Check point1

#include <LWiFi.h>

#include <WiFiClient.h>

#include "MCS.h"

// Assign AP ssid / password here

#define \_SSID "yingde"

#define \_KEY "011701170117"

// Assign device id / key of your test device

MCSDevice mcs("DQMKAdNE", "XiS0KSsXtBBqwP8m");

// Assign chann el id

// The test device should have 2 channel

// the first channel should be "Controller" - "On/Off"

// the secord channel should be "Display" - "On/Off"

MCSControllerOnOff led("xiaode");

MCSDisplayOnOff remote("Xiaode");

#define LED\_PIN 7

void setup() {

// setup Serial output at 9600

Serial.begin(9600);

// setup LED/Button pin

pinMode(LED\_PIN, OUTPUT);

// setup Wifi connection

while(WL\_CONNECTED != WiFi.status())

{

Serial.print("WiFi.begin(");

Serial.print(\_SSID);

Serial.print(",");

Serial.print(\_KEY);

Serial.println(")...");

WiFi.begin(\_SSID, \_KEY);

}

Serial.println("WiFi connected !!");

// setup MCS connection

mcs.addChannel(led);

mcs.addChannel(remote);

while(!mcs.connected())

{

Serial.println("MCS.connect()...");

mcs.connect();

}

Serial.println("MCS connected !!");

// read LED value from MCS server

while(!led.valid())

{

Serial.println("read LED value from MCS...");

led.value();

}

Serial.print("done, LED value = ");

Serial.println(led.value());

digitalWrite(LED\_PIN, led.value() ? HIGH : LOW);

}

void loop() {

// call process() to allow background processing, add timeout to avoid high cpu usage

Serial.print("process(");

Serial.print(millis());

Serial.println(")");

mcs.process(100);

// updated flag will be cleared in process(), user must check it after process() call.

if(led.updated())

{

Serial.print("LED updated, new value = ");

Serial.println(led.value());

digitalWrite(LED\_PIN, led.value() ? HIGH : LOW);

if(!remote.set(led.value()))

{

Serial.print("Failed to update remote");

Serial.println(remote.value());

}

}

// check if need to re-connect

while(!mcs.connected())

{

Serial.println("re-connect to MCS...");

mcs.connect();

if(mcs.connected())

Serial.println("MCS connected !!");

}

}

Check point2

#include <LWiFi.h>

#include <WiFiClient.h>

#include "MCS.h"

// Assign AP ssid / password here

#define \_SSID "yingde"

#define \_KEY "011701170117"

// Assign device id / key of your test device

MCSDevice mcs("DQMKAdNE", "XiS0KSsXtBBqwP8m");

// Assign channel id

// The test device should have 2 channel

// the first channel should be "Controller" - "On/Off"

// the secord channel should be "Display" - "On/Off"

MCSControllerOnOff led("xiaode");

MCSDisplayOnOff remote("Xiaode");

MCSDisplayInteger Temp("0172");

#define LED\_PIN 7

void setup() {

// setup Serial output at 9600

Serial.begin(9600);

// setup LED/Button pin

pinMode(LED\_PIN, OUTPUT);

// setup Wifi connection

while(WL\_CONNECTED != WiFi.status())

{

Serial.print("WiFi.begin(");

Serial.print(\_SSID);

Serial.print(",");

Serial.print(\_KEY);

Serial.println(")...");

WiFi.begin(\_SSID, \_KEY);

}

Serial.println("WiFi connected !!");

// setup MCS connection

mcs.addChannel(led);

mcs.addChannel(remote);

mcs.addChannel(Temp);

while(!mcs.connected())

{

Serial.println("MCS.connect()...");

mcs.connect();

}

Serial.println("MCS connected !!");

// read LED value from MCS server

while(!led.valid())

{

Serial.println("read LED value from MCS...");

led.value();

}

Serial.print("done, LED value = ");

Serial.println(led.value());

digitalWrite(LED\_PIN, led.value() ? HIGH : LOW);

}

void loop() {

// call process() to allow background processing, add timeout to avoid high cpu usage

Serial.print("process(");

Serial.print(millis());

Serial.println(")");

mcs.process(100);

// updated flag will be cleared in process(), user must check it after process() call.

if(led.updated())

{

Serial.print("LED updated, new value = ");

Serial.println(led.value());

digitalWrite(LED\_PIN, led.value() ? HIGH : LOW);

if(!remote.set(led.value()))

{

Serial.print("Failed to update remote");

Serial.println(remote.value());

}

}

if(!Temp.set(analogRead(A0)))

{

Serial.print("Temp: ");

Serial.println(Temp.value());

}

// check if need to re-connect

while(!mcs.connected())

{

Serial.println("re-connect to MCS...");

mcs.connect();

if(mcs.connected())

Serial.println("MCS connected !!");

}

}