

Concevoir-Vérifier

Dossier de fabrication

Lucas COQUARD

Table des matières

1. Introduction.....	2
2. Réalisation du PCB.....	3
3. Nomenclature des composants.....	5
4. Programme de test.....	5
4.1 Ouverture du port série.....	5
4.2 Acquisition des messages	6
4.2.1 Arduino	6
4.2.2 Qt Creator.....	7
4.3 Partie Wi-Fi.....	7
4.4 Partie Capteurs.....	9
4.5 Partie relais.....	10
4.6 Partie écran TFT	11
4.7 Partie RFID.....	12
5. Procédure de test.....	13
6. Conclusion	14
Annexe 1 : Fiche suiveuse d'une carte d'assistant domotique.....	15
Annexe 2 : Glossaire	20
Annexe 3 : Nomenclature des composants.....	21

1. Introduction

Ce dossier de fabrication se place dans le cadre de la SAE3, consistant à concevoir un assistant domotique. Dans la première partie de cette dernière, un prototype fonctionnel avait été réalisé. Cependant, en l'état, ce dernier n'était pas commercialisable. En effet, aucune procédure de maintenance n'avait été mise en place. De plus, sans son PCB, la carte n'était pas productible.

Dans ce document, toutes les informations relatives à la réalisation du PCB et des programmes de test ainsi qu'à la procédure de test ont été explicitées.

2. Réalisation du PCB

Comme dit en introduction, pour pouvoir produire cette carte à grande échelle, il est nécessaire d'avoir en sa possession les fichiers Gerber associés à la carte. Afin de les générer, le PCB de cette dernière doit être réalisé. Ici, contrairement au PCB, les fichiers n'ont pas été générés, car dans le cadre du projet, ils n'étaient pas demandés. Ci-dessous, le schéma fonctionnel qui avait été établi sur KiCad est rappelé.

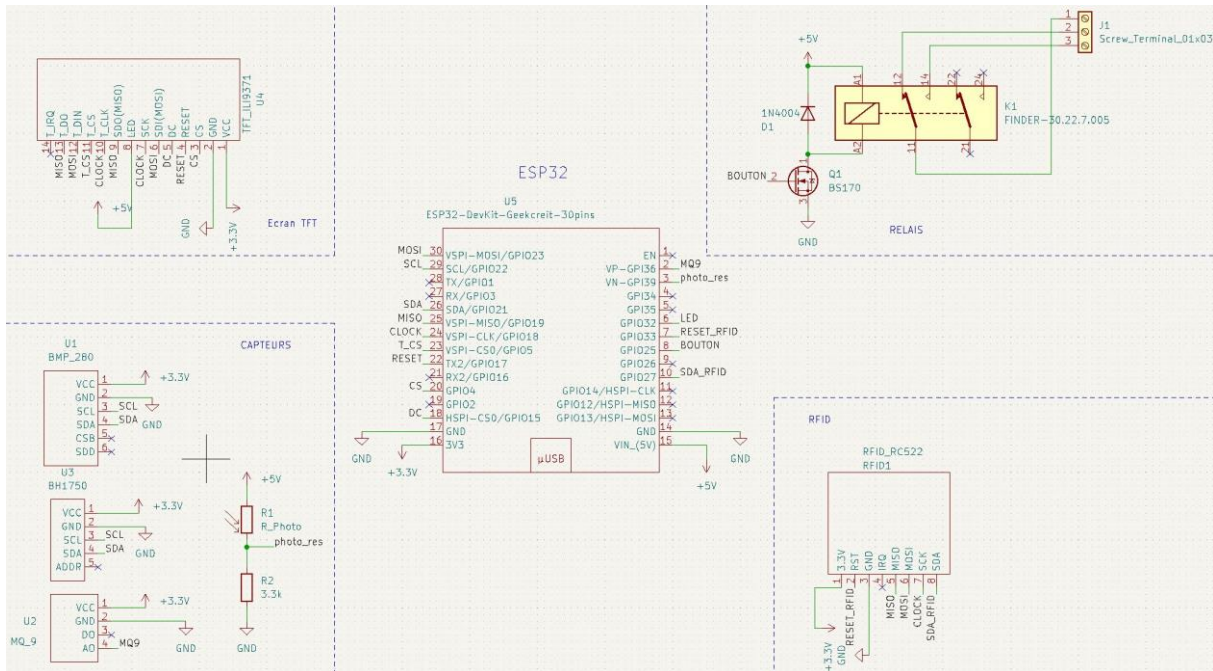


Figure 1 – Schéma fonctionnel de la carte électronique

En partant de ce schéma, le PCB suivant a été établi, en prenant des dimensions de 16 cm x 10 cm :

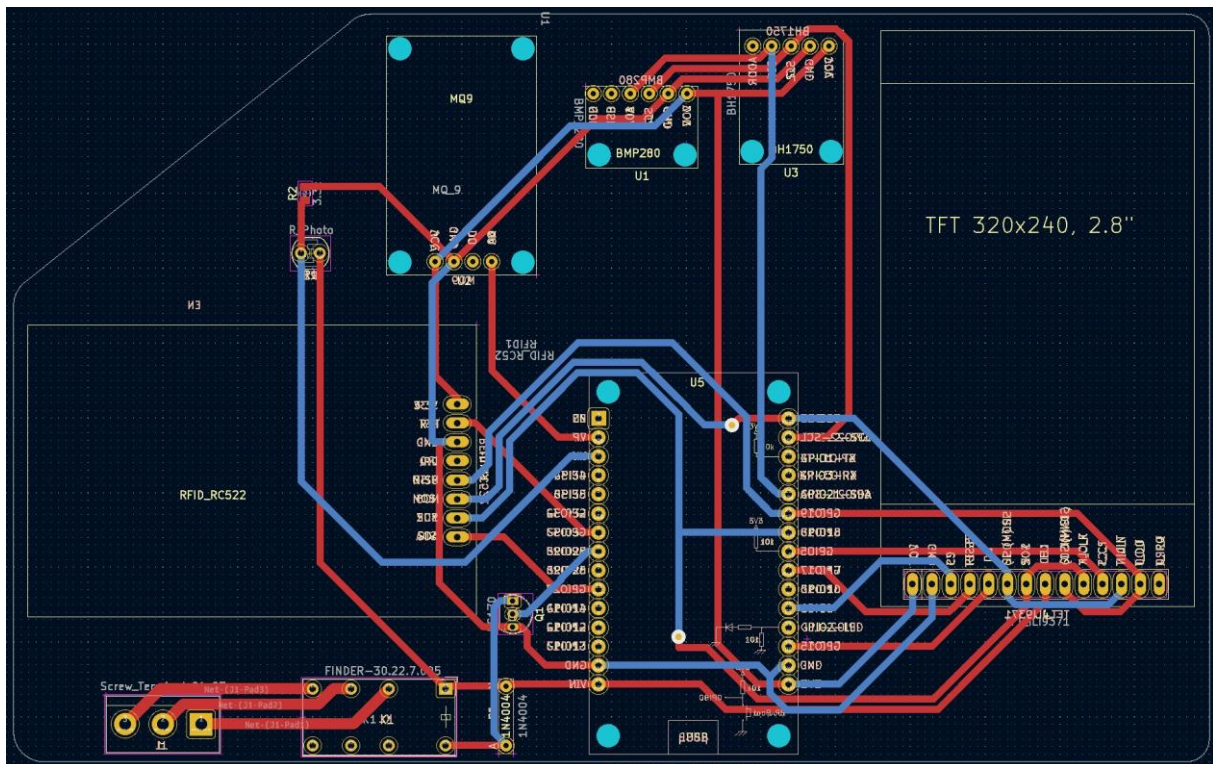


Figure 2 – PCB de la carte électronique

Pour le réaliser, deux tailles de pistes différentes ont été utilisées. Pour la grande majorité des liaisons, les pistes sont de taille 1 mm, car le courant qui y passe n'est pas très important. Cependant, pour les pistes reliant le relais et le bornier à vis, le courant peut monter jusqu'à 2 A. Pour un courant de cette valeur, la norme veut que les pistes soient au minimum de 1.3 mm. Ici, il a été décidé de prendre 1.5 mm.

De plus, pour faciliter la phase de test qui sera abordée plus tard, la face arrière de la carte est sérigraphiée (voir Figure 4) afin d'éviter au technicien d'avoir à retourner la carte sans cesse.

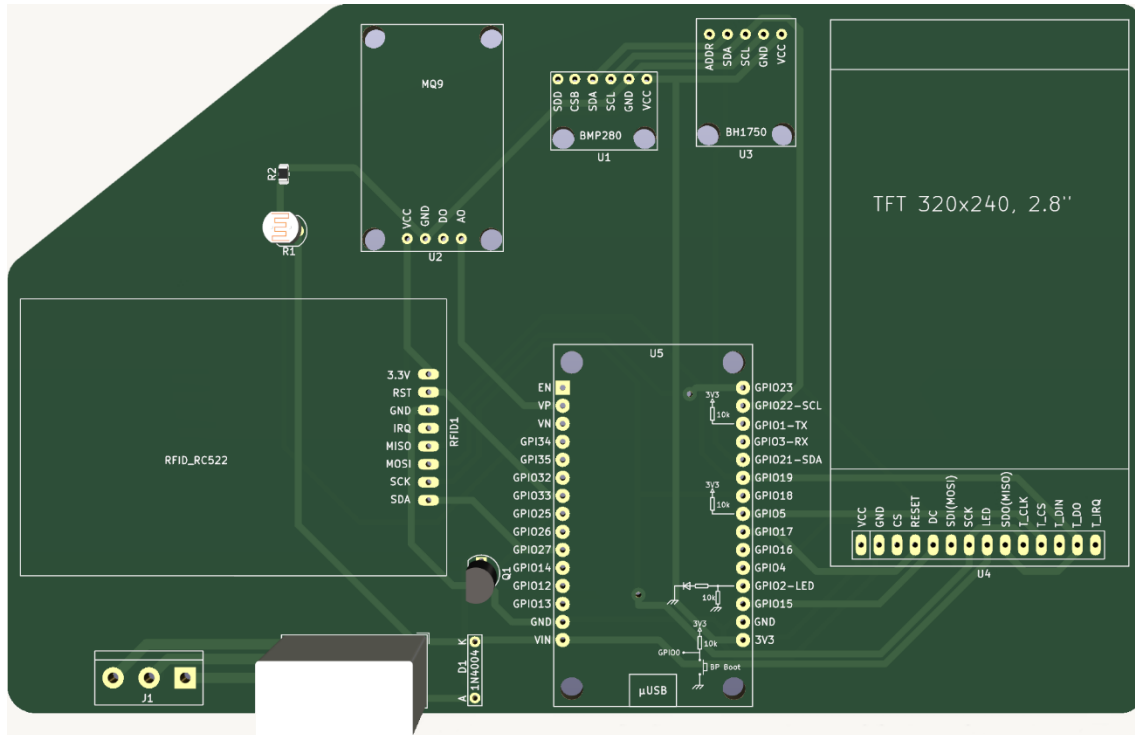


Figure 3 – Visu 3D de la face avant de la carte

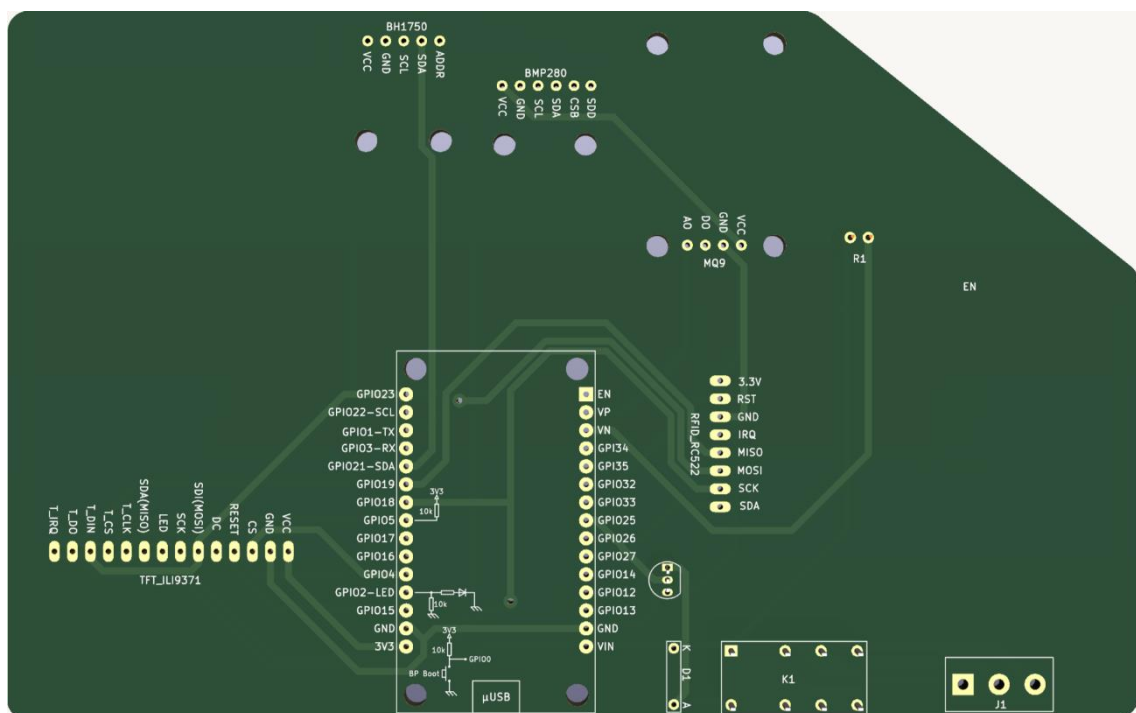


Figure 4 – Visu 3D de la face arrière de la carte

3. Nomenclature des composants

La réalisation d'une nomenclature des composants lors de la fabrication d'une carte électronique offre des avantages tels que la traçabilité, la gestion des stocks, la précision de l'assemblage, l'analyse des coûts, la facilitation de la maintenance, la collaboration entre équipes, et une documentation complète. Cela contribue à améliorer l'efficacité de la production et la qualité du produit final. Ce document se trouve en Annexe 3 : Nomenclature des composants.

4. Programme de test

Pour rendre la procédure de test moins contraignante, une interface graphique a été codée sur Qt Creator en plus du programme principal sur Arduino. Une fois lancés, les deux programmes vont directement communiquer. L'appui de chaque bouton envoie des instructions précises à la partie Arduino qui renvoie les informations demandées.

Dans cette partie du compte rendu, certaines parties de programme ne seront pas expliquées, car elles l'ont déjà été dans le dossier de conception, ou alors, elles n'ont pas été jugées pertinentes ou nécessitant des explications supplémentaires.

4.1 Ouverture du port série

En lançant l'interface pour la première fois, le technicien arrive sur cette page.

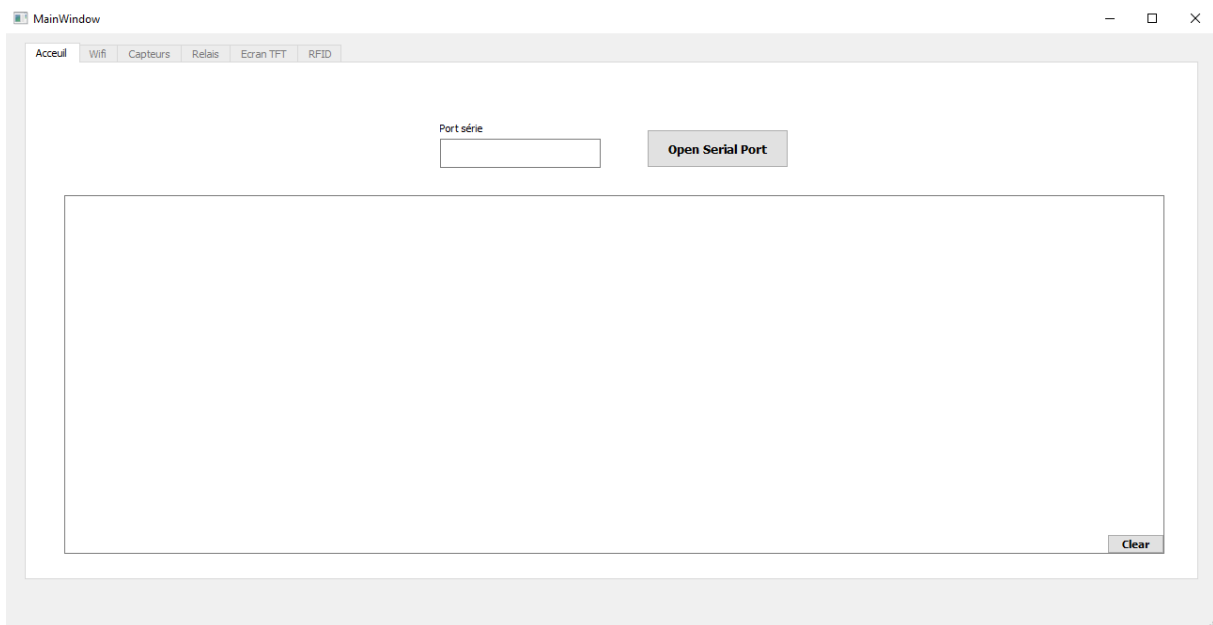


Figure 5 – Fenêtre « Accueil » de l'interface

Ici, il peut ouvrir le port série de son choix en rentrant le nom de ce dernier dans la ligne prévue à cet effet. Si le port s'ouvre correctement, les autres fenêtres lui sont rendues accessibles. Sinon, elles restent désactivées tant qu'aucun port série n'a été ouvert correctement. De plus, l'ouverture du port série a été codée pour que, lorsque qu'un autre port est ouvert via l'application, le précédent se ferme, permettant de le rouvrir plus tard. Enfin, dans l'item « textBox », toutes les informations passant sur le port série sont écrites de sorte que le technicien puisse contrôler facilement les flux de données.

4.2 Acquisition des messages

4.2.1 Arduino

La partie cruciale du programme réside dans l'acquisition des données envoyées sur le port série via l'interface.

```
while (Serial.available() > 0) {  
    // read the incoming byte:  
    incomingByte = Serial.read();  
    if (incomingByte != 10) {  
        msg += incomingByte ;  
    }  
    else {  
        ...  
    }  
}
```

En utilisant cette fonction, tant que quelque chose est lisible sur le port série, les caractères sont ajoutés un par un à la chaîne msg, du moment qu'ils sont différents de "\n". Ici, un while a été utilisé plutôt qu'un if pour pallier les problèmes de latence. Étant donné que les messages transmis sont courts, même si la fonction est bloquante, cela ne pose pas de problèmes majeurs.

Une fois récupérés, les messages doivent être traités pour effectuer des actions spécifiques.

```
sep = msg.indexOf(":");  
if (sep == -1) {  
    commande = msg ;  
    // Serial.println((String)"commande : " + commande);  
}  
else {  
    commande = msg.substring(0, sep);  
    action_Associee = msg.substring(sep + 1, msg.indexOf(":", sep + 1)) ;  
    if (msg.indexOf(":", sep + 1) == -1) {  
        reste_Commande = "" ;  
    }  
    else {  
        reste_Commande = msg.substring(msg.indexOf(":", sep + 1) + 1) ;  
    }  
}
```

Pour communiquer, le séparateur choisi est ":" et les messages prennent la forme suivante :

commande: action_Associee: reste_Commande

Il faut donc commencer par trouver la position des deux points, valeur stockée dans la variable « sep ». Cependant, certaines commandes sont constituées d'un seul mot, dans ce cas le séparateur va prendre la valeur -1. Dans ce cas-là, la commande prendra directement la valeur du message. Sinon, le message est séparé en deux parties. La première, « commande », qui va indiquer quel composant ou fonctionnalité de la carte est concerné par la commande. La seconde, « action_Associee », qui va cette fois déterminer l'information demandée.

De plus, certaines fonctionnalités requièrent plus de précisions dans la requête, c'est pourquoi le reste de la chaîne, s'il y en a une, est stockée dans une variable « reste_Commande » qui sera traitée au cas par cas.

4.2.2 Qt Creator

Étant donné que les deux programmes communiquent, l'interface doit aussi acquérir correctement les données envoyées par le programme Arduino. Comme pour le programme précédent, ce sont les deux points qui sont utilisés comme séparateurs.

```
if(port->canReadLine()) {  
    port->readLine(ch,200);  
    ui->textEdit_Serial_Port->append(ch) ;  
    QString msg=ch;  
    QStringList maListeValeurs = msg.split(":") ;  
    for ( const auto& i : maListeValeurs ) {  
        ...  
    }  
}
```

Avec ce programme, Qt Creator divise directement toute la chaîne de caractères dans un tableau. En utilisant un « for » et en connaissant les messages envoyés, il suffira d'utiliser une machine à état pour afficher les informations aux bons endroits.

4.3 Partie Wi-Fi

Après avoir ouvert le port série, les autres fenêtres sont accessibles. Le technicien passe ainsi sur la fenêtre « Wi-Fi ».

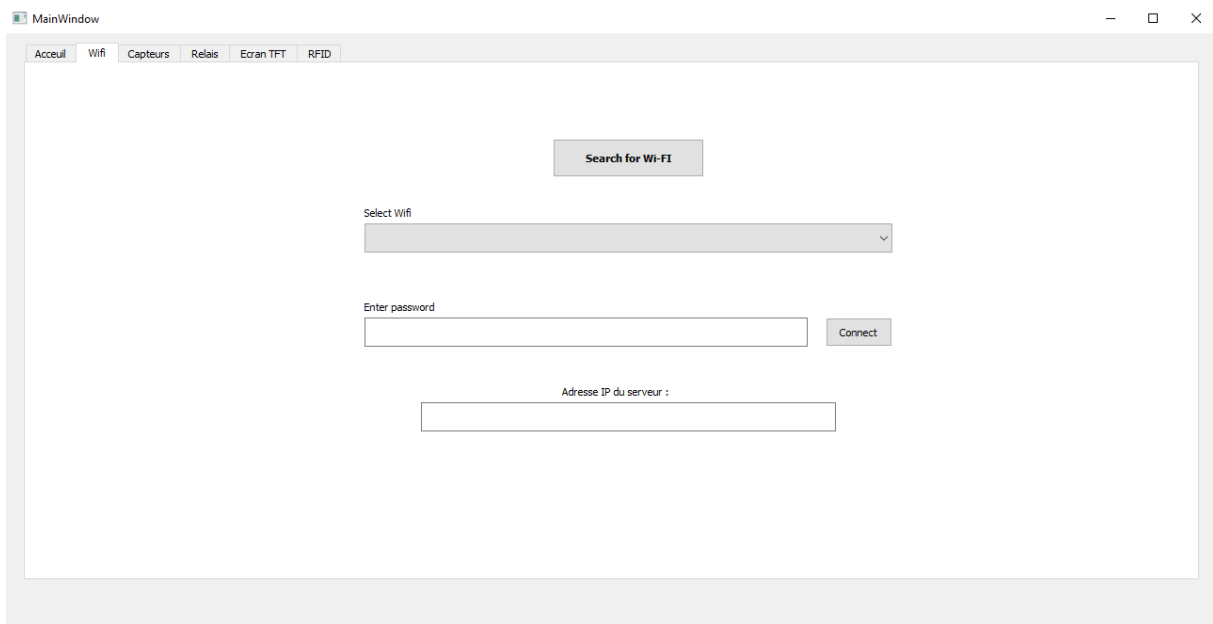


Figure 6 – Fenêtre « Wifi » de l'interface

En cliquant sur le bouton « Search for Wi-Fi », la commande « WIFI :SCAN » est envoyée sur le port série. La fonction suivante s'exécute alors :

```
int n = WiFi.scanNetworks();
if (n == 0) {
    Serial.printf("no networks found\n");
} else {
    for (int i = 0; i < n; ++i) {
        wifi_List += WiFi.SSID(i).c_str() ;
        wifi_List += ":" ;
        tab_Wifi[i] = WiFi.SSID(i).c_str() ;
    }
    tab_Wifi[n] = "\0" ;
    Serial.printf("WIFI:SSID:%s\n", wifi_List.c_str());
}
WiFi.scanDelete();
```

Ce code compte combien de Wifi sont actifs autour de l'ESP32. En utilisant un « for », volontairement bloquant, sinon le code ne s'exécute pas correctement, tous les SSID sont ajoutées à la fois dans un tableau et dans un String. Le String est ensuite envoyé sur le port série pour que le technicien choisisse le réseau auquel il souhaite se connecter.

Une fois la chaîne reçue, l'interface affiche tous les SSID dans une comboBox, dont l'index correspond à celui du tableau précédemment rempli. Le technicien sélectionne ainsi son Wifi et rentre le mot de passe.

En appuyant sur « Connect », le mot de passe ainsi que l'index du réseau choisi sont envoyés à l'ESP32 qui va pouvoir se connecter au réseau.

```
void MainWindow::on_pushButton_Connect_Wifi_clicked()
{
    QString msg ;
    msg.sprintf("WIFI:CONNECT:%d:%s\n", ui->comboBox_Select_Wifi-
>currentIndex(), ui->lineEdit_password->text().toString().c_str()) ;
    port->write(msg.toString().c_str()) ;
}
```

Toutes les secondes, il va essayer de se connecter au réseau. S'il a réussi à se connecter, il retourne l'IP du serveur créé pour qu'il soit affiché sur l'interface.

4.4 Partie Capteurs

La troisième fenêtre regroupe tous les capteurs de la carte. En effet, depuis ici, le technicien sera capable de tester tous les capteurs et de récupérer les adresses des capteurs concernés.

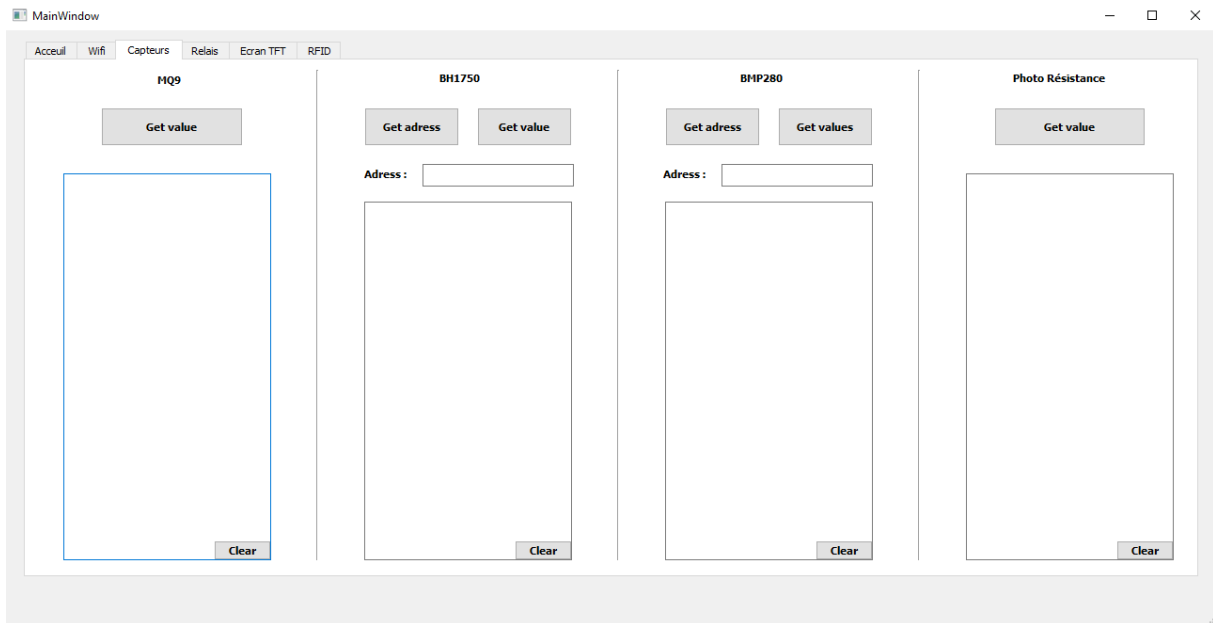


Figure 7 – Fenêtre « Capteurs » de l'interface

En cliquant sur le bouton associé, les fonctions suivantes vont être appelées afin de connaître les valeurs des capteurs :

```
void MQ9() {
    float vOut = analogRead(MQ9_AOUT) * (3.3 / 4095.0);
    int gasValue = 595 * pow((3.3 / vOut - 1) * (996 / 850), -2.24) ;
    Serial.printf("MQ9:%d ppm\n", gasValue);
}

void BMP280() {
    float temperature = bmp.readTemperature();
    float pression = bmp.readPressure() / 100 ;
    Serial.printf("BMP280:VALUE:%.2f hPa ; %.2f °C\n", pression,
temperature);
}

void BH() {
    float lux = lightMeter.readLightLevel();
    Serial.printf("BH1750:VALUE:%.2f lux\n", lux);
}

void RES() {
    float value = analogRead(R_PHOTO) * 3.3 / 4095.0 ;
    float photo_res = 72 * pow(value, 1.429) ;
    Serial.printf("RES:%.2f lux\n", photo_res);
}
```

Les valeurs s'affichent ensuite dans la case qui leur est attribuée. Avoir des zones de textes séparés permet de pouvoir vérifier la cohérence de ses valeurs avec les précédentes sans avoir à les noter quelque part.

Comme dit plus haut, certains capteurs possèdent des adresses. Il est important de les connaître pour être sûr que le programme de l'assistant domotique communique bien avec les capteurs. Lorsque le programme Qt demande les adresses, le code suivant s'exécute :

```
byte scan_address(bool composant) {
    byte error, address;
    byte adr ;
    if (composant == BMP) {
        ...
    }
    else if (composant == BH_1750) {
        for (address = 0x23; address < 0x5C; address++) {
            Wire.beginTransaction(address);
            error = Wire.endTransmission();
            if (error == 0) {
                adr = address ;
            }
        }
    }
    return adr ;
}
```

Selon le composant, le programme va chercher dans une certaine plage d'adresse, comprise entre leurs deux adresses possibles, si un composant I2C communique sur l'une d'elles, l'interface reçoit cette adresse et l'affiche.

4.5 Partie relais

Pour tester le relais, un bouton a été construit. Le bouton change d'état de manière dynamique en fonction de l'état du transistor qui commande.

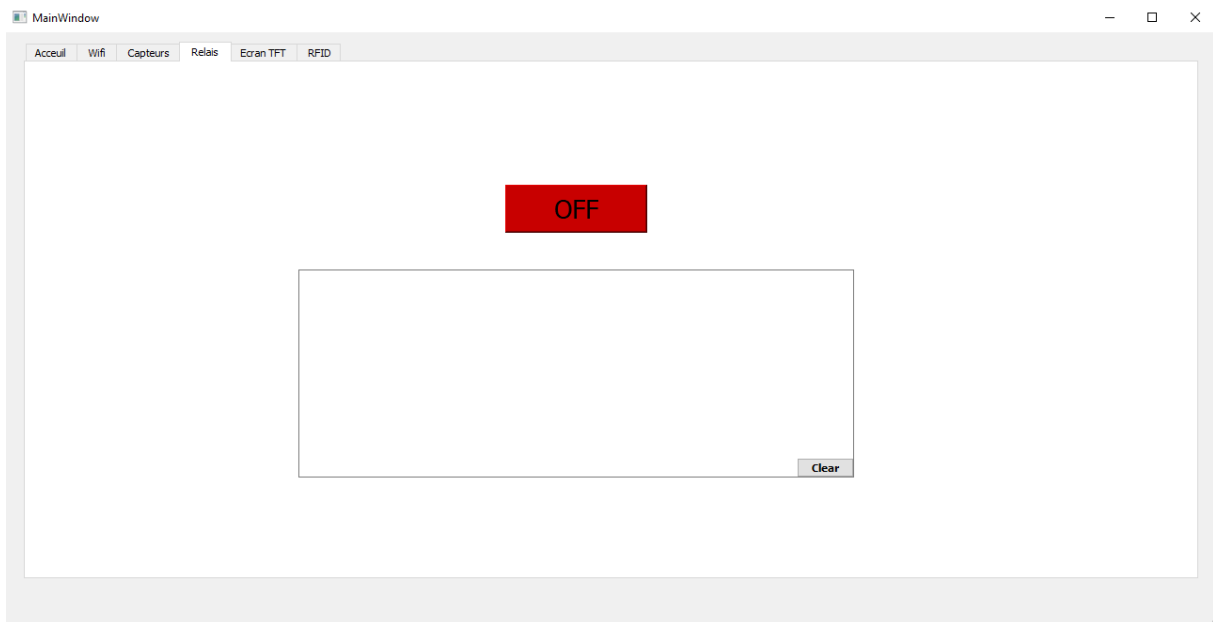


Figure 8 – Fenêtre « Relais » de l'interface

Lorsqu'il est pressé, il transmet la commande associée au nouvel état du bouton, ce qui va faire claquer le transistor. Pour vérifier que les deux programmes sont en phase, le programme Arduino retourne la valeur de relais de son point de vue. Cette dernière sera à comparer avec la valeur du bouton.

4.6 Partie écran TFT

Dans le cadre de son utilisation, l'écran doit assurer 3 fonctions principales :

- Pouvoir de différentes couleurs ;
- Être capable d'afficher toutes les lettres et tous les chiffres correctement ;
- Être tactile.



Figure 9 – Fenêtre « Écran TFT » de l'interface

Lorsqu'une commande de couleur est envoyée, la couleur du texte de l'écran change :

```
if (action_Associee == "RED") {  
    tft.setTextColor(TFT_RED, TFT_BLACK);  
}  
...  
else if (action_Associee == "INVERT") {  
    static bool invert = 0 ;  
    invert = !invert ;  
    tft.invertDisplay(invert);  
}
```

Si le technicien souhaite vérifier que les caractères s'affichent correctement :

```
if (action_Associee == "LETTRES") {  
    tft.println("abcdefghijklmnopqrstuvwxyz") ;  
}  
else if (action_Associee == "CHIFFRES") {  
    tft.println("0 1 2 3 4 5 6 7 8 9") ;  
}
```

Enfin, pour tester la fonction tactile, la fonction ci-dessous, permet de dessiner un rond là où l'écran a été appuyé, à partir des coordonnées, lorsque le bouton « Touch test » est activé :

```

if (touch == 1) {
    uint16_t x = 0, y = 0;
    bool pressed = tft.getTouch(&x, &y);
    if (pressed) {
        tft.fillCircle(x, y, 2, TFT_WHITE);
        i = 0 ;
    }
}

```

En utilisant les coordonnées du touché, ce code dessine un cercle de rayon 2 sur le point. Si, dans une zone de l'écran, aucun rond ne s'affiche, l'écran est défectueux.

4.7 Partie RFID

Le lecteur RFID, utilisé pour activer/désactiver le relais, doit être capable de lire toutes les cartes RFID, surtout celles qui sont autorisées à rentrer.

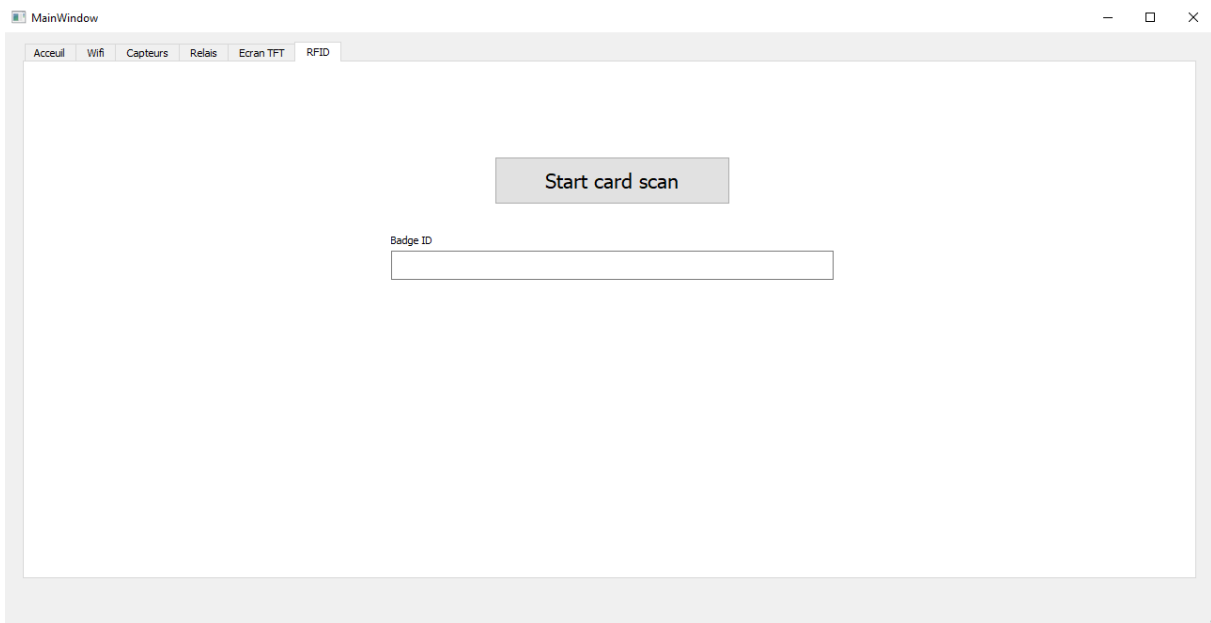


Figure 10 – Fenêtre « RFID » de l'interface

Lorsque le bouton « Start card scan » est pressé, le module RFID passe en mode lecture de carte.

```
void RFID () {
    String strBadgeID = "";
    for (int i = 0; i < 1000 and strBadgeID == ""; i++) {
        if ( !rfid.PICC_IsNewCardPresent() or !rfid.PICC_ReadCardSerial() )
        { }
        else {
            for (byte i = 0; i < 4; i++)
            {
                if (rfid.uid.uidByte[i] < 0x10) {
                    strBadgeID += '0';
                }
                strBadgeID += String(rfid.uid.uidByte[i], HEX);
            }
        }
        delay(10) ;
    }
    if (strBadgeID == "") {
        Serial.printf("RFID:No badge found\n");
    }
    else {
        Serial.printf("RFID:%s\n", strBadgeID.c_str());
    }
}
```

Au bout de 10 secondes où une fois qu'un tag est détecté, la boucle se termine et le programme renvoie l'ID d'une badge ou un message d'erreur le cas échéant.

5. Procédure de test

Le fonctionnement maintenant explicité, passons à la manière dont le technicien devra procéder. Ici sera expliqué, comment le technicien doit procéder pour remplir correctement la feuille, mais aussi les choses à faire en fonction de certains résultats.

Pour commencer, il faut manipuler la carte hors tension. Étant hors tension, c'est à ce moment que tous les tests de continuité sont effectués. Si les tests de continuité ne sont pas bons. Il faut reprendre la feuille depuis le début et vérifier que toutes les soudures sont correctes et que toutes les broches sont bien soudées. Si le problème ne vient pas de là, il se peut qu'un des pistes soit endommagée, auquel cas, il faut prendre des mesures appropriées.

Ensuite, le technicien passe aux tests sous tension. Ici, on regarde si tous les composants sont alimentés correctement et si l'ESP32 fournit bien les tensions attendues. Si les tensions ne correspondent pas, l'ESP32 peut être défectueux, il faut donc le changer dans ce cas-là. Cependant, le problème peut aussi venir du câble USB ou du port USB de l'ordinateur utilisé. On peut donc essayer de changer, le câble, le port USB utilisé ou le PC si aucune des deux solutions précédentes ne fonctionne. Si aucunes de ces trois actions ne résout le problème, il faudra alors envisager de changer l'ESP32.

Si les tensions sont correctes, on peut maintenant programmer la carte. Si lors du téléversement du programme Arduino il y a une erreur, il faut commencer par vérifier que le téléversement s'effectue sur le bon port. Le problème peut aussi venir de la version du programme, vérifier que c'est la bonne et qu'aucune modification n'a été effectuée. Enfin, faire les mêmes vérifications que pour le problème précédent, si rien ne fonctionne, essayer de changer le microcontrôleur.

On passe à présent aux tests via l'application de debug. Pour tous les tests suivants, le programme Qt Creator va envoyer un message en fonction du bouton cliqué, ces messages

s'affichent sur l'écran TFT. Pour tous les tests, vérifier que le bon message est transmis à chaque fois (voir Annexe 2 : Glossaire). Certains boutons n'affichent pas de commande, ceci est volontaire donc pas de problème.

Commencer par ouvrir le même port série que celui sur lequel le programme Arduino a été téléversé. Si le port ne s'ouvre pas, vérifier que le moniteur série n'est pas ouvert sur Arduino. Sinon vérifier l'état du câble et le changer au besoin.

Pour la fenêtre Wi-Fi, appuyez sur le bouton de scan et attendre qu'il se termine. Une fois terminé, s'assurer que le Wi-Fi souhaité est bien dans la liste. S'il n'est pas dans la liste, vérifier s'il est disponible sur un autre point d'accès. S'il l'est, redémarrer l'ESP32. S'il ne l'est pas, activer le Wi-Fi ou travailler sur un autre réseau. Entrer le mot de passe et se connecter. Si une adresse IPV4 s'affiche, un serveur a été créé et la fonctionnalité Wi-Fi est opérationnelle. S'il ne s'est pas connecté, vérifié que le mot de passe est correct, si c'est le cas, réessayer ou changer d'ESP32.

Pour les tests sur les capteurs, si les mesures ne sont pas dans la plage de valeurs indiquée, vérifier que le branchement est correct, s'il l'est essayer un autre capteur. Si le problème persiste, vérifier l'état des pistes et des soudures de la carte. Pour le BMP280 et le BH1750, si les adresses ne sont pas les bonnes, vérifier qu'aucune tension n'est appliquée sur l'une des broches ADDR. Si le problème ne vient pas de là, changer de capteur.

Pour les tests sur le relais, si lors de l'appui sur le bouton, le relais ne claque pas, il faut vérifier que le transistor n'a pas grillé et est branché correctement (voir Figure 1). La bobine du relais peut aussi avoir grillé, vérifier alors que le relais est fonctionnel. Si le problème ne vient pas de là, vérifier les pistes et les soudures.

Pour les tests de l'écran, si les choses ne s'affichent pas correctement, et que les soudures et les pistes sont en bon état, il faut changer l'écran. De même pour le RFID.

6. Conclusion

En conclusion, la carte assistant domotique est désormais commercialisable. En effet, une solution de maintenance est maintenant opérationnelle et la carte est diagnosticable rapidement, sans avoir à ressortir les documentations associées aux composants.

Annexe 1 : Fiche suiveuse d'une carte d'assistant domotique

FICHE SUIVEUSE / FOLLOW UP SHEET

PIE VFCU/ VFCU PEC	N°
-----------------------	----

Référence :

Reference :

Indice :

Issue :

NNO :

NSN :

Document de référence :

Reference document :

OUTILLAGE DE CONTRÔLE

DÉSIGNATION	TYPE UTILISE
- Un câble USB vers micro USB pouvant transférer des données - 1 multimètre - Un PC équipé de : - o Arduino - o Qt Creator - o Un port USB - Un exemplaire supplémentaire de chaque composant (voir Annexe 3 : Nomenclature des composants) - Une source lumineuse - Une source de gaz inflammables comme un briquet - Une source de chaleur - Un thermomètre fiable - Un appareil de mesure de pression fiable - Un luxmètre fiable	

TEST n°	Désignation	Résultats			Observations
		MINI	VALEUR	MAX	
Test à effectuer HORS-TENSION					
1.	Test hors tension				
1.1.	Aspect visuel :				
	- Absence de dégradation sur les composants	-	OK/NOK	-	
	- Soudures correctes	-	OK/NOK	-	
	- Pas de court-circuit apparent	-	OK/NOK	-	
1.2.	Tests de continuité :				
	- Continuité entre les broches 3.3 V	-	OK/NOK	-	
	- Continuité entre les broches 5 V	-	OK/NOK	-	
	- Continuité entre les GND	-	OK/NOK	-	
	- Continuité entre la broche 4 de l'ESP32 et CS TFT	-	OK/NOK	-	
	- Continuité entre la broche de 15 l'ESP32 et DC TFT	-	OK/NOK	-	
	- Continuité entre la broche 17 de l'ESP32 et RESET TFT	-	OK/NOK	-	
	- Continuité entre la broche 5 de l'ESP32 et les T_CS TFT	-	OK/NOK	-	
	- Continuité entre la broche 18 de l'ESP32 et T_CLK TFT et SCK du TFT et du RFID	-	OK/NOK	-	
	- Continuité entre la broche 19 de l'ESP32 et T_DO TFT et MISO du TFT et du RFID	-	OK/NOK	-	
	- Continuité entre la broche 21 de l'ESP32 et SDA BH1750 et BMP280	-	OK/NOK	-	
	- Continuité entre la broche 22 de l'ESP32 et SCL BH1750 et BMP280	-	OK/NOK	-	
	- Continuité entre la broche 23 de l'ESP32 et T_DIN TFT et MOSI du TFT et du RFID	-	OK/NOK	-	
	- Continuité entre la broche 27 de l'ESP32 et SDA RFID	-	OK/NOK	-	
	- Continuité entre la broche 25 de l'ESP32 et broche centrale du transistor BS170	-	OK/NOK	-	
	- Continuité entre la broche 33 de l'ESP32 et RST RFID	-	OK/NOK	-	
	- Continuité entre la broche VN de l'ESP32 et l'intersection entre R1 et R2	-	OK/NOK	-	

TEST n°	Désignation	Résultats			Observations
		MINI	VALEUR	MAX	
	- Continuité entre la broche VP de l'ESP32 et A0 MQ9	-	OK/NOK	-	
	- Discontinuité entre la GND et les broches 3.3 V et 5 V	-	OK/NOK	-	
Tests à effectuer SOUS-TENSION					
2.	Contrôles				
2.1.	Test Alimentations :				
	- Tension 3.3 V entre 3V3 ESP32 et GND	3,135	-	3.465	
	- Tension 3.3 V entre VCC MQ9 et GND	3,135	-	3.465	
	- Tension 3.3 V entre VCC BMP280 et GND	3,135	-	3.465	
	- Tension 3.3 V entre VCC BH1750 et GND	3,135	-	3.465	
	- Tension 3.3 V entre VCC TFT et GND	3,135	-	3.465	
	- Tension 3.3 V entre 3.3 V RFID et GND	3,135	-	3.465	
	- Tension 5 V entre VIN ESP32 et GND	4.75	-	5.25	
	- Tension 5 V entre LED TFT et GND	4.75	-	5.25	
	- Tension 5 V entre cathode de 1N4004 et GND	4.75	-	5.25	
	- Tension 5 V entre A1 relais et GND	4.75	-	5.25	
	- Tension 5 V entre R1 et GND	4.75	-	5.25	
3.	Programmation				
3.1.	Programmation :				
	- Rapport de programmation Arduino sans erreur	-	OK/NOK	-	
	- Rapport de programmation Qt Creator sans erreur	-	OK/NOK	-	
	- Led de vie apparente sur l'ESP32	-	OK/NOK	-	
4.	Port série				
4.1.	Port série :				
	- Port série ouvert	-	OK/NOK	-	
5.	Wi-Fi				
	- Commande affichée sur l'écran	-	OK/NOK	-	
	- Scan terminé	-	OK/NOK	-	
	- Réseau personnel détecté	-	OK/NOK	-	
	- Serveur créé	-	OK/NOK	-	

TEST n°	Désignation	Résultats			Observations
		MINI	VALEUR	MAX	
6.	Capteurs				
6.1	MQ9 : - Commande affichée sur l'écran - Valeur en ppm dans une pièce saine - Valeur en ppm avec une source de gaz inflammable à proximité (briquet)	- - 10	OK/NOK - -	- 1.9 -	
6.2.	BH1750 : - Commande affichée sur l'écran - Adresse - Luminosité avec un flash de téléphone sur le capteur en lux - Luminosité dans une pièce éclairée en lux - Luminosité avec le capteur orienté vers un mur en lux	- - 5000 200 -	OK/NOK 0x23 - - -	- - - 500 10	
6.3.	BMP280 : - Commande affichée sur l'écran - Adresse - Pression en hPa - Température comparée à une mesure fiable	- - 1000 -	OK/NOK 0x76 - OK/NOK	- - 1030 -	
6.4.	Photo Résistance : - Commande affichée sur l'écran - Luminosité avec un flash de téléphone sur le capteur en lux - Luminosité dans une pièce éclairée en lux - Luminosité avec le capteur orienté vers un mur en lux	- 5000 200 -	OK/NOK - - -	- - 500 10	
7.	Relais				
	- Commande affichée sur l'écran - Lors de l'appui sur le bouton, le relais claque - L'état du bouton est en adéquation avec l'état renvoyé par l'ESP - L'état du bouton est en adéquation avec la tension mesurée aux bornes du relais	- - - -	OK/NOK OK/NOK OK/NOK OK/NOK	- - - -	

TEST n°	Désignation	Résultats			Observations
		MINI	VALEUR	MAX	
8.	Ecran TFT				
	- Les messages reçus s'affichent	-	OK/NOK	-	
	- Tous les messages s'affichent correctement en rouge	-	OK/NOK	-	
	- Tous les messages s'affichent correctement en Vert	-	OK/NOK	-	
	- Tous les messages s'affichent correctement en Bleu	-	OK/NOK	-	
	- Tous les messages s'affichent correctement en Blanc	-	OK/NOK	-	
	- Après avoir appuyé sur le bouton « Inversé » toutes les couleurs s'inversent	-	OK/NOK	-	
	- Toutes les lettres s'affichent correctement	-	OK/NOK	-	
	- Tous les chiffres s'affichent correctement	-	OK/NOK	-	
	- Le tactile fonction, absence de zones mortes	-	OK/NOK	-	
	- L'écran s'efface lorsque c'est demandé	-	OK/NOK	-	
9.	RFID				
	- Commande affichée sur l'écran	-	OK/NOK	-	
	- Renvoie l'ID du badge présenté	-	OK/NOK	-	
	- L'ID est bien celui associé au badge	-	OK/NOK	-	

Tableau 1 – Fiche suiveuse de la carte électronique

Annexe 2 : Glossaire

Commande	Bouton associé
WIFI:SCAN	Bouton « Search for Wi-Fi » de la fenêtre « Wi-Fi »
WIFI:CONNECT:index_Wifi:mot_de_passe	Bouton « Connect » de la fenêtre « Wi-Fi »
MQ9	Bouton « Get value » de la fenêtre « Capteurs » pour le MQ9
BH1750:VALUE	Bouton « Get value » de la fenêtre « Capteurs » pour le BH1750
BH1750:ADRESS	Bouton « Get adress » de la fenêtre « Capteurs » pour le BH1750
BMP280:VALUE	Bouton « Get value » de la fenêtre « Capteurs » pour le BMP280
BMP280:ADRESS	Bouton « Get adress » de la fenêtre « Capteurs » pour le BMP280
RES	Bouton « Get value » de la fenêtre « Capteurs » pour la photo résistance
RELAIS:ON	Bouton « OFF » de la fenêtre « Relais »
RELAIS:OFF	Bouton « ON » de la fenêtre « Relais »
TFT:RED	Bouton « Rouge » de la fenêtre « Ecran TFT »
TFT:GREEN	Bouton « Vert » de la fenêtre « Ecran TFT »
TFT:BLUE	Bouton « Bleu » de la fenêtre « Ecran TFT »
TFT:WHITE	Bouton « Blanc » de la fenêtre « Ecran TFT »
TFT:LETTRES	Bouton « Lettres » de la fenêtre « Ecran TFT »
TFT:CHIFFRES	Bouton « Chiffres » de la fenêtre « Ecran TFT »
TFT:TOUCH	Bouton « Touch test » de la fenêtre « Ecran TFT »
RFID	Bouton « Start card scan » de la fenêtre « RFID »

Tableau 2 – Glossaire des messages transmis par le programme Qt Creator

Annexe 3 : Nomenclature des composants

Item	Qté	Rep.	Désignation	Référence Fabricant	Fabricant	Type	Lien
1	1	D1	Diode de redressement standard, 400 V, 1 A, Une, 1 V, 30 A	1N4004-T	Diodes INC	PTH	Cliquer
2	1	J1	Bornier à vis 01x03	3153912784418	CompoZan	PTH	Cliquer
3	1	K1	Finder 30.22.7.005.0000 Relais pour circuits imprimés 5 V/DC 2 A	2050000148408	Finder	PTH	Cliquer
4	1	Q1	TRANSISTOR MOSFET N TO-92	BS170	ONSEMI	PTH	Cliquer
5	1	R1	Photorésistance LDR04	LDR04	Velleman	PTH	Cliquer
6	1	R2	Résistance 3.3 kΩ 0805 (2012M) Metal Film SMD Resistor ±0.1% 0.125W	RN73R2ATTD3301B10	KOA	SMD	Cliquer
7	1	RFID1	Lecteur RFID 13.56 MHz utilisant un module MFRC522	RFID-RC522	AITEXM ROBOT	PTH	Cliquer
8	1	U1	Capteur de pression et de température	BMP280-3.3V	DONSTAR	PTH	Cliquer
9	1	U2	Capteur de méthane	MQ9	AITEXM ROBOT	PTH	Cliquer
10	1	U3	Capteur de luminosité	BH1750	AITEXM ROBOT	PTH	Cliquer
11	1	U4	Écran TFT pour Ardu37, 2.4 pouces avec tactile	TFT_ILI9371	AITEXM ROBOT	PTH	Cliquer
12	1	U5	ESP32 NodeMCU Module WLAN WiFi Dev Kit C Development Board	ESP32	AZDelivery	PTH	Cliquer

Tableau 3 – Nomenclature des composants