// Uncomment the following line to enable serial debug output

#define ENABLE\_DEBUG

#ifdef ENABLE\_DEBUG

#define DEBUG\_ESP\_PORT Serial

#define NODEBUG\_WEBSOCKETS

#define NDEBUG

#endif

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <SinricPro.h>

#include <SinricProSwitch.h>

#include <map>

#define WIFI\_SSID "IOT-2.4G"

#define WIFI\_PASS "2022iedcmbcet"

#define APP\_KEY "08bf5a87-3b03-469b-bc95-09d7af608707" // Should look like "de0bxxxx-1x3x-4x3x-ax2x-5dabxxxxxxxx"

#define APP\_SECRET "c52e7831-4f1d-4b33-a4c4-415f73bb1ca1-534b55ca-e802-4c16-96c5-ccf490ecdfda" // Should look like "5f36xxxx-x3x7-4x3x-xexe-e86724a9xxxx-4c4axxxx-3x3x-x5xe-x9x3-333d65xxxxxx"

//Enter the device IDs here

#define device\_ID\_1 "65005548e2a1e41147659333"

#define device\_ID\_2 "6500557cb1deae87502153be"

#define device\_ID\_3 "65005591b1deae8750215407"

//#define device\_ID\_4 "60764aa148ccc14a4674c047"

// define the GPIO connected with Relays and switches

#define RelayPin1 5 //D1

#define RelayPin2 4 //D2

#define RelayPin3 14 //D5

//#define RelayPin4 12 //D6

#define SwitchPin1 10 //SD3

#define SwitchPin2 0 //D3

#define SwitchPin3 13 //D7

//#define SwitchPin4 3 //RX

#define wifiLed 16 //D0

// comment the following line if you use a toggle switches instead of tactile buttons

//#define TACTILE\_BUTTON 1

#define BAUD\_RATE 9600

#define DEBOUNCE\_TIME 250

typedef struct { // struct for the std::map below

int relayPIN;

int flipSwitchPIN;

} deviceConfig\_t;

// this is the main configuration

// please put in your deviceId, the PIN for Relay and PIN for flipSwitch

// this can be up to N devices...depending on how much pin's available on your device ;)

// right now we have 4 devicesIds going to 4 relays and 4 flip switches to switch the relay manually

std::map<String, deviceConfig\_t> devices = {

//{deviceId, {relayPIN, flipSwitchPIN}}

{device\_ID\_1, { D1, SwitchPin1 }},

{device\_ID\_2, { D2, SwitchPin2 }},

{device\_ID\_3, { D5, SwitchPin3 }},

// {device\_ID\_4, { RelayPin4, SwitchPin4 }}

};

typedef struct { // struct for the std::map below

String deviceId;

bool lastFlipSwitchState;

unsigned long lastFlipSwitchChange;

} flipSwitchConfig\_t;

std::map<int, flipSwitchConfig\_t> flipSwitches; // this map is used to map flipSwitch PINs to deviceId and handling debounce and last flipSwitch state checks

// it will be setup in "setupFlipSwitches" function, using informations from devices map

void setupRelays() {

for (auto &device : devices) { // for each device (relay, flipSwitch combination)

int relayPIN = device.second.relayPIN; // get the relay pin

pinMode(relayPIN, OUTPUT); // set relay pin to OUTPUT

digitalWrite(relayPIN, HIGH);

}

}

void setupFlipSwitches() {

for (auto &device : devices) { // for each device (relay / flipSwitch combination)

flipSwitchConfig\_t flipSwitchConfig; // create a new flipSwitch configuration

flipSwitchConfig.deviceId = device.first; // set the deviceId

flipSwitchConfig.lastFlipSwitchChange = 0; // set debounce time

flipSwitchConfig.lastFlipSwitchState = true; // set lastFlipSwitchState to false (LOW)--

int flipSwitchPIN = device.second.flipSwitchPIN; // get the flipSwitchPIN

flipSwitches[flipSwitchPIN] = flipSwitchConfig; // save the flipSwitch config to flipSwitches map

pinMode(flipSwitchPIN, INPUT\_PULLUP); // set the flipSwitch pin to INPUT

}

}

bool onPowerState(String deviceId, bool &state)

{

Serial.printf("%s: %s\r\n", deviceId.c\_str(), state ? "on" : "off");

int relayPIN = devices[deviceId].relayPIN; // get the relay pin for corresponding device

digitalWrite(relayPIN, !state); // set the new relay state

return true;

}

void handleFlipSwitches() {

unsigned long actualMillis = millis(); // get actual millis

for (auto &flipSwitch : flipSwitches) { // for each flipSwitch in flipSwitches map

unsigned long lastFlipSwitchChange = flipSwitch.second.lastFlipSwitchChange; // get the timestamp when flipSwitch was pressed last time (used to debounce / limit events)

if (actualMillis - lastFlipSwitchChange > DEBOUNCE\_TIME) { // if time is > debounce time...

int flipSwitchPIN = flipSwitch.first; // get the flipSwitch pin from configuration

bool lastFlipSwitchState = flipSwitch.second.lastFlipSwitchState; // get the lastFlipSwitchState

bool flipSwitchState = digitalRead(flipSwitchPIN); // read the current flipSwitch state

if (flipSwitchState != lastFlipSwitchState) { // if the flipSwitchState has changed...

#ifdef TACTILE\_BUTTON

if (flipSwitchState) { // if the tactile button is pressed

#endif

flipSwitch.second.lastFlipSwitchChange = actualMillis; // update lastFlipSwitchChange time

String deviceId = flipSwitch.second.deviceId; // get the deviceId from config

int relayPIN = devices[deviceId].relayPIN; // get the relayPIN from config

bool newRelayState = !digitalRead(relayPIN); // set the new relay State

digitalWrite(relayPIN, newRelayState); // set the trelay to the new state

SinricProSwitch &mySwitch = SinricPro[deviceId]; // get Switch device from SinricPro

mySwitch.sendPowerStateEvent(!newRelayState); // send the event

#ifdef TACTILE\_BUTTON

}

#endif

flipSwitch.second.lastFlipSwitchState = flipSwitchState; // update lastFlipSwitchState

}

}

}

}

void setupWiFi()

{

Serial.printf("\r\n[Wifi]: Connecting");

WiFi.begin(WIFI\_SSID, WIFI\_PASS);

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

{

Serial.printf(".");

delay(250);

}

digitalWrite(wifiLed, LOW);

Serial.printf("connected!\r\n[WiFi]: IP-Address is %s\r\n", WiFi.localIP().toString().c\_str());

}

void setupSinricPro()

{

for (auto &device : devices)

{

const char \*deviceId = device.first.c\_str();

SinricProSwitch &mySwitch = SinricPro[deviceId];

mySwitch.onPowerState(onPowerState);

}

SinricPro.begin(APP\_KEY, APP\_SECRET);

SinricPro.restoreDeviceStates(true);

}

void setup()

{

Serial.begin(BAUD\_RATE);

pinMode(wifiLed, OUTPUT);

digitalWrite(wifiLed, HIGH);

setupRelays();

setupFlipSwitches();

setupWiFi();

setupSinricPro();

}

void loop()

{

SinricPro.handle();

handleFlipSwitches();

}