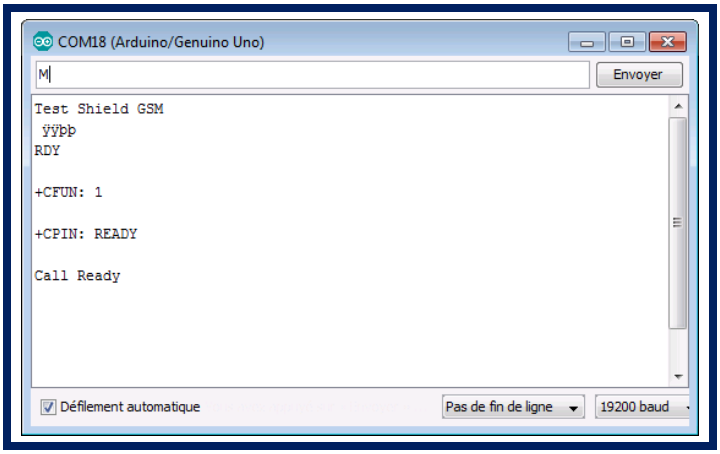


Lorsque la mise sous tension est achevée, le SIM900 envoi le code de résultat (result code) **RDY** pour indiquer qu'il est prêt.

« **+CFUN: 1** » : Le module a trouvé un réseau mobile.

« **+CPIN: 1** » : Pas de nécessité d'entrer un code PIN ou bien l'identification par code PIN a réussie.

« **Call Ready** » : Le module SIM900 est enregistré sur le réseau mobile et il est prêt à l'emploi.



- Exécuter le programme précédant à partir de l'IDLE. Le message est alors envoyé.