Dossier De Conception (DDC)

du projet

Système électronique

DU ENERGIES

Responsabilité documentaire

Action	NOM Prénom	Fonction	Date	Signature
Rédigé par	BORDRON GAMBERT, FEUGAS, BROUSSE, GRILLET	Techniciens	26/10/2023	
Approuvé par	(Equipe DU Energies)	Chef de projet		

Suivi des révisions documentaires

Indice	Date	Nature de la révision
1	26/10/2023	Publication du dossier de conception

Documents de références

Sigle	Référence	Titre	Rév.	Origine
[CDC]	Challenge_DU_Energie_ CDC	Cahier des charges	1	Equipe DU Energies

Table des matières

1. Nature du document

Ce document est un dossier de conception et a pour but de détailler la conception du système électronique. Il apporte ainsi des preuves de la conformité du produit par rapport à l'ensemble des exigences client.

2. Conception détaillée du produit

Ce chapitre décrit l'architecture fonctionnelle du produit. Il apporte les premiers éléments de preuve de la faisabilité du produit vis-à-vis des exigences client.

Rédacteur :BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Relecteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

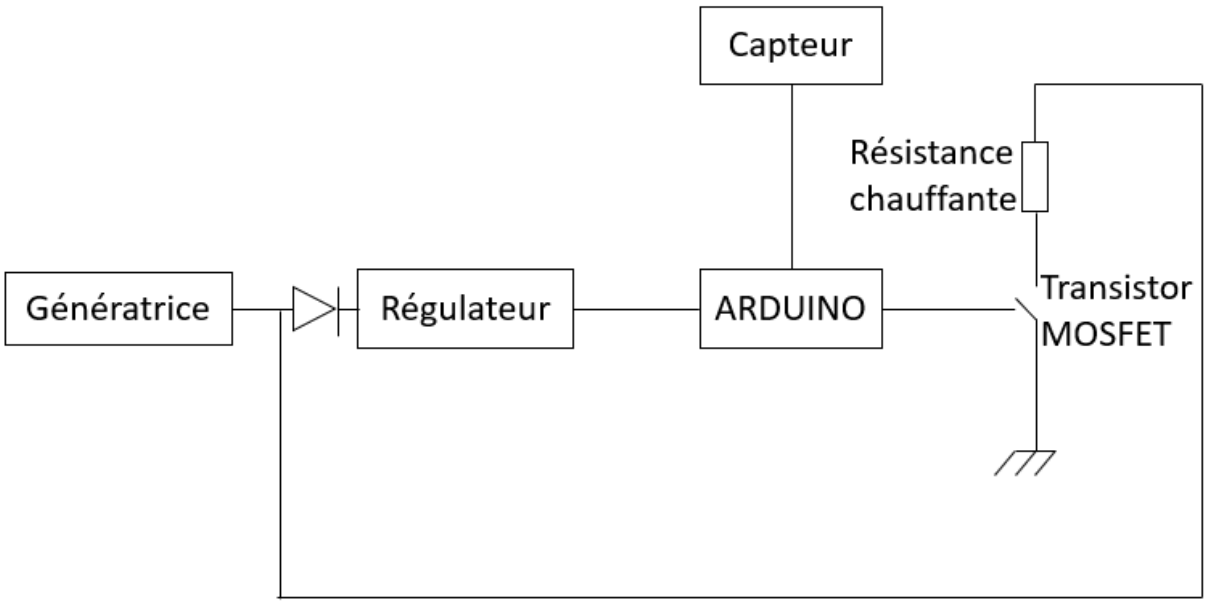

Figure 1 : Synoptique d'architecture du système

Référence de conception détaillée: Mesure de la température

Rédacteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Relecteur :BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Exigences client vérifiées par conception détaillée : Mesure de la température

Pour vérifier le bon fonctionnement de notre système de chauffe ainsi que vérifier la température lors de la chauffe nous avons décidé d'utiliser une sonde de température. Pour ce faire nous allons utiliser une carte Arduino reliée à la sonde de température. Le programme téléversé dans la carte permettra de relever la température et de l'afficher sur le moniteur intégré au logiciel Arduino.

Nous avons récupéré un programme de relevé de température par Arduino sur internet (voir figure 2, 3 et 4) que nous avons commenté ligne par ligne afin de faciliter la compréhension. Ce programme fonctionne uniquement lorsque la bibliothèque OneWire.h est installée.

```
1   #include <OneWire.h>
2   int DS18S20_Pin = 2;  //initialisation de la variable correspondant à broche d'entrée
3
4   OneWire ds(DS18S20_Pin);  // initialisation de la broche d'entrée
5
6   void setup(void) {
7     Serial.begin(9600); //configuration de la communication avec le moniteur
8   }
9   void loop(void) {
10    float temperature = getTemp(); // initialisation de la variable temperature à
11    //la valeur renvoyée par la fonction getTemp() définie figure 3
12    Serial.print(temperature); // envoie au moniteur série la température
13    delay(20); // délai pour facilité la lecture
14  }
```

Figure 2 : Programme principal

```
17  float getTemp() {
18    //returns the temperature from one DS18S20 in DEG Celsius
19    byte data[12];
20    byte addr[8];
21
22    if (!ds.search(addr)) {
23      //no more sensors on chain, reset search
24      Serial.println("no more sensors on chain, reset search!");
25      ds.reset_search();
26      return -1000;
27    }
28
29    if (OneWire::crc8(addr, 7) != addr[7]) {
30      Serial.println("CRC is not valid!");
31      return -1000;
32    }
33
34    if (addr[0] != 0x10 && addr[0] != 0x28) {
35      Serial.print("Device is not recognized");
36      return -1000;
37    }
```

Figure 3 : Fonction getTemp() partie 1/

```
38    ds.reset();
39    ds.select(addr);
40    ds.write(0x44, 1);  // start conversion, with parasite power on at the end
41    byte present = ds.reset();
42    ds.select(addr);
43    ds.write(0xBE);  // Read Scratchpad
44    for (int i = 0; i < 9; i++) {  // we need 9 bytes
45      data[i] = ds.read();
46    }
47    ds.reset_search();
48    byte MSB = data[1];
49    byte LSB = data[0];
50    float tempRead = ((MSB << 8) | LSB);  //using two's compliment
51    float TemperatureSum = tempRead / 16;
52    return TemperatureSum;
53  }
```

Figure 4 : partie 3 du programme

Référence de conception détaillée: Alimentation de la carte Arduino

Rédacteur :BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Relecteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Exigences client vérifiées par conception détaillée : Alimentation de la carte Arduino

La datasheet de la carte Arduino stipule que la tension d'alimentation de celle-ci ne doit pas dépasser 9V. Or nous voulions alimenter la carte directement par la tension générée par la génératrice. Celle-ci étant bien supérieure à 9V nous avons utilisé un régulateur linéaire afin de limiter cette tension d'alimentation. Comme conseillé dans la datasheet du régulateur, nous l'avons monté de cette manière (voir figure 5)

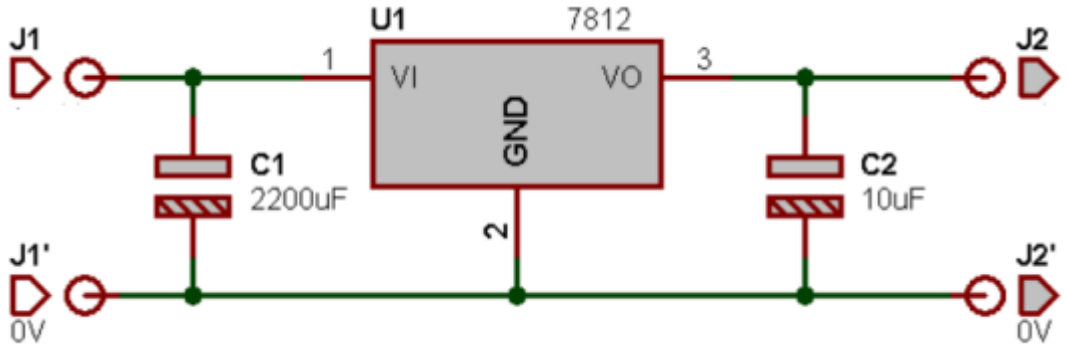

Figure 5 : Montage du régulateur linéaire

La datasheets recommande des condensateurs de découplage de valeurs C1 = 2200µF et C2 = 10µF. Or par soucis de logistique, nous allons utiliser des condensateurs de valeurs C1 = 1000µF et C2 = 100µF, valeurs conseillées par un expert. Ce régulateur linéaire permettra de stabiliser la tension de sortie à 9V afin de ne pas atteindre de valeur dangereuse pour la carte Arduino.

Référence de conception détaillée: Sens de circulation du courant

Rédacteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Relecteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Exigences client vérifiées par conception détaillée: Sens de circulation du courant

Il est important d'imposer le sens du courant car dans ce montage les condensateurs de découplage délivrent une tension. Or si celle-ci arrive en opposition à la tension générée par la génératrice, le montage sera court-circuité et la génératrice se bloquera. Nous avons donc décidé d'utiliser une diode permettant d'imposer le sens de circulation du courant dans le circuit.(voir figure 7)

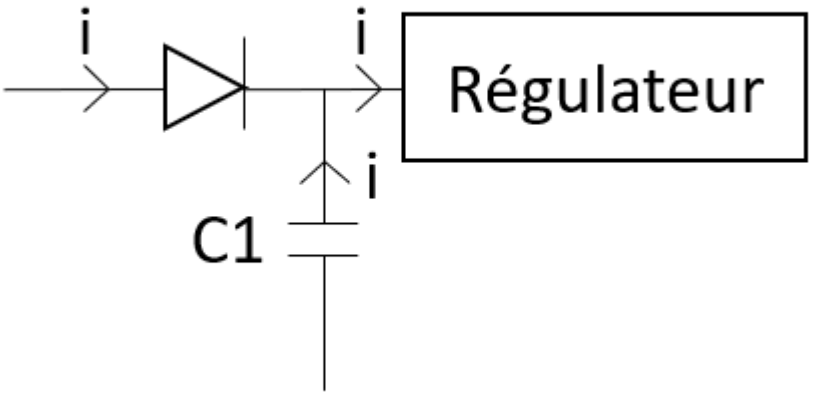

Figure 6 : Représentation du sens du courant avec la diode

On voit sur ce schéma que le courant ne part pas en direction de la génératrice grâce à la diode. La totalité du courant ira donc en direction du régulateur.

Référence de conception détaillée: Témoin de température souhaitée atteinte

Rédacteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Relecteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Exigences client vérifiées par conception détaillée : Témoin de température souhaitée atteinte

Afin de visualiser à l'aide de la carte lorsque nous allons atteindre la température souhaitée nous avons placé une LED verte. Nous avons adapté notre programme afin d'allumer la LED lorsqu'on atteint la température de 54°C (valeur minimale acceptée par le cahier des charges) et que l'on dépasse la température de 56°C.

Pour ce faire, nous avons branché notre LED sur la carte électronique en série avec une résistance. Pour des raison logistiques nous avons choisi une résistance en stock dans nos locaux. N'ayant aucune exigence sur l'intensité lumineuse de la LED et sachant que la valeur 620Ω ne serait pas dangereuse pour la LED nous avons choisi cette résistance. (voir code figure 7)

Nous avons utilisé une résistance de 620Ω.

```
35  if (temperature < 56 && temperature > 54) {
36    digitalWrite(10, HIGH);
37  }
38  else {
39    digitalWrite(10, LOW);
40  }
```

Figure 7 : code d'allumage de la LED verte

Référence de conception détaillée: Stabilisation de la température

Rédacteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Relecteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Exigences client vérifiées par conception détaillée : Stabilisation de la température

Le cahier des charges stipule que lorsque la température de 55°C est atteinte il faut pouvoir la stabiliser pour une durée de 10 minutes. Nous allons utiliser un transistor MOSFET afin de répondre à cette exigence. En effet, le fonctionnement du transistor MOSFET permet une utilisation en tant qu'interrupteur selon une commande. Nous allons commander le transistor à l'aide de la carte Arduino, permettant de chauffer plus ou moins l'eau en fonction de la température de celle-ci. Le transistor sera relié à une broche PWM (Pulse Width Modulation) permettant d'envoyer un signal suivant un cycle avec un rapport de cycle. (voir figure 8)

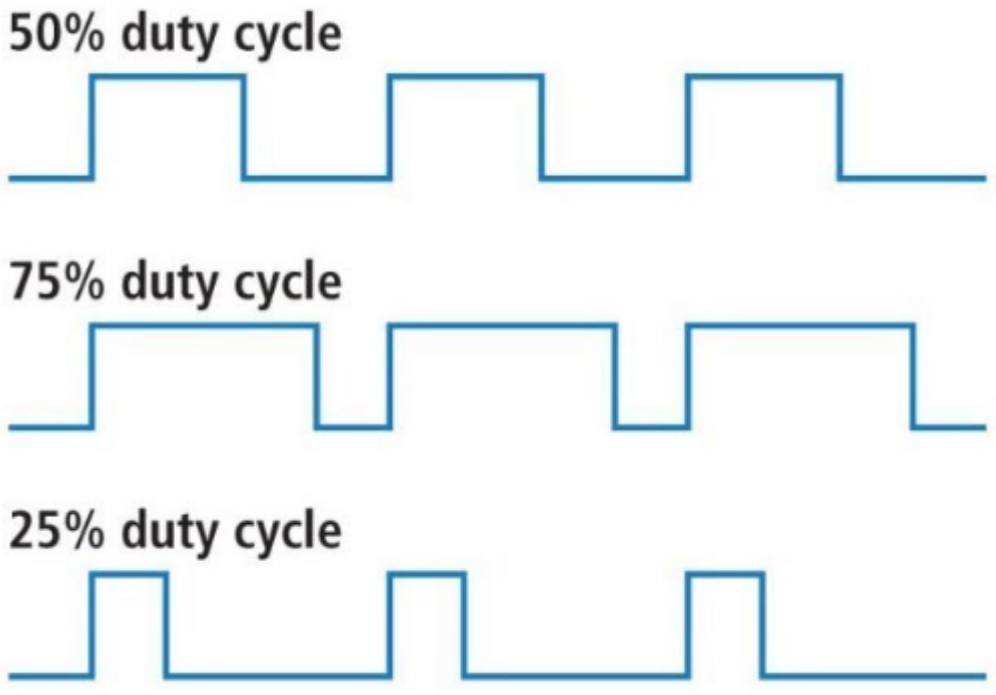

Figure 8: Représentation d'un signal PWM en fonction du rapport de cycle

Nous pouvons envoyer sur une broche PWM un signal compris entre 0 et 255, correspondant respectivement à un rapport de cycle de 0% et 100%

Dans notre cas, si la température est inférieure à 53.5°C nous allons fermer le transistor sur un rapport de cycle de 100%, ce qui correspond à un signal PWM de valeur 255.

Si la température est supérieure à 55.5°C nous allons ouvrir le transistor sur tout le cycle, ce qui correspond à un signal PWM de valeur 0. Nous avons choisis cette valeurs suite à des tests)

Si la température se situe entre 54.5°C et 55.5°C, nous allons envoyer un signal permettant de fermer le transistor sur un rapport cyclique permettant de compenser les pertes. Nous avons estimé précédemment les pertes à 5W. Nous avons mesuré la puissance fournie sur un rapport cyclique de 100%, nous observons une puissance de 60W. Nous avons donc cherché la valeur du signal PWM à envoyer afin de fournir 5W sur un rapport cyclique. Par un produit en croix nous obtenons une valeur de signal PWM de 18.

Si la température se situe entre 53.5°C et 54.5°C alors là nous commençons à ralentir la chauffe. En effet suite à des tests on s'est aperçu que la température de l'eau continuait d'augmenter malgré l'interruption du courant. Ce palier permet donc de chauffer moins vite avant la température d'objectif.

Nous avons par la suite écrit un programme permettant le fonctionnement du transistor selon ces paramètres. (voir figure 9)

Référence de pré-conception: Coût de fabrication

```
#include <OneWire.h>
int DS18S20_Pin = 2;  //DS18S20 Signal pin on digital 2

//Temperature chip i/o
OneWire ds(DS18S20_Pin);  // on digital pin 2

void setup(void) {
  Serial.begin(9600);
  pinMode(6, OUTPUT);
  pinMode(10, OUTPUT);
}
float temperature1;
void loop(void) {
  float temperature = getTemp();
  if (temperature != temperature1) {
    Serial.print("La température est : ");
    Serial.print(temperature);
    if (temperature < 56 && temperature > 54) {
      Serial.println("°C | L'eau est à 55°c +/- 1°C");
    } else if (temperature >= 56) {
      Serial.println("°C | L'eau est trop chaude");
    } else {
      Serial.println("°C | ça chauffe");
    }
  }
  if (temperature < 55.5 && temperature > 54.5) {
    analogWrite(6, 1);
  } else if (temperature < 54.5 && temperature > 53.5) {
    analogWrite(6, 5);
  } else if (temperature > 55.5) {
    analogWrite(6, 0);
  } else {
    analogWrite(6, 255);
```

```
34    }
35    if (temperature < 56 && temperature > 54) {
36      digitalWrite(10, HIGH);
37    } else if (temperature > 56) {
38      digitalWrite(10, LOW);
39    } else {
40      digitalWrite(10, LOW);
41    }
42    temperature1 = temperature;
43  }
44
45
46  float getTemp() {
47    //returns the temperature from one DS18S20 in DEG Celsius
48
49    byte data[12];
50    byte addr[8];
51
52    if (!ds.search(addr)) {
53      //no more sensors on chain, reset search
54      Serial.println("no more sensors on chain, reset search!");
55      ds.reset_search();
56      return -1000;
57    }
58
59    if (OneWire::crc8(addr, 7) != addr[7]) {
60      Serial.println("CRC is not valid!");
61      return -1000;
62    }
63
64    if (addr[0] != 0x10 && addr[0] != 0x28) {
65      Serial.print("Device is not recognized");
66      return -1000;
```

```
67    }
68
69    ds.reset();
70    ds.select(addr);
71    ds.write(0x44, 1);  // start conversion, with parasite power on at the end
72
73    byte present = ds.reset();
74    ds.select(addr);
75    ds.write(0xBE);  // Read Scratchpad
76
77
78    for (int i = 0; i < 9; i++) {  // we need 9 bytes
79      data[i] = ds.read();
80    }
81
82    ds.reset_search();
83
84    byte MSB = data[1];
85    byte LSB = data[0];
86
87    float tempRead = ((MSB << 8) | LSB);  //using two's compliment
88    float TemperatureSum = tempRead / 16;
89
90    return TemperatureSum;
91  }
```

figure 9 : code complet avec programmation du transistor

Rédacteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Relecteur : BORDRON GAMBERT Jules, FEUGAS Julien, BROUSSE Mathis, GRILLET Mathéo

Exigences client vérifiées par préconception : Coût de fabrication du système minimum

Ici nous allons faire un bilan de tous les éléments utilisés pour la fabrication du contenant de notre système, c'est-à-dire toute la partie électronique de notre système.

Désignation	Prix unitaire HT €	Prix unitaire TTC €	Quantité	Coût total TTC €
Carte arduino Uno	19.92	23.9	1	23.9
Joint torique	4.38	5.26	1	5.26

Résistance (4 Ω)	récupération		1	
Transistor mosfet	1.68	2.01	1	2.01
fiche banane male rouge	1.97	2.36	1	2.36
fiche banane male noire	1.83	2.2	1	2.2
Sonde étanche DS18B20	6.63	7.96	1	7.96
Presse étoupe	1.3	1.56	2	3.12
fil noir 0.5m	0.11	0.13	1	0.13
fil rouge 0.5m	0.11	0.13	1	0.13
Bornier à vis Gravity DFR0055	2.17	2.6	1	2.6
LED Verte	0,15	0.18	1	0.18
résistance 620Ω série E24	0.03	0.04	1	0.04
Total	40.28	48.33		49.89