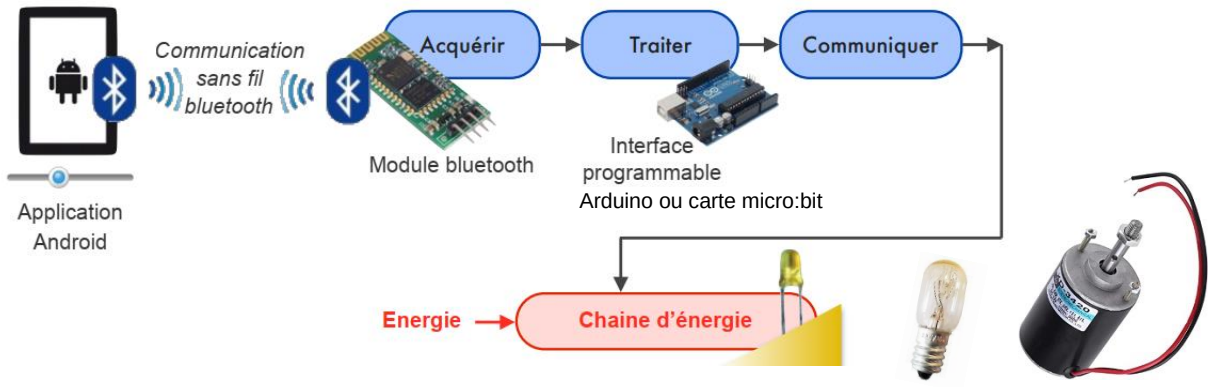
LYCÉE de l'ÉLORN
le choix de votre réussite

Application Android -2- Commande de circuits

Objectif :

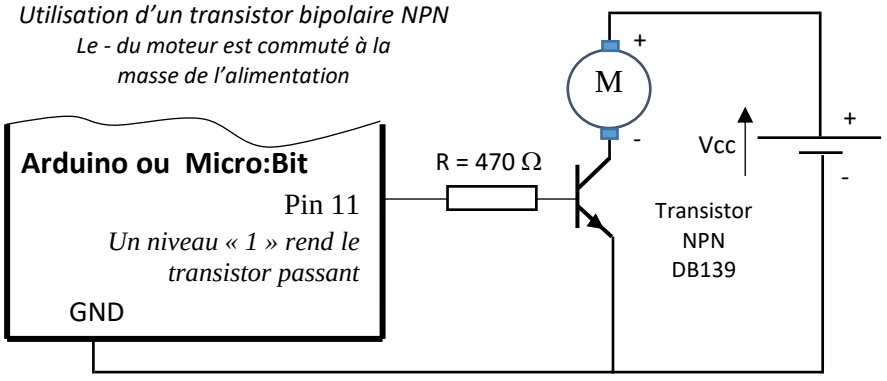
Dans cet exemple il s'agit, de piloter la puissance d'un récepteur électrique depuis une application Android (smartphone ou tablette). Avec cette structure il est possible de faire varier la puissance de n'importe quelle charge analogique (moteur à courant continu, lampe, LED, résistance chauffante …).

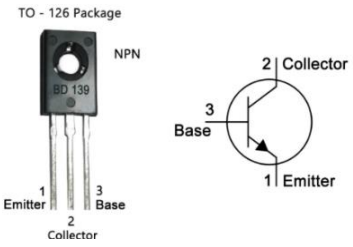


Rappel des schémas de câblage :

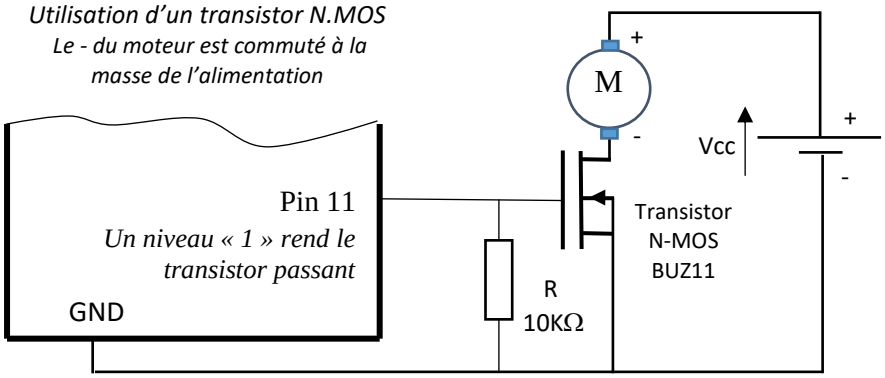
Nous utiliserons un transistor pour assurer la commutation d'énergie. Ce composant se comporte comme un interrupteur commandé par la broche de sortie de la carte. Il n'autorise que du courant continu dans le circuit de puissance.

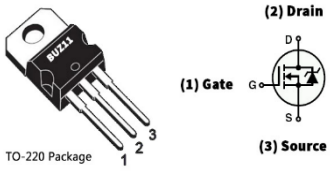
Utilisation d'un transistor bipolaire NPN
Le - du moteur est commuté à la masse de l'alimentation



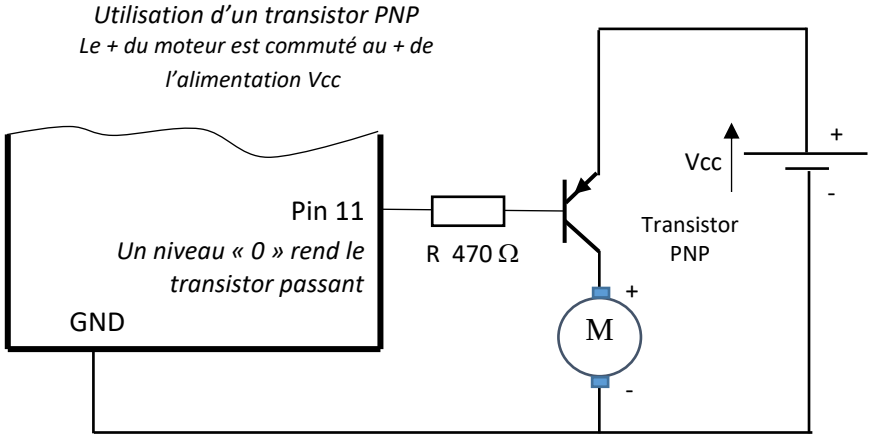


Utilisation d'un transistor N.MOS
Le - du moteur est commuté à la masse de l'alimentation

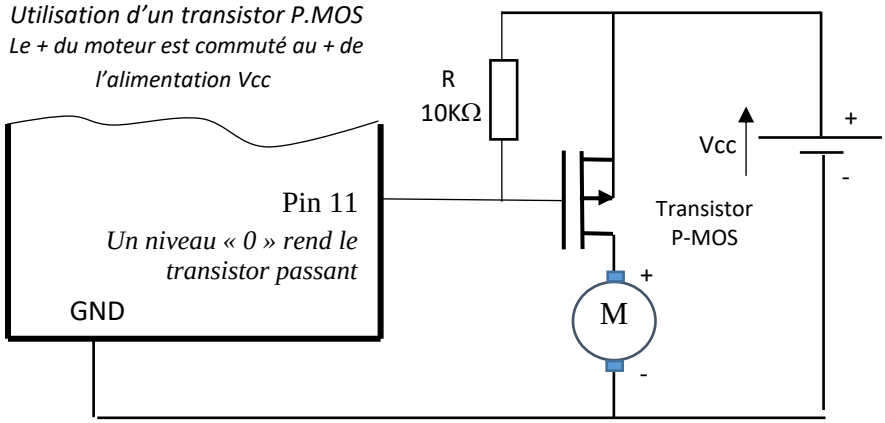




Utilisation d'un transistor PNP
Le + du moteur est commuté au + de l'alimentation Vcc

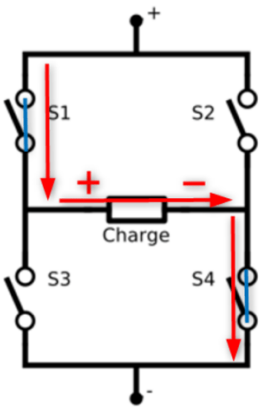


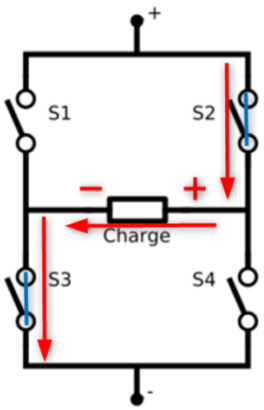
Utilisation d'un transistor P.MOS
Le + du moteur est commuté au + de l'alimentation Vcc



Dans le cas où vous souhaitez polariser le moteur avec des tensions positives ou négatives, vous devrez utiliser une structure de PONT en H.

Les fonctions d'interrupteurs S1, S2, S3 et S4 sont réalisées par les transistors présentés ci-dessus.

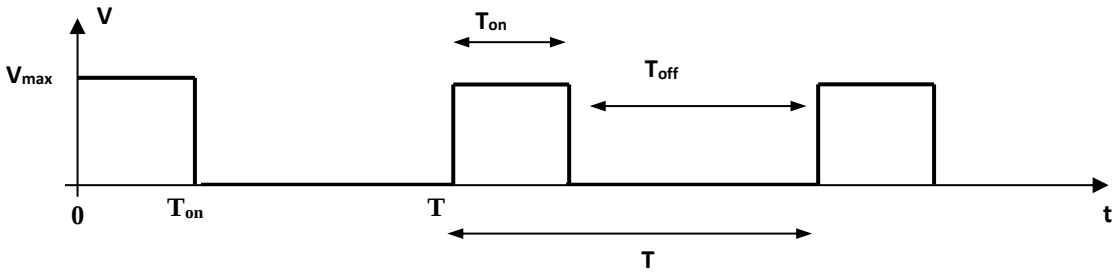




Gestion de la puissance par le PWM.

Une sortie **PWM** (*Pulse Width modulation, Modulation de largeur d'impulsion)* peut générer un signal digital en Tout Ou Rien (0 ou 1) caractérisé par une période constante et un rapport cyclique paramétrable.
La tension moyenne de ce signal est variable. On peut donc faire **évoluer l'énergie** reçue par un récepteur commandé par ce signal.

Rappel des caractéristiques d'une grandeur TOR



V_{max} est la tension matérialisant le niveau « 1 ». Le niveau « 0 » est par défaut matérialisé par le 0V.

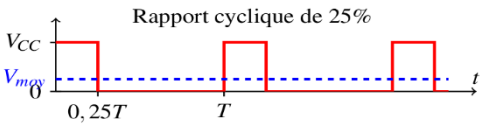
T_{on} la durée du niveau « 1 », T_{off} la durée du niveau « 0 »

La période T en seconde est le temps nécessaire pour que le signal se répète, soit $T = T_{on} + T_{off}$

La fréquence *f* en Hertz est le nombre de périodes par seconde : $f\ (Hz) = \frac{1}{T(s)}$

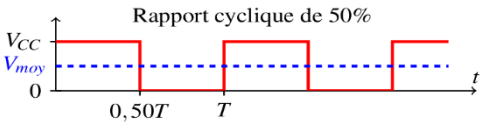
Le rapport cyclique du signal est la grandeur $\alpha = \frac{T_{on}}{T}$

La valeur moyenne du signal est $V_{moy} = \frac{1}{T}\int_0^T v(t)\,dt = V_{max}\frac{T_{on}}{T} = V_{max}\ \alpha$

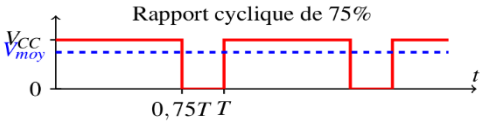


Si $V_{max} = V_{cc} = 5$ V

$V_{moy} = 1{,}25$ V



$V_{moy} = 2{,}5$ V



$V_{moy} = 3,75$ V

Instructions en python sur la carte Micro:Bit

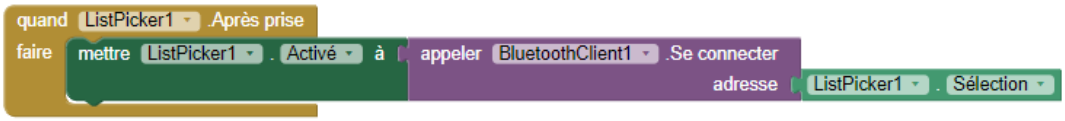
- *write_analog(valeur)* *valeur est un entier sur 10 bits (elle peut évoluer de 0 à 1023)*
- *set_analog_period(période)* *définit la période du signal PWM en ms (valeur minimale 1 ms)*
- *set_analog_period_microseconds(période)* *définit la période du signal PWM en µs (valeur minimale 256 µs)*

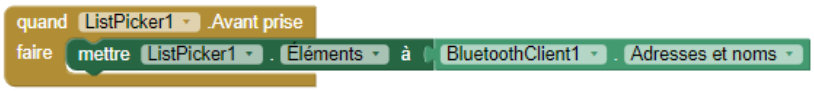
Instruction en C sur la carte Arduino

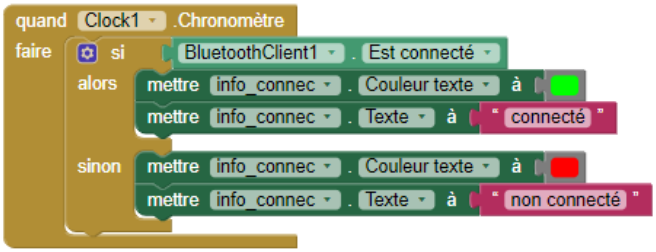
- *analogWrite (valeur)* *valeur est un entier sur 0 bits (elle peut évoluer de 0 à 255)*

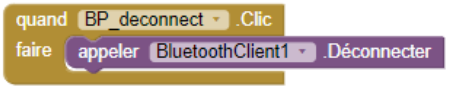
Application N°1 _ Configurer la liaison Bluetooth, transmission de valeurs en tout ou rien (TOR).

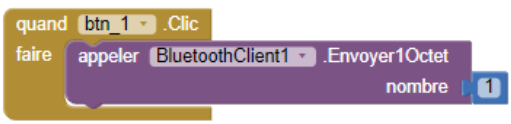
Réaliser sur Appinventor cette première application qui est capable de se connecter à un serveur Bluetooth et de lui envoyer la valeur « 0 » ou « 1 ».

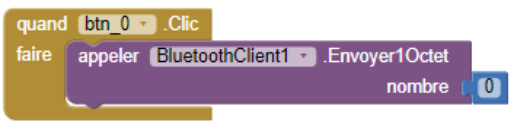












Voici le programme en C pour le client Arduino.

Le programme suivant va mettre à « 1 » la sortie 11 si une action sur le bouton « envoi 1 » est réalisée. La sortie passera à « 0 » par une action sur le bouton « envoi 0 »

```
include <SoftwareSerial.h>   //déclaration d'une liaison série supplémentaire (broches 4 et 5 de l'Arduino)
#define RxD 5
#define TxD 4
#define DEBUG_ENABLED  1
SoftwareSerial blueToothSerial(RxD,TxD);

void setup()
{
 pinMode(11,OUTPUT);
 Serial.begin(38400);
 pinMode(RxD, INPUT);
 pinMode(TxD, OUTPUT);
 setupBlueToothConnection();
}

void loop()
{
 char recvChar;
 while(1){
  if (blueToothSerial.available()){          //Si une donnée est présente sur la liaison série venant du shield Bluetooth
    recvChar = blueToothSerial.read();       //lecture du caractère
    int nb_recu=int(recvChar);               //conversion du String vers un type entier
    Serial.println(nb_recu);                 //affichage sur la console arduino
    if (nb_recu==1) digitalWrite(11,HIGH);   // Mise à 1  de la sortie 11
    if (nb_recu==0) digitalWrite(11,LOW);    // Mise à 0  de la sortie 11
   }
  }
}

void setupBlueToothConnection()       //procédure d'initialisation du shield Bluetooth
{
 blueToothSerial.begin(38400);                          //Définir le débit Bluetooth à 38400 bits/s
 blueToothSerial.print("\r\n+STWMOD=0\r\n");             //Le shield est en esclave
 blueToothSerial.print("\r\n+STOAUT=1\r\n");             // Autorise l'appairage
 blueToothSerial.print("\r\n+STAUTO=0\r\n");             // Auto-connexion est interdite
 delay(2000);                                           // tempo nécessaire
 blueToothSerial.print("\r\n+INQ=1\r\n");
 Serial.println("The slave bluetooth is inquirable!");
 delay(2000);                                           // tempo nécessaire
 blueToothSerial.flush();
}
```

Application N°2 _ Configurer la liaison Bluetooth, transmission de valeurs numériques.

Ajouter un « ascenseur » sur l'interface pour définir une valeur moyenne à la grandeur de sortie puis faite en sorte que cette grandeur soit transmise par Bluetooth.
Modifier en conséquence le code de l'Arduino pour interpréter cette commande.