HELHa

Haute École Louvain
en Hainaut

Technique

RAPPORT

PROJET µS B1 Q2

	Projet Réveil

NOM : Canavaggio-Diana Manon

Année académique 2020-2021

1ère Info

Un projet qui n'est pas terminé n'est pas forcément raté, il faut montrer ce qui a été fait, ce qui tourne même partiellement. On ne fait pas tout en une fois. Il y a peut-être quatre choses qui fonctionnent individuellement mais qui ne tournent pas toutes ensemble.

Le montage final n'est pas terminé faute de matériel, mais le moteur tourne parce que … puis quand on fait … il s'arrête et ….. est alors affiché. OK

Votre capteur ne fonctionne pas, simulez le avec un simple bouton poussoir si cela peut faire fonctionner le reste.

Le projet doit contenir absolument ; SUR CE DOCUMENT ICI

- Une capture de données, d'informations, …..
- Un affichage de résultats (LCD, OLED, Afficheurs 7 segments, matrice, …
- Une partie de puissance, (Moteur p-à-p, résistance chauffante, moteur CC, ……
- ……

1. Motivation du choix de mon projet :

Projet initial :

Pour mon projet, je voulais faire quelque chose qui me serait utile. Alors, comme j'ai énormément de mal à me réveiller le matin, malgré l'enchaînement de réveils plus insupportables les uns que les autres, j'ai pensé à créer le réveil ultime, équipé de cloches, qui joueront une mélodie que l'on pourra sélectionner dans un menu, et qui ne se désactive seulement si on réussit un mini-jeu.

Le projet me semble ambitieux, dans un premier temps, je vais donc me focaliser sur la partie avec les cloches, pour ensuite ajouter progressivement la partie avec le réveil puis avec le mini-jeu si j'ai encore le temps.

En effet, le projet était trop ambitieux ; Je n'ai donc fait qu'un réveil « classique », affichant l'heure et la date, tous les deux réglables, et possédant une fonction alarme, réglable elle aussi.

2. Explication de mon projet.

a. Fonctionnement attendu du projet

Pour faire fonctionner ce réveil, il faut le brancher au secteur. Une fois cela fait, il affichera « «Initialisation… » Et jouera une musique. Le but principal de ceci est de charger le condensateur de la carte PCA8596 pour que les servos aient assez de puissance pour jouer la musique de l'alarme quand elle sonnera. En effet, on notera lors du premier branchement du réveil que toutes les notes ne se jouent pas correctement, mais les fois suivantes, elles sont parfaitement jouées. Une fois la mélodie terminée, Il affichera une date et une heure, dont les valeurs de base sont 01/01/2010 et 00h00m00sec.

En appuyant sur le bouton « Ok » on accède à un menu. Si l'on continue d'appuyer sur ce bouton, on peut voir ses différentes options, qui sont :

- Set time (Régler la date et l'heure)
- Set alarm (Régler l'alarme)
- Activate/deactivate alarm (Si l'alarme est activée, seule l'option Deactivate alarm sera présente. Si elle ne l'est pas, seule l'option Activate alarm sera affichée.) Si on Selectionne « Activate Alarm » sans l'avoir réglée au préalable via le menu « Set Alarm », une alarme par défaut s'activera pour 00h00m00s.
  Si l'on Set une alarme pour 10h30, et qu'on la désactive puis la réactive via ce menu, elle sera toujours réglée sur 10h30 pour en changer la valeur, il faut passer par l'option Setup Alarm.
- Return (Pour retourner à l'affichage de base sans effectuer d'action)

Si l'on souhaite donc régler l'horloge et la date du jour, on entre dans le menu « Set time » en appuyant sur le bouton « Ok ». L'affichage change, et l'affichage nous propose de régler La date en premier, en commençant par l'année, le mois et le jour.

On règle la valeur voulue en appuyant sur le bouton « + » jusqu'à atteindre la valeur voulue, puis appuyer sur « Ok » pour valider la valeur sélectionnée. Evidemment, Si par exemple on voulait régler le mois à 5, mais que nous avons déjà dépassé cette valeur, il suffit de continuer à appuyer sur le bouton « + », puisque la valeur repassera à 0 après avoir atteint le 12 (Pour les jours, la valeur max est réglée sur 31 ou 30 selon le mois sélectionné, et pour les années la valeur max est réglée sur 2040).

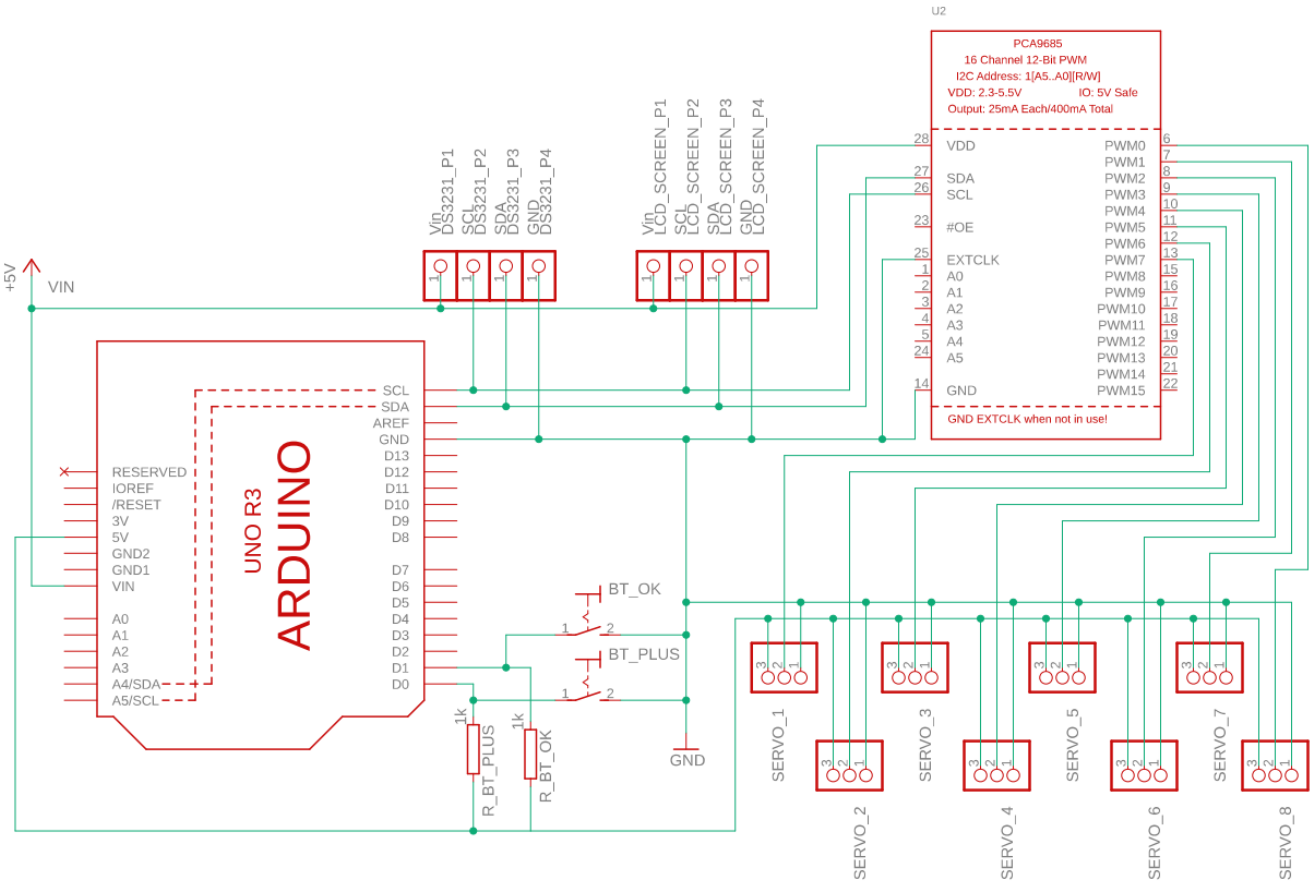
Puis on nous propose de régler l'heure, en commençant par les heures, minutes, puis secondes. Le système est le même que pour la date : Bouton « + » pour faire défiler les valeurs, boutons « Ok » pour la sélectionner et passer à la suivante. Une fois que tout est réglé, l'affichage revient à l'affichage de base, avec les valeurs que l'on vient de lui passer.

Pour régler une alarme, il faut de nouveau appuyer sur le bouton « Ok » pour ouvrir le menu, naviguer dans celui-ci grâce au bouton « + » et sélectionner l'option « Set alarm » Le principe est le même que pour régler l'heure. Une fois l'alarme réglée, on revient à l'affichage de base. On voit qu'une ligne en plus est affichée, montrant à quelle heure l'alarme va sonner.

Une fois que l'heure de la journée est égale à l'heure de l'alarme, les moteurs s'activent, et viennent jouer une mélodie pre-enregistrée, en appuyant sur les cloches. L'écran affiche alors « C'est l'heure ! »

b. Schématisation du model pratique final (dessins, sources internet, dessin personnel, ….)
……Vous êtes en informatique alors les trucs à main levée. NON !!!!!

Voici un schéma de ma conception, réalisé sur Eagle afin de synthétiser le montage au mieux. (à noter que dans le montage final, il s'agit d'un arduino Mega, car mon arduino Uno a décidé de quitter ses fonctions prématurément.)



c. Matériel nécessaire à la réalisation. (Moteur p-à-p, controler (UK), capteurs, pièces 3D, …).

Electronique	Autres
• Arduino Mega • Ecran LCD 20 x4 • Module d'horloge DS3231 • Micro-servo PG90 x8 • PCA9685 16 canaux PWN pour servo-moteurs • Resistance 1KΩ x2 • Bouton poussoir x2 • Alimentation 220V vers 5V • PCB prototypage • Pin-header	• Cloches chromatiques x8 • Morceaux de bois pour le support • Pistolet à colle • Vis • Panneau en aluminum pour les attaches • Ruban adhésif isolant

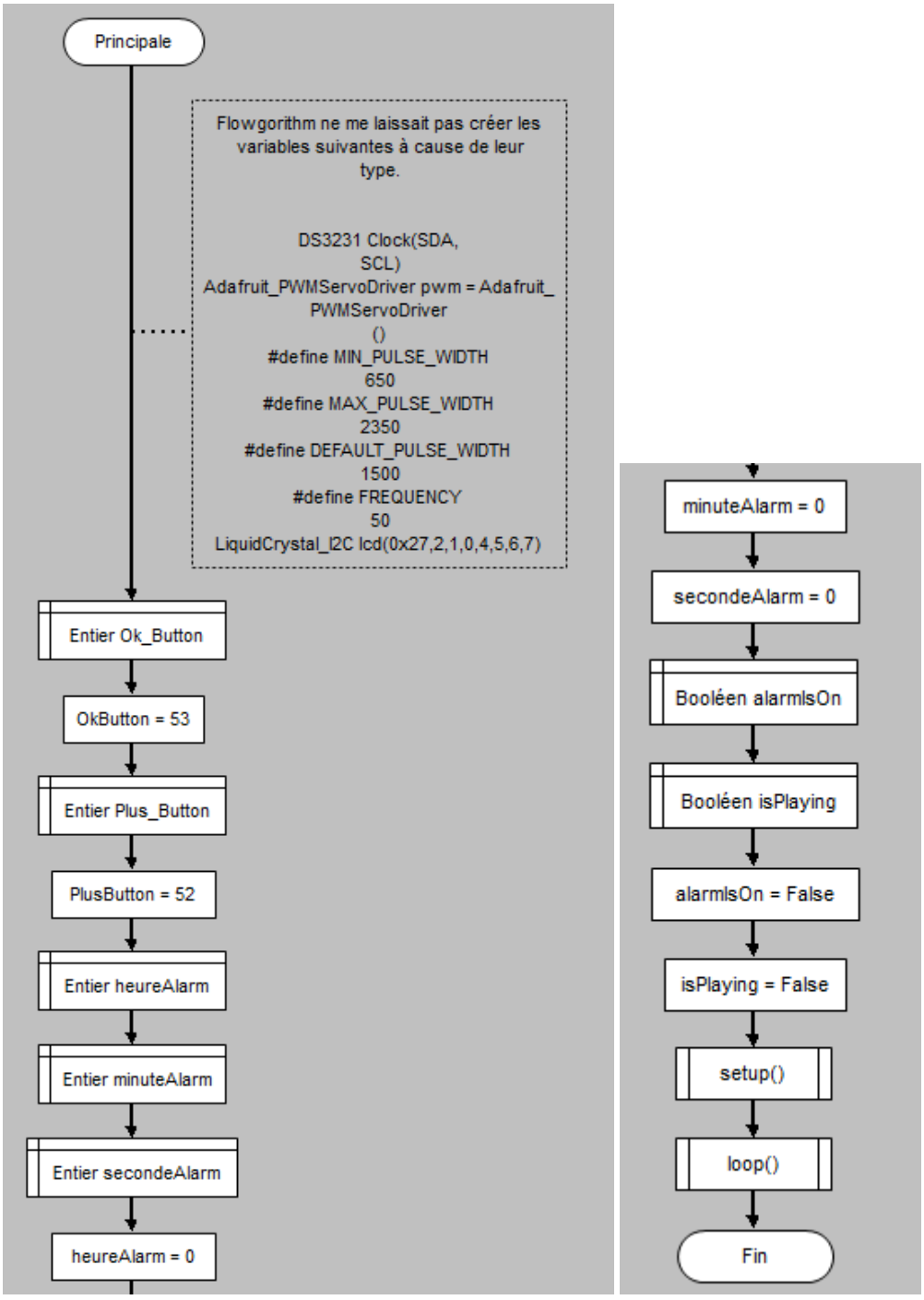
d. **Autres.**

….

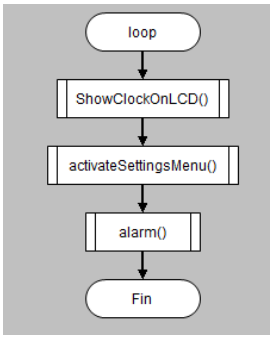
3. Réalisation effectuée.

a. **Un ordinogramme du processus. (Vous avez les outils pour cela, Voir Mr Cascio et Pluquet)**

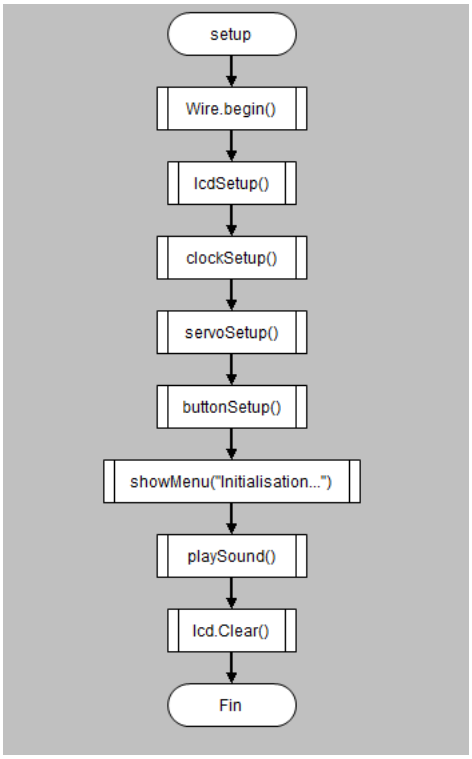
- Fonction principale



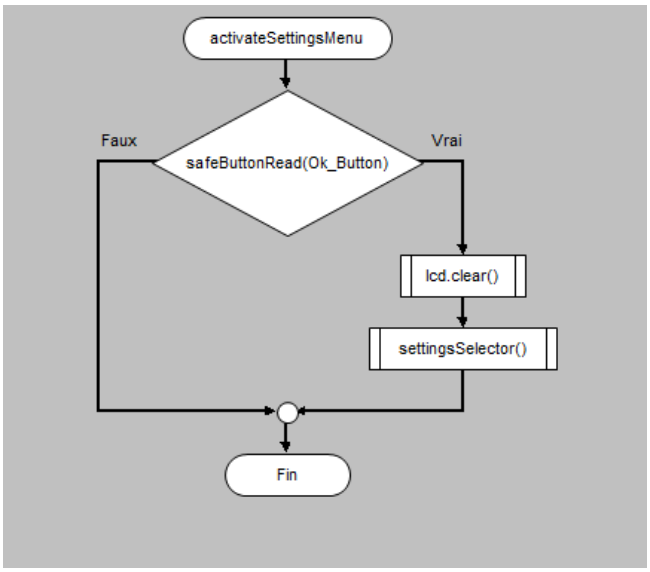
- Fonction loop



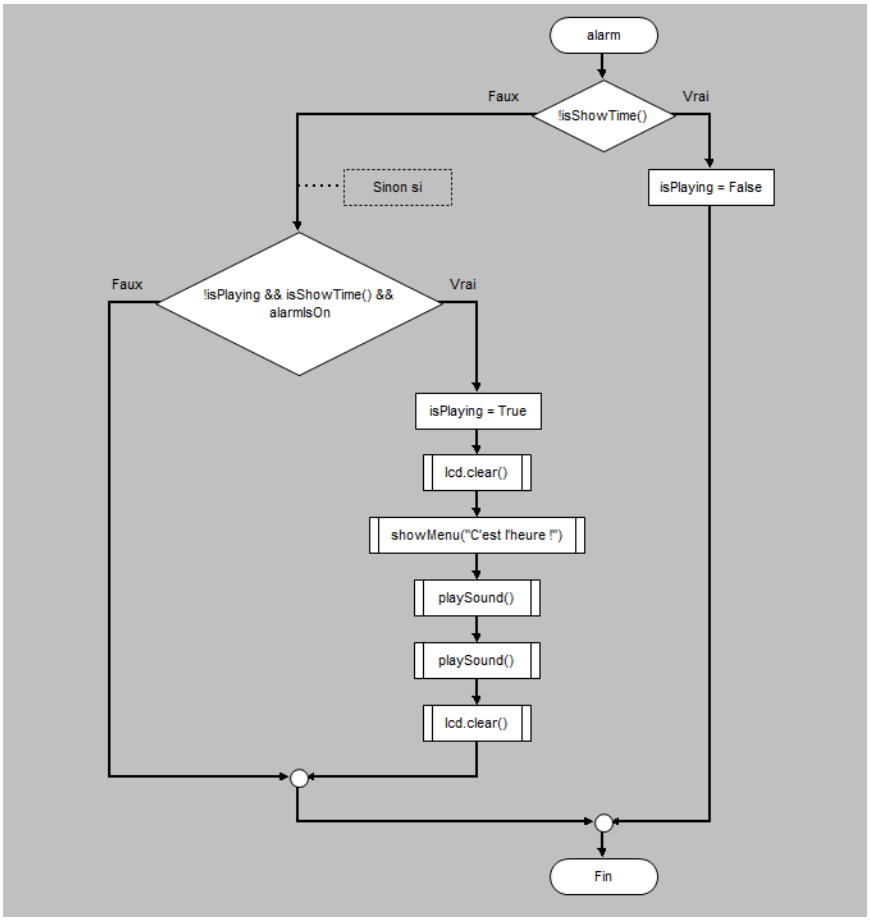
- Fonction setup



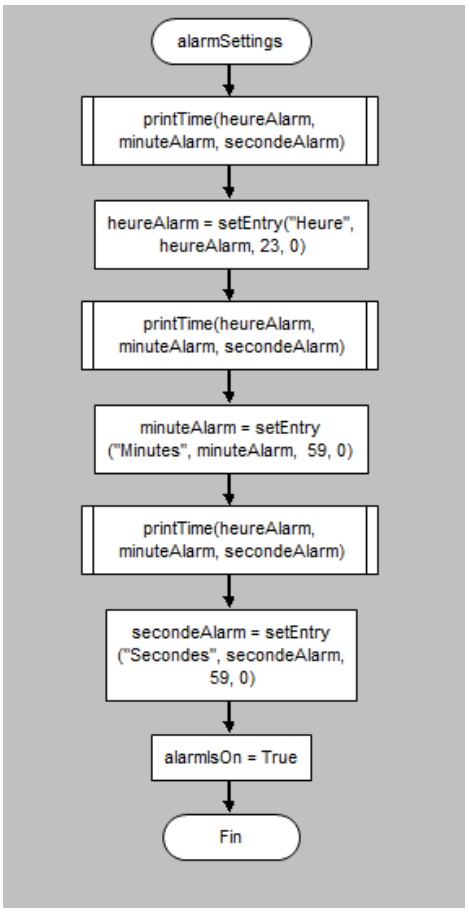
- Fonction activateSettingsMenu



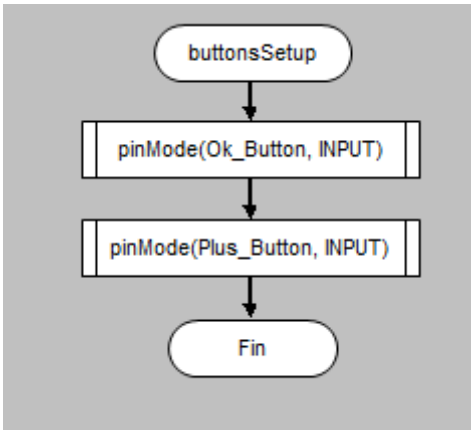
- Fonction alarm



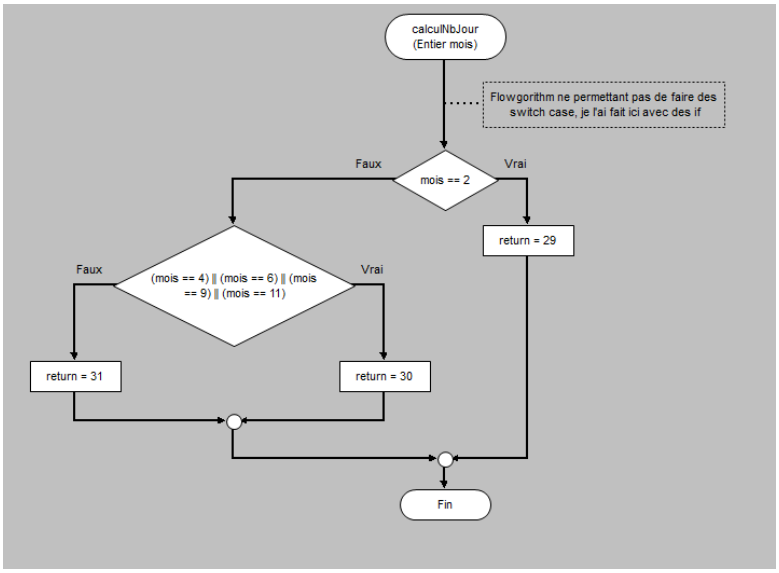
- Fonction alarmSettings



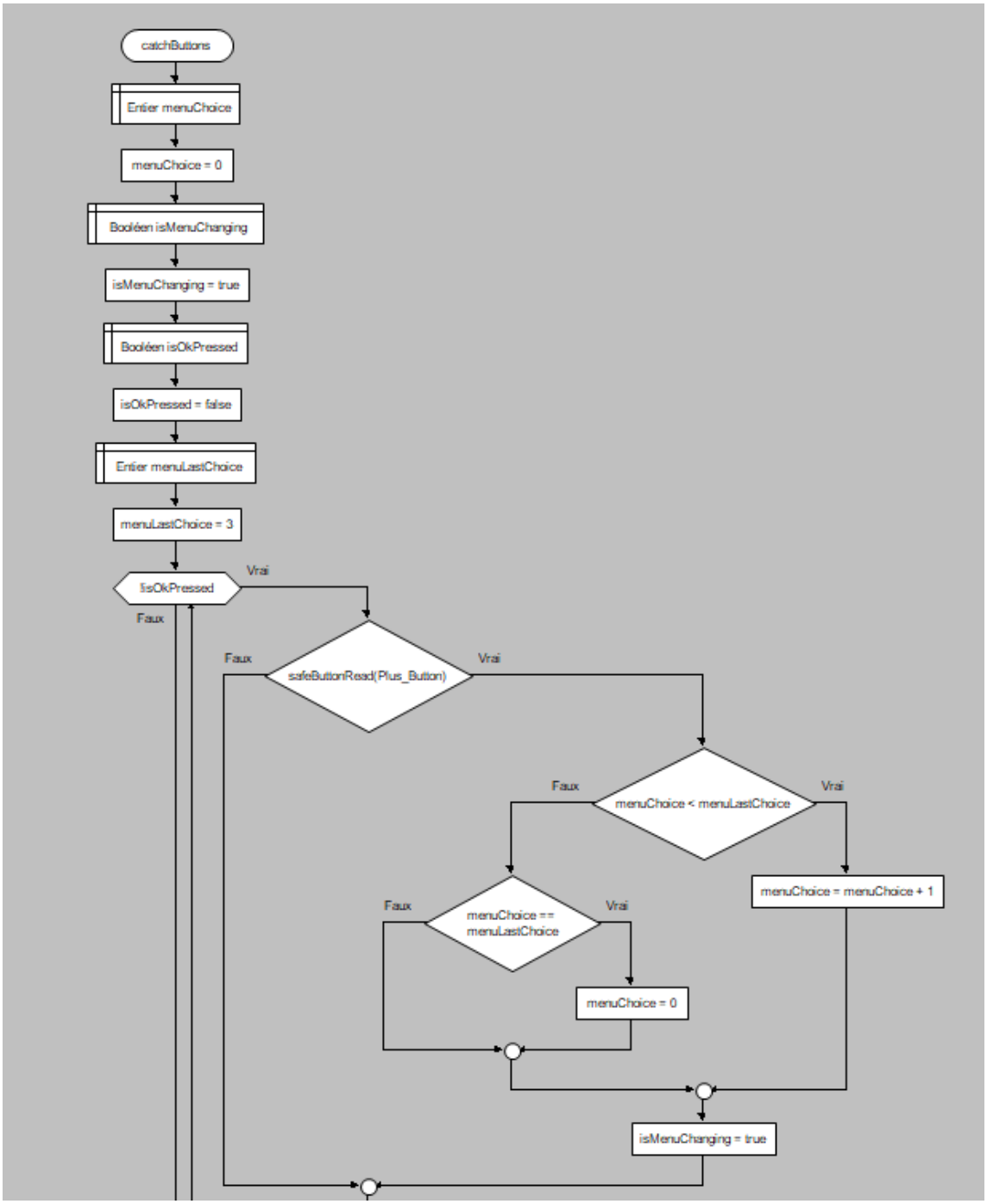
- Fonction buttonsSetup

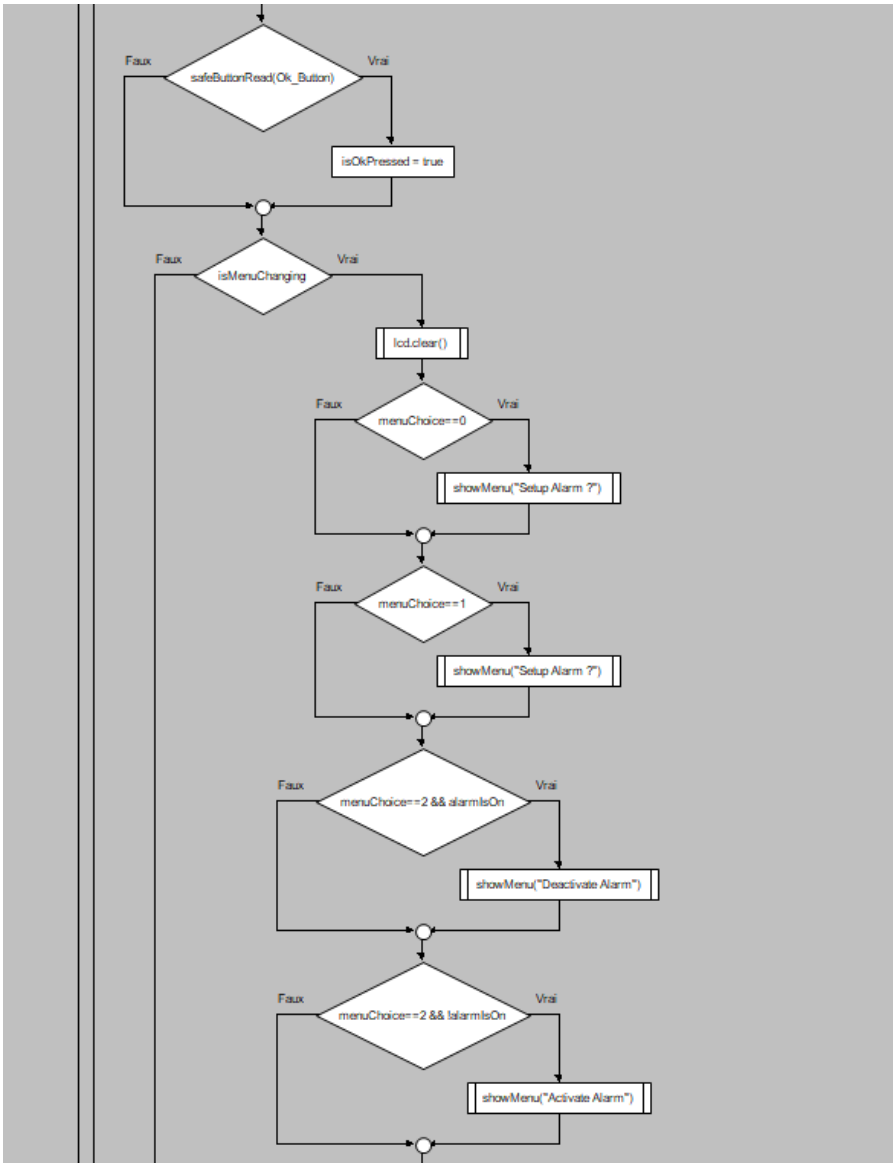


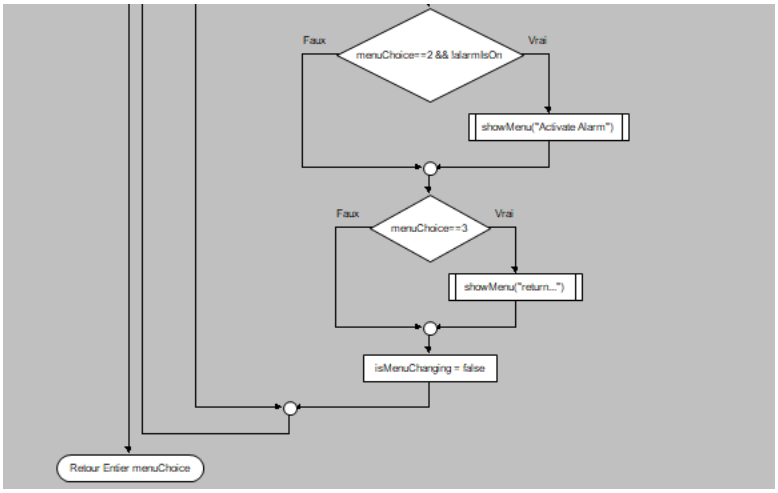
- Fonction calculNbJour



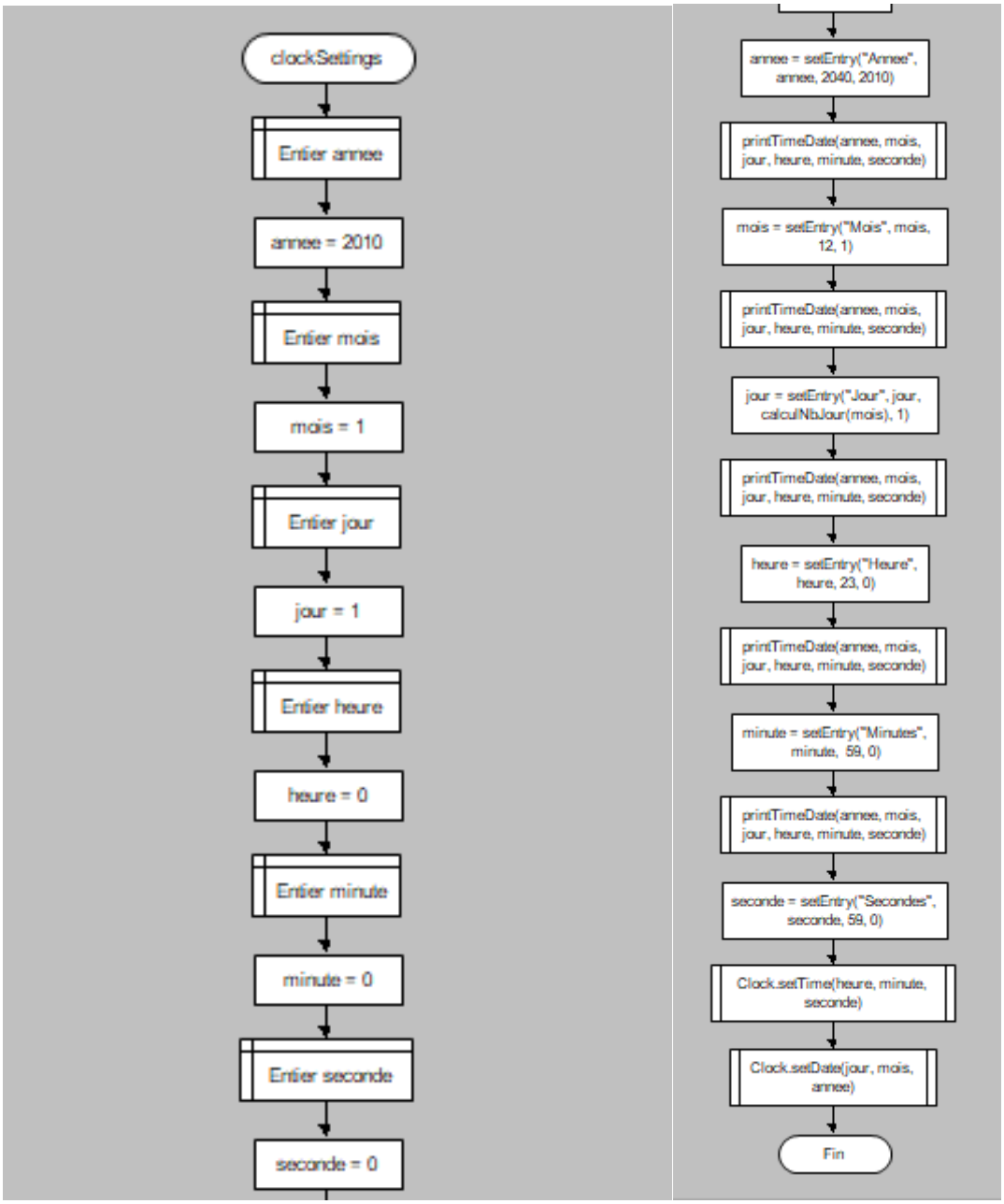
- Fonction catchButton



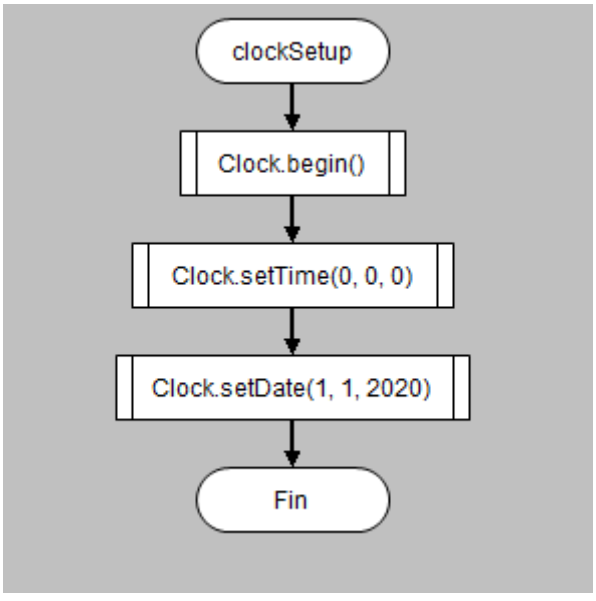

Faux
safeButtonRead(Ok_Button)
Vrai
isOkPressed = true
Faux
isMenuChanging
Vrai
lcd.clear()
Faux
menuChoice==0
Vrai
showMenu("Setup Alarm ?")
Faux
menuChoice==1
Vrai
showMenu("Setup Alarm ?")
Faux
menuChoice==2 && alarmIsOn
Vrai
showMenu("Deactivate Alarm")
Faux
menuChoice==2 && !alarmIsOn
Vrai
showMenu("Activate Alarm")


Faux
menuChoice==2 && !alarmIsOn
Vrai
showMenu("Activate Alarm")
Faux
menuChoice==3
Vrai
showMenu("return...")
isMenuChanging = false
Retour Entier menuChoice

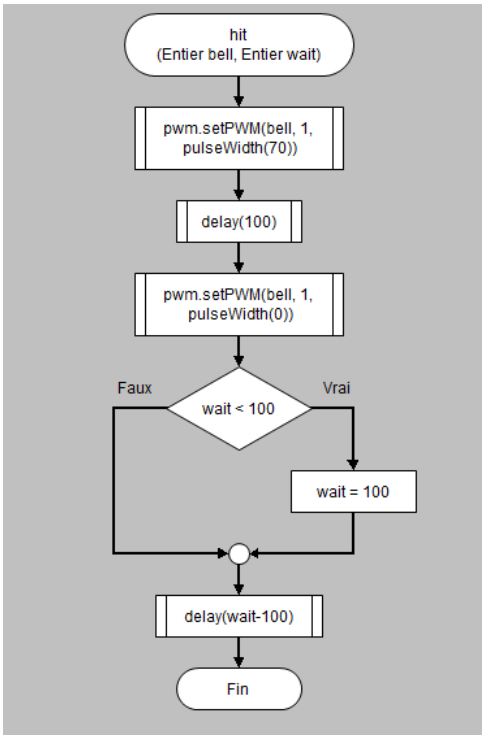
- Fonction ClockSettings



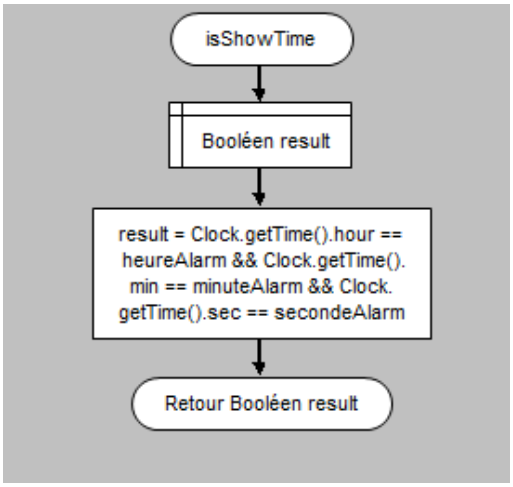
- Fonction clockSetup



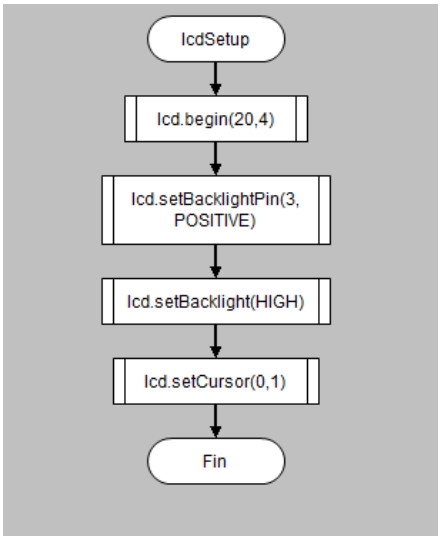
- Fonction hit



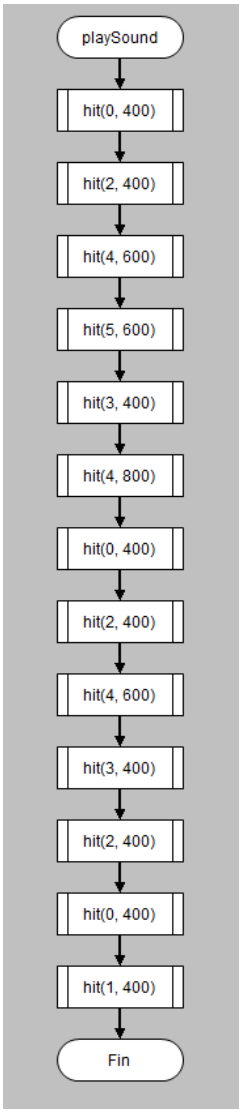
- Fonction isShowTime



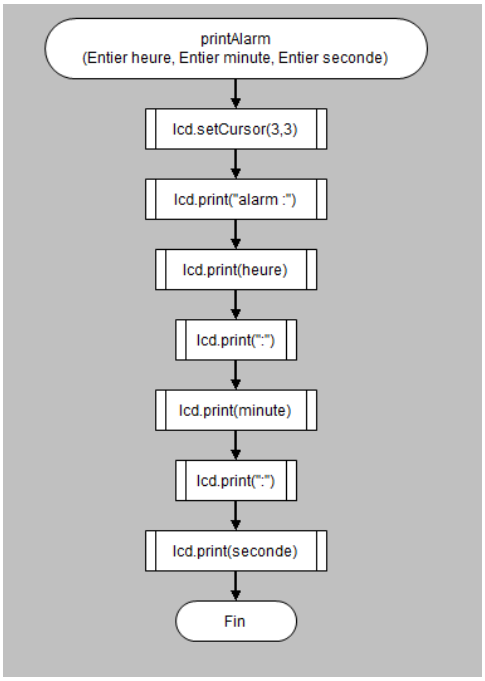
- Fonction lcdSetup



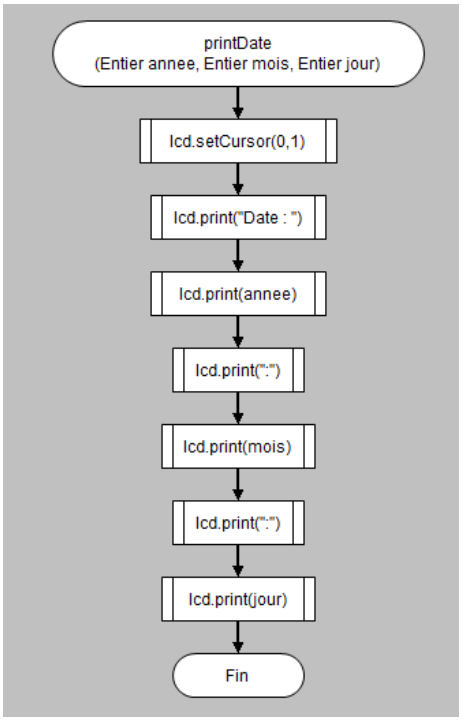
- Fonction playSound



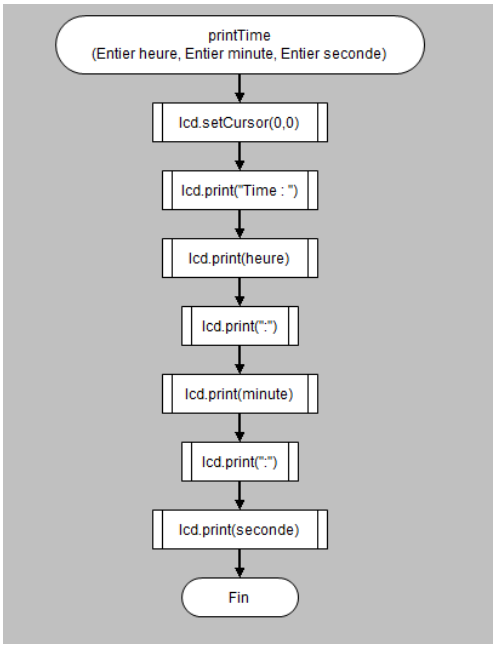
- Fonction printAlarm



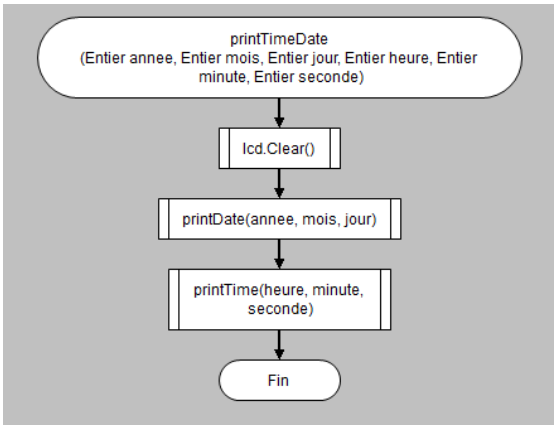
- Fonction printDate



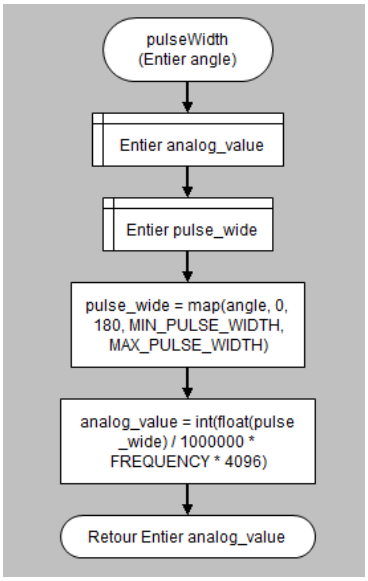
- Fonction printTime



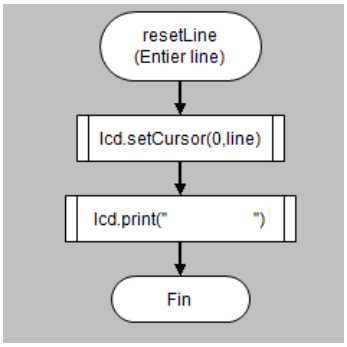
- Fonction printTimeDate



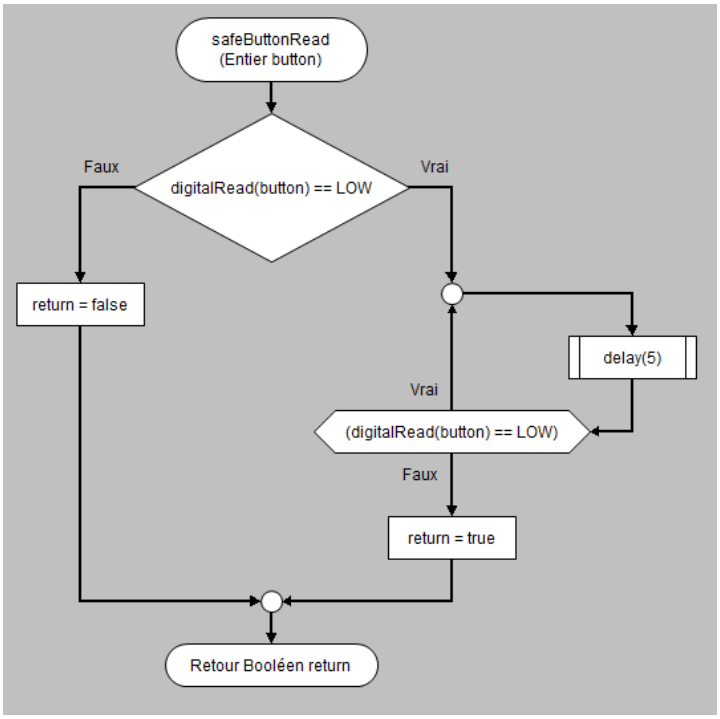
- Fonction pulseWidth



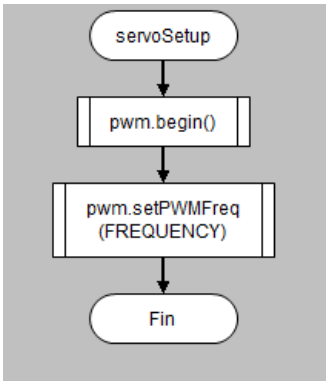
- Fonction resetLine



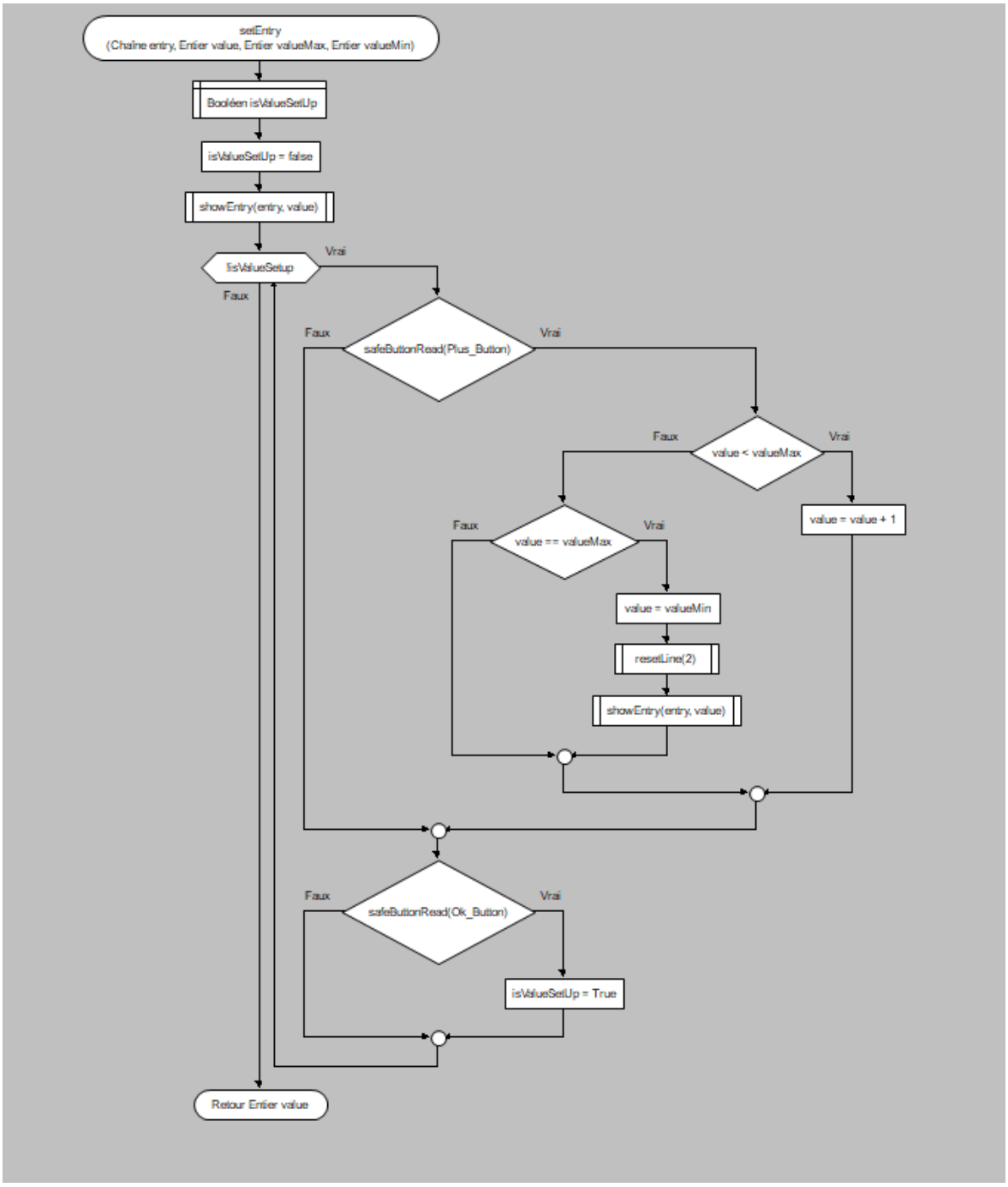
- Fonction safeButtonRead



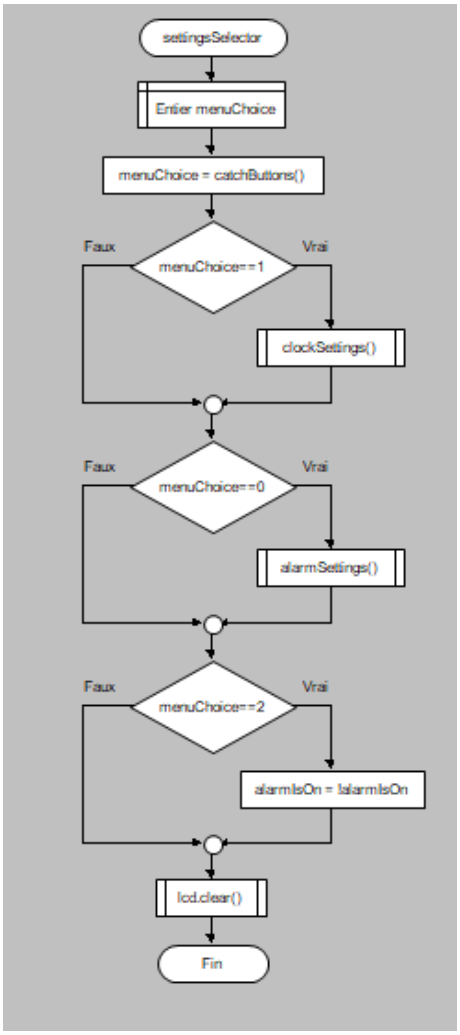
- Fonction servoSetup



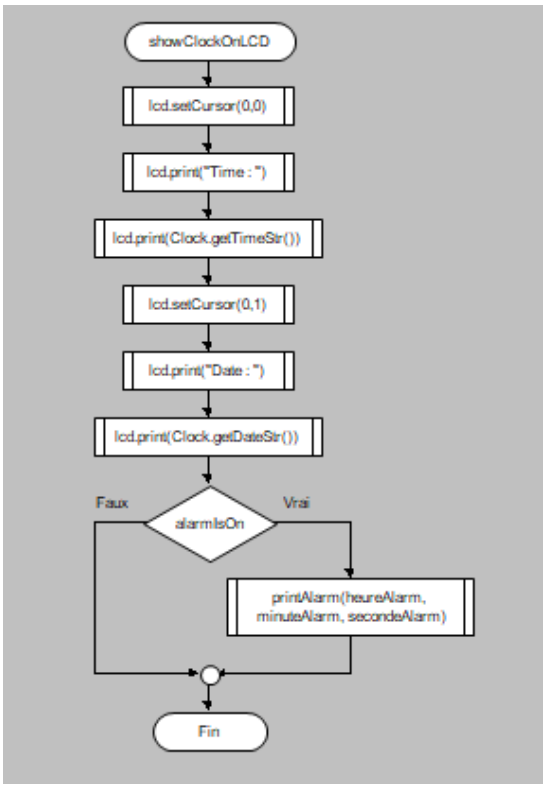
- Fonction setEntry



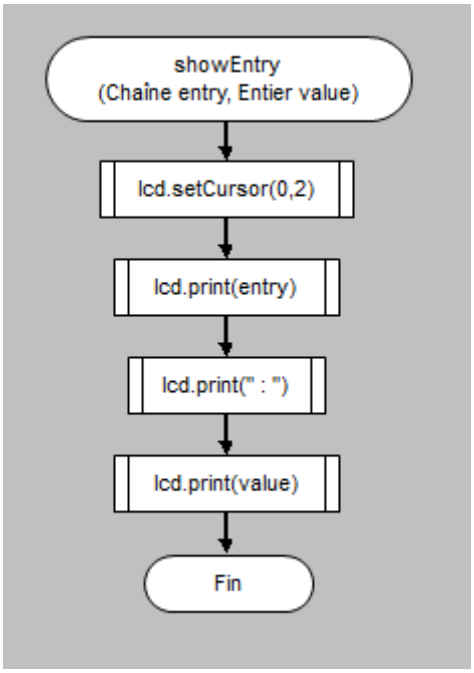
- Fonction settingsSelector



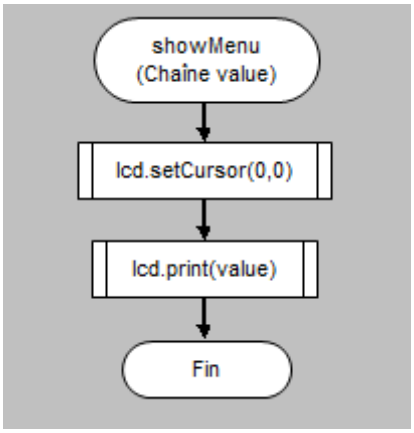
- Fonction showClockOnLCD



- Fonction showEntry



- Fonction showMenu



b. **Le code final bien explicité (signification des variables, des procédures, respecter une casse propre, …)**

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h> //librairie pour la carte PCA9685
#include <LCD.h> // pour l'écran LCD
#include <LiquidCrystal_I2C.h> // pour gérer l'écran LCD avec les pins SDA et SCL
#include <DS3231.h> // librairie de l'horloge

DS3231  Clock(SDA, SCL);// on règle Clock pour utiliser le bus i2c
```

```
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(); //on définit une
variable pwm pour pouvoir utiliser le servo driver

#define MIN_PULSE_WIDTH 650 // largeur minimume de la pulsation
#define MAX_PULSE_WIDTH 2350 //max
#define DEFAULT_PULSE_WIDTH 1500 // default
#define FREQUENCY 50 // fréquence



LiquidCrystal_I2C lcd(0x27,2,1,0,4,5,6,7);//constructeur de LiquidCrystal =>
I2C_ADDR, EN_pin, RW_pin, RS_pin, D4_pin, D5_pin, D6_pin, D7_pin);

const int Ok_Button = 53; // port numérique lié au bouton poussoir
const int Plus_Button = 52; // port numérique lié au bouton poussoir
int heureAlarm, minuteAlarm, secondeAlarm = 0; //On initialise les variables
int heureAlarm, minuteAlarm et secondeAlarm, et on leur donne 0 comme valeur
de base
bool alarmIsOn = false; //On initialise un booleen alarmIsOn, et on le set à
false
bool isPlaying = false; //Met la valeur initiale de isPlaying à false

void lcdSetup(){ //Methode comprenant tout le setup de base de l'écran, qu'on
appellera dans le setup de base
  lcd.begin (20,4); //on règle l'écran
  lcd.setBacklightPin(3,POSITIVE); // règle la lumière de fond de l'affichage
avec les arguments suivants => BL, BL_POL (BackLight, BackLight_Polarity)
  lcd.setBacklight(HIGH); //règle l'intensité de la lumière
  lcd.setCursor(0,1); //curseur à la première case de la deuxième ligne
}

void clockSetup(){//Methode comprenant tout le setup de base de l'horloge,
qu'on appellera dans le setup de base
  Clock.begin(); //init de l'horloge
  Clock.setTime(0, 0, 0); // règle l'horloge(24hr format)
  Clock.setDate(1, 1, 2020); // règle la date JJ/MM/AAAA
}

void servoSetup(){ //Methode permettant d'initialiser les servos, qu'on
appellera dans le setup de base
  pwm.begin();//initialise le servo driver
  pwm.setPWMFreq(FREQUENCY); //on set la fréquence de pwm des servo
}
```

```
void buttonsSetup(){ //Methode permettant d'initialiser les boutons, qu'on appellera dans le setup de base
  pinMode(Ok_Button, INPUT); // réglage du port du bouton en mode ENTREE
  pinMode(Plus_Button, INPUT); // réglage du port du bouton en mode ENTREE
}

int pulseWidth(int angle)//fonction pour régler l'amplitude du servo => son angle
{
  int pulse_wide, analog_value;//on déclare 2 variables
  pulse_wide = map(angle, 0, 180, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH);// on donne les paramètres au servo pour régler l'amplitude
  analog_value = int(float(pulse_wide) / 1000000 * FREQUENCY * 4096); //on définit la pwm et la valeur de retour de la fonction
  return analog_value;//on retourne la valeur à envoyer au servo pour setup la pwm
}

void hit(int bell, int wait) //methode permettant de dire à un moteur quel mouvement faire
{
  pwm.setPWM(bell, 1, pulseWidth(70));//On donne à la fonction : la cloche à faire sonner,1(qui sert à la proportion pour la transition on/off), un angle de 70°
  delay(100);//on attend 100ms pour que le servo ait le temps de se baisser entièrement. Attendre plus longtemps implique que la cloche sonne mal
  pwm.setPWM(bell, 1, pulseWidth(0));//On remet le servo à sa position initiale
  if(wait<100)//protection pour que wait ne soit pas trop petit
    wait = 100;//on définit wait à 100ms
  delay(wait-100);// on soustrait 100 à wait, car on a déjà attendu 100ms au début de cette fonction. Ainsi si on demande d'attendre 3000 ms, on attend 100ms entre les mouvements du servo, puis 2900ms à la fin.
  //On attend un total d'environ 3000ms + le temps des instructions.. (quasi négligeable)
}

void playSound(){ //Méthode permettant de jouer la mélodie
  hit(0, 400); // active le moteur associé à la pin pwm 0, attend 400ms
  hit(2, 400); //idem avec la pin pwm 2,...
  hit(4, 600);
  hit(5, 600);
  hit(3, 400);
  hit(4, 800);
```

```
  hit(0, 400);
  hit(2, 400);
  hit(4, 600);
  hit(3, 400);
  hit(2, 400);
  hit(0, 400);
  hit(1, 400);
}

bool isShowTime(){ //renvoie si l'heure actuelle est égale à celle de
l'alarme
 return Clock.getTime().hour == heureAlarm && Clock.getTime().min ==
minuteAlarm && Clock.getTime().sec == secondeAlarm;//si l'heure vaut l'heure
de l'alarme et ainsi de suite...
}

void alarm(){// fonction qui gère l'alarme
  if(!isShowTime()) //Si c'est pas l'heure, on ne joue pas
    isPlaying = false; // empêche la mélodie d'être jouée
  else if (!isPlaying && isShowTime() && alarmIsOn){ //sinon si ça ne joue
pas, que c'est l'heure et que l'alarme est sur "on"
    isPlaying = true; // on débloque la mélodie
    lcd.clear(); // on vide l'écran
    showMenu("C'est l'heure !"); // on affiche ce message
    playSound(); //on joue la mélodie
    playSound(); // deux fois
    lcd.clear(); // On vide l'écran
  }
}

void ShowClockOnLCD(){ //méthode permettant d'afficher l'heure et la date sur
l'écran LCD
  lcd.setCursor(0,0); //met le curseur à la case 0 sur la ligne 0
  lcd.print("Time : "); // affiche "Time : " sur l'écran lcd
  lcd.print(Clock.getTimeStr()); // Affiche l'heure via la méthode
getTimeStr()
  lcd.setCursor(0,1); //met le curseur à la case 0 sur la ligne 1
  lcd.print("Date : "); //affiche "Date:" sur l'écran LCD
  lcd.print(Clock.getDateStr()); // affiche la date via la méthode
getDateStr()
  if(alarmIsOn) printAlarm(heureAlarm, minuteAlarm, secondeAlarm); // si
l'alarme est sur "on", on l'affiche en appelant printAlarm
}
```

```
void showEntry(String entry, int value){//montre  la valeur à modifier
"entry"
  lcd.setCursor(0,2); // on va à la 2e ligne
  lcd.print(entry); //on montre ce qu'on modifie; entry en argument
  lcd.print(" : "); // on ajoute :
  lcd.print(value); // on ajoute la valeur en argument
}

void resetLine(int line){ // permet d'effacer une ligne sans effacer tout
l'écran
  lcd.setCursor(0,line);//on set le curseur à la ligne reçue en arguement
  lcd.print("                    ");//On écrase la valeur de la ligne
}

int calculNbJour(int mois){ //Permet de calculer la valeur du nombre de jour
pour un mois donné en argument
  switch (mois){
   case 1://Janvier
    return 31;
   case 2://Février
    return 29;
   case 3://Mars
    return 31;
   case 4://Avril Lavigne
    return 30;
   case 5://Mai
    return 31;
   case 6://Juin
    return 30;
   case 7://Juillet
    return 31;
   case 8://Aout
    return 31;
   case 9://Septembre
    return 30;
   case 10://Octobre
    return 31;
   case 11://Novembre
    return 30;
   case 12://Décembre
    return 31;
```

```
  }
}
void printTime(int heure, int minute, int seconde){ //permet d'afficher
l'heure
  lcd.setCursor(0,0); // à la case 0 et à la ligne 0
  lcd.print("Time : "); //affiche time :
  lcd.print(heure);// affiche l'heure
  lcd.print(":"); // afficher :
  lcd.print(minute);//affiche les minutes
  lcd.print(":");//affiche :
  lcd.print(seconde);//affiche les secondes
}
void printDate(int annee, int mois, int jour){//permet d'afficher la date
  lcd.setCursor(0,1);// à la case 0 et à la ligne 1
  lcd.print("Date : ");//affiche date :
  lcd.print(annee);//affiche l'année
  lcd.print(":");// afficher :
  lcd.print(mois);//affiche le mois
  lcd.print(":");// afficher :
  lcd.print(jour);//affiche le jour
}

void printAlarm(int heure, int minute, int seconde){// permet d'afficher
l'alarme
  lcd.setCursor(3,3);// à la ligne 3, avec un alinéa de 3 cases
  lcd.print("alarm : ");//affiche alarm :
  lcd.print(heure);//affiche les heures
  lcd.print(":");// afficher :
  lcd.print(minute);//affiche les minutes
  lcd.print(":");// afficher :
  lcd.print(seconde);//affiche les secondes
}

void printTimeDate(int annee, int mois, int jour, int heure, int minute, int
seconde){//permet d'afficher la date et l'heure
  lcd.clear();//on vide l'écran
  printDate(annee, mois, jour);// on affiche la date
  printTime(heure, minute, seconde);// on affiche l'heure
}

int setEntry(String entry, int value, int valueMax, int valueMin){ // permet
de setup année, mois, Jours ... selon les paramètres donnés
```

```
  bool isValueSetUp = false;// on déclare un booléen qui sert à dire si
l'entrée est correctement setup, elle permet de sortir de la boucle while
  showEntry(entry, value);//on affiche l'entrée sur le lcd "entry : value"
  while(!isValueSetUp){//tant que la value n'est pas setup
    if(safeButtonRead(Plus_Button)){ //si on appuie sur le bouton +
      if(value < valueMax) value++; // si value est plus petite que la valeur
max donnée, on incrémente value
      else if (value == valueMax) value = valueMin; //sinon si elles sont
égales, on reset value en lui donnant sa valeur minimum passée en argument
      resetLine(2);// on reset la ligne 2 pour éviter les résidus quand
l'affichage précédent est plus long
      showEntry(entry, value); // on affiche la valeur et l'entrée modifiée
    }
    if(safeButtonRead(Ok_Button)){//si on appuie sur le bouton ok
      isValueSetUp = true;//on passe la valeur à true, on peut donc sortir de
la boucle
    }
  }
  return value; //on retourne la valeur modifiée
}

void alarmSettings(){//permet de régler les valeurs de l'alarme
  printTime(heureAlarm, minuteAlarm, secondeAlarm);//afficher l'alarme
  heureAlarm = setEntry("Heure", heureAlarm, 23, 0);//on règle l'heure
  printTime(heureAlarm, minuteAlarm, secondeAlarm);//afficher l'alarme mise à
jour
  minuteAlarm = setEntry("Minutes", minuteAlarm,  59, 0);//on règle les
minutes
  printTime(heureAlarm, minuteAlarm, secondeAlarm);// afficher l'alarme mise
à jour
  secondeAlarm = setEntry("Secondes", secondeAlarm, 59, 0);// on règle les
secondes
  alarmIsOn = true;// l'alarme a été paramétrée
}

void clockSettings(){//permet de régler la date et l'heure de l'horloge
  int annee = 2010;//déclaration et définition des valeurs par défauts de la
date et l'heure
  int mois = 1;
  int jour = 1;
  int heure = 0;
  int minute = 0;
  int seconde = 0;
  annee = setEntry("Annee", annee, 2040, 2010);//on règle l'année
```

```
  printTimeDate(annee, mois, jour, heure, minute, seconde);//on affiche la
date et l'heure  à jour
  mois = setEntry("Mois", mois, 12, 1);//on règle le mois
  printTimeDate(annee, mois, jour, heure, minute, seconde);//on affiche la
date et l'heure  à jour
  jour = setEntry("Jour", jour, calculNbJour(mois), 1);//on règle les jours
  printTimeDate(annee, mois, jour, heure, minute, seconde);//on affiche la
date et l'heure  à jour
  heure = setEntry("Heure", heure, 23, 0);//on règle les heures
  printTimeDate(annee, mois, jour, heure, minute, seconde);//on affiche la
date et l'heure  à jour
  minute = setEntry("Minutes", minute,  59, 0);//on règle les minutes
  printTimeDate(annee, mois, jour, heure, minute, seconde);//on affiche la
date et l'heure  à jour
  seconde = setEntry("Secondes", seconde, 59, 0);//on règle les secondes
  Clock.setTime(heure, minute, seconde);     // On utilise les variables
paramétrées précédemment pour set l'heure de l'horloge
  Clock.setDate(jour, mois, annee);   // On utilise les variables paramétrées
précédemment pour set la date de l'horloge
}

bool safeButtonRead(int button){//permet de capturer l'état d'un bouton de
façon sécurisée et efficace
  if(digitalRead(button) == LOW){//Les pins des boutons sont naturellement à
un état haut, puisque la résistance est entre l'input et vcc, c'est donc un
montage en pull up avec résistance externe, on capture donc un état bas,
lorsque l'interrupteur est fermé et que le courant s'échappe à la masse.
    do{//on attend
      delay(5);//5ms
    }while(digitalRead(button) == LOW);//Tant que le bouton est à low, pour
éviter les rebonds et ne sortir de cette boucle uniquement lors d'un
changement d'état du bouton. L'exécution est piégée tant que le bouton est
enfoncé
    return true;// Si le bouton surveillé (passé en argument) a été enfoncé
puis relaché, on répond true,
  }
  return false;//sinon on répond false
}

void activateSettingsMenu(){//Permet d'activer le menu des réglages
  if (safeButtonRead(Ok_Button)) { // si le bouton ok est pressé ...
    lcd.clear();// on vide l'écran
    settingsSelector(); //on affiche le choix des menus
  }
}
```

```
void showMenu(String value){//permet d'afficher un titre sur la première
ligne
  lcd.setCursor(0,0);//on retourne à la ligne 0
  lcd.print(value);  // on affiche le paramètre donné
}

int catchButtons(){// Permet de surveiller l'état des boutons
  int menuChoice = 0;//c'est la variable qui contiendra l'état du menu
  bool isMenuChanging = true;//c'est la variable qui contiendra si le menu a
changé
  bool isOkPressed = false;//c'est la variable qui contiendra si le bouton ok
est pressé
  int menuLastChoice = 3;//le dernier choix du menu avant d'afficher à
nouveau le premier choix
  while(!isOkPressed){//tant que ok n'est pas pressé
    if(safeButtonRead(Plus_Button)){//si + est pressé
      if(menuChoice < menuLastChoice) menuChoice++;//si le choix est plus
petit que le dernier choix possible, on incrémente
      else if(menuChoice == menuLastChoice) menuChoice = 0;// sinon si le
choix actuel est le dernier de la liste, on retourne à 0
      isMenuChanging = true; //Si on passe dans ce if, le menu a changé, donc
on set la valeur à true
    }
    if(safeButtonRead(Ok_Button)){ //Si le bouton Ok est appuyé...
      isOkPressed = true; // set isOkPressed à true
    }
    if(isMenuChanging){//si le menu a changé
      lcd.clear();//on vide l'écran
      if(menuChoice==0) showMenu("Setup Alarm ?");//si le menu est 0, on
affiche
      if(menuChoice==1) showMenu("Setup Time ?");// ...
      if(menuChoice==2 && alarmIsOn ) showMenu("Deactivate Alarm");// si le
menu est 2 et que l'alarme est allumée, on affiche une option pour désactiver
l'alarme
      if(menuChoice==2 && !alarmIsOn ) showMenu("Activate Alarm");// si le
menu est 2 et que l'alarme est éteinte, on affiche l'option pour activer
l'alarme
      if(menuChoice==3) showMenu("return..."); //si menuChoice vaut 2, on
affiche showMenu
      isMenuChanging=false; //set isMenuChanging à false
    }
  }
  return menuChoice; //retourne la valeur de menuChoice
```

```
}

void settingsSelector(){//permet de choisir le menu sélectionné
  int menuChoice = catchButtons(); // récupère la valeur du menu à afficher si on presse un bouton
  if(menuChoice==1) clockSettings(); //si menuChoice vaut 1, on appelle clockSettings
  if(menuChoice==0) alarmSettings(); //si menuChoice vaut 0, on appelle alarmSettings
  if(menuChoice==2) alarmIsOn = !alarmIsOn; //on inverse l'état de l'alarme, si elle était éteinte, on l'allume et inversement
  lcd.clear();//enlève tout l'affichage de l'écran
}

void setup()
{
  Wire.begin();//initialise la lib Wire
  lcdSetup();//initialise la lib lcd
  clockSetup();//initialise l'horloge
  servoSetup();//initialise les servo
  buttonsSetup();//initialise les boutons
  showMenu("Initialisation...");
  playSound();//on joue une fois la musique pour dire bonjour
  lcd.clear();//on vide l'écran
}

void loop() {
  ShowClockOnLCD(); //Affiche l'horloge et la date
  activateSettingsMenu(); //check s'il faut activer les settings et les lance si c'est le cas
  alarm();//on check si l'alarme doit sonner ou pas
}
```

c. **Quelques photos du fonctionnement effectif. Ici dans le document . Et une petite vidéo de quelques secondes, sur votre ONEDRIVE d'étudiant, avec le lien partagé à mon adresse HELHA**
Ou plusieurs petites.

- **Dès que le réveil est branché au secteur :**

- **Réglages par défaut :**

- **Différentes options sélectionnables du menu :**

- **Si on sélectionne Setup Time**

Ici par exemple, j'ai réglé l'heure à 8h10m00s et la date au 15 juin 2021.

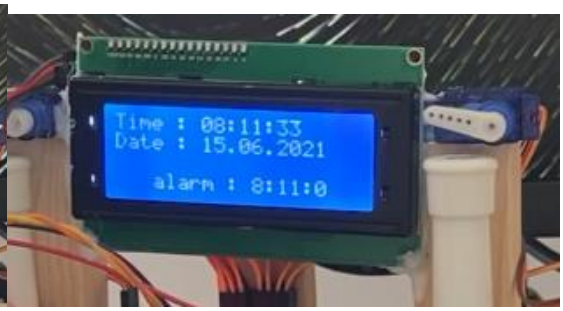
Ces données sont maintenant affichées sur l'écran de base :

- **Si on sélectionne Setup Alarm**

Ici j'ai choisi de régler l'alarme sur 8h11min00sec

L’alarme est désormais visible sur l’écran de base, il n’y a plus qu’à attendre que le réveil sonne !

- **Détail :** On a pu voir sur les screens de sélection des menus qu’il y avait une option « Activate alarm » Maintenant qu’on en a réglé une, le menu affichera à la place de cette option « Deactivate Alarm », l’alarme ne sera plus affichée sur l’écran, et ne sonnera pas. Mais sa valeur est conservée, donc si on retourne dans les menus et qu’on sélectionne « Activate Alarm », elle se réactivera pour 8h11.



La note des parties 1 et 2 sera sur 10

La note du 3a sera sur 10

La note du 3b sera sur 10

La note du 3c sera sur 10.