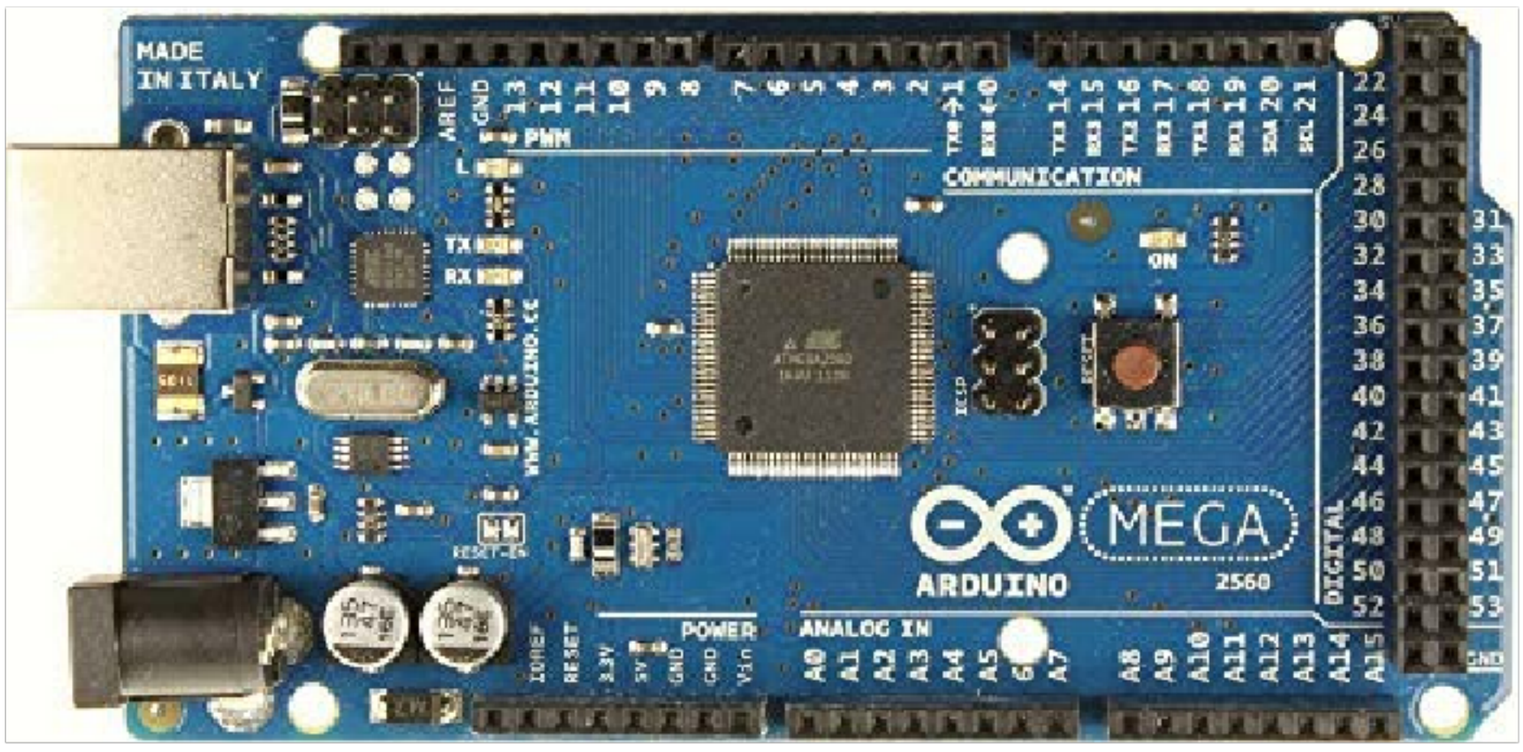
POLYTECH
LILLE

# Projet – tutorat semestre 6
# Microprocesseurs




Tekin Kadir
Soulaimani Jawad
Raouak Haroun

IMA3 TD2
05/04/2019

**Sommaire**

# Introduction

Vendu à plus d'un million d'exemplaire, l'Arduino est un véritable best-seller. Aussi bien utilisé par les particuliers et que par les professionnels, cette carte est connue de tous. Du fait de sa notoriété, une grande communauté s'est bâtie autour de l'Arduino permettant à de nombreux projets de voir le jour. Aujourd'hui, nous allons nous pencher sur une application visant à commander un store en utilisant cette carte, et plus particulièrement avec un modèle d'Arduino sous ATMEGA2560.

# 1. Présentation du cahier des charges

Notre projet consiste à réaliser la commande d'un store permettant d'ombrager une terrasse en période estivale.

Pour réaliser ce projet, nous avons suivi le cahier des charges suivants :

a) le store peut être commandable en mode manuelle ou automatique, le choix doit se faire à l'aide d'un bouton poussoir 2 états :
- en position droite on envoie 5V sur l'ATMEGA, le mode auto est alors activé.
- en position neutre ou gauche, la tension ne circule pas, nous passons ainsi en mode manuelle

b) en mode manuel, l'utilisateur a deux boutons : un pour élever le store, et un pour l'abaisser

c) le store s'ouvre ou se ferme selon le sens de rotation du « moteur tout ou rien »
- une tension positive fait tourner le moteur dans un sens positif, entraînant l'élévation du store,
- et inversement avec une tension négative

d) le moteur est un moteur « tout ou rien » alimenté directement sur le réseau électrique. Plus particulièrement, il doit être monté en pont en H afin de pouvoir recevoir une tension positive ou négative selon l'état des 4 interrupteurs. Les interrupteurs sont contrôlés par 4 pin de l'ATMEGA

e) des capteurs de fin de montée / fin de descente nous renseigne sur la position final du store, la valeur renvoyée par ces capteurs est une valeur numérique

f) dans le cas d'un obstacle ou d'une perturbation quelconque qui bloquerait le moteur, nous devons être en mesure d'arrêter le moteur afin d'éviter d'endommager le dispositif

g) lorsque le mode automatique est activé, les touches de monté/descente sont désactivées

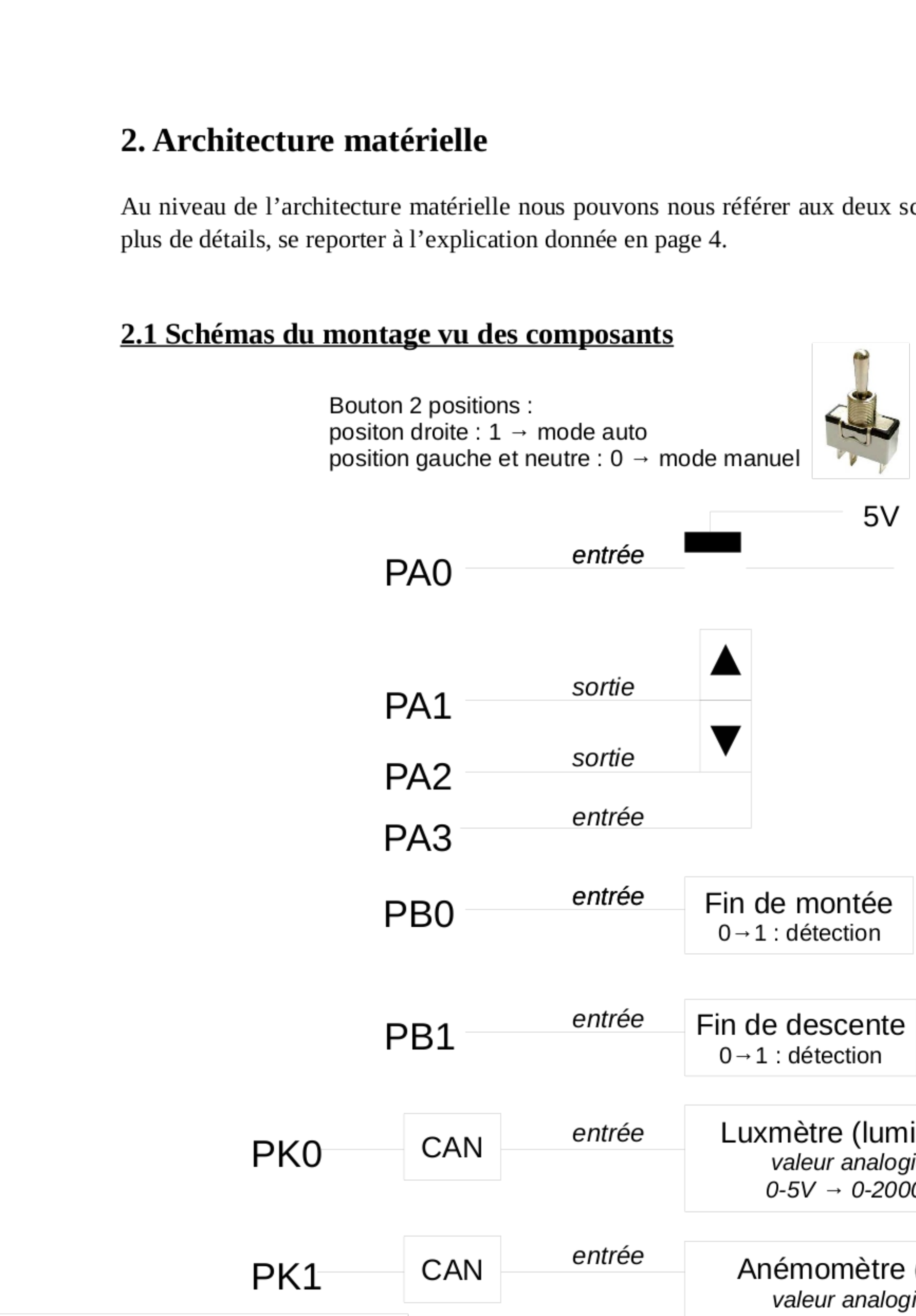
h) en mode automatique, le sore s'ouvre et se ferme en fonction de la lumière et du vent
les valeurs de vents et de lumière sont renseignées à l'ATMEGA par valeur analogique à l'aide d'un luxmètre et d'un anémomètre

<u>important :</u> lorsque le vent est trop fort le store doit obligatoirement se relever quelque soit la luminosité

# 2. Architecture matérielle

Au niveau de l'architecture matérielle nous pouvons nous référer aux deux schémas suivants. Pour plus de détails, se reporter à l'explication donnée en page 4.

## 2.1 Schémas du montage vu des composants

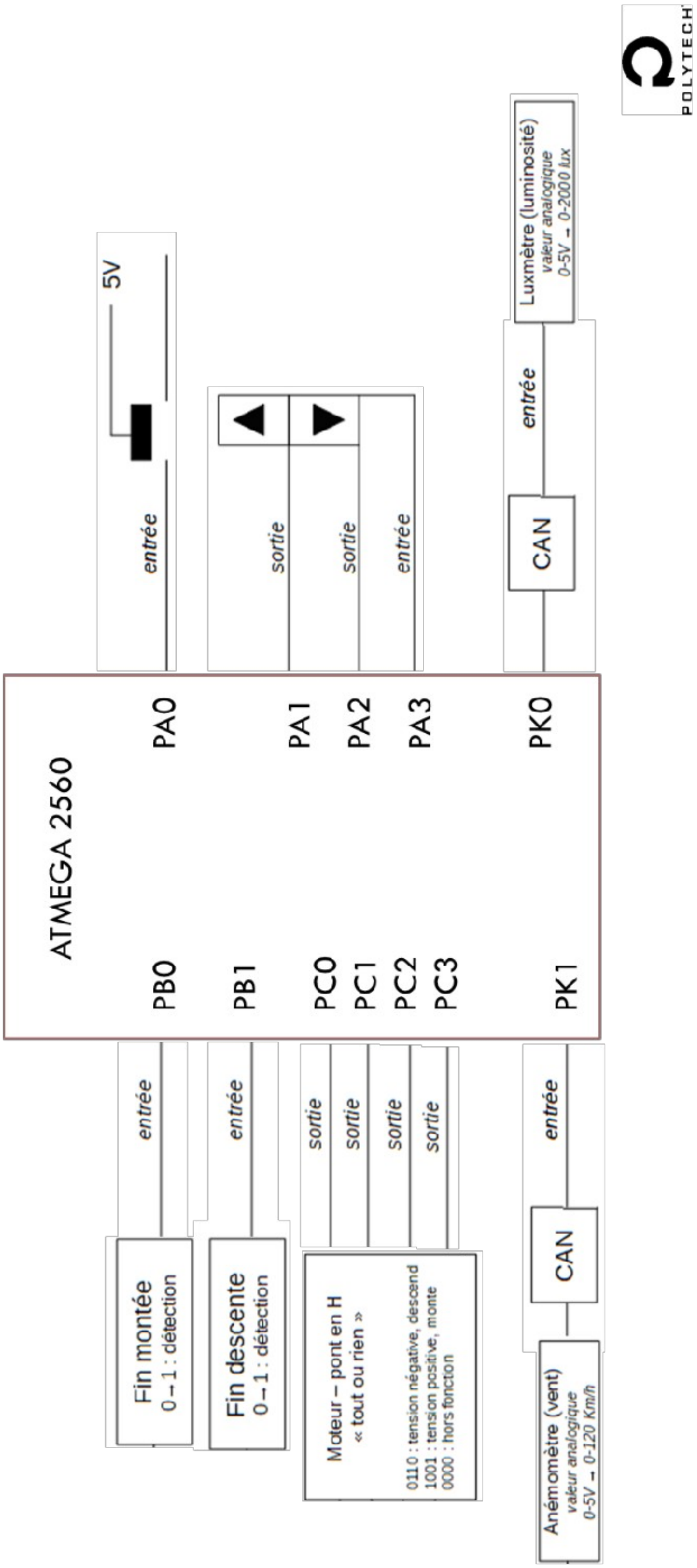






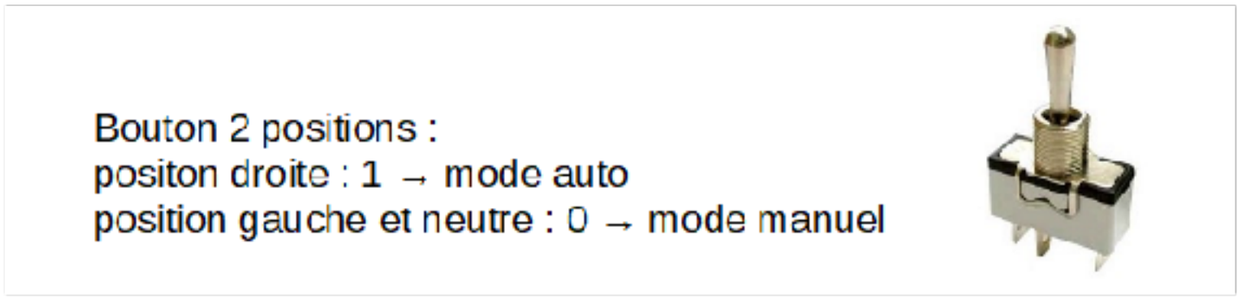
Moteur « tout ou rien » alimenté en pont en H

## 2.2 Schéma du montage vu de l'Arduino ATMEG2560



## 2. 3 Explication concernant l’architecture matérielle

Tout d’abord, pour l’interrupteur nous avons choisi un bouton poussoir qui enverra automatiquement un signal nul ou 5V aussi longtemps que la position de l’interrupteur reste verrouillé. Comme nous allons travailler par scrutation, cela nous facilitera grandement la tâche lors de la conception de l’algorithme. Nous le branchons par défaut sur le pin PA0 (en entrée).




Ensuite, nous réservons 3 pins PA1, PA2, PA3 pour le clavier 2 touches (montée/descente du mode manuel). PA1 et PA2 sont en sorties et PA3 en entrée afin de lire la valeur renvoyé par le clavier selon le code envoyé en PA1, PA2. Lorsque le mode automatique est activé, le clavier est hors-fonction (géré au niveau logiciel).

Au niveau des capteurs, nous avons 2 capteur numériques, qui passent de 0 à 1 au moment où le store est complètement remonté ou abaissé. Comme nous travaillons par scrutation, il n’était pas nécessaire de les brancher sur des broches d’interruption. Nous les branchons donc sur les pins PB0, et PB1.

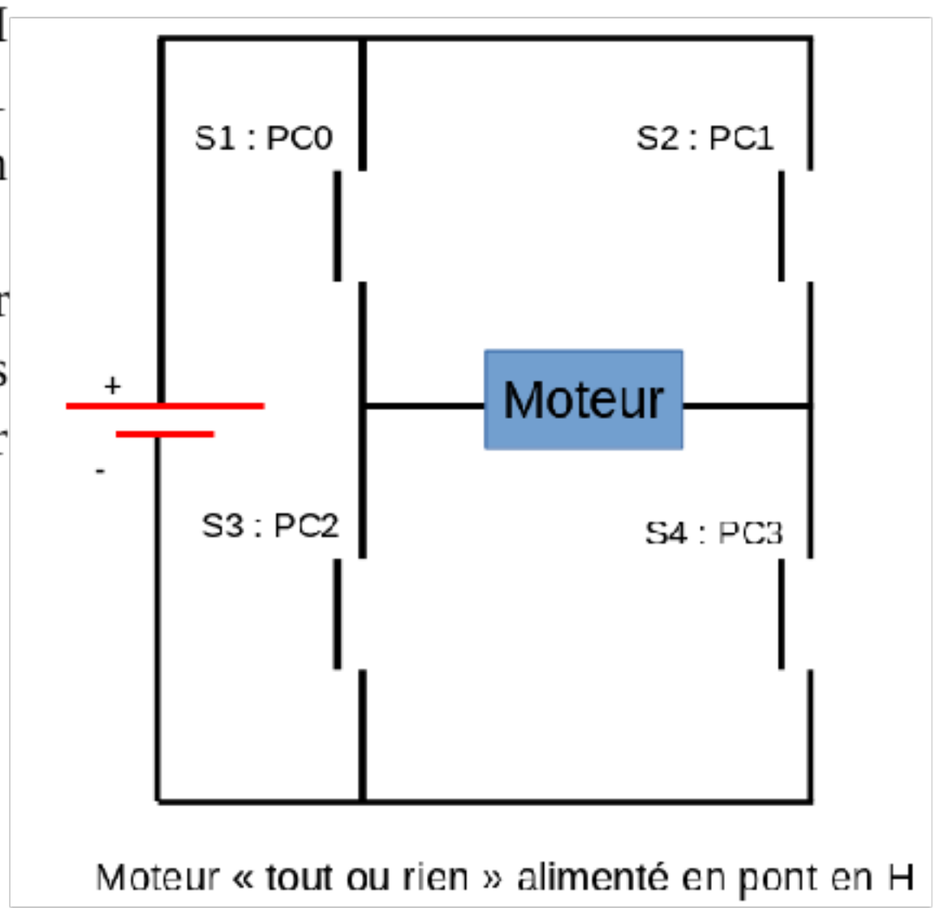
Nous avons changé de port pour plus de visibilité. En effet comme nous n’avons pas besoin de beaucoup de port pour concevoir notre architecture matérielle, nous avons préféré distinguer les ports selon chaque groupe de composant. Nous les avons regroupés selon nos propres critères :

Au niveau du port A, nous avons mis les éléments intervenant dans le choix du mode manuel/ automatique ainsi que la clavier. Pour nous ce sont des éléments qui vont ensembles, car ils fonctionnent indépendamment des capteurs.

Au niveau du port B, nous avons branchés les capteurs numériques qui captent la fin de montée/fin de descente du store.

Pour les capteurs à valeur analogique, nous avons utiliser le port K car il s’agit du port prêt à accueillir des valeurs analogique pour la conversion.

Enfin, nous avons branché les 4 interrupteurs du pont en H (pour le moteur) sur le port C. Lorsque les interrupteur S1 et S4 sont activés le moteur est alimenté par une tension positive, il tourne donc en sens positif (les stores montent). Lorsque S2 et S3 sont activés le moteur est alimenté par une tension négative, il tourne donc en sens négatif (les stores descendent). Lorsque les 4 sont désactivé, le moteur n’est pas alimenté.




Moteur « tout ou rien » alimenté en pont en H

# 3. Algorithme du logiciel

## 3.1 Source du programme en pseudo code

```
.equ DDRA = 0x01
.equ PORTA = 0x02
.equ PINA = 0x00
.equ DDRB = 0x01
.equ PINB = 0x02
.equ DDRC = 0x01
.equ PORTC = 0x02
.equ WDTCSR = 0x60

.def cpt = R20
.equ limite_vent = 255 ; limite vent à 30 km/h  ~ code binaire 255
.equ consigne = 128 ; 250 lux : seuil à partir duquel le store s'abaisse pour faire de l'ombre
; 250lux ~ code binaire 128


.org 0x0000
        jmp init
.org 0x0018
        jmp topHorloge
.org 0x0100

init :
        DDRA ←0b 0000 0110 ; PA1 et PA2 en sortie
        DDRB ←0x00 ; optionnel
        DDRC ←0x0F ; PC0 à PC3 en sortie
        SEI

main :
        if (PINA & 0x01) = 0 ; mode manuelle (utilisation d'un masque, on regarde le premier bit)

                PORTA ← 0b00001010 ; on test la 1ère touche
                if (PINA = PORTA) ; la touche « montée » est appuyée
                        call monte
                        jmp main
                else
                        PORTA ← 0b00001100 ; on test la 2ème touche
                        if (PINA = PORTA) ; la touche « descend » est appuyée
                                call descend
                                jmp main
                        else ; aucune touche ne correspond
                                jmp main

else ; mode automatique
        call lit_vent ; conversion CAN regarder R21 (low) et R22 (high)
        if (R22!=0 || R21 > limite_vent) ; danger trop de vent
                call monte
```

```
                jmp main
        else
                call lit_lumière ; conversion CAN  regarder R21 (low) et R22 (high)
                if (R22!=0 || R21 > consigne) ; trop de soleil, on abaisse le store
                        call descend
                        jmp main
                else ; pas beaucoup de soleil, nuit, etc...
                        call monte
                        jmp main

monte :
        cpt ← 0
        R19 ← 1
        WDTCSR ←0x10 ; désactive protection chien de garde
        WDTCSR ←0b0100 0110 ; chien de garde va dans topHorloge toutes les secondes
        ; R19 : cf topHorloge
        tant que  (PINB & 0x01) = 0 && R19 = 1 ; on teste PB0 (capteur fin montée)
                PORTC ← 0b00001001  ; le moteur tourne en sens positif (les stores montent)
        fin tant que
        PORTC ← 0 ; arrêt du moteur
        WDTCSR ←0x10
        WDTCSR ←0x00 ; arrêt chien de garde
        RET


descend :
        cpt ← 0
        R19 ← 1
        WDTCSR ←0x10
        WDTCSR ←0b0100 0110
        tant que  (PINB & 0x02) = 0 && R19 = 1 ; on teste PB1 (capteur fin descente)
                PORTC ← 0b00000110 ; le moteur tourne en sens négatif (les stores descendent)
        fin tant que
        PORTC ← 0
        WDTCSR ←0x10
        WDTCSR ←0x00
        RET


lit_vent : ; port ADC8 / PK0
        ADMUX ← 01000001
        ADCSRB ← X0XX1000
        ADCSRA ← 11100110
encore :
        if (ADCSRA & 0x10) = 1 ; on teste le flag ADIF : conversion terminée ou pas
                R21 ← ADCL
                R22 ← ADCH
        else
                jmp encore
        RET
```

```
lit_lumière : ; port ADC9 / PK1
        ADMUX ← 01000000
        ADCSRB ← X0XX1000
        ADCSRA ← 11100110
encore :
        if (ADCSRA & 0x10) = 1 ; on teste le flag ADIF : conversion terminée ou pas
                R21← ADCL
                R22 ← ADCH
        else
                jmp encore
        RET
```

```
topHorloge :
        cpt ← cpt +1
        if cpt=20 ; après 20 sec l'ouverture/fermeture ne s'est toujours pas arrêté : problème
                R19 ← 0 ; on arrête le « tant que  »: chien de garde et moteur s'arrête auto à la fin du
                        « tant que »
        RETI
```

## 3.2 Justification du code et éclaircissement

Notre programme est divisé en plusieurs parties : l'initialisation, le main dans lequel on retourne à chaque fin de commande du store, ainsi que les sous-programmes pour le contrôle du moteur et la conversion analogique/numérique des 2 capteurs analogiques.

Tout d'abord, notre programme fonctionne par scrutation, nous regardons constamment la valeur de PA0 pour savoir si l'on est en mode automatique ou manuel. Du fait de notre interrupteur 2 positions, on lira 1 ou 0 aussi longtemps que le mode automatique ou manuelle est enclenchée.

Dans le cas où le mode manuelle est activé, c'est-à-dire que PA0 = 0, nous envoyons le code de la touche 1 « montée du store » en sortie et nous regardons en entrée si le code correspond. Si oui, la touche 1 est activée et nous appelons le sous-programme « monte », si non, on teste la seconde touche, « descente », en envoyant son code et on regarde l'entrée. Si aucune des deux touches n'est appuyées, on retourne au main, et on regarde encore une fois si le mode manuelle est activé et ainsi de suite.

Dans le cas où le mode automatique est activé, c'est-à-dire que PA0=1, nous faisons d'abord appel à la fonction lit_vent afin de déterminer si le vent n'est pas trop fort. Si le vent est trop fort, quelque soit les autres paramètres, le store remonte pour des raisons de sécurité. Si le vent n'est pas trop élevé, on appel lit_lumière pour voir si la luminosité extérieur est supérieur à celle de la consigne, dans ce cas là, nous abaissons le store afin d'ombrager la terrasse. Si non, nous revenons au main sans intervenir sur la position courante du store.

Comme les instructions de montée et de descente de notre store sont répétées plusieurs tout au long de notre code, nous avons jugé préférable de les coder dans deux sous-programmes qu'ils nous suffiraient d'appeler lorsque l'on en a besoin .
Ils fonctionnent de la manière suivante : tout d'abord nous initialisons un compteur de temps à 0, un chien de garde qui réalise un saut dans « topHorloge » toutes les secondes et un registre à 1 qui fait office de valeur booléenne dans notre « tant que ». Ensuite, tant que notre capteur de fin de montée / fin de descente n'a pas détecté le store ou que notre registre booléen égale 1, nous envoyons une instructions via le port C aux 4 interrupteurs du moteur « tout ou rien ». Après cela, si un capteur détecte le store ou si le compteur de temps égal 20 (secondes), nous quittons le tant que. Nous retournons alors dans le main, à l'endroit où nous avions appelé le sous-programme. On quitte le tant que, le moteur s'arrête et le chien de garde est désactivé.

Pour lire les valeurs des capteur de lumière et de vitesse du vent, nous devons appelé les sous-programme lit_lumière et lit_vent. Les valeurs de ces capteurs étant analogiques, nous utilisons un convertisseur analogique/numérique sur 10 bits pour récupérer les données. Pour ce faire nous initialisons les registres afin de correspondre au bon port (PK0 → ADC8 et PK1 → ADC9). Ensuite nous avons décidé de fonctionner en free runing mode, calibré sur une tension externe de 5V, avec une fréquence d'horloge de 8MHz. Pour atteindre une fréquence comprise entre 50kHz et 200kHz, nous avons choisit de diviser la fréquence par 64. En fait, nous attendons qu'une donnée est prête à être lue à l'aide du flag situé sur le 5ème bit du registre ADCSRA. Une fois que le flag passe à 1,

nous enregistrons les bits de poids forts dans le registre R22 et les bits de poids faible dans le registre R21.

Pour comparer les valeurs lues avec les valeurs de consignes, nous devons convertir soit le code lue pour correspondre à  nos consignes (vent et luminosité), soit le code de consigne. Nous avons choisi de modifié le code de consigne. Nos valeurs étant stocké dans 10 bits, nous pouvons aller de 0 à 1023 avec les registres R22 et R21.

En faisant quelques recherches,  nous avons trouvé qu'un vent de 30km/h fait bouger les arbustes et est catégorisé en vent de force 5. Nous fixons donc notre limite à 30km/h, limite à partir de laquelle nous devons obligatoirement remonter le store en mode automatique. Ainsi pour un anémomètre couvrant une plage de 0 à 120km/h avec une consigne fixé à 30km/h, le code binaire correspondant à notre consigne est 30*1023/120 = 255.

De plus, nous avons choisi de fixé la valeur consigne du luxmètre à 250 lux, qui correspond à la luminosité d'une pièce normalement éclairé. De la même manière, le code binaire lue de 0 à 1023 couvrent une plage de 0 à 2000 lux. Ainsi le code correspondant à 250 lux est 128.

Pour comparer les valeurs de consigne aux valeurs lues, nous regardons tout d'abord si R22 (poids forts de ADC) est différents de 0. En effet nos consignes ne sont stockés que sur un seul octet, une valeur différentes de 0 sur les 9eme et 10eme bits implique forcément que la valeur lue est supérieure à notre consigne. Si non, nous comparons normalement le poids faible de ADC avec la valeur de notre consigne (comparaison octet-octet).

## 4. Limites de la solution

Tout d'abord, au niveau matérielle, nous avons une limite au niveau du moteur. En fait, lors de l'établissement du cahier des charges, nous avons choisi d'utiliser un moteur « tout ou rien », qui restreint la mobilité de notre store. En effet, ou il s'abaisse complètement, ou il s'élève entièrement, nous ne pouvons donc pas décider de monter un peu ou d'abaisser un peu le store : il n'existe que deux positions possibles. Pour ce faire, il aurait fallu utilisé un cerveau-moteur avec un capteur de position par exemple.

Ensuite, lors du mode manuel, l'utilisateur est considéré comme maître du système, c'est-à-dire que c'est à l'utilisateur d'abaisser le store si le vent est trop élevé par exemple (en mode manuel). Nous aurions pu décider d'imposer le rabattement des stores par vent violent même en mode manuel, mais dans le cahier des charges nous avons préféré imposer la volonté de l'utilisateur par dessus les valeurs des capteurs de vent et de luminosité. Cependant, afin de protéger le système lors du mouvement du store, nous avons quand même installé un chien de garde qui arrête le mouvement du store au bout de 20 secondes si le mouvement ne s'est toujours pas terminé quelque soit le mode.

Enfin, l’inconvénient du travail par scrutation est que l’on peut perdre du temps à lire une valeur. Cependant comme nous utilisons un boutons poussoir 2 états, avec un clavier basique à 2 touches, et 2 capteurs numériques, le problème ne semble pas se poser pour ces éléments là. Le problème pourrait venir de l’anémomètre et du luxmètre, qui pourrait mettre du temps à envoyer une valeur.

Dans sa globalité, notre solution demeure fiable et robuste, en mode automatique la vitesse du vent impose le mouvement du store, et un chien de garde arrête le mouvement du store au bout de 20 secondes s’il ne s’est toujours pas arrêté à cause d’un obstacle par exemple.

## Conclusion

Pour conclure, au cours de ce tutorat, nous avons eu l’occasion d’utiliser nos compétences dans l’élaboration d’une application concrète, à savoir la commande d’un store. Nous avons élaboré le cahier des charges, en précisant certains points qui n’étaient très précis, où auquel on n’y avait pas pensé. Ensuite, nous avons proposé une architecture matérielle et logicielle en posant des hypothèses à la réalisation de notre application. Dans une application future, on pourra éventuellement améliorer la solution en se basant sur les limites énoncées lors de ce compte-rendu. Pour finir, dans ce récapitulatif, nous vous avons proposé une application qui fonctionne tant au niveau matériel que logiciel avec un ATMEGA2560.