Microcontrôleurs et carte ARDUINO

R. BOCQUET - A. CUISSET - D. SADOVSKII - ULCO

28 septembre 2017

Table des matières

Chapitre 1

Carte ARDUINO

1.1 Introduction

Le but de ces travaux pratiques est de vous initier à l'utilisation des microcontroleurs qui ont littéralement envahis le monde technologique d'aujourd'hui. Un microcontrôleur n'est rien d'autre qu'un microprocesseur à jeu d'instructions limité spécialisé dans les communications avec l'extérieur. On retrouve ce type de composants dans les voitures par exemple où ils gèrent l'ensemble des capteurs et éléments de sécurité du véhicule, dans les drones où ils gèrent les capteurs de vitesse, d'altitude, de lacet, roulis et tangage, dans les robots où bien souvent des radars anti-collisions sont mis en place ou bien encore dans les imprimantes 3D où les moteurs ainsi que les positionnements sont gérés par ce type de composant. Les microcontrôleurs ont été développés dans les années 80 mais ont réellement diffusés dans le grand public depuis 2005 grâce au travail d'un groupe d'italiens dans le monde du logiciel libre qui a développé une plateforme logiciel simplifiant l'utilisation de ces composants. Il s'agit des développements connus sous la bannière ARDUINO et repris maintenant sous le nom GENUINO, adhérant à la charte du développement "libre".

Nous utiliserons le microcontôleur ATmega328 dont le brochage et les caractéristiques générales sont données dans les figures suivantes (doc ATMEL). Rassurez vous cependant, vous n'aurez à utiliser le microcontrôleur seul que dans les phases finales d'intégration de vos développements, lors des projets si vous en avez le temps. Vous utiliserez la plupart du temps le microcontrôleur dans son environnement de développement ARDUINO. La documentation complète et les explications des différentes broches et entrées sorties sont données sur le site du frabicant [1], quant à la carte ARDUINO, l'ensemble des documentations ainsi qu'une mine d'exemples sont compulsables sur le site ARDUINO [2]. D'ores et déjà vous pouvez noter l'existence d'un port de communication série (RXD et TXD broches 2 et 3), de convertisseurs analogiques-digitaux (broches ADCi),

1. ATMEL : http ://www.microchip.com/wwwproducts/en/ATMEGA328P
2. ARDUINO : https ://www.arduino.cc

d'un bus I2C (SDA et SDC) très utilisé avec les capteurs, d'une interface série synchrone dénommée SPI (MISO, MOSI et SCK) et de broches dénommées digitales à collecteur ouvert.

Enfin, dans cette introduction, notez que vous trouvez nombre de tutoriaux sur le « net » pour vous aider à découvrir le matériel et le logiciel [3].

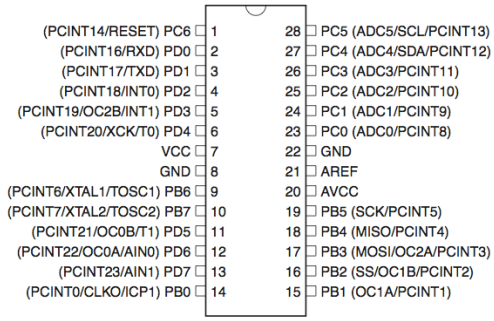


FIGURE 1.1: brochage ATmega328

3. Exemple de tutorial : http ://eskimon.fr

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48PA/88PA/168PA/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48PA/88PA/168PA/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48PA/88PA/168PA/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I^2C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48PA/88PA/168PA/328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - 0 - 20 MHz @ 1.8 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48PA/88PA/168PA/328P:
  - Active Mode: 0.2 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.75 µA (Including 32 kHz RTC)

FIGURE 1.2: Caractéristiques ATmega328

1.2 Carte ARDUINO Uno

Comme indiqué précédemment vous utiliserez l'environnement de développement ARDUINO qui va vous permettre d'écrire un programme en langage proche du langage C. Les instructions sont exécutées par le microcontrôleur placé sur une carte ARDUINO Uno reliée par un câble USB à un ordinateur sur lequel l'environnement de développement (IDE) est installé. L'alimentation de la carte est fournie par l'ordinateur via le câble USB. Vous utiliserez également les plaquettes d'essai disponibles en salle d'électronique pour réaliser vos montages d'électronique.

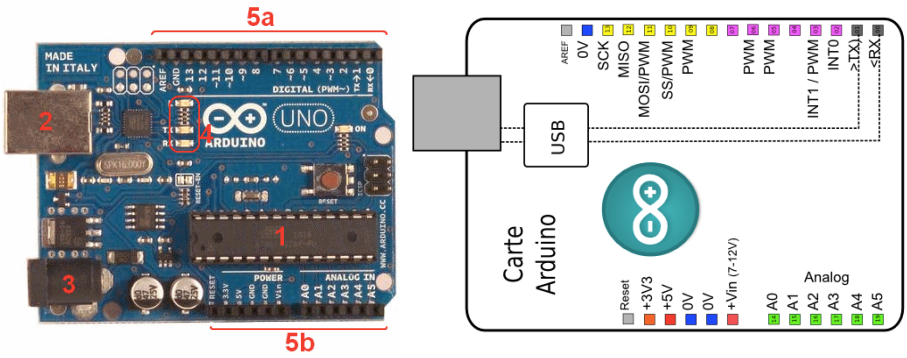


FIGURE 1.3: Carte ARDUINO Uno

Sur la figure 1.3 les numérotations correspondent à :

- 1 : microcontrôleur ATmega328
- 2 : connecteur USB pour relier l'ordinateur
- 3 : jack de connection d'une alimentation extérieure
- 4 : diodes de fonctionnement de la carte dont TX et RX pour visualiser le passage de données dans le port série
- 5a et 5b : connecteurs d'entrée/sortie pour « contrôler le monde extérieur »
  - 5a : entrées sorties digitales qui fonctionnent en niveaux TTL 5V
  - 5b : entrées-sorties analogiques
    - broches des tensions d'alimentation (3V, 5V, masse)
    - Vin permet d'entrer une tension d'alimentation extérieure pour la carte
    - A0 ... A5 : broches d'entrées analogiques

Une des conséquences de l'utilisation de la carte ARDUINO est que les broches digitales sont des niveaux TTL 5V ce qui signifie que le niveau logique 1 correspond à une tension comprise entre 2,4 et 5 V et que le niveau logique bas correspond à une tension comprise entre 0 et 1,4 V. Chaque sortie numérique est limitée en courant à 20 mA. Enfin vous disposez d'une mémoire flash d'une capacité de 32 kO pour stocker votre programme de manière permanente mais

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

FIGURE 1.4: Caractéristiques carte Uno (https ://www.arduino.cc)

bien évidemment effaçable.

1.3 Langage de programmation

Il est basé sur le langage C et vous trouvez sur le site ARDUINO une aide en ligne des principales fonctions. Vous verrez que tout programme présente la même structure :

— une partie définition des constantes qui seront utilisées dans tout le programme
— une partie « void setup() » qui n'est exécutée qu'une seule fois au tout début du programme. C'est là qu'il faut définir les broches que vous allez utiliser (les numéros et si elles sont utilisées en entrée ou sortie).
— une partie « void loop() » qui sera exécutée en boucle de manière infinie, cela constitue le programme principal.

Chapitre 2

Séances de travaux pratiques

Ce chapitre rassemble quelques exemples qui seront traités à l'aide de la carte ARDUINO. Il a comme seul but de vous familiariser avec le matériel et le logiciel dont vous disposez, pour vous préparer aux projets que vous développerez au cours du 2ième semestre. ARDUINO est issu du « monde libre » et bénéficie de ce fait de développements et d'applications innombrables que l'on retrouve sur le « net ». Prenez donc l'habitude lorsque vous développez une application ou lorsque vous installez un composant de rechercher sur le réseau s'il existe une bibliothèque pour votre composant ou si une application similaire a déjà été écrite. Cette façon de procéder permet de développer plus rapidement. Vous aurez une première séance de TP où l'accent sera mis sur la programmation, suivie de 2 séances centrées sur l'utilisation du bus I2C de la carte et des communications pour finir sur une séance d'apprentissage à la gestion de projets collectifs. Les évaluations de ce module seront faites durant l'ensemble des séances par la mise à jour d'un cahier de compétences rempli par un des enseignants à la vue de la réalisation des objectifs. Vous trouvez en fin de topo les critères qui seront retenus.

2.1 Séance 1 : Programmation

Le microcontrolleur Arduino est programmé en `c++` dans un environment specialement dedié. Nous allons dans cette première séance couvrir les bases de programmation et de l'utilisation de cet environment en développant un petit code qui nous permettra de façon évolutive :

1. brancher et reconnaitre votre carte, faire clignoter la LED propre à Arduino (Uno)
2. modifier le code pour faire le signal SOS avec la LED
3. allumer les 4 LED's externes un par un en sequence
4. passer les messages via port série/usb (aka «Terminal» ou «Console»)
5. communiquer avec Arduino via son Terminal par les touches a 1 caractère

6. interpreter les touches, retourner leur codes ASCII
7. pour les touches 0..9, a..f, ou A..F signaler les comme les chiffres hexadecimaux
8. donner la valeur 0..16 du chiffre correspondant
9. donner sa representation binaire (en 4 bits), utiliser opérations «bitshift» et «bitmask»
10. allumer les 4 LED's externes (voir point 3) selon cette representation binaire

2.1.1 Branchement de la carte ARDUINO à l'ordinateur

Vous devez avoir à votre disposition :
— un ordinateur sur lequel le programme de développement ARDUINO est installé
— une carte ARDUINO UNO ou équivalent avec le cable USB idoine
— une plaquette d'essais électronique à trous pour les montages électroniques
— des fils dénudés à chaque bout ainsi que les composants électroniques pour le montage envisagé

Nous allons tester le matériel à disposition en faisant l'expérience d'allumage de la diode électroluminescente de la carte ARDUINO. Ceci est le pendant pour les microcontrôleurs du programme « Hello World » des langages de programmation.

1. Brancher la carte ARDUINO à l'ordinateur par l'intermédiaire du câble USB et démarrer le programme ARDUINO
2. Charger le programme Blink dans Fichiers/Exemples/01.Basics fournis par ARDUINO
3. Dans Outils/types de cartes/ : choisir la carte Genuino Uno
4. Téléverser le programme dans la carte, le port USB de l'ordinateur sur lequel la carte est branché devrait être reconnu automatiquement
5. La diode de la carte doit clignoter à une fréquence de 1 Hz indiquant que l'installation de la carte et du logiciel sont corrects. Vous êtes alors fonctionnels et prêts à utiliser l'environnement ARDUINO.

A noter ici la syntaxe `c,c++` de base, terminaison obligatoire de ligne par `;` et les commentaires ainsi que les deux parties standard du codage Arduino :
— l'initialisation `setup` qui tourne une fois
— le programme principal `loop` qui tourne sans cesse après l'initialisation.

2.1.2 Programme Blink-SOS

On modifie le programme précédant pour faire clignoter la LED suivant un message morse «SOS». Ce signal de détresse consiste en trois appels courts (lettre S) suivis par trois appels longs (lettre O) puis encore 3 courts. On apprend à organiser une boucle `for` :

```
/*
  Programme Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino model, check
  the Technical Specs of your board  at https://www.arduino.cc/en/Main/Products
  This example code is in the public domain.
*/
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

```
// Modification du programme Blink
for(i=3; i; i--) {
    digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(200);                        // wait for a 200 milliseconds
    digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
    delay(200);
  }
  delay(200);
```

ici `i` est déclaré auparavant comme une variable du type `int`.

2.1.3 Allumage - LED extérieures

Propos liminaires Pour ceux qui n'ont pas suivi le module d'électronique en L2, une LED est une diode électroluminescente c'est à dire qui s'éclaire lorsqu'un courant la parcourt. Elle sert beaucoup pour visualiser de façon très simple l'état logique d'une ligne (haut/bas ou vrai/faux). C'est un dipôle non symétrique constitué d'une anode et d'une cathode. Lorsque la diode est polarisée dans le sens direct (+V sur l'anode et 0V sur la cathode) la diode est passante et un courant circule dans la diode, lorsqu'elle est polarisée en inverse, la diode est bloquée, aucun courant ne circule. La figure 2.1 donne les caractéristiques courant-tension de LED de diverses couleur. Vous remarquerez que lorsque la diode est passante, une tension est présente à ses bornes. On appelle cette tension, la tension de coude et on peut en première approximation prendre une valeur de 1,8 V pour les LED et vertes ou rouges que vous avez en TP. La cathode est répérée par la patte la plus courte de la diode.

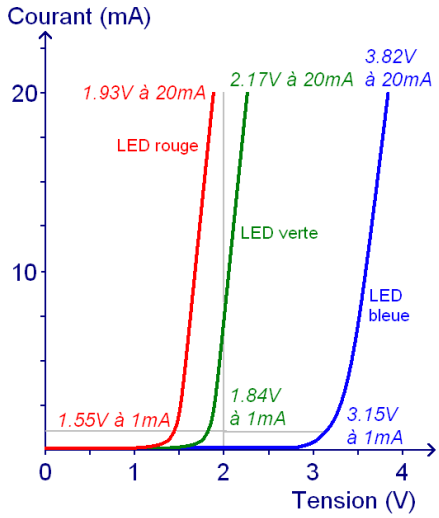
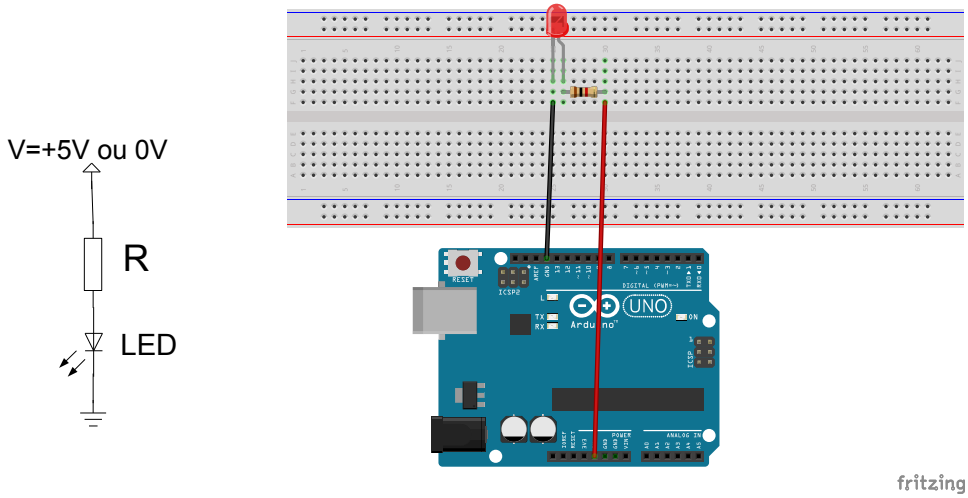


FIGURE 2.1: Caractéristiques courant - tension de LED (https ://www.astuces-pratiques.fr)

Avant de commencer le TP, familiarisez vous avec la diode en faisant un petit montage comme indiqué figure 2.2.

Vous utiliserez l'alimentation de la carte ARDUINO pour fournir l'alimentation et vous calculerez la valeur de la résistance R pour limiter le courant de diode à 15 mA. En branchant alternativement le fil rouge au 5V et la masse, vérifiez que vous allumez et éteignez la diode. Vérifiez également que si vous



(a) Schéma de montage LED (b) Montage de la LED

FIGURE 2.2: Montage préliminaire à LED

inversez le sens de la diode dans le montage, la diode reste éteinte. Expliquer pourquoi la diode reste éteinte dans ce dernier cas.

Clignotement d'une LED Vous avez vu avant que la LED peut être commandée en envoyant +5V (allumée) ou 0V (éteinte) sur le montage précédent. Plutôt que faire cette commutation à la main, nous allons utiliser une sortie digitale de la carte ARDUINO et programmer la carte de telle sorte que la sortie fournisse alternativement un niveau haut (+5V) puis un niveau bas (0V).

En reprenant et modifiant le programme « Blink », il s'agit d'allumer une LED dont l'anode sera branchée sur une des sorties numériques de la carte, par exemple la sortie n°8 comme indiqué dans la figure 2.3. Vous limiterez le courant de la LED à 15 mA. Vous adapterez le programme pour avoir un clignotement à une fréquence de l'ordre du Hertz. Essayez d'autres fréquences et d'autres sorties digitales.

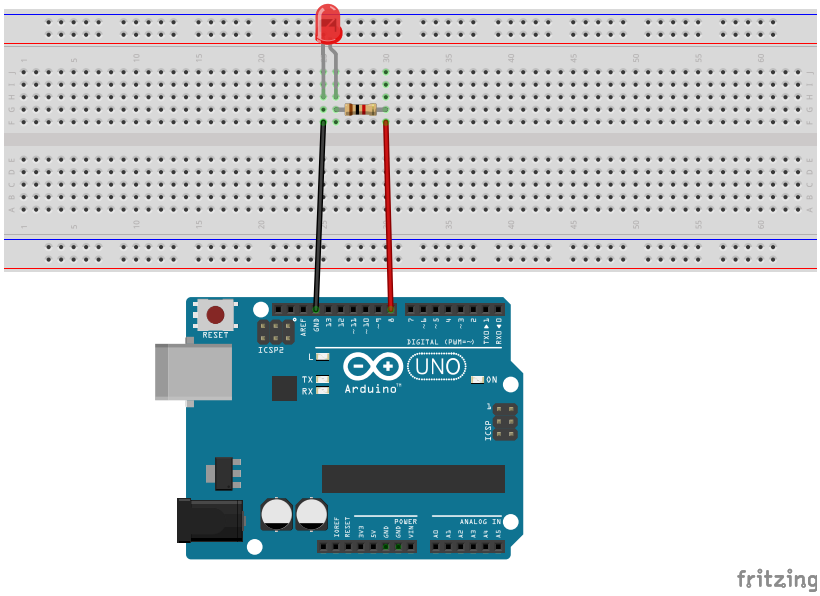


FIGURE 2.3: Allumage d'une LED

Montage à 4 LED's Vous utiliserez 4 LED branchées sur des sorties digitales et des résistances entre les cathodes de LED et GND d'une valeur de 220 Ohm.

Dans la partie `setup`, nous pouvons initier les quatres sorties digitales avec une petite boucle :

```
for(j=PIN_BASE,i=4; i--; pinMode(j++, OUTPUT));
```

où l'utilisation des operations `++` et `--` est à noter. Et par la suite, nous pouvons aussi allumer nos LED's une par une :

```
for (j=PIN_BASE, i=4; i--; j++) {
    digitalWrite(j, HIGH);    // turn the LED on (HIGH is the voltage level)
    delay(200);               // wait for a 200 milliseconds
    digitalWrite(j, LOW);     // turn the LED off by making the voltage LOW
    delay(200);
  }
```

pour tester notre montage.

2.1.4 Interaction avec le port série - Lecture analogique

Le port série classique est crée par Arduino à travers son interface USB. Pour initier ce port et travailler avec la vitesse de 9600 baud (la vitesse typique des anciens terminaux et modems sur le porter série RS232), nous ajoutons dans le programme d'initialisation `setup`

```
Serial.begin(9600);
```

Pour communiquer les informations par ce port, une fois que notre carte a été identifiée (comme `ttyACM0` dans ce texte, ou `COM1`, `COM2`, etc sous WINDOWS), nous ouvrons le terminal de l'environment Arduino IDE. Notez, qu'à l'ouverture du terminal, ainsi qu'après le téléchargement d'un nouvel exécutable, la carte se redémarre en exécutant le `setup`.

Sortir les informations : le texte (string) Nous pouvons sortir un message de demarrage, par exemple

```
Serial.println("\nSOS blinking and Serial interaction via keys");
```

Notez, qu'à la difference de `Serial.print` la commande `Serial.println` y ajoute automatiquement le changement de ligne. Avec ces deux commandes, nous pouvons communiquer le texte, dit «string», entouré par les signes double de citation `"`.

Le caractère, le byte, le bit .. Chaque string consiste d'un ou plusieurs caractères, et chaque caractère est representé par un nombre 0..255 (où les caractères imprimables/lisibles demarrent à 32). On tel nombre occupe un *byte*, c. à d., *huit* registres binaires 0/1 dits *bits* ($255 = 2^8 - 1$). Dans le language `c,c++`, on déclare ces nombres avec le type `char` ou plus précisement `unsigned char`. Il existe plusieures façons de definire la correspondance (codage) entre les caractères et les ces nombres. Actuellement, le plus repondu et simple est le codage ASCII.

NB : les types représentant les nombres entiers A la différence de `unsigned char`, le type `char` définit les nombres entiers dans l'intervalle $-127 \ldots 127$ qui occupent aussi un byte dont les 7 bits inférieures gardent la valeur ($127 = 2^7 - 1$) et le 8me bit sert à donner le signe. Les nombre entiers, quant à eux, sont définis par le class `int` ou enfin `unsigned int` et occupent *deux* bytes. Par consequent, ces nombres vont jusqu'à $\pm(2^{15} - 1)$ ou bien $2^{16} - 1$.

Entrée d'information Pour envoyer des information à la carte, nous les entrons dans la ligne de terminal suivies par la touch «Enter» qui les envoye. Les information sont transmises comme un texte qui est mis en attente dans le tampon de l'interface série/usb dit «buffer». Pour savoir si les informations nous attendent à l'entrée, on appelle `Serial.available()`

Montage du potentiomètre Branchez les extrémités dn potentiomètre de $R = 10$ à 100 KOhm entre GND et la sortie 5V coté analogique (A0), prévoyez une résistance de limitation de courant montée en série (faites attention à repérer chacune des connexions du potentiomètre, on veut l'utiliser en résistance variable). La programmation est simple, voir l'exemple de code «AnalogInput» dans « Fichiers/Exemples/Analog/ ». La partie essentielle du code est

```
int sensorPin = A0, sensorValue = 0;
  sensorValue = analogRead(sensorPin);
```

où nous appelons `analogRead` pour lire le voltage présent sur l'entrée analogique.

Remarque : Vous utiliserez la fonction analogRead() pour mesurer la tension présente sur la broche A0. Cette fonction effectue la conversion analogique numérique (A/N) de la tension avec un convertisseur sur 10 bits. Cela signifie que l'on transforme la tension présente sur l'entrée A0 en une valeur lisible et exploitable par le micro-contrôleur. La conversion consiste à comparer cette tension à une valeur de référence fixée à 5 V par défaut. Il est possible de comparer par rapport à une tension comprise entre 0 et 5 V qui est appliquée à la borne Aref de la carte. Il faut alors le spécifier lors de l'utilisation de la commande analogRead(reftype). Ainsi les tensions de $0 \ldots 5$V correspondent aux valeurs numériques entre 0 et 1023, respectivement.

2.2 Séances 2 et 3 : Initiation au micro-controleur

2.2.1 Clignotement d'une LED - « pull up »

On peut faire également le montage correspondant au schéma de la figure 2.4. Cette fois on amène une tension d'alimentation à la LED, de l'extérieur et la commutation de la sortie digitale de l'ARDUINO permettra de faire passer alternativement la diode de l'état passant à l'état bloqué.

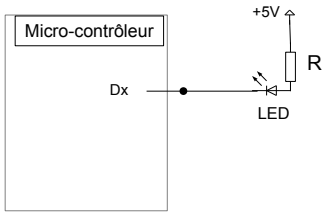
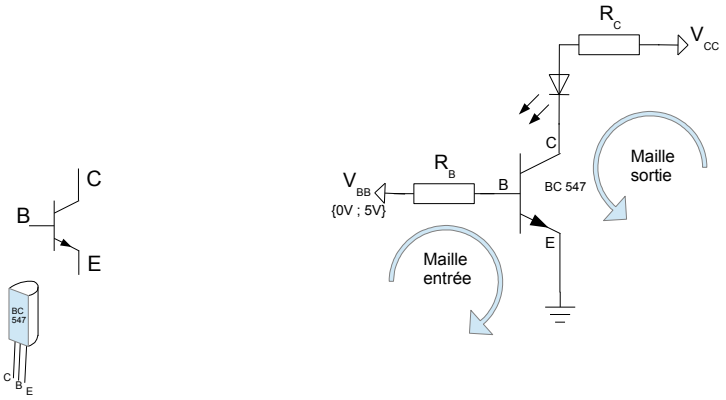


FIGURE 2.4: LED pull up

Faites le montage correspondant et modifiez le programme pour avoir un clignotement de la diode à 1 Hz. Vous utiliserez le 5V issu de la carte ARDUINO pour la tension d'alimentation. Expliquer le fonctionnement du montage, notamment à quelle condition la LED s'allume.

2.2.2 LED forte puissance

On est parfois obligé de commander des instruments ou des équipements qui peuvent consommer des puissances incompatibles avec les sorties du micro-contrôleur. Recherchez sur les caractéristiques techniques de la carte ARDUINO et du fabricant ATMEL du micro-contrôleur, les courants maximum que peuvent fournir les sorties digitales et le courant maximum total que peut fournir la carte UNO. On est amené dans le cas de fortes consommations à découpler les circuit d'utilisation (forte intensité) et de commande (faible intensité). Pour cela on utilise classiquement un transistor bipolaire en commutation. C'est un élément électronique comportant 3 broches dénommées « Collecteur », « Base » et « Emetteur ». Nous prendons ici l'exemple de la commande d'une LED.



(a) Brochage du BC 547 (b) Schéma typique d'utisation

FIGURE 2.5: Transistor NPN BC547

Le transistor bipolaire NPN Il s'agit d'un composant « transcourant » [1] à 3 broches, créé par l'empilement de semi-conducteurs de type N, P et N, pour ainsi dire équivalent à 2 diodes montées en tête bêche. La figure 2.5b donne un schéma typique d'utilisation du transistor dont le fonctionnement est régi par 3 équations :

1. transcourant : le courant de sortie est proportionnel au courant d'entrée mais a été amplifié

1. Equation de la maille d'entrée donnant le courant d'entrée : $V_{BB} = R_B I_B - V_{BE} \Rightarrow I_B = \frac{V_{BB}-V_{BE}}{R_B}$
2. Equation de la maille de sortie donnant le courant de sortie : $I_C = \frac{V_{CC}-V_{LED}-V_{CE}}{R_C}$
3. Equation du transistor : $\begin{cases} I_C &= \beta_{cc} I_B \\ I_E &= I_C + I_B \end{cases}$, $\beta_{cc} \approx 200 \Rightarrow I_C \approx I_E$

On peut apporter des simplifications en tenant compte du fait que les tensions V_{BE} et V_{LED} sont des tensions de coude de diodes polarisées en direct valant respectivement 0,65 V et 1,8 V. La première peut-être négligée devant les autres tensions.

Equations simplifiées du transistor : $\begin{cases} I_B &= \frac{V_{BB}}{R_B} \\ I_C &= \frac{V_{CC}-V_{LED}-V_{CE}}{R_C} \end{cases}$ et $I_C = \beta_{cc} I_B$

Remarquons plusieurs choses :

1. si $V_{BB} = 0 \Rightarrow I_B = 0 \Rightarrow I_C = 0$: le transistor est bloqué, aucun courant ne circule
2. Le courant de sortie maximum noté I_{sat} est obtenu pour $V_{CE} = 0 \Rightarrow I_{sat} = \frac{V_{cc}-V_{LED}}{R_C}$ et correspond à l'état saturé du transistor

Dans l'utilisation qui sera faite du transistor, on souhaite qu'il travaille en commutation pour les niveaux d'entrée TTL correspondant aux sorties digitales du micro-contrôleur soit $V_{BB} \in \{0\,V,\ 5\,V\}$. Le niveau bas correspond à l'état bloqué du transistor, reste à calculer les valeurs des résistances et de la tension V_{CC} pour le courant de saturation que l'on souhaite :

$$I_{sat} = Inf\left\{ \frac{V_{CC}-V_{LED}}{R_C}\ ,\ \beta_{cc}\frac{V_{BB}}{R_B} \right\}$$

Dans la pratique on s'arrange pour prendre une valeur de résistance de base ($R_B = 1\,k\Omega$) pour que le courant de base soit de l'ordre du mA et que le courant de saturation soit limité par $I_{sat} = \frac{V_{CC}-V_{LED}}{R_C}$. En effet, dans ce cas, si on prend une valeur standard $\beta_{cc} = 200$, pour l'état logique haut ($V_{BB} = 5V$), une tension $V_{CC} = 20\,V$, une résistance $R_C = 100\,\Omega$, nous obtenons :

$$\begin{aligned} I_{satB} &= \beta_{cc}\frac{V_{BB}}{R_B} = 1\,A \\ I_{satC} &= \frac{V_{CC}-V_{LED}}{R_C} = 200\,mA \end{aligned}$$

Le courant circulant dans la diode est donc limité par V_{CC} et R_C tandis que le courant circulant dans la sortie du micro-contrôleur est de l'ordre de 5 mA.

Montage Réaliser un montage tel que proposé figure 2.5b en choisissant les valeurs de tension et de résistance pour limiter le courant de diode à 30 mA. La tension d'entrée du montage, VEE sera donnée par une sortie digitale que l'on peut commander.

2.2.3 Moteur à courant continu

Sur la base du montage précédent, on se propose de commander un moteur à courant continu qui est un exemple typique d'élément qu'il n'est pas possible d'alimenter directement par les sorties digitales de la carte ARDUINO. Les moteurs dont vous disposez peuvent fonctionner avec des tensions d'alimentation comprises entre 2,5 V et 6 V. Vous utiliserez une tension extérieure d'alimentation stabilisée à 5 V car les consommations du moteur vont de 0,2 A à 1,5 A selon la charge.

Le shéma de principe du montage est donné figure 2.6 où Dx est une sortie digitale de la carte notée d'un tilda (∼) correspondant à des sorties PWM (voir montage suivant). Il s'agit pour la carte UNO des E/S digitales 3, 5, 6, 9, 10 et 11.

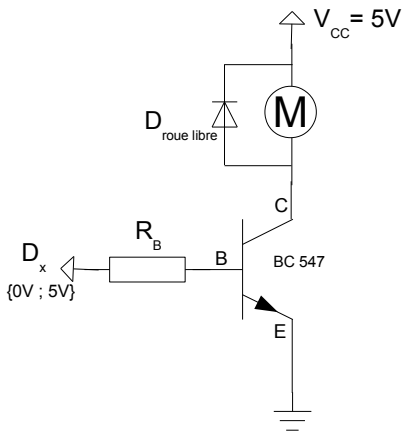


FIGURE 2.6: Montage moteur à CC

Vous prendrez une résistance de base de l'ordre de 1 $k\Omega$ pour limiter le courant de la sortie digitale à une quarantaine de mA. Vous remarquerez l'utilisation d'une diode dite de « roue libre » en parallèle sur le moteur. Elle permet d'être inutilisée lorsque l'état logique de la sortie digitale est haut mais permet d'absorber le courant créé par le moteur en mode « générateur » lorsque la sortie digitale est au niveau bas. En effet, dans ce dernier cas, le transistor est bloqué

et le moteur n'est plus alimenté. Cependant il continue à tourner à cause de son inertie et fournit alors un courant qui serait susceptible de détruire le transistor si la diode de roue libre n'était pas mise.

Montage : Vous réliserez le montage et modifierez un programme pour faire tourner et arrêter le moteur de façon cyclique.

2.2.4 Modulation de la vitesse du moteur à CC

L'idée pour moduler la vitesse du moteur est d'agir sur la puissance moyenne délivrée au moteur. Plutôt que d'alimenter tout le temps le moteur, nous allons le démarrer pendant un certain temps, puis couper l'alimentation également pendant un certain temps et recommencer de façon cyclique cette procédure. Si la période du cycle est suffisamment courte, le moteur n'a pas le temps de s'arrêter et on obtient un fonctionnement du moteur avec une vitesse proportionnelle à la moyenne de puissance temporelle. L'exemple du cycle de commande du moteur est donné figure 2.7.

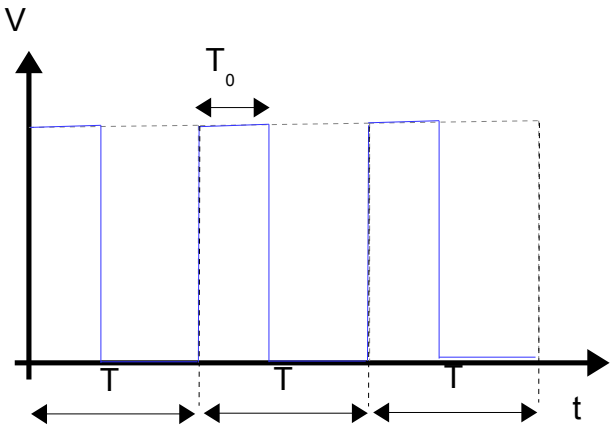


FIGURE 2.7: Modulation de la commande de moteur

En réglant le rapport cyclique $\frac{T_0}{T}$ de 0 à 100% on pourra moduler la vitesse du moteur de 0 à Vmax. C'est justement ce qu'est capable de faire la carte ARDUINO avec les sorties PWM pour Pulse Width Modulation notées par un tilda (∼). Ces sorties se commandent dans le mode PWM par l'instruction :

```
analogWrite(pin,valeur);
```

avec « pin » le numéro de la broche utilisée qui aura été définie dans le setup en sortie et « valeur » un nombre compris entre 0 et 255 correspondant au rapport cyclique de la sortie, respectivement 0 à 100 %. Le taux de répétition du signal de sortie est d'environ 500 Hz.

Montage : Reprenez le montage précédent et modifiez le programme pour moduler la vitesse de rotation du moteur. Vous pourrez par exemple créer une boucle avec temporisation qui incrémente la vitesse par pallier de 25%.

2.2.5 Capteur de température

Le micro-contrôleur est réellement adapté pour l'utilisation de capteurs au sens large. Il s'agit bien souvent de mesurer une grandeur électrique (tension ou courant) ou un temps qui sont proportionnels à la grandeur physique que mesure le capteur. Nous vous proposons d'utiliser et de mettre en oeuvre un capteur de température basé sur une thermistance. Vous mettrez en place la liaison série vers le moniteur série pour visualiser les données de votre capteur.

Capteur La thermistance dont vous disposez est une CTN (Coefficient de Température Négative) c'est à dire que la résistance diminue avec l'augmentation de la température. Il existe également des CTP (Coefficient de Température Positif). Elle a une résistance qui varie en fonction de la température suivant une loi définie par un polynome de degré 3 donné par la relation de Steinhart-Hart :

$$\frac{1}{T} = a + b.Log(R_T) + c.\left[Log(R_T)\right]^3$$

Les coefficients du polynome sont :

$$a = ;\ b = ;\ c =$$

On peut également utiliser l'approximation suivante de la formule de Steinhart-Hart, donnée pour une plage de température plus restreinte :

$$\frac{R_T}{R_0} = exp\left[B\left(\frac{1}{T} - \frac{1}{T_0}\right)\right]$$

Les valeurs de B sont référencées par le fabricant pour une gamme de température donnée en kelvin. La thermistance que vous avez à votre disposition présente une résistance de $100\,k\Omega$ à 25°C et les caractéristiques électriques sont données ci-dessous :

Electrical specification and ordering codes

R_{25} Ω	No. of R/T characteristic	$B_{25/85}$ K	$B_{0/100}$ K	$B_{25/100}$ K
5 k	8326	3483	3456	3497 ±1%
10 k	7003	3612	3586	3625 ±1%
10 k	8307	3478	3450	3492 ±1%
100 k	8304	4072	4036	4092 ±1%

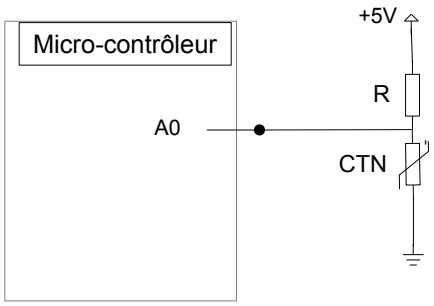


FIGURE 2.8: Montage diviseur de tension pour la CTN

Montage diviseur de tension En utilisant le +5V de la carte, réalisez un montage diviseur de tension comme indiqué figure 2.8 en prenant $R = 100\,k\Omega$. Vous utiliserez l'entrée analogique A0 de la carte pour mesurer la tension aux bornes de la résistance. Pour une variation de température de 25 à 30°C, calculez les valeurs de thermistance attendues et les valeurs de tension correspondantes sur l'entrée A0 de la carte.

25 ; 100 ; 2,5 ; 30 ; 80 ; 1,11

Convertisseur A/N Vous utiliserez la fonction analogRead() pour mesurer la tension présente sur la broche A0. Cette fonction effectue la conversion analogique numérique (A/N) de la tension avec un convertisseur sur 10 bits. Cela signifie que l'on transforme la tension présente sur l'entrée A0 en une valeur lisible et exploitable par le micro-contrôleur. La conversion consiste à comparer cette tension à une valeur de référence fixée à 5 V par défaut. Il est possible de comparer par rapport à une tension comprise entre 0 et 5 V qui est appliquée à la borne Aref de la carte. Il faut alors le spécifier lors de l'utilisation de la commande analogRead(reftype).

Montage Réaliser le montage et modifier ou écrire un programme vous permettant de lire la valeur du capteur et afficher sur le moniteur série cette valeur ainsi que la T° calculée.

2.2.6 Utilisation du bus I2C pour affichage

Vous utiliserez le bus I2C pour visualiser la mesure de T° de votre montage précédent, sur un écran alphanumérique équipé I2C.

Beaucoup de capteurs et d'équipements sont embarqués sur une minicarte comportant un micro-contrôleur mettant en oeuvre des moyens de communica-

tion évolués. On retrouve en général des capteurs avec bus I2C ou SPI. Vous pouvez trouver de nombreuses références sur la mise en place de ces bus. Nous donnerons ici quelques explications succintes sur le bus I2C, le plus léger à mettre en place et qui est souvent utilisé avec les capteurs ARDUINO. La bibliothèque « Wire » fournie par ARDUINO permet d'utiliser simplement ce bus.

Le bus I2C I2C est l'acronyme de « Inter Integrated Circuit » et est un protocole de communication entre un maître (master en anglais) et un ou plusieurs esclaves (slaves en anglais) qui n'utilise que 2 fils :

— SDA pour les données
— SCL pour l'horloge.

L'idée est simple : un maître envoie des ordres à un esclave qui les exécute et renvoie éventuellement un message en retour. Pour identifier l'esclave visé dans le message, chaque esclave possède une adresse unique, définie soit de façon logicielle pour des systèmes intelligents comme ARDUINO, soit de façon matérielle par la commutation d'interrupteurs présents sur le capteur, soit il est prédéfini par le constructeur du composant ou du capteur. Le programme « Scanning_I2C_bus » que l'on trouve sur le net permet de trouver l'ensemble des capteurs et leur adresse présents sur le bus I2C.

La carte ARDUINO UNO que vous avez à votre disposition utilise les broches A4 et A5 respectivement pour SDA et SCL. Lisez la documentation de la bibliothèque Wire pour de plus amples explications et pour découvrir les fonctions accessibles.

Affichage de la T° vers un écran alphanumérique Vous avez à votre disposition un écran de 2 lignes de 16 caractères basé sur le composant I2C/SPI 2 x 16 EF03111. Il présente des interfaçages possibles en I2C et SPI. Nous utiliserons le bus I2C. Recherchez sur le site du frabicant les notices techniques et d'utilisation. Vous devrez utiliser les bibliothèques Wire et LiquidCrystal.

Régler l'adresse I2C de l'écran à la valeur que vous souhaitez, reprenez le montage précédent en y adjoignant l'écran et créez un programme qui affiche la T° sur l'écran alphanumérique.

2.2.7 Capteur température et pression

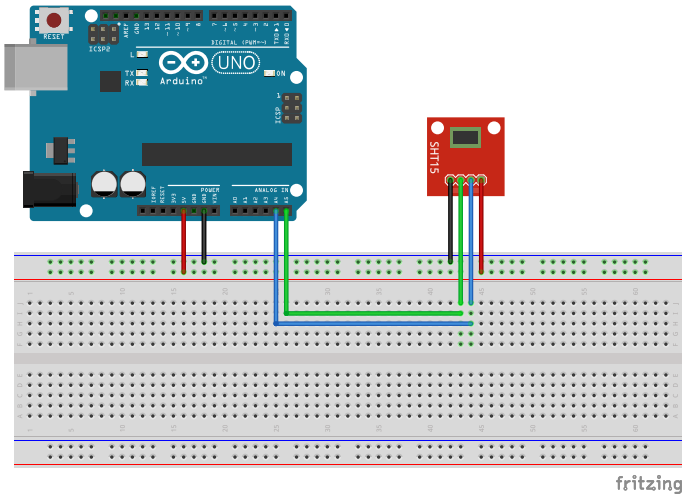


FIGURE 2.9: Capteur sur bus I2C

Documentez vous sur le capteur dont vous disposez et téléchargez la bibliothèque Wire dans votre programme.

Réalisez le montage de base comme indiqué figure 2.9. Sur la carte UNO, les connexions sont les suivantes :

— SDA : A4
— SCL : A5

Repérez bien sur votre capteur les connexions SDA et SCL du bus I2C.

Trouver l'adresse du capteur L'adresse en hexadécimal du capteur a dû être programmée en usine et nous allons la lire à l'aide d'un programme. Recherchez sur le net le programme « Scanning_I2C_bus » le télécharger et le faire tourner. Il va scanner l'ensemble des adresses I2C possibles et communiquer sur le moniteur les résultats trouvés.

Lire les données du capteur Mettez en place la liaison série et affichez sur le moniteur la pression et la température.

2.3 Séance 4 : Organisation de projets

Il s'agira lors de cette séance d'acquérir les méthodes et les outils fondamentaux de la gestion de projet pour piloter un projet avec succès et se doter d'une boîte à outils. A l'issue de la séance, les apprenants seront capables de :

- S'approprier les notions clés de la gestion de projet
- Identifier le rôle et les responsabilités des pilotes
- Identifier les étapes clés d'un projet et le processus de mise en œuvre
- Conduire un projet en mettant en œuvre une méthode et des outils opérationnels
- Identifier les ressources pour la réussite d'un projet
- Débloquer les situations difficiles dans la gestion de projet

Enfin, la deuxième partie de la séance sera consacrée à la présentation des projets que vous réaliserez au 2 ième semestre puis à la constitution de groupes et enfin à la préparation de l'organisation du projet. Vous aurez à appliquer sur le projet les méthodes vues avant. Notamment vous aurez à :

- Appliquer un QQOQCP ; (méthode couramment utilisée en entreprise à appliquer ici sur un « petit projet »)
- Définir un plan d'actions ; (décomposition en tâches et sous-tâches, attribution des rôles, identifications des ressources, risques et parades...)
- Organiser ce plan d'actions dans le temps et définir des livrables (diagramme de Gantt) ;

Ces trois points seront évalués par l'intermédiaire du cahier de compétences.

Chapitre 3

Cahier de compétences

Les compétences seront évaluées tout au long des séances de TP par les enseignants responsables, sur la base de jalons définis ci-après. Ils ne correspondent pas forcément à un montage de TP mais nécessitent de montrer à l'enseignant que vous maîtrisez les compétences attendues. La note finale du module sera basée sur le taux de réalisation des compétences. Bien évidemment vous n'êtes pas obligé d'atteindre toutes les compétences. Travaillez à votre rythme, l'essentiel étant de comprendre ce que vous faites.

Compétences et savoirs	**A ou NA**	**Remarques**	**Enseignant**
Démarrer la carte Arduino			D. Sadovskii
Coder un message SOS sur la LED interne			D. Sadovskii
Allumer 4 LED			D. Sadovskii
Utiliser le Port série et la conversion A/N		communication série ordinateur - carte	D. Sadovskii
Allumer une LED en pull up			R. Bocquet
Montage forte puissance - exemple d'une LED		dissocier la commande de l'utilisation	R. Bocquet
Démarrer un moteur à CC		montage forte puissance à transistor	R. Bocquet
Contrôler la vitesse de rotation du moteur à CC		Pulse Width Modulation (PWM) de ARDUINO	R. Bocquet
Mesurer la température à l'aide d'une thermistance		bibliothèque Math.h et la conversion A/N	R. Bocquet
Afficher la mesure de température sur un écran LCD		bibliothèques Wire.h et LiquidCrystal.h ainsi que l'utilisation du bus I2C	R. Bocquet
Mesurer la pression à l'aide d'un capteur I2C, visualisation sur écran LCD		considérer cette compétence comme optionnelle	R. Bocquet
Appliquer un QQOQCP			A. Cuisset D. Sadovskii
Définir un plan d'actions			A. Cuisset D. Sadovskii
Diagramme de Gant			A. Cuisset D. Sadovskii

TABLE 3.1: Cahier de compétences (A pour acquis, NA sinon)