```
#include "RMaker.h"
#include "WiFi.h"
#include "WiFiProv.h"

unsigned long previousReconnectAttempt = 0;
const unsigned long reconnectInterval = 10000;  // 10 seconds
const char *service_name = "PROV_12345";
const char *pop = "1234567";

// define the Device Names
char deviceName_1[] = "Switch1";
char deviceName_2[] = "Switch2";
char deviceName_3[] = "Switch3";
char deviceName_4[] = "Switch4";
char deviceName_5[] = "Switch5";
char deviceName_6[] = "Switch6";


static uint8_t RelayPin1 = 23;  //D23
static uint8_t RelayPin2 = 22;  //D22
static uint8_t RelayPin3 = 21;  //D21
static uint8_t RelayPin4 = 19;  //D19
static uint8_t RelayPin5 = 18;  //D18
static uint8_t RelayPin6 = 5;   //D5

static uint8_t SwitchPin1 = 13;  //D13
static uint8_t SwitchPin2 = 12;  //D12
static uint8_t SwitchPin3 = 14;  //D14
static uint8_t SwitchPin4 = 27;  //D27
static uint8_t SwitchPin5 = 33;  //D33
static uint8_t SwitchPin6 = 32;  //D32

static uint8_t wifiLed    = 2;   //D2
static uint8_t gpio_reset = 0;

bool toggleState_1 = LOW;
bool toggleState_2 = LOW;
bool toggleState_3 = LOW;
bool toggleState_4 = LOW;
bool toggleState_5 = LOW;
bool toggleState_6 = LOW;


bool SwitchState_1 = LOW;
bool SwitchState_2 = LOW;
bool SwitchState_3 = LOW;
```

```
bool SwitchState_4 = LOW;
bool SwitchState_5 = LOW;
bool SwitchState_6 = LOW;


static Switch my_switch1(deviceName_1, &RelayPin1);
static Switch my_switch2(deviceName_2, &RelayPin2);
static Switch my_switch3(deviceName_3, &RelayPin3);
static Switch my_switch4(deviceName_4, &RelayPin4);
static Switch my_switch5(deviceName_5, &RelayPin5);
static Switch my_switch6(deviceName_6, &RelayPin6);

void sysProvEvent(arduino_event_t *sys_event)
{
    switch (sys_event->event_id) {
        case ARDUINO_EVENT_PROV_START:
#if CONFIG_IDF_TARGET_ESP32
        Serial.printf("\nProvisioning Started with name \"%s\" and PoP \"%s\" on
BLE\n", service_name, pop);
        printQR(service_name, pop, "ble");
#else
        Serial.printf("\nProvisioning Started with name \"%s\" and PoP \"%s\" on
SoftAP\n", service_name, pop);
        printQR(service_name, pop, "softap");
#endif
        break;
        case ARDUINO_EVENT_WIFI_STA_CONNECTED:
        Serial.printf("\nConnected to Wi-Fi!\n");
        digitalWrite(wifiLed, true);
        break;
    }
}

void write_callback(Device *device, Param *param, const param_val_t val, void
*priv_data, write_ctx_t *ctx)
{
    const char *device_name = device->getDeviceName();
    const char *param_name = param->getParamName();

    if(strcmp(device_name, deviceName_1) == 0) {

      Serial.printf("Lightbulb = %s\n", val.val.b? "true" : "false");

      if(strcmp(param_name, "Power") == 0) {
          Serial.printf("Received value = %s for %s - %s\n", val.val.b? "true" :
"false", device_name, param_name);
        toggleState_1 = val.val.b;
```

```
      (toggleState_1 == false) ? digitalWrite(RelayPin1, HIGH) :
digitalWrite(RelayPin1, LOW);
      param->updateAndReport(val);
    }

  } else if(strcmp(device_name, deviceName_2) == 0) {

    Serial.printf("Switch value = %s\n", val.val.b? "true" : "false");

    if(strcmp(param_name, "Power") == 0) {
      Serial.printf("Received value = %s for %s - %s\n", val.val.b? "true" :
"false", device_name, param_name);
      toggleState_2 = val.val.b;
      (toggleState_2 == false) ? digitalWrite(RelayPin2, HIGH) :
digitalWrite(RelayPin2, LOW);
      param->updateAndReport(val);
    }

  } else if(strcmp(device_name, deviceName_3) == 0) {

    Serial.printf("Switch value = %s\n", val.val.b? "true" : "false");

    if(strcmp(param_name, "Power") == 0) {
      Serial.printf("Received value = %s for %s - %s\n", val.val.b? "true" :
"false", device_name, param_name);
      toggleState_3 = val.val.b;
      (toggleState_3 == false) ? digitalWrite(RelayPin3, HIGH) :
digitalWrite(RelayPin3, LOW);
      param->updateAndReport(val);
    }

  } else if(strcmp(device_name, deviceName_4) == 0) {

    Serial.printf("Switch value = %s\n", val.val.b? "true" : "false");

    if(strcmp(param_name, "Power") == 0) {
      Serial.printf("Received value = %s for %s - %s\n", val.val.b? "true" :
"false", device_name, param_name);
      toggleState_4 = val.val.b;
      (toggleState_4 == false) ? digitalWrite(RelayPin4, HIGH) :
digitalWrite(RelayPin4, LOW);
      param->updateAndReport(val);
    }

  } else if(strcmp(device_name, deviceName_5) == 0) {

    Serial.printf("Lightbulb = %s\n", val.val.b? "true" : "false");
```

```
      if(strcmp(param_name, "Power") == 0) {
        Serial.printf("Received value = %s for %s - %s\n", val.val.b? "true" :
"false", device_name, param_name);
        toggleState_5 = val.val.b;
        (toggleState_5 == false) ? digitalWrite(RelayPin5, HIGH) :
digitalWrite(RelayPin5, LOW);
        param->updateAndReport(val);
      }

    } else if(strcmp(device_name, deviceName_6) == 0) {

      Serial.printf("Switch value = %s\n", val.val.b? "true" : "false");

      if(strcmp(param_name, "Power") == 0) {
        Serial.printf("Received value = %s for %s - %s\n", val.val.b? "true" :
"false", device_name, param_name);
        toggleState_6 = val.val.b;
        (toggleState_6 == false) ? digitalWrite(RelayPin6, HIGH) :
digitalWrite(RelayPin6, LOW);
        param->updateAndReport(val);
      }

    }

}

void manual_control()
{
  if (digitalRead(SwitchPin1) == LOW && SwitchState_1 == LOW) {
    digitalWrite(RelayPin1, LOW);
    toggleState_1 = 1;
    SwitchState_1 = HIGH;
    my_switch1.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_1);
    Serial.println("Switch-1 on");
  }
  if (digitalRead(SwitchPin1) == HIGH && SwitchState_1 == HIGH) {
    digitalWrite(RelayPin1, HIGH);
    toggleState_1 = 0;
    SwitchState_1 = LOW;
    my_switch1.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_1);
    Serial.println("Switch-1 off");
  }
  if (digitalRead(SwitchPin2) == LOW && SwitchState_2 == LOW) {
    digitalWrite(RelayPin2, LOW);
```

```
    toggleState_2 = 1;
    SwitchState_2 = HIGH;
    my_switch2.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_2);
    Serial.println("Switch-2 on");
  }
  if (digitalRead(SwitchPin2) == HIGH && SwitchState_2 == HIGH) {
    digitalWrite(RelayPin2, HIGH);
    toggleState_2 = 0;
    SwitchState_2 = LOW;
    my_switch2.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_2);
    Serial.println("Switch-2 off");
  }
  if (digitalRead(SwitchPin3) == LOW && SwitchState_3 == LOW) {
    digitalWrite(RelayPin3, LOW);
    toggleState_3 = 1;
    SwitchState_3 = HIGH;
    my_switch3.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_3);
    Serial.println("Switch-3 on");
  }
  if (digitalRead(SwitchPin3) == HIGH && SwitchState_3 == HIGH) {
    digitalWrite(RelayPin3, HIGH);
    toggleState_3 = 0;
    SwitchState_3 = LOW;
    my_switch3.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_3);
    Serial.println("Switch-3 off");
  }
  if (digitalRead(SwitchPin4) == LOW && SwitchState_4 == LOW) {
    digitalWrite(RelayPin4, LOW);
    toggleState_4 = 1;
    SwitchState_4 = HIGH;
    my_switch4.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_4);
    Serial.println("Switch-4 on");
  }
  if (digitalRead(SwitchPin4) == HIGH && SwitchState_4 == HIGH) {
    digitalWrite(RelayPin4, HIGH);
    toggleState_4 = 0;
    SwitchState_4 = LOW;
    my_switch4.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_4);
    Serial.println("Switch-4 off");
  }
  if (digitalRead(SwitchPin5) == LOW && SwitchState_5 == LOW) {
```

```
    digitalWrite(RelayPin5, LOW);
    toggleState_5 = 1;
    SwitchState_5 = HIGH;
    my_switch5.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_5);
    Serial.println("Switch-5 on");
  }
  if (digitalRead(SwitchPin5) == HIGH && SwitchState_5 == HIGH) {
    digitalWrite(RelayPin5, HIGH);
    toggleState_5 = 0;
    SwitchState_5 = LOW;
    my_switch5.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_5);
    Serial.println("Switch-5 off");
  }
  if (digitalRead(SwitchPin6) == LOW && SwitchState_6 == LOW) {
    digitalWrite(RelayPin6, LOW);
    toggleState_6 = 1;
    SwitchState_6 = HIGH;
    my_switch6.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_6);
    Serial.println("Switch-6 on");
  }
  if (digitalRead(SwitchPin6) == HIGH && SwitchState_6 == HIGH) {
    digitalWrite(RelayPin6, HIGH);
    toggleState_6 = 0;
    SwitchState_6 = LOW;
    my_switch6.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
toggleState_6);
    Serial.println("Switch-6 off");
  }

}

void setup()
{
    WiFi.setAutoReconnect(true);  // Auto reconnect enabled
    WiFi.persistent(true);        // Save WiFi credentials

    uint32_t chipId = 0;

    Serial.begin(115200);

    // Set the Relays GPIOs as output mode
    pinMode(RelayPin1, OUTPUT);
    pinMode(RelayPin2, OUTPUT);
    pinMode(RelayPin3, OUTPUT);
```

```
pinMode(RelayPin4, OUTPUT);
pinMode(RelayPin5, OUTPUT);
pinMode(RelayPin6, OUTPUT);
pinMode(wifiLed, OUTPUT);

// Configure the input GPIOs
pinMode(SwitchPin1, INPUT_PULLUP);
pinMode(SwitchPin2, INPUT_PULLUP);
pinMode(SwitchPin3, INPUT_PULLUP);
pinMode(SwitchPin4, INPUT_PULLUP);
pinMode(SwitchPin5, INPUT_PULLUP);
pinMode(SwitchPin6, INPUT_PULLUP);
pinMode(gpio_reset, INPUT);

// Write to the GPIOs the default state on booting
digitalWrite(RelayPin1, !toggleState_1);
digitalWrite(RelayPin2, !toggleState_2);
digitalWrite(RelayPin3, !toggleState_3);
digitalWrite(RelayPin4, !toggleState_4);
digitalWrite(RelayPin5, !toggleState_5);
digitalWrite(RelayPin6, !toggleState_6);
digitalWrite(wifiLed, LOW);

Node my_node;
my_node = RMaker.initNode("ESP32_Relay_6");

my_switch1.addCb(write_callback);
my_switch2.addCb(write_callback);
my_switch3.addCb(write_callback);
my_switch4.addCb(write_callback);
my_switch5.addCb(write_callback);
my_switch6.addCb(write_callback);

my_node.addDevice(my_switch1);
my_node.addDevice(my_switch2);
my_node.addDevice(my_switch3);
my_node.addDevice(my_switch4);
my_node.addDevice(my_switch5);
my_node.addDevice(my_switch6);

RMaker.enableOTA(OTA_USING_PARAMS);

RMaker.enableTZService();
RMaker.enableSchedule();


for(int i=0; i<17; i=i+8) {
```

```
    chipId |= ((ESP×getEfuseMac() >> (40 - i)) & 0xff) << i;
  }

  Serial.printf("\nChip ID:  %d Service Name: %s\n", chipId, service_name);

  Serial.printf("\nStarting ESP-RainMaker\n");
  RMaker.start();

  WiFi.onEvent(sysProvEvent);
#if CONFIG_IDF_TARGET_ESP32
  WiFiProv.beginProvision(WIFI_PROV_SCHEME_BLE,
WIFI_PROV_SCHEME_HANDLER_FREE_BTDM, WIFI_PROV_SECURITY_1, pop,
service_name);
#else
  WiFiProv.beginProvision(WIFI_PROV_SCHEME_SOFTAP,
WIFI_PROV_SCHEME_HANDLER_NONE, WIFI_PROV_SECURITY_1, pop,
service_name);
#endif

  my_switch1.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
  my_switch2.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
  my_switch3.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
  my_switch4.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
  my_switch5.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
  my_switch6.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
false);
}

void loop()
{
  if(digitalRead(gpio_reset) == LOW) {
    Serial.printf("Reset Button Pressed!\n");
    delay(100);
    int startTime = millis();
    while(digitalRead(gpio_reset) == LOW) delay(50);
    int endTime = millis();

    if ((endTime - startTime) > 10000) {
      Serial.printf("Reset to factory.\n");
      RMakerFactoryReset(2);
    } else if ((endTime - startTime) > 3000) {
      Serial.printf("Reset Wi-Fi.\n");
```

```
          RMakerWiFiReset(2);
          RMakerWiFiReset(2);
        }
    }
    delay(100);

    if (WiFi.status() != WL_CONNECTED)
    {
      digitalWrite(wifiLed, false);
    }
    else
    {
      //Serial.println("WiFi Connected");
      digitalWrite(wifiLed, true);
    }

    manual_control();

static unsigned long lastPrint = 0;
unsigned long currentMillis = millis();

if (WiFi.status() != WL_CONNECTED) {
    digitalWrite(wifiLed, LOW);

    if (currentMillis - previousReconnectAttempt >= reconnectInterval) {
        Serial.println("WiFi disconnected. Trying to reconnect...");
        WiFi.begin();  // Try reconnect
        previousReconnectAttempt = currentMillis;
    }
} else {
    digitalWrite(wifiLed, HIGH);

    if (currentMillis - lastPrint >= 30000) {  // Log every 30 seconds
        Serial.println("WiFi still connected.");
        lastPrint = currentMillis;
    }
}
}
```