COMPTE RENDU – SÉANCE 5

TRAVAIL RÉALISÉ LORS DE LA SÉANCE

Lors de cette cinquième séance, nous avons essayé, avec Elisa, de trouver des solutions aux problèmes rencontrés et réfléchi à l'élaboration de la boite.

J'ai donc essayé de joindre les deux cartes via software serial, les deux programmes pour chaque cartes (ESP32 et UNO) compilaient mais l'ESP32 ne parvenait pas à se connecter au réseau. On obtenait plusieurs lignes de code défilant sur le moniteur série en boucle. Il se trouve que le problème rencontré précédemment sur la compatibilité de la bibliothèque software serial avec l'esp32 était dû à la présence de trop nombreuses bibliothèque software qui interféraient entre elles.

Nous avons donc envoyé un mail à monsieur Peter qui nous a conseillé d'utiliser une nouvelle bibliothèque : Hardware serial sur le programme de l'esp32. Cette bibliothèque permet entre autres de sélectionner soi-même les ports pour la connexion rx/tx sur l'esp32.

CODE

POUR LA CARTE UNO :

```
#include <Keypad.h>
#include <Servo.h>
#include <SoftwareSerial.h>
#define LIGNES 4
#define COLONNES 4
#define CODE 4

// variables communication cartes
const int tx = 01;
const int rx = 00;
SoftwareSerial MySerial(rx, tx);

String message_recu = "";
String result[4];
int k = 0;


// variables led
const int led_VERTE = 13;
const int led_ROUGE = 11;
const int led_ORANGE = 12;


const char kp4x4Keys[LIGNES][COLONNES]  = {
  {'1', '2', '3', 'A'}, {'4', '5', '6', 'B'}, {'7', '8', '9', 'C'}, {'*', '0', '#', 'D'}};
byte lignePin [4] = {9, 8, 7, 6};
byte colonnePin [4] = {5, 4, 3, 2};
char tab[CODE];
char codebon[]="1234";
char codeferme[]="0000";
int i=0;
int incomingByte;
int tentative=0;
int chance= 3;
Servo servo_10;//servo branché sur l'entrée 10

//Variables
Keypad kp4x4  = Keypad(makeKeymap(kp4x4Keys), lignePin, colonnePin, LIGNES, COLONNES);

void setup() {
  Serial.begin(9600);
  MySerial.begin(4800);
  servo_10.attach(10);
  Serial.println("Entrez votre code :");

  pinMode(led_ORANGE, OUTPUT);
  pinMode(led_ROUGE, OUTPUT);
  pinMode(led_VERTE, OUTPUT);

  digitalWrite(led_ORANGE,LOW);
  digitalWrite(led_VERTE,LOW);
  digitalWrite(led_ROUGE,LOW);
}
```

```
void loop() {
  lecturechiffre();

  /*
  readData();
  for(int j=0; j<3;j++){ // on imprime res
    Serial.println(result[j]);}
  k=0;// on reset k */
}

void lecturechiffre(){
  char transformechiffre = kp4x4.getKey(); //récupère le chiffre du keypad
  if (transformechiffre) {
    tab[i]=transformechiffre;//mets les chiffres à la suite des autres
    i=i+1;
    if(i==CODE){
      Serial.println("****");
      i=0;

      if((strcmp(tab,codebon)==0)){ //compare le code entré et celui attendu
        Serial.println("Code bon !");
        digitalWrite(led_ORANGE,HIGH);
        if (result[0]==1){
          servo_10.write(-90); //ouvre le loquet
          digitalWrite(led_ORANGE,LOW);
          digitalWrite(led_ROUGE,LOW);
          digitalWrite(led_VERTE,HIGH);}
      }

      else if(strcmp(tab,codeferme)==0){ //compare le code entré et celui attendu (code de fermeture : ici 0000)
        Serial.println("Code fermeture !");
        digitalWrite(led_ORANGE,LOW);
        digitalWrite(led_VERTE,LOW);
        digitalWrite(led_ROUGE,LOW);
        servo_10.write(90); //ouvre le loquet
      }

      else{
        tentative ++;
        chance --;
        Serial.print("Code faux, il vous reste ");
        Serial.print(chance);
        Serial.println(" essai(s)");
        digitalWrite(led_ROUGE, HIGH);
        delay(2000);
        digitalWrite(led_ROUGE, LOW);
        digitalWrite(led_VERTE, LOW);
        digitalWrite(led_ORANGE,LOW);
        if(chance==0){
          Serial.println("Merci d'attendre 20 sec avant de recommmencer");
          for(int j; j<20; j++){
            digitalWrite(led_ROUGE,HIGH);
            delay(1000);
            digitalWrite(led_ROUGE,LOW);}
          chance = 3;}
      }
   }
 }
}

void readData() {
  while (MySerial.available()) {
    char inChar = (char)MySerial.read();
    if (inChar != '\n') {
      message_recu += inChar;}
    else{
      result[k] = message_recu;
      message_recu = ""
      k++;}
  }
}

void addToResult(){
  if (k <3){
    result[k] = message_recu;
    message_recu = "";
    k++;}
  else{
    k=0;
    result[k] = message_recu;
    message_recu = "";}
}
```

Ce code permet de récupérer un message envoyé par l'esp32 et de le stocker dans un tableau. Ensuite, on récupère les éléments de ce tableau pour effectuer des actions ou remplacer des variables (comme le code d'ouverture qui pourra par la suite être réinitialisé à distance). Par exemple, si result[0] == 1, cela veut dire que le propriétaire de la boite à validé l'ouverture de la boite à distance et que la boite peut donc s'ouvrir physiquement.

CODE POUR L'ESP32

```
#include <SPIFFS.h>
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
```

```
#include <HardwareSerial.h>

HardwareSerial MySerial(1);

// Replace with your network credentials
const char* ssid = "Freebox_BNBHERE";
const char* password = "bnbhereguest";

// varaible communication RX/TX


// variables pour stocker les messages
int valuesToSend[4];
String message_recu = "";

// variables etat leds
int etat_LV = 0;
int etat_LR = 0;
int etat_LO = 0;

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);


// Replaces placeholder with LED state value
String processor(const String& var){
  if(var == "STATE"){
  return String();}
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);
  MySerial.begin(4800, SERIAL_8N1, 17, 16);
  message_recu.reserve(200);


  // Initialize SPIFFS
  if(!SPIFFS.begin(true)){
    Serial.println("An Error has occurred while mounting SPIFFS");
    return;
  }

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }

  // Print ESP32 Local IP Address
  Serial.print("Adresse IP: ");
  Serial.println(WiFi.localIP());

  // Route for root / web page
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  // Route to load style.css file
  server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/style.css", "text/css");
  });

  // Route to set GPIO to HIGH
  server.on("/oui", HTTP_GET, [](AsyncWebServerRequest *request){
    etat_LV = 1;
    etat_LO = 0;
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  // Route to set GPIO to LOW
  server.on("/non", HTTP_GET, [](AsyncWebServerRequest *request){
    etat_LR = 1;
    etat_LO = 0;
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  // Start server
  server.begin();
}


void sendData(int msg){
  MySerial.println(msg);
}


void loop(){
  valuesToSend[0] = etat_LV ; // etat_LV
  valuesToSend[1] = etat_LO ; //etat_LO
  valuesToSend[2] = etat_LR ; //etat_LR

  if(MySerial.available()){
    for (int j=0;j<3;j++){
      MySerial.write(valuesToSend[j]);}
  }
}
```

Ce code permet de stocker des variables dans un tableau et de les envoyer à la carte UNO.