

```
/* -----
- - - - - */
// DESCRIPTIONS AND GENERAL SETTINGS
/* -----
- - - - - */
/*
https://github.com/ceviana/esp8266-pc-power-control/blob/master/README.md
/* -----
- - - - - */
/* LOCAL
ESP8266 based PC power controller - This project is
super-easy and
lightweight Wi-Fi connected ESP8266 board that polls
remote server and
allows to turn PC power ON and OFF, reset or perform
graceful shutdown.*/
/* -----
- - - - - */
/* PHP SERVER SIDE
Usage - Install example PHP task server from php_server
directory so
it is accesible on the URL you've set in firmware.ino. */
/* -----
- - - - - */
// INCLUDE EXTERNAL LIBRARIES
/* -----
- - - - - */
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
/* -----
- - - - - */
// DEFINES TIMES TO TASKS
/* -----
```

```

- - - - - */
#define USE_SERIAL Serial
// How long the PowerOFF button should be pressed to
power off PC forcefully
#define PWR_OFF_TIME 4500
// How long the button should be pressed to REBOOT, POWER
ON or RESET
#define PUSH_TIME 400
// How often the tasks server should be polled, ms
#define POLL_INTERVAL 20000
/* - - - - -
- - - - - */
// DEFINES ALL NEEDED I/O PINS
/* - - - - -
- - - - - */
/* ORIGINALS:
    #define PWR_PIN 12    #define RST_PIN 14    #define
STATUS_PIN 16    */
/* - - - - -
- - - - - */
// Pin for the PWR signal line
#define PWR_PIN 14
// Pin for the RST signal line
#define RST_PIN 12
// Pin for the status LED signal line
#define STATUS_PIN 16
// GPIO-11 LD0 >--> LED GREEN - PO2
#define LD_GRN 5
// GPIO-13 LD1 >----> LED RED - PO2
#define LD_RED 13
/* - - - - -
- - - - - */
// Wi-Fi network settings
/* - - - - -
- - - - - */

```

```

const char* ssid = "ESP_GUEST";
const char* password = "01234567890";
// URL to task providing script, that will be polled each
POLL_INTERVAL ms.
// The endpoint script should return one of the following
strings as plain text
// in order to execute desired actions: PWR_ON, PWR_OFF,
RESET, SHUTDOWN
const char* endpoint =
"http://cevana.com/pc-power/endpoint.php";
unsigned times = 0;    // counter times var
/* - - - - -
- - - - - */
// SETUP FUNCTION
/* - - - - -
- - - - - */
void setup() {
    USE_SERIAL.begin(115200);
    USE_SERIAL.setDebugOutput(true);
    USE_SERIAL.println();
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        USE_SERIAL.print('.');
        delay(1000);}
    Serial.println("");
// LED GREEN AND RED DEFINED AS OUTPUT AND INITIAL STATE
OFF
pinMode(LD_GRN, OUTPUT);
pinMode(LD_RED, OUTPUT);
digitalWrite(LD_RED, LOW);    // LD_RED OFF
digitalWrite(LD_GRN, LOW);    // LD_GRN OFF
// Synchronize time from NTP servers
configTime(3 * 3600, 0, "ua.pool.ntp.org", "time.nist.
gov");}
/* - - - - -
- - - - - */

```

```
- - - - - */
// MAIN LOOP FUNCTION
/* - - - - -
- - - - - */
void loop() {
digitalWrite(LD_GRN, HIGH); // LD_GRN ON
digitalWrite(LD_RED, LOW); // LD_RED OFF
delay(POLL_INTERVAL);
digitalWrite(LD_RED, HIGH); // LD_RED ON
digitalWrite(LD_GRN, LOW); // LD_GRN OFF
reportStatus();
pollTasks();}
/* - - - - -
- - - - - */
// FUNCTION TO SEND REPORT STATUS
/* - - - - -
- - - - - */
void reportStatus() {
Serial.print("> Status: power ");
if (isPoweredOn()) {
Serial.println("ON");
} else {
Serial.println("OFF");}}
/* - - - - -
- - - - - */
// FUNCTION
/* - - - - -
- - - - - */
void pollTasks() { times++;
if (WiFi.status() != WL_CONNECTED) {
USE_SERIAL.print("> Wi-Fi is not connected. Can not
check tasks.\n");}
HTTPClient http;
http.begin(endpoint);
http.addHeader("Content-Type",
```

```

"application/x-www-form-urlencoded; charset=UTF-8");
    char request[64];
    snprintf(request, sizeof request, "status=%d",
isPoweredOn());
    int httpCode = http.POST(request);
    USE_SERIAL.printf("|> [HTTP] POST success. Code: %d\n",
httpCode);
    USE_SERIAL.printf("|> Number of resquests: %d\n",
times);
    if (httpCode != HTTP_CODE_OK) {
        USE_SERIAL.printf("|> [HTTP] POST failed, error:
%s\n",http.errorToString(httpCode).c_str());
        http.end();
        return;}
    String payload = http.getString();
    USE_SERIAL.print("|>----->  ");
    USE_SERIAL.println(payload);
    if (payload == "PWR_OFF") {
        do_powerOff();
    } else if (payload == "PWR_ON") {
        do_powerOn();
    } else if (payload == "SHUTDOWN") {
        do_shutdown();
    } else if (payload == "RESET") {
        do_reset();} http.end();}

/* - - - - -
- - - - - */
// FUNCTION TO VERIFY THAT PC IS POWERED ON OR NOT
/* - - - - -
- - - - - */

bool isPoweredOn() {
    return digitalRead(STATUS_PIN);}

void do_powerOff() {
    if (!isPoweredOn()) {
        Serial.println("|> System is already off. Skipping

```

```

task.");
    return;}

togglePin(PWR_PIN, PWR_OFF_TIME);
Serial.println("|> Power OFF signal was sent");
reportStatus();}

/* - - - - -
- - - - - */

// FUNCTION TO POWER ON PC IF IT IS NOT
/* - - - - -
- - - - - */

void do_powerOn() {
    if (isPoweredOn()) {
        Serial.println("|> System is already ON. Skipping
task.");
        return;}
    togglePin(PWR_PIN, PUSH_TIME);
    Serial.println("|> Power ON signal was sent");
    reportStatus();}

/* - - - - -
- - - - - */

// FUNCTION TO SHUTDOWN PC IF IT IS ON
/* - - - - -
- - - - - */

void do_shutdown() {
    if (!isPoweredOn()) {
        Serial.println("|> System is already off. Skipping
task.");
        return;}
    togglePin(PWR_PIN, PUSH_TIME);
    Serial.println("|> Shutdown signal was sent");
    reportStatus();}

/* - - - - -
- - - - - */

// FUNCTION TO RESET PC IF IT IS ON
/* - - - - -
- - - - - */

```

```

- - - - - */
void do_reset() {
    if (!isPoweredOn()) {
        Serial.println("|> System is turned off. Skipping
task.");}
    togglePin(RST_PIN, PUSH_TIME);
    Serial.println("|> Reset signal was sent");
    reportStatus();}

/* - - - - -
- - - - - */

// FUNCTION

/* - - - - -
- - - - - */

void togglePin(int pin, int ms){
    digitalWrite(pin, LOW);
    pinMode(pin, OUTPUT);
    delay(ms);
    pinMode(pin, INPUT);}

/* - - - - -
- - - - - */

// END

/* - - - - -
- - - - - */

/*
digitalWrite(LD_RED, LOW);    // LD_RED OFF
digitalWrite(LD_GRN, HIGH);   // LD_GRN ON
digitalWrite(LD_RED, HIGH);   // LD_RED ON
digitalWrite(LD_GRN, LOW);    // LD_GRN OFF

*/

```