Лабораторная работа №4

**Цель работы:** научиться создавать веб интерфейсы с помощью библиотеки Settings

**Задачи:**

* Освоить создание веб интерфейса
* Изучить управление мотором постоянного тока через драйвер L9110

**Полезные ссылки:**

* <https://github.com/GyverLibs/Settings/tree/main> - Библиотека для создания простого веб-интерфейса настроек на esp8266/esp32
* <https://kit.alexgyver.ru/tutorials/motor-driver/> - общая информация о работе драйвера
* <https://3d-diy.ru/blog/drajver-dvigatelya-l9110s/?srsltid=AfmBOoqKF6SX_8SaGSBpZo61uW9lMfMCgTqWbjffUoWGYQ26cOKQzloo> - дополнительная информация о работе драйвера

**Ход работы**

1. **Установка библиотек**

Для работы библиотеки settings нужно установить не только ее, но и зависимые библиотеки

* [GTL v1.2.1+](https://github.com/GyverLibs/GTL)
* [GyverDB v1.2.3+](https://github.com/GyverLibs/GyverDB)
* [StringUtils v1.4.30+](https://github.com/GyverLibs/StringUtils)
* [GyverHTTP v1.0.23+](https://github.com/GyverLibs/GyverHTTP)
* [BSON v2.0.5+](https://github.com/GyverLibs/BSON)
* [Stamp v1.4.0+](https://github.com/GyverLibs/Stamp)
* [Table v1.1.0+](https://github.com/GyverLibs/Table)
* [StreamIO v1.0.5+](https://github.com/GyverLibs/StreamIO)

1. **Редактирование кода и прошивка микроконтроллера**

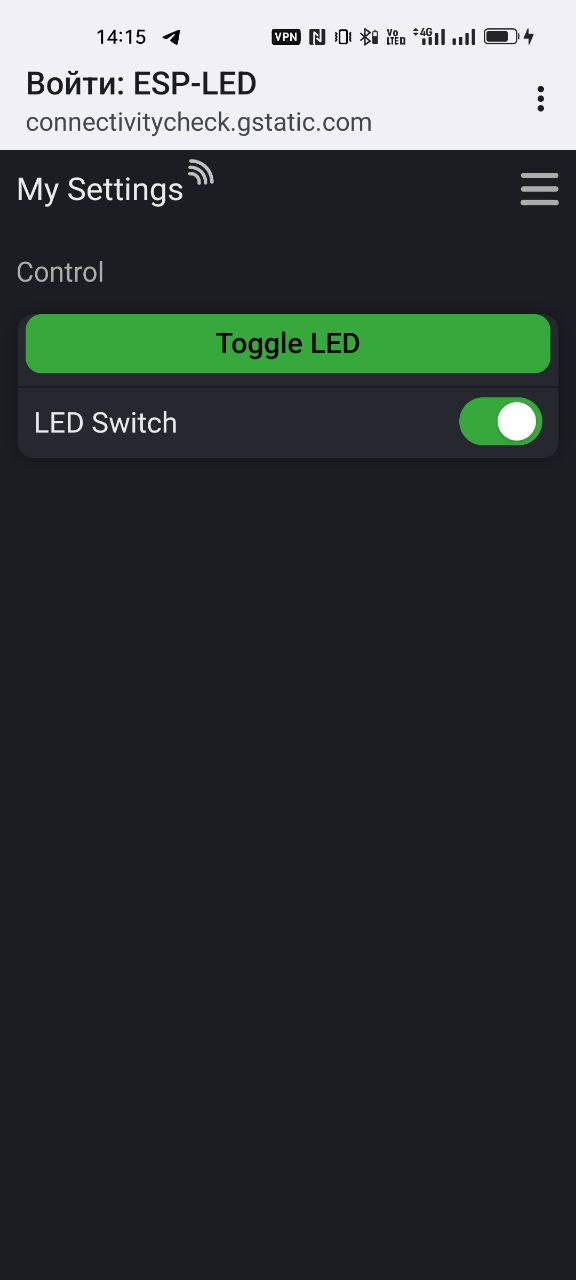
После установки библиотек необходимо прошить микроконтроллер следующим кодом, обратите внимание, **имя** устройство желательно сделать индивидуальным

| #include <Arduino.h> #include <GyverDBFile.h> #include <LittleFS.h> #include <SettingsGyver.h> #include <ESP8266WiFi.h> // или #include <WiFi.h> для ESP32  #define LED\_PIN 2 // Встроенный светодиод (D4 на ESP8266)  GyverDBFile **db**(&LittleFS, "/data.db"); SettingsGyver **sett**("My Settings", &db);  enum kk : size\_t {  toggle, };  bool ledState = false;  void **build**(sets::Builder& b) {  if (b.beginGroup("Control")) {  if (b.Button("Toggle LED")) {  ledState = !ledState;  digitalWrite(LED\_PIN, !ledState);  db.set(kk::toggle, ledState); // Правильный метод для установки значения  db.update(); // Сохранение изменений  }  b.Switch(kk::toggle, "LED Switch");  b.Label(kk::toggle, "LED State", ledState ? "ON" : "OFF");  b.endGroup();  } }  void **update**(sets::Updater& upd) {  upd.update(kk::toggle, ledState); }  void **setup**() {  Serial.begin(115200);  pinMode(LED\_PIN, OUTPUT);  LittleFS.begin();  db.begin();   // Инициализируем переменную в базе, если её нет  db.init(kk::toggle, false);  ledState = db.get(kk::toggle).toBool(); // Приведение к bool  digitalWrite(LED\_PIN, ledState);   WiFi.mode(WIFI\_AP);  WiFi.softAP("ESP-LED", "12345678");  Serial.print("AP IP: ");  Serial.println(WiFi.softAPIP());   sett.begin();  sett.onBuild(build);  sett.onUpdate(update); }  void **loop**() {  sett.tick(); } |
| --- |

После внесенных изменений необходимо прошить микроконтроллер.

1. **Протестировать работу устройства**

После прошивки устройства среди доступных WI-FI сетей появится сеть с именем указанным в коде. Подключитесь к сети введя пароль, указанный в коде. Если все выполнено правильно, появится вот такой интерфейс.



По нажатию на кнопку Toggle LED будет загораться и гаснуть светодиод на плате. LED Switch лишь показывает состояние светодиода.

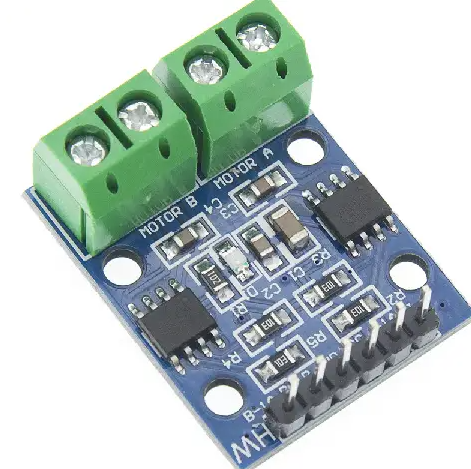
**Самостоятельное задание**

Сделать устройство открытия двери/штор. Подразумевается что имеется мотор, который приводит в движение двери/шторы в одно из крайних положений открыто/закрыто до срабатывания концевиков. Дополнительный функционал по желанию.



Рис. - Пример автоматических дверей

Устройство управляет двигателем (можно два) постоянного тока через драйвер L9110.



Иметь возможность

* установки скорости вращения мотора.
* отправки в одно из крайних положений.
* отображение его текущего местоположения и сопутсвующих параметров (скорость, режимы например: открыто/закрыто/откроется/закрывается)

Устройство должно иметь два концевика (2 физических кнопки, при нажатии которые устройство понимает что оно в крайнем положении).

#include <Arduino.h>

#include <GyverDBFile.h>

#include <LittleFS.h>

#include <SettingsGyver.h>

#include <ESP8266WiFi.h>

// Конфигурация пинов

#define MOTOR\_A1 D1 // Пин A1 драйвера L9110

#define MOTOR\_A2 D2 // Пин A2 драйвера L9110

#define BUTTON1 D5 // Кнопка 1 (концевик/управление)

#define BUTTON2 D6 // Кнопка 2 (концевик/управление)

GyverDBFile db(&LittleFS, "/data.db");

SettingsGyver sett("Motor Control", &db);

enum dbKeys : size\_t {

motorSpeed,

motorState,

};

int speed = 1023; // Скорость (0-1023)

String motorStatus = "Stopped";

bool motorStopped = true;

unsigned long lastChangeTime = 0;

const unsigned long delayTime = 1000; // Задержка 1 секунда

void stopMotor() {

analogWrite(MOTOR\_A1, 0);

analogWrite(MOTOR\_A2, 0);

motorStatus = "Stopped";

motorStopped = true;

}

void rotateCW() { // По часовой

analogWrite(MOTOR\_A1, speed);

analogWrite(MOTOR\_A2, 0);

motorStatus = "Rotating CW";

motorStopped = false;

}

void rotateCCW() { // Против часовой

analogWrite(MOTOR\_A1, 0);

analogWrite(MOTOR\_A2, speed);

motorStatus = "Rotating CCW";

motorStopped = false;

}

void build(sets::Builder& b) {

if (b.beginGroup("Motor Control")) {

b.Slider(dbKeys::motorSpeed, "Speed", 0, 1023);

b.Label(dbKeys::motorState, "Status", motorStatus);

b.endGroup();

}

}

void update(sets::Updater& upd) {

upd.update(dbKeys::motorSpeed, speed);

upd.update(dbKeys::motorState, motorStatus);

}

void setup() {

Serial.begin(115200);

pinMode(MOTOR\_A1, OUTPUT);

pinMode(MOTOR\_A2, OUTPUT);

pinMode(BUTTON1, INPUT\_PULLUP);

pinMode(BUTTON2, INPUT\_PULLUP);

LittleFS.begin();

db.begin();

db.init(dbKeys::motorSpeed, 1023);

speed = db.get(dbKeys::motorSpeed).toInt();

WiFi.mode(WIFI\_AP);

WiFi.softAP("Motor-Control", "12345678");

Serial.print("AP IP: ");

Serial.println(WiFi.softAPIP());

sett.begin();

sett.onBuild(build);

sett.onUpdate(update);

stopMotor();

}

void loop() {

sett.tick();

// Чтение состояния кнопок (концевиков)

bool btn1 = !digitalRead(BUTTON1);

bool btn2 = !digitalRead(BUTTON2);

// Если обе кнопки нажаты - остановка

if (btn1 && btn2) {

stopMotor();

return;

}

// Если двигатель не остановлен и нажат концевик

if (!motorStopped && (btn1 || btn2)) {

stopMotor();

lastChangeTime = millis();

return;

}

// Если прошла задержка после остановки

if (motorStopped && (millis() - lastChangeTime) > delayTime) {

if (btn1) {

rotateCCW(); // Кнопка 1 - вращение против часовой

} else if (btn2) {

rotateCW(); // Кнопка 2 - вращение по часовой

}

}

delay(10);

}

### Особенности работы:

1. **Две кнопки** выполняют двойную роль:
   * Управление направлением вращения
   * Концевики (при нажатии двигатель останавливается)
2. **Логика работы**:
   * Нажатие кнопки 1 → вращение против часовой (после задержки)
   * Нажатие кнопки 2 → вращение по часовой (после задержки)
   * Нажатие обеих кнопок → мгновенная остановка
   * При достижении концевика (нажатии кнопки) → остановка + задержка 1с
3. **Веб-интерфейс**:
   * Регулировка скорости
   * Отображение текущего состояния
4. **Защита**:
   * Задержка 1 секунда при смене направления
   * Мгновенная остановка при нажатии обеих кнопок

Подключите кнопки между пинами D5/D6 и GND (используются внутренние подтягивающие резисторы). Двигатель подключается к пинам D1 и D2 через драйвер L9110.

1. **Подключение драйвера L9110S**:
   * A1A → GPIO5 (D1 на NodeMCU)
   * A1B → GPIO4 (D2 на NodeMCU)
   * VCC → 5V от NodeMCU (или внешний источник)
   * GND → Общая земля
2. **Подключение двигателя**:
   * К клеммам Motor A на L9110
3. **Подключение кнопок (концевиков)**:
   * Кнопка 1:
     + Один контакт → GPIO14 (D5)
     + Второй контакт → GND
   * Кнопка 2:
     + Один контакт → GPIO12 (D6)
     + Второй контакт → GND
4. **Питание**:
   * Для слабых двигателей (≤200mA) можно питать от 5V NodeMCU
   * Для мощных двигателей нужен отдельный блок питания:
     + Подключить + к VCC L9110
     + - к общему GND (связать с GND NodeMCU)

#include <Arduino.h>

#include <GyverDBFile.h>

#include <LittleFS.h>

#include <SettingsGyver.h>

#include <ESP8266WiFi.h>

#include "GyverMotor.h"

// Конфигурация выводов

#define DRIVER\_PIN1 D1 // Вывод 1 драйвера

#define DRIVER\_PIN2 D2 // Вывод 2 драйвера

#define SENSOR\_UP D5 // Датчик верхнего положения

#define SENSOR\_DOWN D6 // Датчик нижнего положения

// Инициализация двигателя

GMotor curtainMotor(DRIVER2WIRE, DRIVER\_PIN1, DRIVER\_PIN2, HIGH);

// База данных и настройки

GyverDBFile storage(&LittleFS, "/config.db");

SettingsGyver webPanel("Smart Curtain", &storage);

enum ConfigIDs : size\_t {

DIRECTION\_FLAG,

MOTOR\_POWER

};

bool movingUp = true; // Направление движения

int powerLevel = 180; // Мощность двигателя (0-255)

void createWebInterface(sets::Builder& ui) {

if (ui.beginGroup("Управление")) {

// Регулятор мощности

ui.Slider(ConfigIDs::MOTOR\_POWER, "Сила движения", 50, 255, 5, "", powerLevel);

// Кнопка управления

if (ui.Button(movingUp ? "Опустить штору" : "Поднять штору")) {

if (movingUp) {

// Движение вниз

while (digitalRead(SENSOR\_DOWN) == HIGH) {

curtainMotor.setMode(AUTO);

curtainMotor.setSpeed(-powerLevel);

Serial.println("Движение вниз");

delay(15);

}

curtainMotor.setMode(STOP);

Serial.println("Нижнее положение");

} else {

// Движение вверх

while (digitalRead(SENSOR\_UP) == HIGH) {

curtainMotor.setMode(AUTO);

curtainMotor.setSpeed(powerLevel);

Serial.println("Движение вверх");

delay(15);

}

curtainMotor.setMode(STOP);

Serial.println("Верхнее положение");

}

movingUp = !movingUp;

storage.set(ConfigIDs::DIRECTION\_FLAG, movingUp);

storage.update();

}

// Индикатор направления

ui.Switch(ConfigIDs::DIRECTION\_FLAG, "Текущее направление");

ui.endGroup();

}

}

void syncSettings(sets::Updater& sync) {

sync.update(ConfigIDs::DIRECTION\_FLAG, movingUp);

sync.update(ConfigIDs::MOTOR\_POWER, powerLevel);

}

void initializeSystem() {

Serial.begin(115200);

// Инициализация файловой системы

if (!LittleFS.begin()) {

Serial.println("Ошибка инициализации памяти");

} else {

Serial.println("Файловая система готова");

}

// Настройка датчиков

pinMode(SENSOR\_UP, INPUT\_PULLUP);

pinMode(SENSOR\_DOWN, INPUT\_PULLUP);

// Настройка WiFi

WiFi.mode(WIFI\_AP);

WiFi.softAP("Smart-Curtain", "home1234");

Serial.print("Адрес панели: ");

Serial.println(WiFi.softAPIP());

// Инициализация хранилища

storage.begin();

// Запуск веб-интерфейса

webPanel.begin();

webPanel.onBuild(createWebInterface);

webPanel.onUpdate(syncSettings);

}

void mainLoop() {

// Обслуживание веб-интерфейса

webPanel.tick();

}