

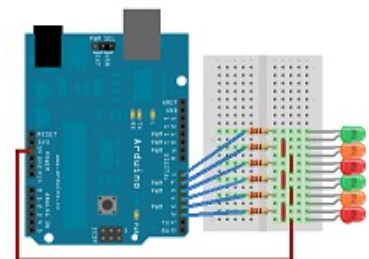
TP introduction Arduino

avec shield d'extension E/S

Table des matières

1. Installation et configuration de la carte Arduino.....	2
1.1. Carte Arduino.....	2
1.2. Shield d'extension E/S.....	2
1.3. Structure d'un programme.....	4
1.4. Installation de l'interface de développement.....	5
2. Clignotement d'une LED.....	5
3. Utilisation d'un interrupteur switch.....	6
4. Utilisation de l'entrée analogique.....	6
4.1. Variation de la fréquence d'allumage d'une LED en fonction de la tension d'entrée.....	6
4.2. Mesure de l'éclairement.....	6
4.3. Mesure de la température.....	7
5. Affichage d'une grandeur sur un afficheur LCD.....	8
6. Commande d'un moteur.....	9
6.1. Servo moteur.....	9
6.2. Moteur à courant continu en PWM.....	11
7. Références.....	15

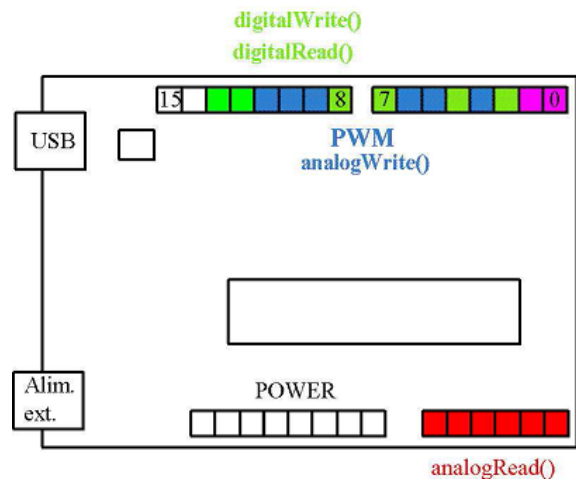
Ce TP de 10 heures est une introduction au projet SI (70 heures jusqu'au mois de mai). Il a pour but d'introduire l'utilisation de la carte Arduino. À l'issue des 10 heures, vous devrez rendre un compte rendu par binôme donnant les sources de vos programmes commentés, les mesures faites, une photo et/ou une vidéo des montages, les explications du fonctionnement des capteurs, de la variation de vitesse d'un moteur à courant continu...



1. Installation et configuration de la carte Arduino

1.1. Carte Arduino

Nous utiliserons une carte Arduino Uno. Elle emploie un microcontrôleur ATMEGA328P alimenté en 5 V. Il y a 14 entrées/sorties numériques dont 6 sont utilisables en PWM¹. Il y a 6 entrées analogiques. Le microcontrôleur possède un CAN² avec 10 bits de résolution. Sur la carte, il y a un circuit qui permet de gérer facilement l'USB³ qui peut alimenter la carte.



- Mémoire Flash 32 ko
- Mémoire RAM 2 ko
- Mémoire EEPROM 1 ko
- Fréquence d'horloge 16 MHz
- Courant max. E/S 40 mA

Pour en savoir plus, consultez la page : <http://arduino.cc/en/Main/ArduinoBoardUno>

1.2. Shield d'extension E/S

Ce shield d'extension vous permet le branchement simple et rapide de nombreux modules. Pour carte Arduino Uno, Mega et Due à 14 I/O + support I2C, Xbee, APC220, Bluetooth, SD Card.

Caractéristiques :

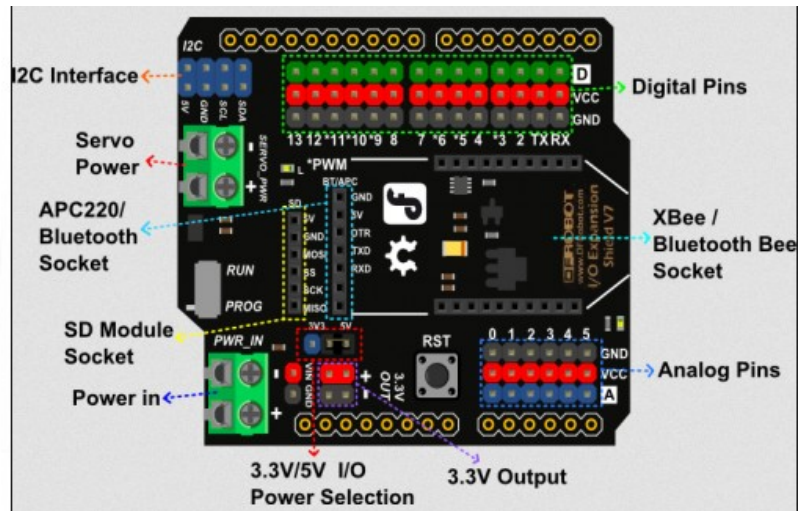
- Connecteurs colorés pour identifier les I/O numériques et analogiques
- Interface à 14 I/O
- Interface de transmission de données sans fil
- Compatible avec les Arduino UNO, Mega, DUE...
- Interface de sortie sélectionnable en 3.3V

¹ Pulse Width Modulation (ou MLI pour modulateur de largeur d'impulsion)

² Convertisseur Analogique Numérique

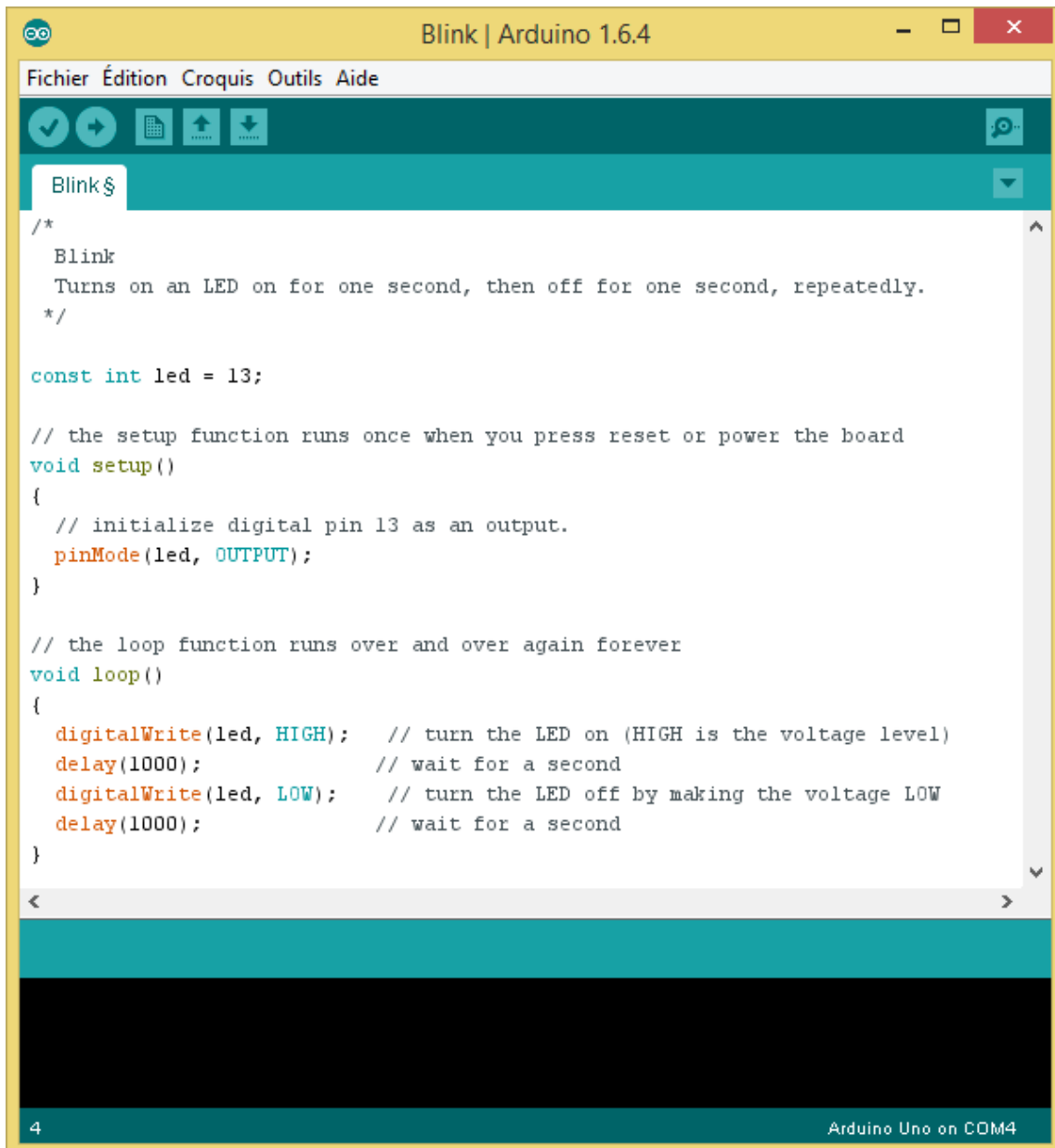
³ Universal Serial Bus

- interface I2C
- Support Xbee (Xbee pro)
- Support Bluetooth
- Support APC220
- Support SD card Lecture/Ecriture



1.3. Structure d'un programme

Un exemple de programme est donné ci-dessous.

The image shows a screenshot of the Arduino IDE window titled "Blink | Arduino 1.6.4". The menu bar includes "Fichier", "Édition", "Croquis", "Outils", and "Aide". Below the menu is a toolbar with icons for saving, undo, redo, and uploading. The main text area contains the following code:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
  */

const int led = 13;

// the setup function runs once when you press reset or power the board
void setup()
{
  // initialize digital pin 13 as an output.
  pinMode(led, OUTPUT);
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

At the bottom of the window, there is a status bar showing "4" on the left and "Arduino Uno on COM4" on the right.

- Le commentaire de description du programme, appelé sketch, débute par `/*` et se termine par `*/`.
- Puis il est placé la définition des constantes et des variables avec les instructions `const` et `int`.
- Vient ensuite la configuration des entrées / sorties avec l'instruction `void setup()`. La suite d'instructions est précédé de `{` et se termine par `}`.
- On définit par l'instruction `void loop()` la programmation des interactions et comportements. La suite d'instructions est précédé de `{` et se termine par `}`.

1.4. Installation de l'interface de développement

Pour utiliser l'interface de développement, [télécharger le logiciel](#) Arduino (version windows).

1. Ouvrir le logiciel Arduino.
2. Vérifier que la carte Arduino Uno est bien prise en compte par : Outils > Type de carte > Arduino Uno.
3. Brancher la carte sur un port USB. Vérifier que le port COM est bien configuré. Outils > Port série > COM X.
4. Vérifier que le port COM X est bien reconnu par windows : Panneau de Configuration > Système et sécurité > Gestionnaire de périphériques > Ports : Arduino Uno (COM X).

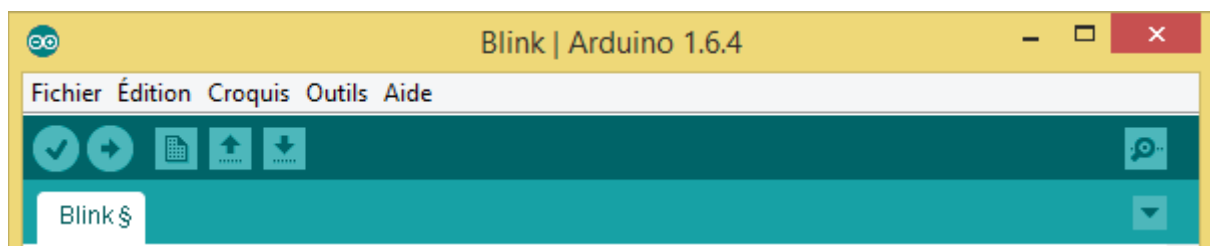
2. Clignotement d'une LED

Ouvrir le fichier Blink par : Fichier > Exemples > 01.Basics > Blink

Pour comprendre la syntaxe et le rôle d'une instruction, utiliser la page :

<http://arduino.cc/en/Reference/HomePage>

1. Modifier le programme de telle façon que la sortie sur laquelle on connectera la LED soit la sortie 11. Modifier les instructions pour que la période soit de 1 s et la durée d'allumage de la LED 100 ms.
2. Lancer l'exécution du programme en cliquant sur le bouton en dessous Édition :



Pour voir le résultat, cliquer sur l'image

3. Modifier le programme pour que deux LED s'allument en opposition de phase comme dans le tableau suivant :

LED 1	Allumée	Éteinte
LED 2	Éteinte	Allumée



Pour voir le résultat, cliquer sur l'image

3. Utilisation d'un interrupteur switch

1. Écrire un programme pour que la carte Arduino détecte la pression du bouton poussoir.
2. Modifier ce programme pour que la carte Arduino commande la LED lors de la pression du bouton.

Pour voir le résultat, cliquer sur l'image



4. Utilisation de l'entrée analogique

4.1. Variation de la fréquence d'allumage d'une LED en fonction de la tension d'entrée

On placera un potentiomètre qui sera connecté sur l'entrée Analog IN A0. On utilisera l'instruction **YY = *analogRead* (XX)**, XX étant la valeur lue sur l'entrée Analog IN A0.

1. Écrire un programme pour faire varier la fréquence d'allumage de la LED.
L'instruction ***delay* (YY)** permettra de faire varier la fréquence de commande de la LED.

Pour voir le résultat, cliquer sur l'image



2. Modifier ce programme pour que l'intensité lumineuse de la LED varie en fonction de la position du potentiomètre.

Pour voir le résultat, cliquer sur l'image



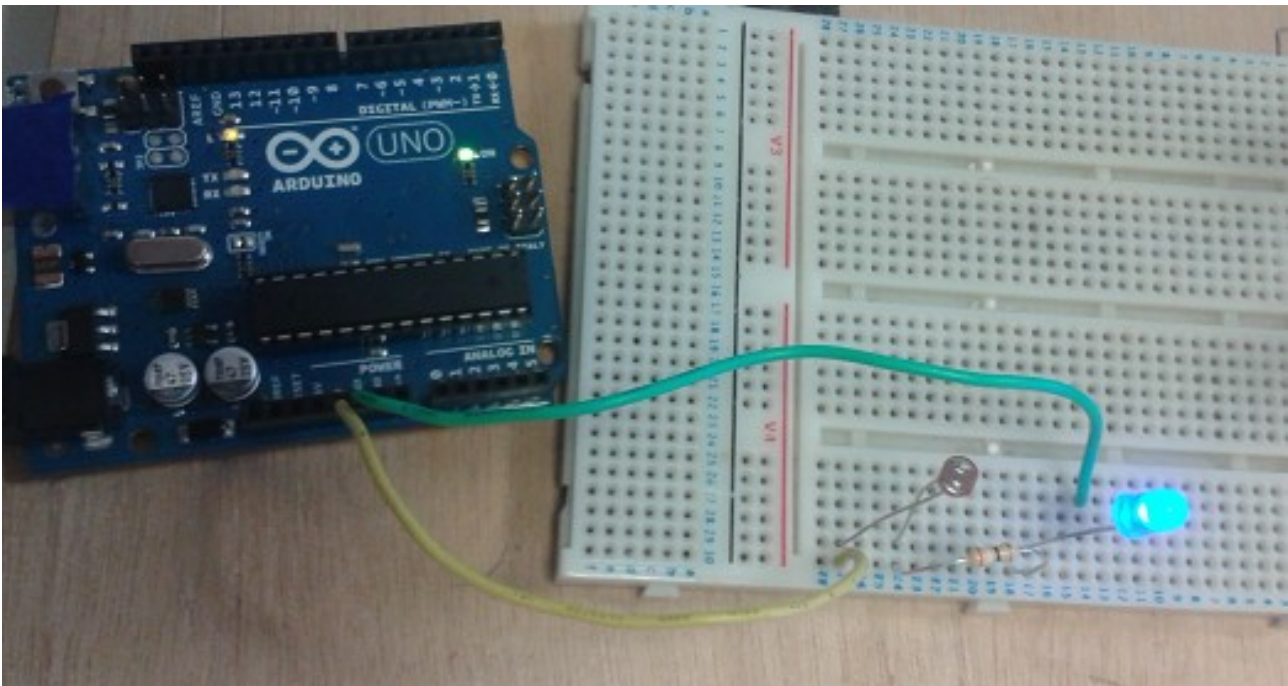
4.2. Mesure de l'éclairement

Le capteur de luminosité utilisé est le LDR VT935G (ou équivalent série VT900).

1. Télécharger la [documentation du capteur](#) et comprendre son fonctionnement.
2. Indiquer le principe de fonctionnement du capteur.
3. Indiquer la tension maximale d'utilisation.
4. Indiquer la signification de la sensibilité.
5. Proposer un montage permettant de détecter le passage d'un objet.
6. Modifier ce montage pour que la détection du passage soit gérée par la carte Arduino (utiliser la liaison série du port USB et l'objet Serial de la bibliothèque Arduino).

Pour voir le résultat, cliquer sur l'image

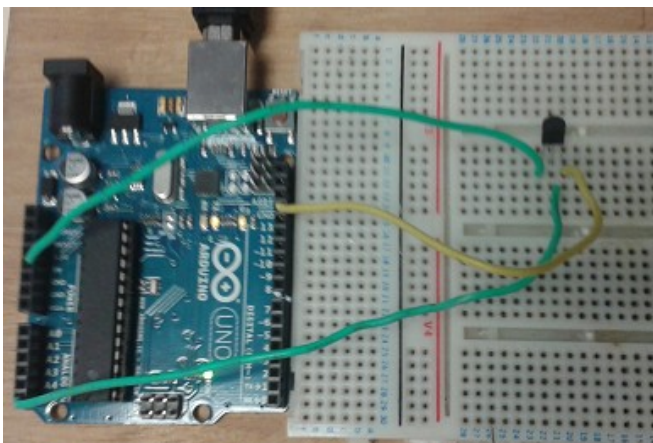
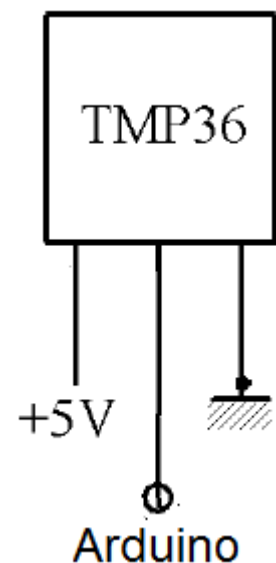




4.3. Mesure de la température

Le capteur de température utilisé est le TMP 36.

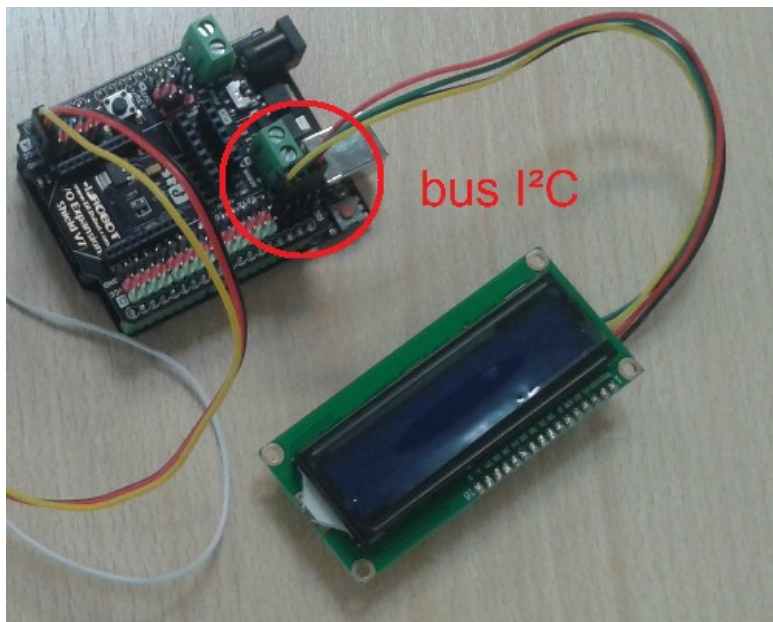
1. Télécharger la [documentation du capteur](#) et comprendre son fonctionnement.
2. Indiquer le principe de fonctionnement du capteur.
3. Indiquer la tension de sortie pour une température nulle.
4. Donner la sensibilité du capteur.
5. Alimenter le capteur à partir de la carte Arduino.
6. Connecter sa sortie sur une entrée analogique.
7. Afficher sa valeur sur le port série.
8. Vérifier la compatibilité de la valeur numérique lue avec la température et la valeur de la tension continue en sortie du capteur.



5. Affichage d'une grandeur sur un afficheur LCD

L'afficheur utilisé est décrit sur [cette page](#).

1. Ouvrir dans l'interface Arduino ouvrir le programme HelloWorld :
Fichiers > Exemples > LiquidCrystal > HelloWorld.
2. Connecter l'afficheur sur le bus I²C selon le schéma ci-dessous.



Broche A0 : capteur
température

Broche A2 : capteur humidité

Bus I²C : LCD

3. Détecter l'adresse utilisée par l'afficheur LCD à l'aide du programme ci-dessous :

```
//
// scanner I2C
//

#include <Wire.h>

void setup()
{
    Wire.begin();

    Serial.begin(9600);
    while ( !Serial);           // Leonardo: wait for serial monitor
    Serial.println("\nI2C Scanner");
}

void loop()
{
    Serial.println("Scanning...");

    int nDevices(0);
    for (int address(1); address < 127; address++) {
        // The i2c_scanner uses the return value of
        // the Write.endTransmission to see if
        // a device did acknowledge to the address.
        Wire.beginTransmission(address);
        int error = Wire.endTransmission();

        if ( error == 0 ) {
            Serial.print("dispositif I2C détecté à l'adresse : 0x");
        }
    }
}
```



```

        if ( address < 16 )
            Serial.print("0");
        Serial.println(address,HEX);

        nDevices++;
    }

    if ( nDevices == 0 )
        Serial.println("Aucun dispositif I2C trouvé\n");
    else
        Serial.println("terminée\n");

    delay(5000);          // wait 5 seconds for next scan
}

```

4. Tester le programme HelloWorld.
5. Modifier le programme afin d'afficher la température en °C et l'humidité en %.

6. Commande d'un moteur

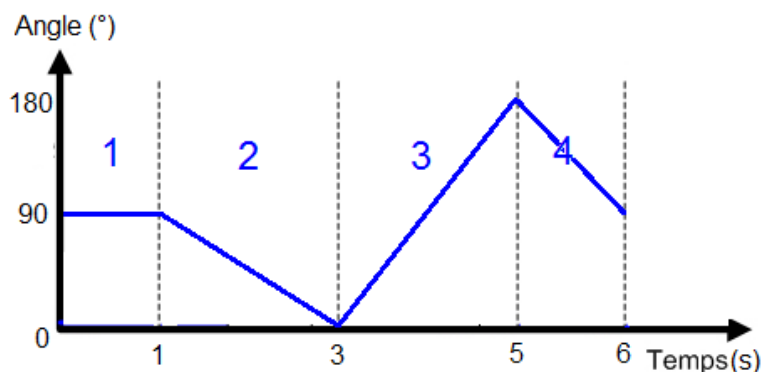
6.1. Servo moteur

Un servomoteur est un système motorisé capable d'atteindre des positions prédéterminées, puis de les maintenir. La position est : dans le cas d'un moteur rotatif, une valeur d'angle et, dans le cas d'un moteur linéaire une distance.

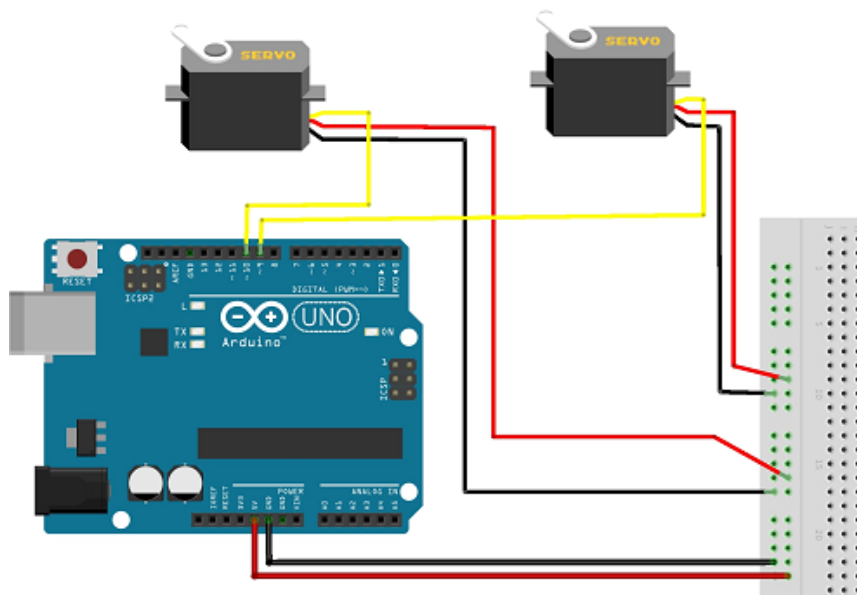
On veut commander 2 servo moteurs.

Un premier servo moteur doit effectuer le cycle ci-contre.

Le deuxième doit être en opposition de phase.



1. Réaliser le montage ci-dessous.



Broche 9 : servo 1
Broche 10 : servo 2

2. Modifier le programme afin de respecter le cahier des charges ci-dessus.

```
/*
  commande servo moteurs
*/

#include <Servo.h>

const int pin_servo1 = 9;           // broche 9 servo 1
const int pin_servo2 = 10;          // broche 10 servo 2
const int button = 3;               // broche 3 pour le bouton poussoir

//Création des objets servo pour les contrôler
Servo serv1, serv2;

void setup()
{
  // On connecte les broches aux servos
  serv1.attach(pin_servo1);
  serv2.attach(pin_servo2);

  // initialisation position servo moteurs
  initServo();
}

void loop()
{
  if ( digitalRead(button) == HIGH )
    init();
  else {
    const int tempo = 10; // 10 ms

    // position 90° pensant 1s
    delay(1000);

    // accélération servo 2s = 2000ms
    float acc = 90 * tempo / 2000;
    for (float angle(0); angle <= 90; angle += acc) {
      serv1.write(90 - (int) angle);
      serv1.write(90 + (int) angle);
      delay(tempo);
    }
  }
}
```

```
// cycle 3
acc = 180 * tempo / 2000;
for (float angle(0); angle <= 180; angle += acc) {
    servol.write((int) angle);
    servol.write(180 - (int) angle);
    delay(tempo);
}

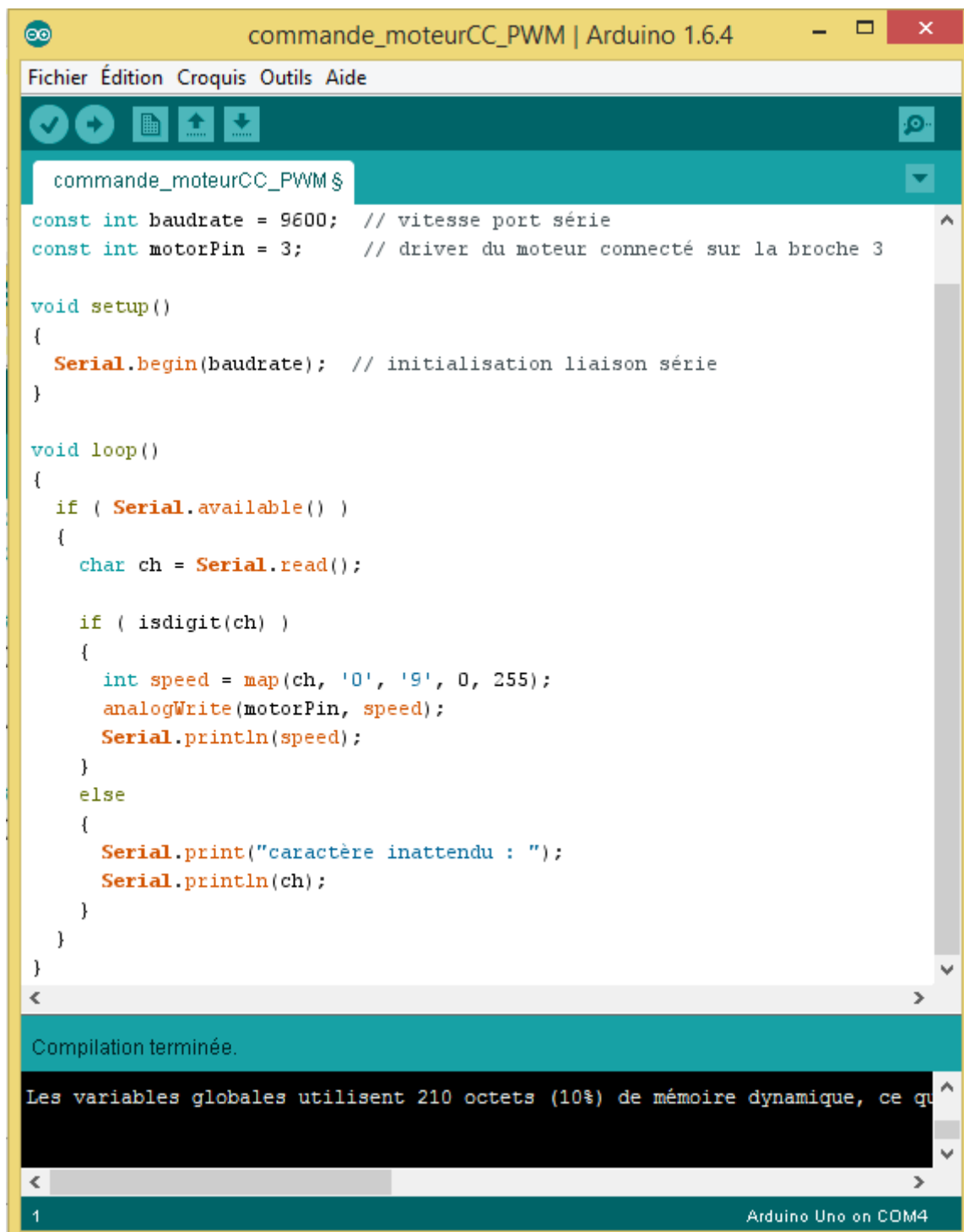
// cycle 8
acc = 90 * tempo / 1000;
for (float angle(0); angle <= 90; angle += acc) {
    servol.write(180 - (int) angle);
    servol.write((int) angle);
    delay(tempo);
}
}

void initServo()
{
    // initialisation positions des servo moteurs à 90°
    servol.write(90);
    servo2.write(90);

    delay(500); // attente 500ms pour que les servos se mettent en position
}
```

6.2. Moteur à courant continu en PWM

Le programme suivant est utilisé pour commander en PWM la vitesse d'un moteur en courant continu. La vitesse est entrée au clavier par un caractère ch compris entre 0 et 9 qui est alors converti en valeur de 0 à 255 avec l'instruction *map*. Pour voir la valeur, ouvrir le Moniteur Série par Outils > Moniteur Série.



```
commande_moteurCC_PWM | Arduino 1.6.4
Fichier Édition Croquis Outils Aide

commande_moteurCC_PWM $
const int baudrate = 9600; // vitesse port série
const int motorPin = 3;    // driver du moteur connecté sur la broche 3

void setup()
{
  Serial.begin(baudrate); // initialisation liaison série
}

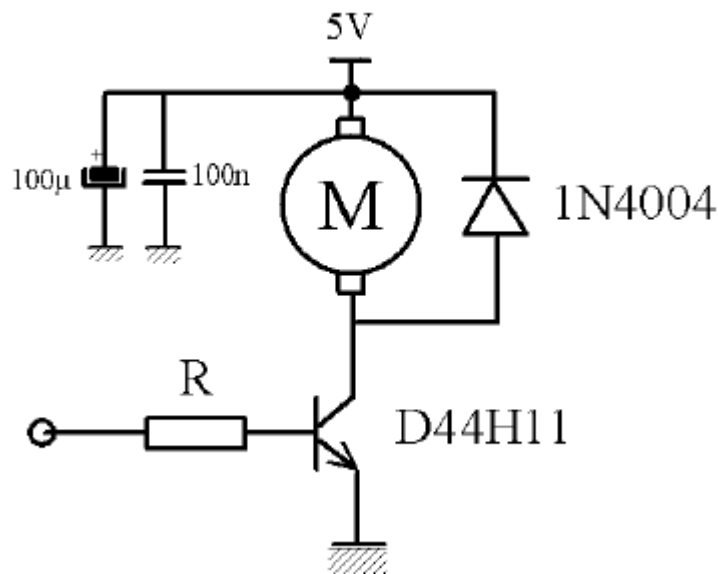
void loop()
{
  if ( Serial.available() )
  {
    char ch = Serial.read();

    if ( isdigit(ch) )
    {
      int speed = map(ch, '0', '9', 0, 255);
      analogWrite(motorPin, speed);
      Serial.println(speed);
    }
    else
    {
      Serial.print("caractère inattendu : ");
      Serial.println(ch);
    }
  }
}

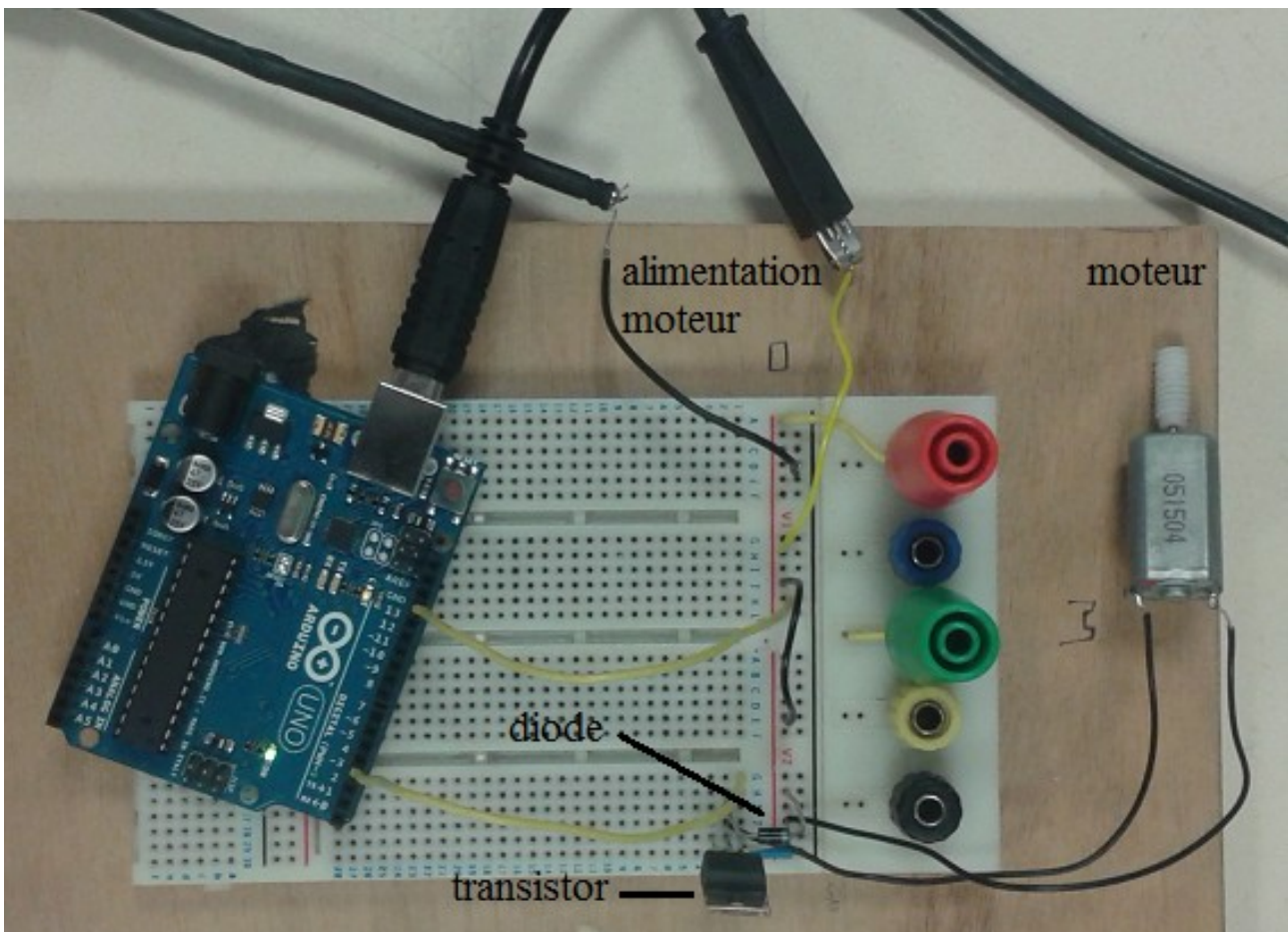
Compilation terminée.

Les variables globales utilisent 210 octets (10%) de mémoire dynamique, ce qu
```

On connectera la sortie Arduino utilisée au montage suivant :

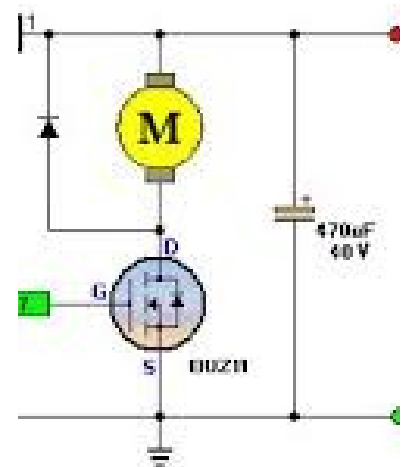
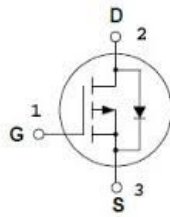
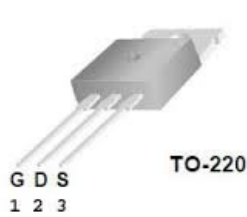


Une alimentation extérieure délivre la tension 5V nécessaire pour alimenter le moteur. Attention au sens de branchement de la capacité de 100 μ F car elle est polarisée (pole positif et pole négatif). Les masses de l'alimentation et de l'Arduino doivent être reliées.



1. Déterminer la valeur de la résistance R à partir du gain en courant β_{\min} ou hFE_{\min} du [transistor D44H11](#), ou équivalent, et du courant circulant dans le moteur qu'on estime à 100 mA. Voir le professeur pour le calcul et la ressource cours sur les transistors. Autre solution

utiliser un transistor MOS comme le IRF520N.



2. Indiquer le rôle de la diode et des capacités de découplages ([aide en ligne.](#))
3. Observer à l'oscilloscope Mesurer la tension délivrée par l'Arduino et la tension V_{CE} du transistor.
4. Mesurer au voltmètre RMS_True la tension délivrée par l'Arduino et la tension V_{CE} du transistor.

Pour voir le résultat, cliquer sur l'image

7. Références

- [Tutoriel Arduino](#)
- [Manuel de référence Aduino](#)
- [langage C](#)
- [langage C++](#)
- [la résistance](#)
- [la diode](#)
- [le Transistor](#)
- [Travaux Pratique Arduino](#)
- [Référence Langage Arduino](#)
- [les datashhets](#)