**Lycée Raymond Queneau**

18

**Annexe**

**Annexe du projet domotique**

**Chauveau, Dufresn, Guyader, Girard**

Sommaire

[Programme de la centrale de gestion : 3](#_Toc514861079)

[Programme principal : 3](#_Toc514861080)

[capteur.h 6](#_Toc514861081)

[capteur.cpp 7](#_Toc514861082)

[controleActionneur.h 11](#_Toc514861083)

[controleActionneur.cpp 11](#_Toc514861084)

[horodatageConsomation.h 14](#_Toc514861085)

[horodatageConsomation.cpp 15](#_Toc514861086)

[gestionMaison.h 17](#_Toc514861087)

[gestionMaison.cpp 18](#_Toc514861088)

[Programme du serveur WEB : 21](#_Toc514861089)

[Programme de l'application mobile : 21](#_Toc514861090)

[Câblage du système : 23](#_Toc514861091)

# Programme de la centrale de gestion :

## Programme principal :

#include "capteur.h"

#include "controleActionneur.h"

#include "horodatageConsomation.h"

#include "gestionMaison.h"

#include <avr/io.h>

#include <avr/interrupt.h>

Maison maison;

void initTimer2 (void);

unsigned short interruptePinCompteur = 3; //Variable Compteur

unsigned short flagCompteurEnergie=1;

unsigned short flagCompteurTimer=0;

volatile unsigned int cpt=0;

unsigned short interruptBouton=2;

void setup(){

initSerial();

initTimer2();

initActionneur();

initHorodatageConsomation();

initCapteur();

//Setup Compteur d'energie interruption

pinMode(interruptePinCompteur, INPUT\_PULLUP);

attachInterrupt(digitalPinToInterrupt(interruptePinCompteur), interruptionCompteur, LOW);

//setup bouton alarme interruption

pinMode(interruptBouton, INPUT);

attachInterrupt(digitalPinToInterrupt(interruptBouton), interruptionBouton, RISING);

}

void loop(){

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Le drapeau 1 du Timer1 se "lève" toute les 5 sec\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (flagCompteurTimer==1){

maison.temperature=capteurTemperature(); //récupération de la température dans un attribue

maison.humidite=capteurHumidite();

maison.etatRadiateur=etatThermostat();

maison.emissionTrame(); //emission de la trame pour le serveur WEB et la tablette

flagCompteurTimer=0; //rermise du drapeau 1 a 0 "on baisse le drapeau".

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Le drapeau 2 du Timer1 se "lève" toute les 10 sec\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (flagCompteurTimer==2){

envoieSMS(maison.incendie, maison.mouvement); //émission du sms d'alerte si alerte

maison.qualiteAir=capteurQualiteAir();

maison.temperature=capteurTemperature();

maison.humidite=capteurHumidite();

maison.etatRadiateur=etatThermostat();

maison.luminosite=capteurLuminosite();

maison.incendie = false; //on remet les valeurs des attribue d'alerte a l'état passif

maison.mouvement = false;

maison.emissionTrame();//emission de la trame pour le serveur WEB et la tablette

flagCompteurTimer=0; //rermise du drapeau 2 a 0 "on baisse le drapeau".

}

maison.lectureTablette(); //on récupère les valeurs valeurs envoyer par la tablette

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Partie sécurité\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (maison.incendie == false){

maison.incendie=capteurIncendie();

}

if (maison.alarme==true){

if (maison.mouvement == false){

maison.mouvement=capteurMouvement();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Controle des actionneur en fonction des commande de l'utilisateur\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

controleThermostat(maison.radiateurMode, maison.temperature, maison.temperatureUtilisateur, maison.radiateur);

maison.volet1=controleVolet1(maison.volet1Etat, maison.voletMode);

maison.volet2=controleVolet2(maison.volet2Etat, maison.voletMode);

maison.lumiere=controleLumiere(maison.lumiereEtat);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Le drapeau 1 de l'interruption compteur\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (flagCompteurEnergie==1){

maison.consomation=consomation();

maison.annee=annee();

maison.mois=mois();

maison.jour=jour();

maison.heure=heure();

maison.minutes=minutes();

maison.seconde=seconde();

horodatage(maison.consomation);

flagCompteurEnergie=0;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void interruptionBouton(){

if (maison.alarme==true){

maison.alarme=false;

Serial.println("eteint");

}

else{

maison.alarme=true;

Serial.println("allume");

}

}

void interruptionCompteur(){

delay(50);

if (digitalRead(interruptePinCompteur)==0){

flagCompteurEnergie=1;

while(digitalRead(interruptePinCompteur)!=1);

}

}

void initTimer2 (){

//config timer2

TCCR2A=0x00;

TCCR2B|=(1<<CS22)|(1<<CS21)|(1<<CS20);

TIMSK2|=(1<<TOIE2);

sei();

}

ISR(TIMER2\_OVF\_vect){

cpt++;

if (cpt==305){

flagCompteurTimer=1;

}

if (cpt==610){

flagCompteurTimer=2;

cpt=0;

}

}

## capteur.h

#ifndef CAPTEUR\_H

#define CAPTEUR\_H

void initCapteur (void);

double capteurTemperature (void);

double capteurHumidite (void);

int capteurQualiteAir (void);

float capteurLuminosite(void);

bool capteurIncendie(void);

bool capteurMouvement(void);

void envoieSMS(bool, bool);

int sendATcommand (char\*, char\*, unsigned int);

#endif

## capteur.cpp

#include "capteur.h"

#include "Arduino.h"

#include <HIH6130.h>

#include <AirQuality.h>

#include <Wire.h>

#include <Adafruit\_Sensor.h>

#include <Adafruit\_TSL2561\_U.h>

#define CAPTEUR\_INCENDIE 35

#define CAPTEUR\_MOUVEMENT 36

Adafruit\_TSL2561\_Unified tsl = Adafruit\_TSL2561\_Unified(TSL2561\_ADDR\_FLOAT, 12345);

HIH6130 rht(0x27);

AirQuality airqualitysensor;

int current\_quality =-1;

int answer;

char aux\_string[30];

char phone\_number[]="+33648977501";

void initCapteur(){

Serial.println("initialisation des capteurs...");

rht.begin();

pinMode(CAPTEUR\_INCENDIE, INPUT);

pinMode(CAPTEUR\_MOUVEMENT, INPUT);

airqualitysensor.init(A0);

delay (5000);

Serial.println("Capteurs initialises.");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fonction Capteur : Temperature, Humiditée, Qualitée d'air, Luminosité, Incendie, Mouvement\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

double capteurTemperature(){

rht.readRHT();

return rht.temperature;

}

double capteurHumidite(){

rht.readRHT();

return rht.humidity;

}

int capteurQualiteAir(){

current\_quality=airqualitysensor.slope();

return current\_quality;

}

float capteurLuminosite(void){

tsl.setGain(TSL2561\_GAIN\_16X);

sensors\_event\_t event;

tsl.getEvent(&event);

return event.light;

}

bool capteurIncendie(void){

int fumeeDetecte = digitalRead(CAPTEUR\_INCENDIE);

if(fumeeDetecte == HIGH){

return true;

}

else{

return false;

}

}

bool capteurMouvement(void){

int mouvementDetecte = digitalRead(CAPTEUR\_MOUVEMENT);

if(mouvementDetecte == HIGH){

return true;

}

else{

return false;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fonction d'envoie du SMS d'alerte\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void envoieSMS(bool incendie, bool mouvement){

//while( (sendATcommand("AT+CREG?", "+CREG: 0,1", 500) || sendATcommand("AT+CREG?", "+CREG: 0,5", 500)) == 0 );

// Activation du mode texte pour les SMS.

sendATcommand("AT+CMGF=1", "OK", 1000);

sprintf(aux\_string,"AT+CMGS=\"%s\"", phone\_number);

// Envoi du numéro de téléphone au module GSM.

sendATcommand(aux\_string, ">", 2000);

if(mouvement==true){

Serial.println("un mouvement est detecte !");

Serial1.println("un mouvement est detecte !");

Serial1.write(0x1A);

}

if(incendie==true){

Serial1.println("un incendie est detecte !");

Serial1.write(0x1A);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fonction Timer et Commande AT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int sendATcommand(char\* ATcommand, char\* expected\_answer, unsigned int timeout){

int x=0, answer=0;

char response[100];

unsigned long previous;

// Initialisation de la chaine de caractère (string).

memset(response, '\0', 100);

delay(50);

// Initialisation du tampon d'entrée (input buffer).

while( Serial1.available() > 0) Serial1.read();

// Envoi des commandes AT

Serial1.println(ATcommand);

x = 0;

previous = millis();

// Cette boucle attend la réponse du module GSM.

do{

// Cette commande vérifie s'il y a des données disponibles dans le tampon.

//Ces données sont comparées avec la réponse attendue.

if(Serial1.available() != 0){

response[x] = Serial1.read();

x++;

// Comparaison des données

if (strstr(response, expected\_answer) != NULL)

{

answer = 1;

}

}

// Attente d'une réponse.

}while((answer == 0) && ((millis() - previous) < timeout));

//Serial.println(response); //Cette ligne permet de debuguer le programme en cas de problème !

return answer;

}

ISR(TIMER1\_OVF\_vect) //timer

{

if(airqualitysensor.counter==61)//une fois que le nombre de marqueur

//atteind 61 (~30ms) on exécute les

//lignes de commande suivante

{

/\*On récupére la dernière valeur avant de la remplacer par

\*une valeur plus récentes puis on remet le nombre de marqueur

\* à zéro et on relance le timer\*/

airqualitysensor.last\_vol=airqualitysensor.first\_vol;

airqualitysensor.first\_vol=analogRead(A0);

airqualitysensor.counter=0;

airqualitysensor.timer\_index=1;

PORTB=PORTB^0x20;

/\*La bibliothèque compare les deux valeur last\_vol et

\*first\_vol pour nous indiquer la qualité de l'air\*/

}

else

{

airqualitysensor.counter++;

}

}

## controleActionneur.h

#ifndef CONTROLEACTIONNEUR\_H

#define CONTROLEACTIONNEUR\_H

void initActionneur(void);

void controleThermostat (bool, double, double, bool);

bool controleLumiere(int);

bool controleVolet1(bool, bool);

bool controleVolet2(bool, bool);

bool etatThermostat (void);

#endif

## controleActionneur.cpp

#include "controleActionneur.h"

#include "Arduino.h"

#include "capteur.h"

int cmdUp = 24;

int cmdDown = 25;

int switchUp = 26;

int switchDown = 27;

int cmdUp2 = 28;

int cmdDown2 = 29;

int switchUp2 = 30;

int switchDown2 = 31;

int pinThermostat = 4;

int pinLumiere = 33;

char valeurEtat;

void initActionneur(){

pinMode(pinThermostat, OUTPUT);

pinMode(pinLumiere, OUTPUT);

pinMode(cmdUp, OUTPUT);

pinMode(cmdDown, OUTPUT);

pinMode(switchUp, INPUT);

pinMode(switchDown, INPUT);

pinMode(cmdUp2, OUTPUT);

pinMode(cmdDown2, OUTPUT);

pinMode(switchUp2, INPUT);

pinMode(switchDown2, INPUT);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fonction Controle des actionneurs : Thermostat, Lumiere, Volet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void controleThermostat(bool mode, double temperature, double temperatureUtilisateur, bool valeurEtat){

if (mode==false){

if (valeurEtat==true){

digitalWrite(pinThermostat,HIGH);

}

if (valeurEtat==false){

digitalWrite(pinThermostat,LOW);

}

}

if (mode==true){

if(temperature<=temperatureUtilisateur-1){

digitalWrite(pinThermostat,HIGH);

}

if(temperature>=temperatureUtilisateur){

digitalWrite(pinThermostat,LOW);

}

}

}

bool controleLumiere(int positionLumiere){

bool etat=false;

if(positionLumiere==1){

digitalWrite(pinLumiere, HIGH);

etat=true;

}

if(positionLumiere==0){

digitalWrite(pinLumiere, LOW);

etat=false;

}

return etat;

}

bool controleVolet1(bool sens, bool mode){

bool positionVolet;

if(mode==true){

if(sens==true){ //ouvrir

while(digitalRead(switchUp)==HIGH){

digitalWrite(cmdUp,HIGH);

}

digitalWrite(cmdUp,LOW);

positionVolet = true;

}

if(sens==false){ //fermer

while(digitalRead(switchDown)==HIGH){

digitalWrite(cmdDown,HIGH);

}

digitalWrite(cmdDown,LOW);

positionVolet = false;

}

}

if(mode==false){

if(capteurLuminosite()>35.00){

while(digitalRead(switchUp)==HIGH){

digitalWrite(cmdUp,HIGH);

}

digitalWrite(cmdUp,LOW);

positionVolet = true;

}

if(capteurLuminosite()<10.00){

while(digitalRead(switchDown)==HIGH){

digitalWrite(cmdDown,HIGH);

}

digitalWrite(cmdDown,LOW);

positionVolet = false;

}

}

return positionVolet;

}

bool controleVolet2(bool sens, bool mode){

bool positionVolet;

if(mode==true){

if(sens==true){ //ouvrir

while(digitalRead(switchUp2)==HIGH){

digitalWrite(cmdUp2,HIGH);

}

digitalWrite(cmdUp2,LOW);

positionVolet = true;

}

if(sens==false){ //fermer

while(digitalRead(switchDown2)==HIGH){

digitalWrite(cmdDown2,HIGH);

}

digitalWrite(cmdDown2,LOW);

positionVolet = false;

}

}

if(mode==false){

if(capteurLuminosite()>35.00){

while(digitalRead(switchUp2)==HIGH){

digitalWrite(cmdUp2,HIGH);

}

digitalWrite(cmdUp2,LOW);

positionVolet = true;

}

if(capteurLuminosite()<10.00){

while(digitalRead(switchDown2)==HIGH){

digitalWrite(cmdDown2,HIGH);

}

digitalWrite(cmdDown2,LOW);

positionVolet = false;

}

}

return positionVolet;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Récupération de létat du thermostat : allumer/etteint\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool etatThermostat(){

bool val = digitalRead(pinThermostat);

return val;

}

## horodatageConsomation.h

#ifndef HORODATAGECONSOMATION\_H

#define HORODATAGECONSOMATION\_H

void initHorodatageConsomation(void);

void horodatage(float);

void initSD(void);

void initRTC(void);

float consomation(void);

int annee(void);

int jour(void);

int mois(void);

int heure(void);

int minutes(void);

int seconde(void);

#endif

## horodatageConsomation.cpp

#include <SPI.h> //

#include <SdFat.h> //Bibliothèque carte SD

#include <RTClib.h> //Bibliothèque RTC

#include "horodatageConsomation.h"

#include "gestionMaison.h"

/\*Le module SD focntion en liason SPI et les pin SPI sont différente en fonction de la carte

carte Arduino utilisé. cf : https://www.arduino.cc/en/Reference/SPI \*/

#define BUFFER\_SIZE 250 //déffinition de la taille du buffer.

SdFat sd; //

uint8\_t buf[BUFFER\_SIZE]; //Variable SD

float pulsion = 0;

RTC\_DS1307 rtc; //Variable RTC

void initHorodatageConsomation(){

//Setup SD

Serial.println("init SD");

if(!sd.begin()){

Serial.println("erreur init");

return;

}

rtc.begin();

rtc.isrunning();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fonction qui horodate la consomation d'énergie et sauvegarde le tout sur une carte SD\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void horodatage(float consomation){

SdFile fichier;

pulsion=pulsion+0.1;

//ecriture dans le fichier txt Compteur\_Elec dans la SD

if(!fichier.open(&sd, "Compteur\_bis\_bis\_bis.txt", O\_RDWR|O\_TRUNC|O\_AT\_END)){

Serial.println("Erreur");

return;

}

fichier.print(consomation);

fichier.print("KWh ");

DateTime now = rtc.now();

//affichage de la date

fichier.print(now.year());

fichier.print('/');

fichier.print(now.month());

fichier.print('/');

fichier.print(now.day());

fichier.print(" ");

//affichage de l'heure

fichier.print(now.hour());

fichier.print(':');

fichier.print(now.minute());

fichier.print(':');

fichier.println(now.second());

fichier.close();

//sd.ls("/", LS\_SIZE|LS\_R);

//lecture du contenue du fichier txt dans la SD

if(!fichier.open(&sd, "Compteur\_bis\_bis\_bis.txt", O\_READ)){

Serial.println("erreur");

return;

}

fichier.read(buf, sizeof(buf));

String myString = String ((char \*)buf);

myString.trim();

Serial.println(myString);

fichier.close();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fonction de comptage de la consomation\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float consomation(){

pulsion=pulsion+0.1;

return pulsion;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fonctions de récupération de la date et de l'heure\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int annee(){

DateTime now = rtc.now();

return now.year();

}

int mois(){

DateTime now = rtc.now();

return now.month();

}

int jour(){

DateTime now = rtc.now();

return now.day();

}

int heure(){

DateTime now = rtc.now();

return now.hour();

}

int minutes(){

DateTime now = rtc.now();

return now.minute();

}

int seconde(){

DateTime now = rtc.now();

return now.second();

}

## gestionMaison.h

#ifndef \_\_MAISON\_H\_\_

#define \_\_MAISON\_H\_\_

class Maison

{

public:

bool lumiere=false;

bool volet1=false;

bool volet2=false;

float luminosite;

bool alarme=false;

bool incendie;

bool mouvement;

double temperatureUtilisateur=21;

bool radiateur=false;

double temperature;

int qualiteAir;

double humidite;

float consomation;

bool etatRadiateur;

bool lumiereEtat=false; //\_false=0=etteind|true=1=allumer

bool volet1Etat=false; //\_false=0=etteind|true=1=allumer

bool volet2Etat=false; //\_false=0=etteind|true=1=allumer

bool voletMode=true; //\_true=0=mode manuel|false=1=mode automatique

bool radiateurMode=false;//\_false=0=mode manuel|true=1=mode automatique

int annee;

int jour;

int mois;

int heure;

int minutes;

int seconde;

void emissionTrame (void);

void lectureTablette (void);

private:

};

void initSerial (void);

#endif

## gestionMaison.cpp

#include "gestionmaison.h"

#include "Arduino.h"

void initSerial (){

Serial.begin(9600);

Serial1.begin(9600);

Serial2.begin(9600);

Serial3.begin(9600);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fonction de récupération des donnée de la tablette\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Maison::lectureTablette (){

while(Serial2.available() > 0) {

double c = Serial2.read();

if (c==0){ //lumiere eteinte

if (lumiere==true){

lumiereEtat=false;

}

}

if (c==1){ //lumiere allumer

if (lumiere==false){

lumiereEtat=true;

}

}

if (c==2){ //volet1 fermer

if (volet1Etat==true){

volet1Etat=false;

}

}

if (c==3){ //volet1 ouvrir

if (volet1Etat==false){

volet1Etat=true;

}

}

if (c==4){ //volet2 fermer

if (volet2Etat==true){

volet2Etat=false;

}

}

if (c==5){ //volet2 ouvrir

if (volet2Etat==false){

volet2Etat=true;

}

}

if (c==6){ //chauffage etteind

if (radiateur==true){

radiateur=false;

}

}

if (c==7){ //chauffage allumer

if (radiateur==false){

radiateur=true;

}

}

if (c==8){ //chauffage manuel

radiateurMode=true;

}

if (c==9){ //chauffage auto

radiateurMode=false;

}

if (c>=10 && c<=30){

temperatureUtilisateur = c;

}

if (c==31){ //volet manuel

voletMode=true;

}

if (c==32){ //volet auto

voletMode=false;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Fonction d'emission des valeurs au serveur Web et a la Tablette\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Maison::emissionTrame (){

String trame;

//tablette

trame += lumiere;

trame += "!";

trame += luminosite;

trame += "!";

trame += volet1Etat;

trame += "!";

trame += volet2Etat;

trame += "!";

trame += etatRadiateur;

trame += "!";

trame += temperature;

trame += "!";

trame += consomation;

trame += "!";

trame += humidite;

trame += "!";

trame += qualiteAir;

//relevé date

trame += "!";

trame += jour;

trame += "!";

trame += mois;

trame += "!";

trame += annee;

trame += "!";

trame += heure;

trame += "!";

trame += minutes;

trame += "!";

Serial.println(trame);

Serial2.println(trame);

Serial3.println(trame);

trame = "";

}

# Programme du serveur WEB :

## Programme principal :

#include <SPI.h> //bibliothèqe pour SPI

#include <Ethernet.h> //bibliothèque pour Ethernet

#include "decodage.h"

#include "maison.h"

Maison maison;

String adresse = "";

int valeur[10] = {0, 2, 0, 1, 0, 6, 7, 8, 9, 10};

int date[8] = {1, 2, 0, 3, 2, 0, 1, 8};

int heure[6] = {1, 2, 4, 2, 5, 9};

byte mac[] = {0x90, 0xA2, 0xDA, 0xCC, 0xDF, 0xAB}; // tableau pour l'adresse MAC de votre carte

byte ip[] = {192, 168, 190, 46}; //tableau pour l'adresse IP

EthernetServer serveur(80); // déclare l'objet serveur au port d'écoute 80

String trame = ""; // a String to hold incoming data

boolean trameComplete = false; // whether the string is complete

char c = 0;

void setup() {

Serial.begin (9600); //initialisation de communication série

Serial1.begin(9600); //initialisation de communication bluetooth

Ethernet.begin (mac, ip); //initialisation de la communication Ethernet

Serial.print("\nLe serveur est sur l'adresse : ");

Serial.println(Ethernet.localIP()); //on affiche l'adresse IP de la connexion

serveur.begin(); // démarre l'écoute

}

void loop() {

if(trameComplete == true){ // Quand on recois une trame complete on l'affiche sur le site

recuperationTrame(maison, trame, trameComplete); // Fonction qui recupère la trame recue

traductionTrame(maison); // Fonction qui interprete la trame

} //on écoute le port

EthernetClient client = serveur.available(); // On attend la conexion d'un utilisateur au site

if (client) { //si client connecté // Si il y a un client

Serial.println("Client en ligne\n"); //on le dit...

if (client.connected()) { // tant qu'il a des infos à transmettre

while (client.available()) { // tant que le client et connecté

char c=client.read(); // on lit le caractère

Serial.write(c); // on l'écrit sur le moniteur série

delay(1); //delai de lecture

}

//réponse au client et affichage du site

client.println("HTTP/1.1 200 OK"); // type du HTML

client.println("Connection: close"); // fermeture de la connexion quand toute la réponse sera envoyée

client.println("Refresh: 5"); // rafraîchissement automatique de la page toutes les 5 secondes

client.println(); //saut de ligne obligatoire avant la suite

client.println("<!DOCTYPE HTML>"); // informe le navigateur du type de document à afficher

client.println("<html>"); //début du code html

client.println("<head>"); //entête

client.println("<title>La Maison du Futur connectée</title>"); //titre de la fenêtre

//client.println("link rel=""icon"" type=""image/png"" href=""image/favicon.png""/>");

client.println("<meta charset=""UTF-8"">");

// CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS //CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS

client.println("<style type=text/css>body{margin-left:auto;margin-right:auto;font-family:""Arial Narrow"",serif;text-align:center;background-color :#F9F9F9;max-width:800px;}");

client.println("#entete{text-align:center;margin-left:20%;margin-right:auto;min-width:600px;padding-bottom:20px;}");

client.println("#valeur {padding-top:40px;margin-left:auto;padding-bottom:340px;font-size:24px;margin-right:auto;min-width:600px;}");

client.println("h2{color:#07578a;}");

client.println("h1{font-size:200%;}");

client.println(".valeur {background-color:#07578a;border-style:solid;border-width:3px;vertical-align:middle;text-align:center;color:#f9f9f9}</style>");

// CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS //CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS // CSS

// REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU //

client.println("</head>");//fin d'entête

client.println("<body>"); //corps

client.println("<div id=""CentrePage"">");

client.println("<h1>La Maison du Futur connectée</h1>"); //titre en grosse lettres

client.println("<h2>Voici le dernier relevé reçu le ");

client.print(date[0]);

client.print(date[1]);

client.print("/");

client.print(date[2]);

client.print(date[3]);

client.print("/");

client.print(date[4]);

client.print(date[5]);

client.print(date[6]);

client.print(date[7]);

client.print(" à ");

client.print(heure[0]);

client.print(heure[1]);

client.print(":");

client.print(heure[2]);

client.print(heure[3]);

client.print(":");

client.print(heure[4]);

client.print(heure[5]);

client.print(".</h2>");

client.println("<hr>"); //ligne horizontale

client.println("<table id=""valeur""><tr>");

client.println("<td class=""valeur"" colspan=""2""><h3> Détail du relevé <br/></h3></td></tr>");

client.println("<tr><td class=""valeur""> Lumière :</td>");

client.println("<td class=""valeur"">");

if(maison.lumiere == 0){

client.print("Fermer");

}

if(maison.lumiere == 1){

client.print("Ouvert");

}

client.print(" </td></tr>");

client.println("<tr><td class=""valeur""> Luminosité extérieur :</td>");

client.println("<td class=""valeur"">");

client.print(maison.lumex);

client.print(" Lux</td></tr>");

client.println("<tr><td class=""valeur""> Volet n°1 :</td>");

client.println("<td class=""valeur"">");

if( maison.volet1 == 0){

client.print("Fermer");

}

if( maison.volet1 == 1){

client.print("Ouvert");

}

client.print(" </td></tr>");

client.println("<tr><td class=""valeur""> Volet n°2 :</td>");

client.println("<td class=""valeur"">");

if( maison.volet2 == 0){

client.print("Fermer");

}

if( maison.volet2 == 1){

client.print("Ouvert");

}

client.print(" </td></tr>");

client.println("<tr><td class=""valeur""> Radiateur :</td>");

client.println("<td class=""valeur"">");

if(maison.radiateur == 0){

client.print("Eteint");

}

if(maison.radiateur == 1){

client.print("Allumer");

}

client.print(" </td></tr>");

client.println("<tr><td class=""valeur""> Température :</td>");

client.println("<td class=""valeur"">");

client.print(maison.temperature);

client.print(" °c</td></tr>");

client.println("<tr><td class=""valeur""> Consomation d'énergie :</td>");

client.println("<td class=""valeur"">");

client.print(maison.air);

client.print(" W/h</td></tr>");

client.println("<tr><td class=""valeur""> Taux d'humidité :</td>");

client.println("<td class=""valeur"">");

client.print(maison.humidite);

client.print(" %</td></tr>");

client.println("<tr><td class=""valeur""> Qualité de l'air :</td>");

client.println("<td class=""valeur"">");

client.print(maison.conso);

client.print(" (1 à 10)</td>");

client.print("</tr></table>");

// REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU // REMPLISSAGE TABLEAU //

client.println("</div>");

client.println("<hr>"); //ligne horizontale

client.println("<div id=""piedpage""><h3>Bonjour. Si Vous souhaitez interagir avec votre maison via votre télèphone télècharger l'application !</h3></div>");

client.println("</body>"); //fin du corps

client.println("</html>"); //fin du code html

client.stop(); //on déconnecte le client

Serial.println("Fin de communication avec le client");

void resetTrame(Maison); // Fonction qui supprime le contenue de la trame precedente

}

}

}

void serialEvent1(){ // Programme d'interuption qui se lance a la fin de chaque loop

while (Serial1.available()) {

// get the new byte:

char lettre = (char)Serial1.read(); // Lecture du charactère envoyer via bluetooth

// add it to the inputString:

trame += lettre; // Ajout a la trame

// if the incoming character is a newline, set a flag so the main loop can

// do something about it:

if (lettre == '\n') { // Si la phrase et terminer

trameComplete = true; // La trame est complète

Serial.println(trame); // On l'affiche

}

}

}

## decodage.h

#ifndef \_\_DECODAGE\_H\_\_

#define \_\_DECODAGE\_H\_\_

#include"Arduino.h"

#include "maison.h"

void recuperationTrame(Maison, String, boolean);

void traductionTrame(Maison);

void resetTrame(Maison);

Maison\* getAdresse();

#endif

## decodage.cpp

#include "decodage.h"

#include "arduino.h"

#include "maison.h"

void recuperationTrame(Maison maison, String trame,boolean trameComplete){

Serial.println(trame);

char valeur;

for(int i=0;i<=trame.length();i++){

if(maison.selecteur==0 && trame[i]!= '!'){

valeur = trame[i];

maison.trameLumiere += valeur;

}

if(maison.selecteur==1 && trame[i]!= '!'){

valeur = trame[i];

maison.lumex += valeur;

}

if(maison.selecteur==2 && trame[i]!= '!'){

valeur = trame[i];

maison.trameVolet1 += valeur;

}

if(maison.selecteur==3 && trame[i]!= '!'){

valeur = trame[i];

maison.trameVolet2 += valeur;

}

if(maison.selecteur==4 && trame[i]!= '!'){

valeur = trame[i];

maison.trameRadiateur += valeur;

}

if(maison.selecteur==5 && trame[i]!= '!'){

valeur = trame[i];

maison.temperature += valeur;

}

if(maison.selecteur==6 && trame[i]!= '!'){

valeur = trame[i];

maison.air += valeur;

}

if(maison.selecteur==7 && trame[i]!= '!'){

valeur = trame[i];

maison.humidite += valeur;

}

if(maison.selecteur==8 && trame[i]!= '!'){

valeur = trame[i];

maison.conso += valeur;

}

if(maison.selecteur==9 && trame[i]!= '!'){

valeur = trame[i];

maison.energie += valeur;

}

if(trame[i]== '!'){

maison.selecteur++;

}

}

Serial.print("lumiere : ");

Serial.println(maison.trameLumiere);

Serial.print("lumiere exterieur : ");

Serial.println(maison.lumex);

Serial.print("volet 1 : ");

Serial.println(maison.trameVolet1);

Serial.print("volet 2 : ");

Serial.println(maison.trameVolet2);

Serial.print("radiateur : ");

Serial.println(maison.trameRadiateur);

Serial.print("temperature : ");

Serial.println(maison.temperature);

Serial.print("air : ");

Serial.println(maison.air);

Serial.print("humidite : ");

Serial.println(maison.humidite);

Serial.print("consomation electrique : ");

Serial.println(maison.conso);

Serial.print("energie : ");

Serial.println(maison.energie);

maison.selecteur = 0;

trame="";

trameComplete = false;

}

void traductionTrame(Maison maison){

if(maison.trameLumiere==0){

maison.lumiere=false;

}

if(maison.trameLumiere==1){

maison.lumiere=true;

}

if(maison.trameVolet1==0){

maison.volet1=false;

}

if(maison.trameVolet1==1){

maison.volet1=true;

}

if(maison.trameVolet2==0){

maison.volet2=false;

}

if(maison.trameVolet2==1){

maison.volet2=true;

}

if(maison.trameRadiateur==0){

maison.radiateur=false;

}

if(maison.trameRadiateur==1){

maison.radiateur=true;

}

}

void resetTrame(Maison maison){

maison.lumiere="";

maison.lumex="";

maison.volet1="";

maison.volet2="";

maison.radiateur="";

maison.temperature="";

maison.air="";

maison.humidite="";

maison.conso="";

maison.energie="";

}

## maison.h

#ifndef \_\_MAISON\_H\_\_

#define \_\_MAISON\_H\_\_

#include"Arduino.h"

#include"decodage.h"

class Maison

{

public:

Maison();

int adresse;

bool lumiere;

String lumex;

bool volet1;

bool volet2;

bool radiateur;

String temperature;

String air;

String humidite;

String conso;

String energie;

String trameLumiere;

String trameVolet1;

String trameVolet2;

String trameRadiateur;

int selecteur;

~Maison();

void recuperationTrame(Maison, String, boolean);

void traductionTrame(Maison);

void resetTrame(Maison);

Maison\* getAdresse();

private:

int\* adressage(Maison);

};

#endif

## maison.cpp

#include "decodage.h"

#include "arduino.h"

#include "maison.h"

Maison::Maison(){

lumiere = false;

lumex = "";

volet1 = false;

volet2 = false;

radiateur = false;

temperature = "";

air = "";

humidite = "";

conso = "";

energie = "";

trameLumiere = "";

trameVolet1 = "";

trameVolet2 = "";

trameRadiateur = "";

selecteur = 0;

}

Maison::~Maison(){

}

void recuperationTrame(Maison maison, String trame,boolean trameComplete){

Serial.println(trame);

char valeur;

for(int i=0;i<=trame.length();i++){

if(maison.selecteur==0 && trame[i]!= '!'){

valeur = trame[i];

maison.trameLumiere += valeur;

}

if(maison.selecteur==1 && trame[i]!= '!'){

valeur = trame[i];

maison.lumex += valeur;

}

if(maison.selecteur==2 && trame[i]!= '!'){

valeur = trame[i];

maison.trameVolet1 += valeur;

}

if(maison.selecteur==3 && trame[i]!= '!'){

valeur = trame[i];

maison.trameVolet2 += valeur;

}

if(maison.selecteur==4 && trame[i]!= '!'){

valeur = trame[i];

maison.trameRadiateur += valeur;

}

if(maison.selecteur==5 && trame[i]!= '!'){

valeur = trame[i];

maison.temperature += valeur;

}

if(maison.selecteur==6 && trame[i]!= '!'){

valeur = trame[i];

maison.air += valeur;

}

if(maison.selecteur==7 && trame[i]!= '!'){

valeur = trame[i];

maison.humidite += valeur;

}

if(maison.selecteur==8 && trame[i]!= '!'){

valeur = trame[i];

maison.conso += valeur;

}

if(maison.selecteur==9 && trame[i]!= '!'){

valeur = trame[i];

maison.energie += valeur;

}

if(trame[i]== '!'){

maison.selecteur++;

}

}

Serial.print("lumiere : ");

Serial.println(maison.trameLumiere);

Serial.print("lumiere exterieur : ");

Serial.println(maison.lumex);

Serial.print("volet 1 : ");

Serial.println(maison.trameVolet1);

Serial.print("volet 2 : ");

Serial.println(maison.trameVolet2);

Serial.print("radiateur : ");

Serial.println(maison.trameRadiateur);

Serial.print("temperature : ");

Serial.println(maison.temperature);

Serial.print("air : ");

Serial.println(maison.air);

Serial.print("humidite : ");

Serial.println(maison.humidite);

Serial.print("consomation electrique : ");

Serial.println(maison.conso);

Serial.print("energie : ");

Serial.println(maison.energie);

maison.selecteur = 0;

trame="";

trameComplete = false;

}

void traductionTrame(Maison maison){

if(maison.trameLumiere==0){

maison.lumiere=false;

}

if(maison.trameLumiere==1){

maison.lumiere=true;

}

if(maison.trameVolet1==0){

maison.volet1=false;

}

if(maison.trameVolet1==1){

maison.volet1=true;

}

if(maison.trameVolet2==0){

maison.volet2=false;

}

if(maison.trameVolet2==1){

maison.volet2=true;

}

if(maison.trameRadiateur==0){

maison.radiateur=false;

}

if(maison.trameRadiateur==1){

maison.radiateur=true;

}

}

void resetTrame(Maison maison){

maison.lumiere="";

maison.lumex="";

maison.volet1="";

maison.volet2="";

maison.radiateur="";

maison.temperature="";

maison.air="";

maison.humidite="";

maison.conso="";

maison.energie="";

}

# Programme de l'application mobile :

## Q:\Projet_Domotique_GITBTS\Rapport\ANNexe\Application_Code_1.pngBlock :

## Interface graphique :

# Câblage du système :Montage_final.png