Project hardware testing ( relay and current sensor )

// Include libraries

#include <Bridge.h>

#include <YunServer.h>

#include <YunClient.h>

#include <Console.h>

// Create Yun server

YunServer server;

// Define current sensor & relay pins

#define CURRENT\_SENSOR A0

#define RELAY\_PIN 8

// Define measurement variables

float amplitude\_current;

float effective\_value;

float effective\_voltage = 5; // Set voltage to 230V (Europe) or 110V (US)

float effective\_power;

float zero\_sensor;

void setup(void)

{

// Start bridge

Bridge.begin();

// Initialize serial connection

Console.begin();

// Wait for Console port to connect

while (!Console);

Console.println("Hi,wifi is connect !!");

// Initialize relay pin

pinMode(RELAY\_PIN,OUTPUT);

// Start server

server.listenOnLocalhost();

server.begin();

// Calibrate sensor with null current

zero\_sensor = getSensorValue();

Console.print("Zero point sensor: ");

Console.println(zero\_sensor);

Console.println("");

}

void loop(void)

{

// Get clients coming from server

YunClient client = server.accept();

// There is a new client?

if (client) {

// Process request

process(client);

// Close connection and free resources.

client.stop();

}

// Perform power measurement

float sensor\_value = getSensorValue();

Console.print("Sensor value: ");

Console.println(sensor\_value);

// Convert to current

amplitude\_current=(float)(sensor\_value-zero\_sensor)/1024\*5/185\*1000000;

effective\_value=amplitude\_current/1.414;

// Plot data

Console.println("Current amplitude (in mA): ");

Console.println(amplitude\_current,1);

Console.println("Current effective value (in mA)");

Console.println(effective\_value,1);

Console.println("Effective power (in W): ");

Console.println(abs(effective\_value\*effective\_voltage/1000),1);

Console.println("");

// Poll every 50ms

delay(5000);

}

// Process incoming command

void process(YunClient client) {

// read the command

String command = client.readStringUntil('/');

// is "digital" command?

if (command == "digital") {

digitalCommand(client);

}

// is "analog" command?

if (command == "analog") {

analogCommand(client);

}

// is "mode" command?

if (command == "mode") {

modeCommand(client);

}

}

void digitalCommand(YunClient client) {

int pin, value;

// Read pin number

pin = client.parseInt();

// If the next character is a '/' it means we have an URL

// with a value like: "/digital/13/1"

if (client.read() == '/') {

value = client.parseInt();

digitalWrite(pin, value);

}

else {

value = digitalRead(pin);

}

// Send feedback to client

client.print(F("Pin D"));

client.print(pin);

client.print(F(" set to "));

client.println(value);

// Update datastore key with the current pin value

String key = "D";

key += pin;

Bridge.put(key, String(value));

}

void analogCommand(YunClient client) {

int pin, value;

// Read pin number

pin = client.parseInt();

// If the next character is a '/' it means we have an URL

// with a value like: "/analog/5/120"

if (client.read() == '/') {

// Read value and execute command

value = client.parseInt();

analogWrite(pin, value);

// Send feedback to client

client.print(F("Pin D"));

client.print(pin);

client.print(F(" set to analog "));

client.println(value);

// Update datastore key with the current pin value

String key = "D";

key += pin;

Bridge.put(key, String(value));

}

else {

// Read analog pin

value = analogRead(pin);

// Send feedback to client

client.print(F("Pin A"));

client.print(pin);

client.print(F(" reads analog "));

client.println(value);

// Update datastore key with the current pin value

String key = "A";

key += pin;

Bridge.put(key, String(value));

}

}

void modeCommand(YunClient client) {

int pin;

// Read pin number

pin = client.parseInt();

// If the next character is not a '/' we have a malformed URL

if (client.read() != '/') {

client.println(F("error"));

return;

}

String mode = client.readStringUntil('\r');

if (mode == "input") {

pinMode(pin, INPUT);

// Send feedback to client

client.print(F("Pin D"));

client.print(pin);

client.print(F(" configured as INPUT!"));

return;

}

if (mode == "output") {

pinMode(pin, OUTPUT);

// Send feedback to client

client.print(F("Pin D"));

client.print(pin);

client.print(F(" configured as OUTPUT!"));

return;

}

client.print(F("error: invalid mode "));

client.print(mode);

}

// Get the reading from the current sensor

float getSensorValue()

{

int sensorValue;

float avgSensor = 0;

int nb\_measurements = 100;

for (int i = 0; i < nb\_measurements; i++) {

sensorValue = analogRead(CURRENT\_SENSOR);

avgSensor = avgSensor + float(sensorValue);

}

avgSensor = avgSensor/float(nb\_measurements);

return avgSensor;

}