

Robot Véronica

Rapport technique

Nous attestons que ce travail est original, que nous citons en référence toutes les sources utilisées et qu'il ne comporte pas de plagiat.

Table des matières

Présentation de l'équipe

Présentation du projet

Solution technique

Cahier des charges

Démarche utilisée

- 1) **Pilotage du Robot**
- 2) **Relevés de température**
- 3) **Communication entre les deux cartes**
- 4) **Communication avec l'utilisateur**

État du projet et points d'amélioration

Conclusion et remerciements

Annexes

Présentation de l'équipe

Nous sommes un groupe de 4 étudiants de l'Institut d'Optique Graduate School :

- Hajar ELAZRI
- Barbara BENAMIRA
- Baptiste BOUHET
- Léo GUERNION.

Nos compétences en programmation, en automatisation et en systèmes de communication et de détection nous ont permis de répondre aux missions de l'entreprise Solec et de réaliser l'un de ses projets qui est intitulé Robot Véronica.

Présentation du projet

L'idée du projet consiste à **guider un robot à distance pouvant réceptionner et transmettre des mesures de température et d'humidité.**

Ce robot sera réalisé en vue de l'envoyer sur Mars, afin d'explorer son sol particulier. Son parcours, composé de tronçons de ligne droite et de virages, sera transmis au fur et à mesure depuis une base terrienne. Les données collectées seront enregistrées et transmises à intervalle régulier.

Dans ce sens, nous nous sommes fixés quelques objectifs à atteindre :

- piloter le robot en vitesse et en rotation ;
- réceptionner et transmettre les résultats des mesures de température à intervalles de temps réguliers ;
- gérer l'autonomie du robot ;
- mettre en place une interface graphique d'utilisation intuitive.

Solution technique

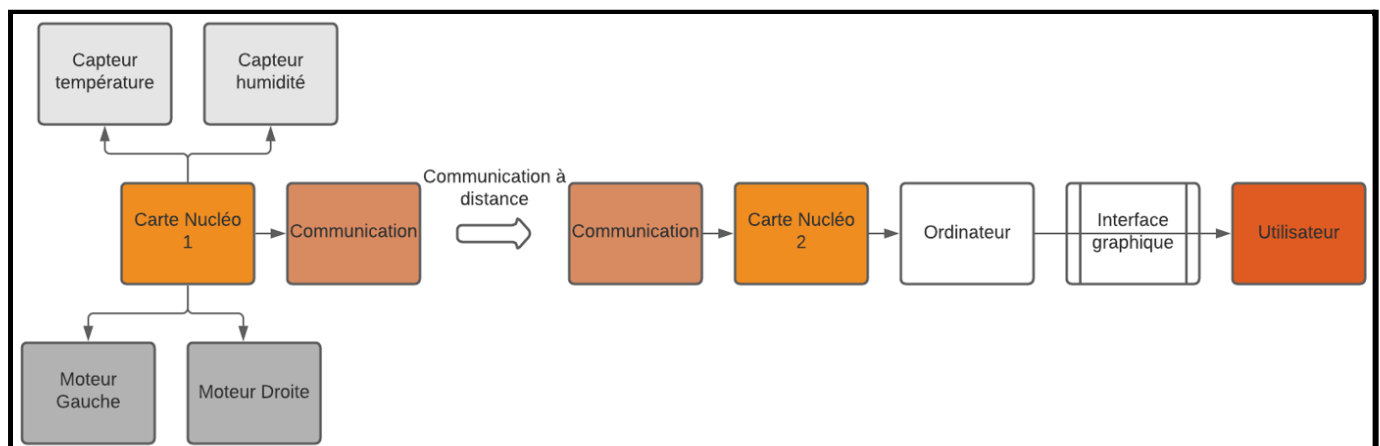


Schéma-bloc de la solution technique retenue

Nous avons divisé le problème en deux parties :




- la carte Nucléo 1, qui sera la carte à déposer sur le robot, et qui gèrera les capteurs, pilotera les moteurs et communiquera avec l'ordinateur ;
- la carte Nucléo 2, reliée à l'ordinateur, qui va gérer la communication avec le robot, et avec l'utilisateur à travers l'interface graphique.

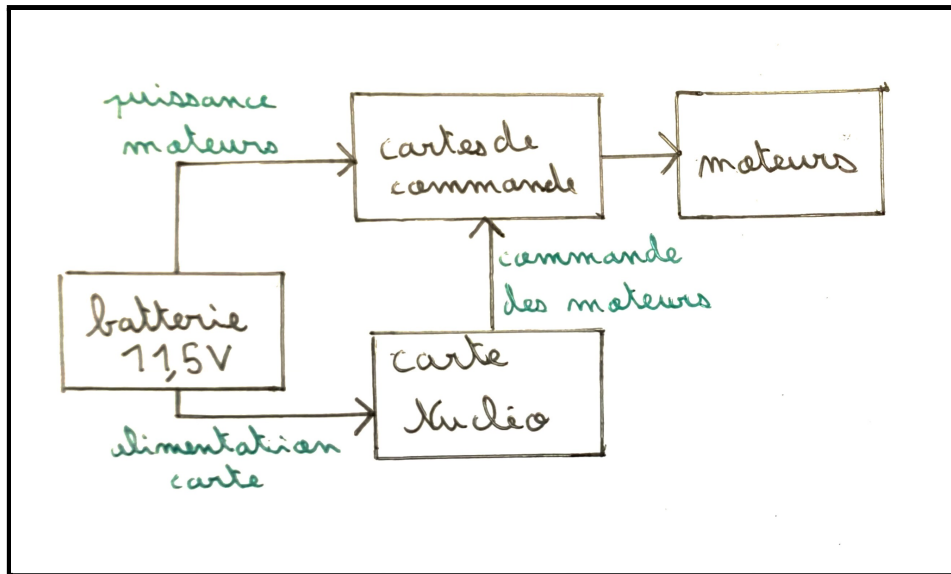
Cahier des charges

- récupération des mesures de température tous les 1 m, avec une précision de $\pm 0.25^{\circ}C$
- vitesse constante du robot de 40 ± 10 cm/s
- erreur maximale sur la position à chaque pas de 1 m de 5 cm et de 10° en angle (sans asservissement du robot)
- le robot peut réaliser un parcours d'environ 1 km avec notre batterie sans que cette dernière ne soit rechargée
- l'interface Humain-Machine permettent de transmettre les ordres de parcours peut être utilisée sans aucune formation préalable
- enfin, les données sont affichées en fonction de la distance ou du temps.

Démarche utilisée

1) Pilotage du Robot

Pour le pilotage du robot, nous avons un moteur associé à chaque roue (). Dans une optique de simplifier la programmation, chaque moteur est commandé par une carte : la *PmodHB5* () qui nous permet de contrôler simplement chaque moteur. Le pin "DIR" de cette carte nous permet de choisir le sens de rotation du moteur et le pin "EN" nous permet de choisir la vitesse de rotation du moteur. Ceci nous a donc permis de commander les moteurs indépendamment afin de réaliser les translations ou les rotations nécessaires à la trajectoire d'un point A à un point B du robot. Pour ce qui est de l'alimentation, nous avons choisi d'utiliser une batterie lithium ion de 11.5V () nous permettant d'assurer une puissance suffisamment élevée tout en évitant les chutes de tension. Cette batterie servira aussi à alimenter la carte Nucléo donnant ainsi au robot Véronica une totale autonomie. Le schéma bloc impliquant la partie liée à la batterie du montage est le suivant :



Cependant, une amélioration du prototype est envisagée.

En effet, pour garantir un guidage encore plus précis du robot, il faudrait mettre en place un asservissement des moteurs. En effet, chaque moteur a des coefficients de friction différents et réagit différemment en fonction du niveau de charge de la batterie. Autrement dit, même avec une commande identique, les moteurs vont réagir différemment et le robot risque de ne pas suivre exactement la trajectoire prévue. Heureusement, chaque moteur est doté de deux signaux (signal A et signal B) nous renseignant sur la vitesse de rotation, la position et le sens de rotation. Nous avons donc pensé à mettre en place un asservissement avec un correcteur proportionnel intégral dérivé (PID) (nous avons premièrement commencé uniquement avec un correcteur proportionnel qui n'était alors pas satisfaisant, puisque des emballements des moteurs pouvaient se produire avant de se diriger vers un asservissement plus complet, pas encore satisfaisant à l'issue du projet mais en bonne voie). Cette voie d'amélioration explorée nous semble très prometteuse pour le robot, et pourrait résoudre les problèmes de déplacement du robot Véronica précédemment détaillés.

2) Relevés de température

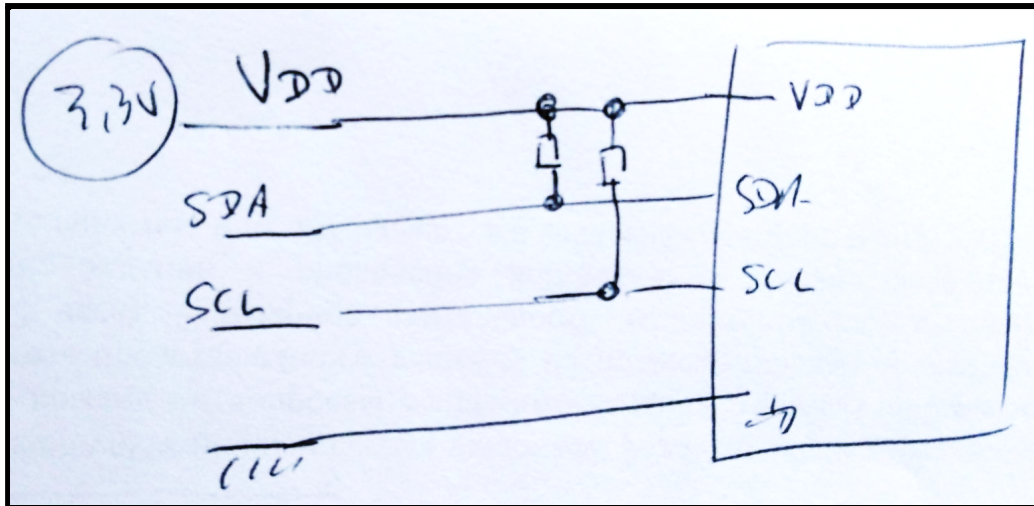
Pour les capteurs, nous nous sommes restreints à acquérir les données de température, vu que nous n'avions pas à disposition un capteur d'humidité, mais le principe est le même.

Nous avons utilisé une carte I2C (Inter Integrated Circuit Bus) qui permet de relier facilement un microprocesseur et différents circuits, notamment un capteur de température.

La carte utilisée est de référence *MCP 9800 PICKit Serial I2C Demo Board*. (fiche technique en annexe 1)

Son constructeur indique que le capteur peut mesurer des températures de -40°C à 125°C avec $\pm 0.25^{\circ}\text{C}$ de précision.

Afin de l'utiliser avec la carte Nucléo, on la branche aux deux pôles SDA et SCL, avec 2 résistances de $1\text{k}\Omega$, comme indiqué sur le schéma suivant :



Le code utilisé pour récupérer les mesures de température est donné en annexe.

3) Communication entre les deux cartes

Afin de faire communiquer les cartes entre elles, nous avons utilisé un émetteur-récepteur radiofréquence par console *RF Solutions KAPPA-M868 111-0159*. (fiche technique en annexe 2)

Pour les utiliser sur les consoles Nucléo L 476-RG, on réalise le schéma électrique suivant où $R1 = 1\text{ k}\Omega$ et $R2 = 56\text{ k}\Omega$.

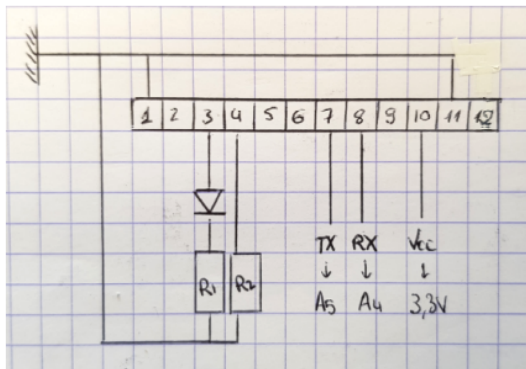


Figure 1 - Schéma électrique de la connexion entre le module et la carte Nucléo

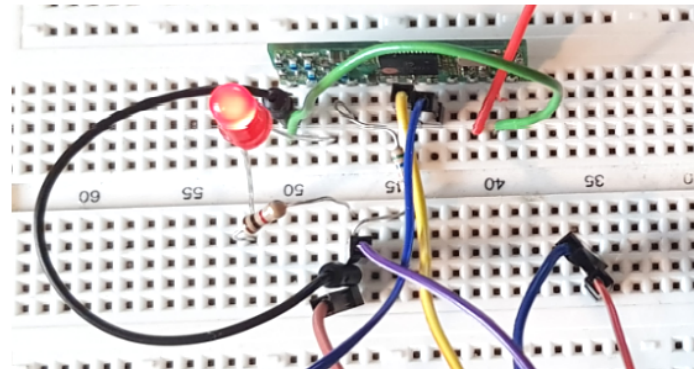


Figure 2 - Photo des branchements du module à la carte Nucléo

Ces modules fonctionnent à 19 200 bauds, donc pour éviter tout problème d'horloge, on règle le fonctionnement de la carte Nucléo et du terminal TeraTerm à 19 200 bauds également.

Le code utilisé pour l'envoi et la réception des données est donné en annexe

4) Communication avec l'utilisateur

L'objectif du cahier des charges est de réaliser une interface graphique simple d'utilisation

Etat du projet et points d'amélioration

Tous nos objectifs ont été atteints : nous avons réussi à piloter le moteur en translation et en rotation et à tracer sa trajectoire, nous avons pu récupérer fiablement des mesures de température à intervalles de temps réguliers, nous avons fait communiquer les deux cartes entre elles, à la fois pour envoyer les commandes au moteur, et pour recevoir les mesures du capteur, et enfin, nous avons réalisé une interface graphique permettant d'entrer la commande du moteur, d'afficher sa trajectoire, et d'afficher les mesures de température.

Cependant, notre pilotage du moteur n'est pas encore parfait, car la trajectoire réellement effectuée ne correspond pas parfaitement à la commande. En effet, nous observons qu'au bout d'un certain temps, la vitesse du moteur diminue, ou les deux roues ne vont pas à la même vitesse. Ces problèmes pourraient être résolus par un asservissement des roues du moteur, chose que nous avons commencé pendant plusieurs séances mais pas fini faute de temps. Un autre point d'amélioration serait d'optimiser le montage de notre prototype. Notre robot actuel est assez encombré avec les différentes cartes, les fils et la batterie ; en plus, la petite roue arrière n'est pas bien fixée et influence sur la direction d'avancement et sur la rotation du robot.

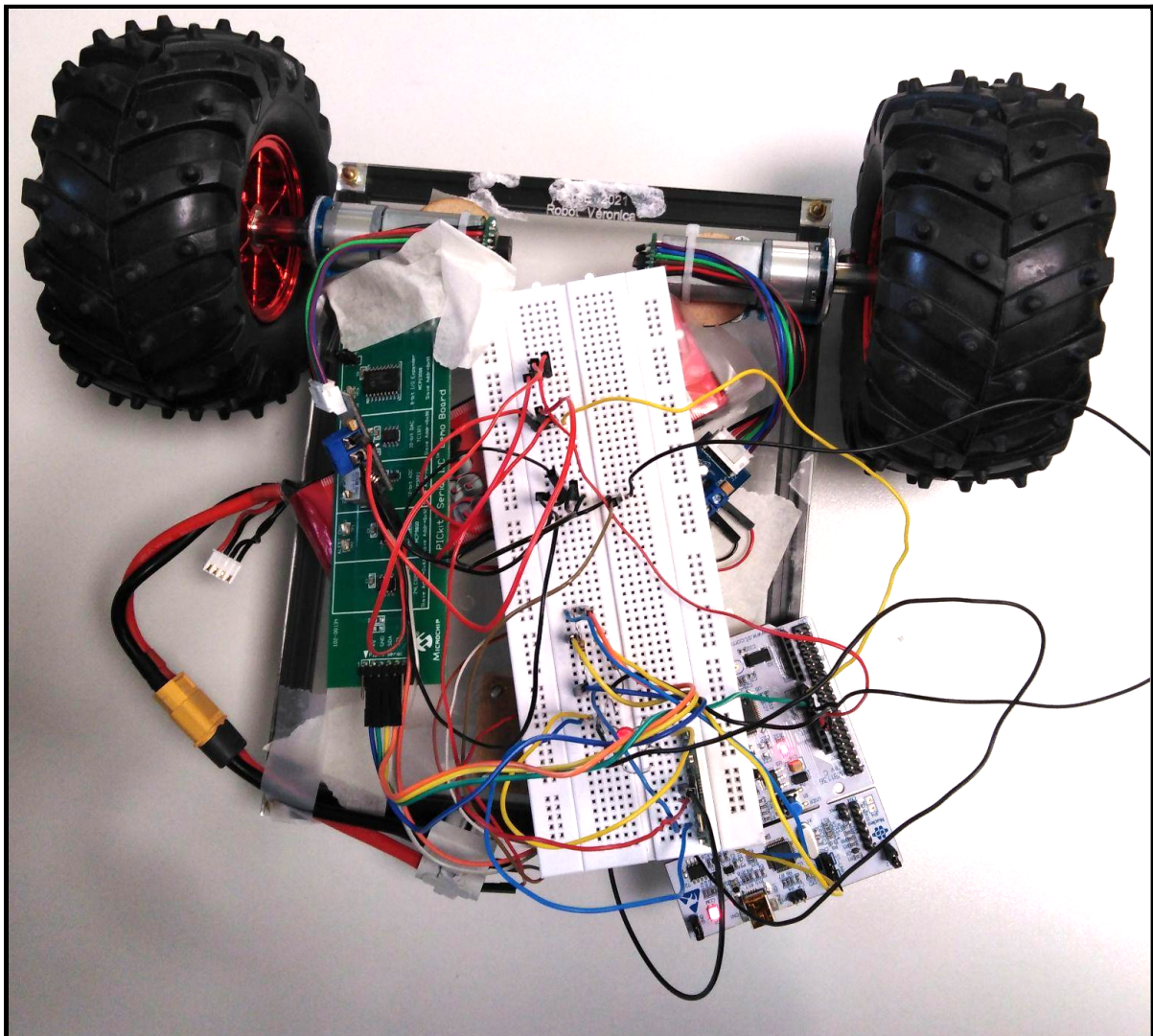


Image du prototype final du robot

Conclusion et remerciements

En somme, ce projet a été très enrichissant pour notre équipe.

Non seulement nous avons acquis plusieurs compétences techniques, mais nous avons surtout pu développer un bon esprit d'équipe, en nous répartissant les tâches tout en restant en communication permanente grâce aux différents outils de travail collaboratif que nous avons utilisés (Microsoft Teams et Google Workspace). Nous avons également vaincu la peur de réaliser un projet du début à la fin et appris à nous débrouiller pour trouver des solutions aux problèmes rencontrés. Toutefois, nous tenons à remercier chaleureusement Mme. Bernard, M. Villemejeane et M. Benisty, M. Hecquet et M. Adam pour leur aide tout au long de la réalisation du prototype.

Merci pour votre dévouement depuis le processus de brainstorming jusqu'au test final.

Annexes

Annexe 0 : Code final du projet

→ Captures d'écran (aperçu global du code)

```

1 /* mbed Microcontroller Library
2  * Copyright (c) 2019 ARM Limited
3  * SPDX-License-Identifier: Apache-2.0
4  */
5
6 // programme qui lit et transmet la température
7
8 #include "mbed.h"
9 #include "platform/mbed_thread.h"
10
11 // BRANCHEMENTS
12
13 Serial communication(D8, D2);
14 Serial pc(USBTX, USBRX);
15 PwmOut moteur_dir_g(D12);
16 PwmOut moteur_en_g(D11);
17 PwmOut moteur_dir_d(D9);
18 PwmOut moteur_en_d(D10);
19 I2C capteur(I2C_SDA, I2C_SCL); //déclaration
20
21 // VARIABLES
22
23 double rcg=0.3; // rapport cyclique du moteur gauche
24 double rcd=0.3; // rapport cyclique du moteur droit
25 int sens_MG=0; // variable valant 0,-1 ou 1 selon le sens de rotation (ou non)
26 int sens_MD=0; // variable valant 0,-1 ou 1 selon le sens de rotation (ou non)
27
28 const int addr = 0x92;
29 char instruct[100];
30 char c=' ';
31 int i=0;
32 int av; // variable contenant la distance dont on veut avancer
33 int tourn; // variable contenant l'angle dont on veut tourner
34
35 // FONCTIONS
36
37 void rotation_moteurs(){ // fonction permettant de mettre en marche les moteurs en fonction du sens de rotation et de la vitesse voulus
38     if (sens_MG==1){ // moteur gauche entraîne la roue vers l'avant avec une vitesse de rotation associée au rapport cyclique rcg
39         moteur_dir_g.write(1);
40         moteur_en_g.write(rcg);
41     }
42     if (sens_MG==-1){ // moteur gauche entraîne la roue vers l'arrière
43         moteur_dir_g.write(0);
44         moteur_en_g.write(rcg);
45     }

```

```

46     if (sens_MG==0){ // roue gauche immobile
47         moteur_dir_g.write(0);
48         moteur_en_g.write(0);
49     }

50     if (sens_MD==-1){ // moteur droit entraine la roue vers l'avant
51         moteur_dir_d.write(1);
52         moteur_en_d.write(rcd);
53     }
54     if (sens_MD==1){ // moteur droit entraine la roue vers l'arriere
55         moteur_dir_d.write(0);
56         moteur_en_d.write(rcd);
57     }
58     if (sens_MD==0){ // roue droite immobile
59         moteur_dir_d.write(0);
60         moteur_en_d.write(0);
61     }
62 }
63
64 void avancer(double distance){ // fait avancer le robot de la distance algebrigue "distance"
65     int k=1000000;
66     if (distance<0){ // robot recule
67         rcg=0.24;
68         rcd=0.24;
69         sens_MD=1;
70         sens_MG=1;
71         rotation_moteurs();
72         wait_us(k*distance);
73         sens_MD=0;
74         sens_MG=0;
75         rotation_moteurs();
76     }
77     if (distance>0){ // robot avance
78         rcg=0.24;
79         rcd=0.24;
80         sens_MD=-1;
81         sens_MG=-1;
82         rotation_moteurs();
83         wait_us(k*distance);
84         sens_MD=0;
85         sens_MG=0;
86         rotation_moteurs();
87     }
88 }
89
90 void tourner(double angle){ //fait tourner le robot de l'angle algebrigue "angle"
91     int k=1000000;
92     if (angle>0){ // rotation gauche
93         rcg=0.24;
94         rcd=0.24;

```



```

95     sens_MD=1;
96     sens_MG=-1;
97     rotation_moteurs();
98     wait_us(k*angle);
99     sens_MD=0;
100    sens_MG=0;
101    rotation_moteurs();
102  }
103  if (angle<0){ // rotation droite
104    rcg=0.24;
105    rcd=0.24;
106    sens_MD=-1;
107    sens_MG=1;
108    rotation_moteurs();
109    wait_us(k*angle);
110    sens_MD=0;
111    sens_MG=0;
112    rotation_moteurs();
113  }
114 }
115
116 void temperature(){ // relevés de température
117   char cmd[2];
118   cmd[0] = 0x00;
119   capteur.write(addr, cmd, 1);
120   capteur.read(addr, cmd, 2);
121   // wait(600); // transmet l'information toutes les 10 min
122   pc.printf("Temp = %d\n", cmd[0]);
123
124   communication.putc(cmd[0]); // transmet la température à l'interface graphique
125 }
126
127 double angle_vers_temps(int tourn){ // donne le temps de rotation en fonction de l'angle de rotation commandé
128   return (tourn + 18.4) /55.1;
129 }
130
131 // MAIN
132
133 int main()
134 {
135   pc.baud(115200);
136   communication.baud(19200); // fréquence de communication avec l'ordinateur
137   capteur.frequency(100000); // fréquence de transmission
138
139   while (1) {

```

```

129     }
130
131 // MAIN
132
133 int main()
134 {
135     pc.baud(115200);
136     communication.baud(19200); // fréquence de communication avec l'ordinateur
137     capteur.frequency(100000); // fréquence de transmission
138
139     while (1) {
140
141         while(c!='F') {
142             c = communication.getc();
143             instruct[i]=c; //instruct contient les instructions pour le robot sous forme d'une liste de caractères
144             i++;}
145
146         for (int j=0;100;j++){ // boucle pour lire les instructions dans instruct[j]
147             if (instruct[j]=='A'){ // le robot avance (A)
148                 j=j+1;
149                 av=instruct[j]-'0';
150                 temperature();
151                 for (int k = 0 ;k<av ;k++){
152                     avancer(2.30);
153                     temperature(); // un relevé de température est réalisé tous les mètres
154                     wait(3);
155                 }
156             }
157             // wait_us(1000000); // un wait pour attendre que le robot s'arrete complètement avant de tourner
158             if (instruct[j]=='T'){ // le robot tourne (T)
159                 tourn=10*(instruct[j+1]-'0')+(instruct[j+2]-'0');
160                 j=j+2;
161                 int t=angle_vers_temps(tourn);
162                 tourner(t);
163                 temperature();
164                 wait(1);
165             }
166         }
167         c=' ';
168     }
169 }
170 //explications V
171 //putc un caractère >127 avant cmd[0] pour indiquer valeur 1
172 //if (communication.readable(<127) ne pas lire
173 //else stocker la valeur 1

```

→ Code brut

```

/* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */

// programme qui lit et transmet la tempÃ©rature

#include "mbed.h"
#include "platform/mbed_thread.h"

// BRANCHEMENTS

Serial communication(D8, D2);
Serial pc(USBTX, USBRX);
PwmOut moteur_dir_g(D12);
PwmOut moteur_en_g(D11);
PwmOut moteur_dir_d(D9);
PwmOut moteur_en_d(D10);
I2C capteur(I2C_SDA, I2C_SCL); //dÃ©claration

// VARIABLES

double rcg=0.3; // rapport cyclique du moteur gauche
double rcd=0.3; // rapport cyclique du moteur droit
int sens_MG=0; // variable valant 0,-1 ou 1 selon le sens de rotation (ou non) souhaitÃ© du
moteur gauche, 1 pour que le moteur entraÃªne la roue vers l'avant
int sens_MD=0; // variable valant 0,-1 ou 1 selon le sens de rotation (ou non) souhaitÃ© du
moteur droit, 1 pour que le moteur entraÃªne la roue vers l'avant

const int addr = 0x92;
char instruct[100];
char c=' ';
int i=0;
int av; // variable contenant la distance dont on veut avancer
int tourn;// variable contenant l'angle dont on veut tourner

// FONCTIONS

void rotation_moteurs(){ // fonction permettant de mettre en marche les moteurs en fonction
du sens de rotation et de la vitesse voulu
    if (sens_MG==1){ // moteur gauche entraÃªne la roue vers l'avant avec une vitesse
de rotation associÃ©e au rapport cyclique rcg
        moteur_dir_g.write(1);
        moteur_en_g.write(rcg);
    }
    if (sens_MG==-1){ // moteur gauche entraÃªne la roue vers l'arriÃ¨re

```

```

moteur_dir_g.write(0);
moteur_en_g.write(rcg);
}
if (sens_MG==0){ // roue gauche immobile
moteur_dir_g.write(0);
moteur_en_g.write(0);
}
if (sens_MD==-1){ // moteur droit entraîne la roue vers l'avant
moteur_dir_d.write(1);
moteur_en_d.write(rcd);
}
if (sens_MD==1){ // moteur droit entraîne la roue vers l'arrière
moteur_dir_d.write(0);
moteur_en_d.write(rcd);
}
if (sens_MD==0){ // roue droite immobile
moteur_dir_d.write(0);
moteur_en_d.write(0);
}
}
}

```

```

void avancer(double distance){ // fait avancer le robot de la distance algebrique "distance"
int k=1000000;
if (distance<0){ // robot recule
rcg=0.24;
rcd=0.24;
sens_MD=1;
sens_MG=1;
rotation_moteurs();
wait_us(k*distance);
sens_MD=0;
sens_MG=0;
rotation_moteurs();
}
if (distance>0){ // robot avance
rcg=0.24;
rcd=0.24;
sens_MD=-1;
sens_MG=-1;
rotation_moteurs();
wait_us(k*distance);
sens_MD=0;
sens_MG=0;
rotation_moteurs();
}
}

```

```

void tourner(double angle){ //fait tourner le robot de l'angle algebrique "angle"

```

```

    int k=1000000;
    if (angle>0){ // rotation gauche
    rcg=0.24;
    rcd=0.24;
    sens_MD=1;
    sens_MG=-1;
    rotation_moteurs();
    wait_us(k*angle);
    sens_MD=0;
    sens_MG=0;
    rotation_moteurs();
    }
    if (angle<0){ // rotation droite
    rcg=0.24;
    rcd=0.24;
    sens_MD=-1;
    sens_MG=1;
    rotation_moteurs();
    wait_us(k*angle);
    sens_MD=0;
    sens_MG=0;
    rotation_moteurs();
    }
}

void temperature(){ // relevés de température
    char cmd[2];
    cmd[0] = 0x00;
    capteur.write(addr, cmd, 1);
    capteur.read(addr, cmd, 2);
    // wait(600); // transmet l'information toutes les 10 min
    pc.printf("Temp = %d\n", cmd[0]);

    communication.putc(cmd[0]); // transmet la température à l'interface graphique
}

double angle_vers_temps(int tourn){ // donne le temps de rotation en fonction de l'angle de
rotation commandé
    return (tourn + 18.4) /55.1;
}

// MAIN

int main()
{
    pc.baud(115200);
    communication.baud(19200); // fréquence de communication avec l'ordinateur
    capteur.frequency(100000); // fréquence de transmission

```



```

while (1) {

while(c!='F') {
c = communication.getc();
instruct[i]=c; //instruct contient les instructions pour le robot sous forme d'une liste de
caractères
i++;}

for (int j=0;100;j++) { // boucle pour lire les instructions dans instruct[j]
if (instruct[j]=='A'){ // le robot avance (A)
j=j+1;
av=instruct[j]-'0';
temperature();
for (int k = 0 ;k<av ;k++){
avancer(2.30);
temperature(); // un relevé de température est réalisé tous les mètres
wait(3);
}
}
// wait_us(1000000); // un wait pour attendre que le robot s'arrete complètement
avant de tourner
if (instruct[j]=='T'){ // le robot tourne (T)
tourn=10*(instruct[j+1]-'0')+(instruct[j+2]-'0');
j=j+2;
int t=angle_vers_temps(tourn);
tourner(t);
temperature();
wait(1);
}
}
c=' ';
}
}
//explications V
//putc un caractère >127 avant cmd[0] pour indiquer valeur 1
//if (communication.readable())<127) ne pas lire
//else stocker la valeur 1

```

Annexe 1 : Fiche technique de la carte I2C

[PKSERIAL-I2C1 Microchip](#)

Annexe 2 : Fiche technique du module émetteur-récepteur

[KAPPA-M868](#)