MAKERS FOR LIFE

MakAir

Respirateur PPC

BOUTRY Loan

Alternant sur le projet MakAir

Ce document fait partie d'une série de livrables concernant la conception et le prototypage d'un respirateur à pression positive continue pour diminuer l'apnée du sommeil d'un patient

Ce livrable a pour but d'expliquer les différentes versions du Respirateur PPC. Chaque version a une fonctionnalité différente (commande en vitesse, commande en débit, mode auto-piloté).

Table des matières

Table des figures

Les fonctions principales

Nous allons aborder les fonctions principales mises en place sur les différentes versions du respirateur PPC afin de comprendre la logique de programmation.

Machine à état

Une machine à état est une machine qui lit un ensemble d'entrées et passe à un état différent en fonction de ces entrées. Un état est une description de l'état d'un système en attente d'exécution d'une transition, cela peut correspondre un état de repos (machine à l'arrêt) ou un état actif (envoie d'air à un seuil défini).

La machine à état va être différente par rapport aux différentes versions du respirateur. Nous allons prendre celle du mode auto-piloté pour comprendre tous les états possibles du système :

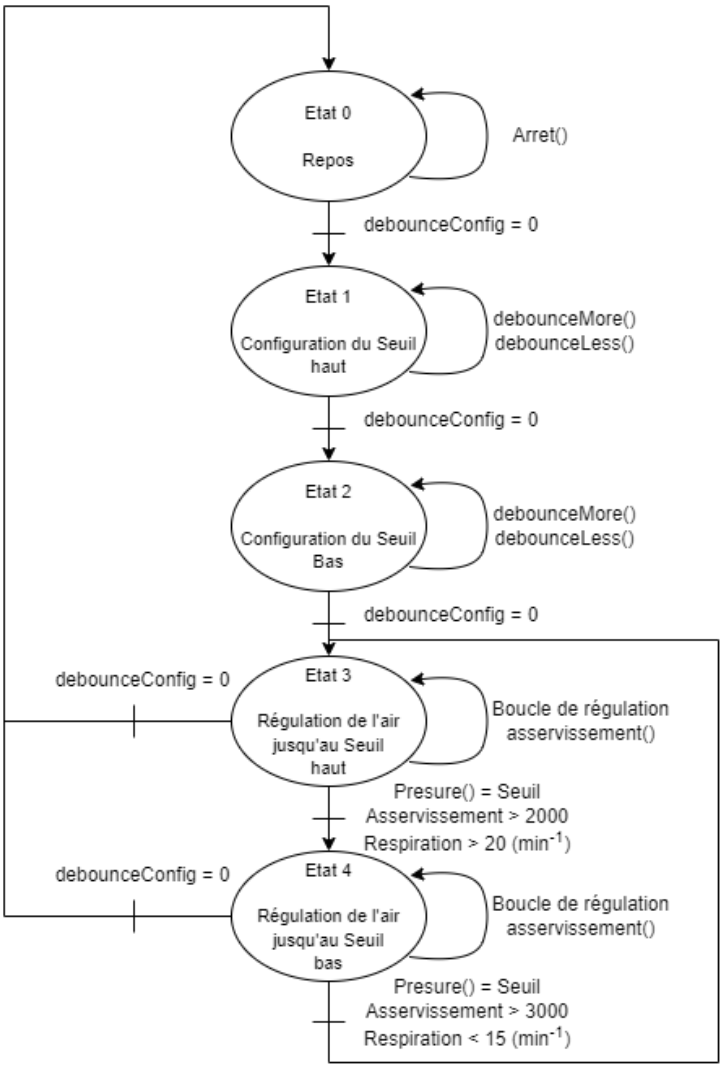

Figure 1 : Machine à état du mode auto-piloté

La machine à état reste linéaire puisque la transition se fait avec le bouton « Config » et suit toujours le même cheminement. On peut voir aussi que les états actifs (état 3 et 4) peuvent revenir à l'état repos grâce au bouton « Config » pour éteindre le système au moment de la configuration des paramètres.

Boucle de régulation

Une boucle de régulation est une fonction permettant de réguler le moteur afin d'atteindre le seuil défini par l'utilisateur. Tant que le système n'a pas atteint le seuil demandé, il va augmenter la vitesse du moteur si les mesures sont inférieures au seuil ou va diminuer la vitesse si les mesures sont supérieures au seuil. Nous ajoutons un taux d'erreur possible car le système n'est pas idéal et ne pourra donc pas atteindre exactement la valeur du seuil. Par exemple, nous pouvons mettre une erreur possible de 0,2 mbar sur la pression. La fonction « delayMicroseconds » est ajoutée dans la boucle afin d'éviter une accélération trop importante du moteur et donc de rendre le système instable. Pour chaque boucle de régulation, on jauge 3 paramètres importants : la stabilité, la précision et la rapidité.

Voici un exemple d'une boucle de régulation sur la pression de l'air :

```
while ((pressionLimite - intervalle) >= getPres() || (pressionLimite +
intervalle) <= getPres()) { //Boucle de régulation modifiant la vitesse
jusqu'à atteindre le seuil configuré
    if (pressionLimite > getPres()) vitesse++;
    if (pressionLimite < getPres()) vitesse--;
    setVitesse(vitesse); //Mise en place de la nouvelle vitesse
    delayMicroseconds(acceleration);
}
```

Figure 2 : Code d'une boucle de régulation

Asservissement Moteur

L'asservissement d'un moteur consiste à vérifier si le moteur répond au seuil demandée, si ce n'est pas le cas, l'asservissement corrige sa position, sa vitesse ou son accélération. Dans notre cas l'asservissement va corriger la vitesse du moteur en revenant dans la boucle de régulation tant que les paramètres (pression et débit d'air) ne sont pas atteintes. Dès que l'asservissement valide la vitesse du moteur, celui-ci gardera cette vitesse fixe jusqu'au changement d'état.

Voici un exemple d'un asservissement du moteur :

```
while ((pressionLimite - intervalle) >= getPres() || (pressionLimite +
intervalle) <= getPres()) { //Boucle modifiant la vitesse jusqu'à atteindre
le seuil configuré
      if ((pressionLimite - intervalle) > getPres()) vitesse++;
      if ((pressionLimite + intervalle) < getPres()) vitesse--;
      setVitesse(vitesse);
      delayMicroseconds(acceleration);
    }
    asservissement++;
    if (asservissement >= 2000) { //dès que le seuil est bien atteint, la
configuration est finie
      vitessePilot = vitesse;
      configuration = 4;
      asservissement = 0;
    }
```

Figure 3 : Code d'un asservissement de moteur

Les différentes versions du Respirateur PPC

Avant de commencer, il est important de préciser que certaines fonctions utilisées dans les différentes versions du respirateur PPC sont retrouvables sur le Github du projet :

https://github.com/makers-for-life/makair-cpap/tree/master/Firmware-CPAP

Version 1 : Commande en vitesse

Cette première version va permettre de commander le respirateur en vitesse. C'est un paramètre assez simple à configurer puisqu'elle commande que le blower. Cela mène à un programme assez ressemblant au code d'intégration du projet :

```
void cmdVitesse() {
  Speed = map(vitesse, 180, 1800, 0, 100); //Valeur de la vitesse en % (0 à 100%)
  setAffichagex2("Vitesse: ", Speed, " %"); //Fonction d'affichage de la vitesse sur la 3ème
ligne du LCD
  configuration = debounceConfig(PA5, 50, configuration, 1); //Fonction de configuration pour
changer d'état
  if (configuration == 1) {
    arret();
    vitesse = debounceMore(PB5, 50, vitesse, 16, 1800); //Fonction de configuration pour
augmenter la valeur
    vitesse = debounceLess(PA8, 50, vitesse, 16, 180); //Fonction de configuration pour
diminuer la valeur
    configuration = debounceConfig(PA5, 50, configuration, 2);
    Serial.println(vitesse);
  }
  else if (configuration == 2) {
    setVitesse(vitesse); //Mise en place de la nouvelle vitesse
    configuration = 0;
  }
}
```

Figure 4 : Code de la commande en vitesse

On commande la vitesse en % de vitesse pour avoir des valeurs plus accessibles pour un utilisateur. On rajoute la valeur configurée sur l'afficheur pour communiquer visuellement à l'utilisateur. On retrouve 2 états : l'état en fonction et l'état en configuration, cette dernière permet d'arrêter le blower et de configurer la vitesse avec l'IHM. Après la configuration, le blower change de vitesse en fonction de la nouvelle valeur de vitesse. On a donc notre système qui est commandé en vitesse.

Figure 5 : Photos du respirateur PPC en commande de vitesse (avec poumon artificiel)

Version 2 : Commande en débit d'air

Pour cette deuxième version, on va commander le débit d'air envoyé par le blower. Cela va être un paramètre un peu plus complexe puisqu'on a le débit et la vitesse à prendre en compte. On ajoute une boucle de régulation afin de gérer le blower en fonction du débit d'air souhaité :

```
void cmdAirflow(int minimun, int maximum, int acceleration) {
  setAffichagex2("Seuil: ", seuil, " SLPM"); //Fonction d'affichage de la vitesse sur la 3ème
ligne du LCD
  configuration = debounceConfig(PA5, 50, configuration, 1); //Fonction de configuration pour
changer d'état
  if (configuration == 1) {
    arret();
    vitesse = 0;
    seuil = debounceMore(PB5, 50, seuil, 1, maximum); //Fonction de configuration pour
augmenter la valeur
    seuil = debounceLess(PA8, 50, seuil, 1, minimun); //Fonction de configuration pour
diminuer la valeur
    configuration = debounceConfig(PA5, 50, configuration, 2);
  }
  else if (configuration == 2) {
    while (seuil != getAirflow()) { //Boucle de régulation modifiant la vitesse jusqu'à
atteindre le seuil configuré
      if (seuil > getAirflow()) vitesse++;
      if (seuil < getAirflow()) vitesse--;
      setVitesse(vitesse); //Mise en place de la nouvelle vitesse
      delayMicroseconds(acceleration);
    }
    asservissement++;
    if (asservissement >= 200) { //dès que le seuil est bien atteint, la configuration est
finie
      configuration = 0;
      asservissement = 0;
    }
  }
}
```

Figure 6 : Code de la commande en débit d'air

On retrouve le même schéma que la commande en vitesse sauf qu'on y ajoute une boucle de régulation. On mesure le débit toutes les 200 ms afin de corriger la vitesse actuelle du blower pour atteindre le seuil configuré par l'utilisateur. Une condition d'asservissement s'ajoute pour changer d'état dès que la vitesse du blower est correctement régulé.

Figure 7 : Photos du respirateur PPC en commande de débit d'air (sans poumon artificiel)

Version 3 : Commande en pression

Pour cette troisième version, on va commander la pression de l'air envoyé par le blower. Cela va être un paramètre un peu plus complexe puisqu'on a la pression et la vitesse à prendre en compte. On ajoute une boucle de régulation afin de gérer le blower en fonction de la pression souhaitée :

```
void cmdPressure(double minimum, double maximum, int acceleration, double intervalle) {
  setAffichagex2("Limite: ", pressionLimite, " mbar"); //Fonction d'affichage de la vitesse
sur la 3ème ligne du LCD
  configuration = debounceConfig(PA5, 50, configuration, 1); //Fonction de configuration pour
changer d'état
  if (configuration == 1) {
    arret();
    vitesse = 0;
    pressionLimite = debounceMore(PB5, 50, pressionLimite, 0.1 , maximum); //Fonction de
configuration pour augmenter la valeur
    pressionLimite = debounceLess(PA8, 50, pressionLimite, 0.1, minimum); //Fonction de
configuration pour diminuer la valeur
    configuration = debounceConfig(PA5, 50, configuration, 2);
  }
  else if (configuration == 2) {
    while ((pressionLimite - intervalle) >= getPres() || (pressionLimite + intervalle) <=
getPres()) { //Boucle de régulation modifiant la vitesse jusqu'à atteindre le seuil configuré
      if (pressionLimite > getPres()) vitesse++;
      if (pressionLimite < getPres()) vitesse--;
      setVitesse(vitesse); //Mise en place de la nouvelle vitesse
      delayMicroseconds(acceleration);
    }
    asservissement++;
    if (asservissement >= 50) { //dès que le seuil est bien atteint, la configuration est
finie
      configuration = 0;
      asservissement = 0;
    }
  }
}
```

Figure 8 : Code de la commande en pression

On retrouve le même programme que la commande du débit d'air sauf que cette fois on commande la pression de l'air envoyé.

Figure 9 : Photos du respirateur PPC en commande de pression (avec poumon artificiel)

Version finale : Commande auto-pilotée

Pour cette version finale, on va s'approcher d'une commande autopilotée d'un respirateur PPC. Cette fonction se commande en pression et varie en fonction de la respiration du patient. On va donc avoir plusieurs boucles de régulation et on va introduire le débit d'air qui permettra de capter la respiration du patient. On va aussi avoir 2 états qui s'ajoutent aux précédentes : l'état de pression basse et l'état de pression haute. Le code du mode auto-piloté est retrouvable en Annexe :

Etat Repos

Etat 1

Etat 2

Etat 3

Etat 4 (Respiration)

Etat 4 (Seuil bas)

Figure 10 : Photos du respirateur PPC en mode auto-piloté (avec poumon artificiel)

Annexe

Fichier .h

```
#include <Arduino.h>

#include "airflowSensor.h"
#include "blower.h"
#include "IHM.h"
#include "pressureSensor.h"

void PPC_init(int vitesseInit);
void cmdVitesse();
void cmdAirflow(int minimun, int maximum, int acceleration);
void cmdPressure(double minimum, double maximum, int acceleration, double
intervalle);
void autoPilot(double minimum, double maximum, int acceleration, double
intervalle, int frequence);
```

Fichier .cpp

```
#include <Arduino.h>

#include "airflowSensor.h"
#include "blower.h"
#include "IHM.h"
#include "pressureSensor.h"
#include "PPC.h"

int configuration = 0;
int vitesse = 0;
int vitessePilot = 0;
int Speed = map(vitesse, 180, 1800, 0, 100);
int seuil = 0;
double pressionLimite = 0.00;
int asservissement = 0;
int respiration = 0;
double traitement = 0;
int pilote = 1;

void PPC_init(int vitesseInit) {
  vitesse = map(vitesseInit, 0, 100, 180, 1800);
  setVitesse(vitesse);
}

void cmdVitesse() {
  Speed = map(vitesse, 180, 1800, 0, 100); //Valeur de la vitesse en % (0 à
100%)
  setAffichagex2("Vitesse: ", Speed, " %"); //Fonction de l'affichage de la
vitesse sur la 3ème ligne du LCD
  configuration = debounceConfig(PA5, 50, configuration, 1); //Fonction de
configuration pour changer d'état
  if (configuration == 1) {
    arret();
    vitesse = debounceMore(PB5, 50, vitesse, 16, 1800); //Fonction de
configuration pour augmenter la valeur
    vitesse = debounceLess(PA8, 50, vitesse, 16, 180); //Fonction de
configuration pour diminuer la valeur
    configuration = debounceConfig(PA5, 50, configuration, 2);
    Serial.println(vitesse);
  }
  else if (configuration == 2) {
```

```
    setVitesse(vitesse); //Mise en place de la nouvelle vitesse
    configuration = 0;
  }
}

void cmdAirflow(int minimun, int maximum, int acceleration) {
  setAffichagex2("Seuil: ", seuil, " SLPM"); //Fonction de l'affichage de
la vitesse sur la 3ème ligne du LCD
  configuration = debounceConfig(PA5, 50, configuration, 1); //Fonction de
configuration pour changer d'état
  if (configuration == 1) {
    arret();
    vitesse = 0;
    seuil = debounceMore(PB5, 50, seuil, 1, maximum); //Fonction de
configuration pour augmenter la valeur
    seuil = debounceLess(PA8, 50, seuil, 1, minimun); //Fonction de
configuration pour diminuer la valeur
    configuration = debounceConfig(PA5, 50, configuration, 2);
  }
  else if (configuration == 2) {
    while (seuil != getAirflow()) { //Boucle de régulation modifiant la
vitesse jusqu'à atteindre le seuil configuré
      if (seuil > getAirflow()) vitesse++;
      if (seuil < getAirflow()) vitesse--;
      setVitesse(vitesse); //Mise en place de la nouvelle vitesse
      delayMicroseconds(acceleration);
    }
    asservissement++;
    if (asservissement >= 200) { //dès que le seuil est bien atteint, la
configuration est finie
      configuration = 0;
      asservissement = 0;
    }
  }
}

void cmdPressure(double minimum, double maximum, int acceleration, double
intervalle) {
  setAffichagex2("Limite: ", pressionLimite, " mbar"); //Fonction de
l'affichage de la vitesse sur la 3ème ligne du LCD
  configuration = debounceConfig(PA5, 50, configuration, 1); //Fonction de
configuration pour changer d'état
  if (configuration == 1) {
    arret();
    vitesse = 0;
    pressionLimite = debounceMore(PB5, 50, pressionLimite, 0.1 , maximum);
//Fonction de configuration pour augmenter la valeur
    pressionLimite = debounceLess(PA8, 50, pressionLimite, 0.1, minimum);
//Fonction de configuration pour diminuer la valeur
    configuration = debounceConfig(PA5, 50, configuration, 2);
  }
  else if (configuration == 2) {
    while ((pressionLimite - intervalle) >= getPres() || (pressionLimite +
intervalle) <= getPres()) { //Boucle de régulation modifiant la vitesse
jusqu'à atteindre le seuil configuré
      if (pressionLimite > getPres()) vitesse++;
      if (pressionLimite < getPres()) vitesse--;
      setVitesse(vitesse); //Mise en place de la nouvelle vitesse
      delayMicroseconds(acceleration);
    }
    asservissement++;
```

```
    if (asservissement >= 50) { //dès que le seuil est bien atteint, la
configuration est finie
      configuration = 0;
      asservissement = 0;
    }
  }
}

void autoPilot(double minimum, double maximum, int acceleration, double
intervalle, int frequence) {
  // commencer en appuyant sur le bouton, ensuite configurer la moyenne de
la pression qui va sortir et appuyer sur le bouton, ensuite autopiloter la
pression et le débit envoyé et enfin pouvoir appuyer sur le bouton pour
reconfigurer
  Serial.println(configuration);

  if (configuration == 0) {
    arret();
    configuration = debounceConfig(PIN_BUTTON_CONFIG, 50, configuration,
1);
  }

  //  if (configuration > 0) {
  //    setAffichagex2("Seuil : ", pressionLimite, " mbar");
  //    setAffichagex3("Pilot : ", traitement, " mbar");
  //  }

  if (configuration == 1) {
    setAffichagex2("Seuil: ", pressionLimite, " mbar");
    pressionLimite = debounceMore(PIN_BUTTON_MORE, 50, pressionLimite, 0.5
, maximum);
    pressionLimite = debounceLess(PIN_BUTTON_LESS, 50, pressionLimite, 0.5,
minimum);
    configuration = debounceConfig(PIN_BUTTON_CONFIG, 50, configuration,
2);
  }

  if (configuration == 2) {
    setAffichagex3("Pilot: ", traitement, " mbar");
    traitement = debounceMore(PIN_BUTTON_MORE, 50, traitement, 0.5 ,
pressionLimite);
    traitement = debounceLess(PIN_BUTTON_LESS, 50, traitement, 0.5,
minimum);
    configuration = debounceConfig(PIN_BUTTON_CONFIG, 50, configuration,
3);
  }

  else if (configuration == 3) {
    while ((pressionLimite - intervalle) >= getPres() || (pressionLimite +
intervalle) <= getPres()) { //Boucle modifiant la vitesse jusqu'à atteindre
le seuil configuré
      if ((pressionLimite - intervalle) > getPres()) vitesse++;
      if ((pressionLimite + intervalle) < getPres()) vitesse--;
      setVitesse(vitesse);
      delayMicroseconds(acceleration);
    }
    asservissement++;
    if (asservissement >= 2000) { //dès que le seuil est bien atteint, la
configuration est finie
      vitessePilot = vitesse;
      configuration = 4;
```

```
      seuil = getAirflow();
      asservissement = 0;
    }
  }
  else if (configuration == 4) {
    configuration = debounceConfig(PIN_BUTTON_CONFIG, 50, configuration,
5);
    if (seuil - 5 >= getAirflow()) {
      respiration++;
      delay(100);
    }
    if (respiration >= frequence && pilote == 1) {
      while ((traitement - intervalle) >= getPres() || (traitement +
intervalle) <= getPres()) { //Boucle modifiant la vitesse jusqu'à atteindre
le seuil configuré
        if ((traitement - intervalle) > getPres()) vitessePilot++;
        if ((traitement + intervalle) < getPres()) vitessePilot--;
        setVitesse(vitessePilot);
        delayMicroseconds(acceleration);
      }
      asservissement++;
    }

    if (asservissement >= 2000 && pilote == 1) { //dès que le seuil est
bien atteint, la configuration est finie
      asservissement = 0;
      seuil = getAirflow();
      respiration = 0;
      pilote = 0;
    }

    if (respiration >= frequence && pilote == 0) {
      while ((pressionLimite - intervalle) >= getPres() || (pressionLimite
+ intervalle) <= getPres()) { //Boucle modifiant la vitesse jusqu'à
atteindre le seuil configuré
        if ((pressionLimite - intervalle) > getPres()) vitessePilot++;
        if ((pressionLimite + intervalle) < getPres()) vitessePilot--;
        setVitesse(vitessePilot);
        delayMicroseconds(acceleration);
      }
      asservissement++;
    }

    if (asservissement >= 3000 && pilote == 0) { //dès que le seuil est
bien atteint, la configuration est finie
      asservissement = 0;
      seuil = getAirflow();
      respiration = 0;
      pilote = 1;
    }
  }
  else if (configuration == 5) {
    arret();
    asservissement = 0;
    respiration = 0;
    configuration = 1;
  }
}
```