



Руководство программиста (HotbedAgroControl)

Оглавление

Структура ПО.....	2
Основной модуль HotbedAgroControl-V1.cpp.....	2
Функция setup().....	2
Функция loop().....	2
Web-страница устройства.....	3
Control Panel.....	4
Debug Page.....	5
Oscilloscope.....	5
Web-страница AutoConnect.....	6
Web.cpp.....	6
Сайт Home Assistant.....	7
SD_MQTT.cpp.....	12
Работа с датчиками и оборудованием микроконтроллера.....	13
Реализация автоматов.....	13
Класс AMT1001.....	16
Класс EC_meter.....	16
Класс LUX_meter.....	17
Класс PhAndTemperature.....	18
RelayControl.....	19
SensorDS18B20.....	20
SensorLevel.....	20

Структура ПО

Программная структура проекта содержит один основной программный модуль, код которого содержит файл `HotbedAgroControl-V1.cpp`, и множество других модулей. В базовом каталоге проекта `src`, кроме основного модуля, находятся модули `HtmlDebug.cpp`, `HtmlMainPage.cpp`, `HtmlOscilloscope.cpp` и `SendHtml.cpp`, содержащие код web-страниц в формате языка HTML, и два каталога `Evgen` и `SWITCH`.

Каталог `Evgen` проекта включает программный модуль, реализующий режим `AutoConnect` для определения параметров связи по умолчанию по WiFi при включении устройства – имя и пароль. Все это реализует код файла `Web.cpp`. Другие модули данного каталога в сумме обеспечивают работу по протоколу `Home Assistant` в формате протокола MQTT. Среди них основной модуль представлен файлом `SD_MQTT.cpp`.

Каталог `SWICH` проекта содержит программные модули, которые собственно и реализуют логику работы с оборудованием платы контроллера, к которым относятся разнообразные датчики, два реле и вентилятор охлаждения платы контроллера.

Основной модуль `HotbedAgroControl-V1.cpp`

Данный модуль – это базовый модуль проекта, содержащий стандартные функции `setup()` и `loop()`. Он содержит основные объекты, реализующие логику работы с датчика. Это объекты с именами: 1) `Lux_meter` – измерение освещенности; 2) `Amt1001` – работа с датчиком измерения температуры и влажности `AMT1001`; 3) `EC_Meter` – измерение электропроводности; 4) `PHAndTemperature` – измерение PH и температуры; 5) `DS1820` – измерение температуры датчиком `DS18B20`; 6) `sensorLevel` – трехпозиционный контроль уровня жидкости.

Кроме упомянутых объектов, модуль содержит объекты, реализующие функции фильтрации данных, поступающих от датчиков. Это соответственно объекты: `DgtFilterT_Air` – фильтр для температуры воздуха, `DgtFilterH_Air` – фильтр значения влажности, `DltFilterLux` – фильтр значения освещенности, `DgtFolter18B20` – фильтр датчика `DS18B20`, `DgtFiterEC` – фильтр значения датчика `EC`, `DgtFilterPH` – фильтр значения датчика `PH`.

Функция `setup()`

Данная функция выполняет настройку цифровых фильтров. Для этого полям цифрового фильтра с именами `pVarX`, `pVarY` и `nN` устанавливаются соответственно – адрес входной переменной, адрес выходной переменной и глубину цифрового фильтра. В процессе своей работы цифровой фильтр помещает входное значение в буфер заданного размера, из которого на каждом такте работы фильтра извлекается усредненное значение, вычисленное по всем значениям буфера фильтра.

Здесь же настраиваются входные и выходные пины микроконтроллера, аналоговый вход (разрешение АЦП) и вызывается функция `setup_web_common()` программного модуля `Web.cpp` для настройки режима `AutoConnect`.

Функция `loop()`

Если функция `setup()` выполняется только один раз при включении устройства, то функции `loop()` запускается циклически, реализуя дискретный цикл работы программных компонентов

микроконтроллера. При этом минимальная величина дискретного такта работы устройства определяется текущим временем работы функции loop().

В рамках запуска данной функции работают цифровые фильтры. Для этого вызывается их метод run() и методы run() объектов, реализующих логику работы с датчиками.

Здесь же, но уже со своим дискретным временем, запускается функция loop2(), которая реализует индикацию работы платы миганием светодиода, размещенного на плате микроконтроллера. Здесь же выполняется запуск функции loop_web() модуля Web.cpp, реализующей соединение по WiFi.

В рамках loop2() происходит также вызов метода loop() класса SD_Termo, который организует оперативное сохранение во внутренней памяти микроконтроллера параметров настроек устройства, а именно, подключение к внешнему сайту по протоколу MQTT, определяемых соответствующим диалогом AutoConnect.

Web-страница устройства

После выполнения процедуры подключения устройства по WiFi вы получаете IP-адрес страницы, набрав который в браузере она будет иметь следующий вид (см. рис.1):

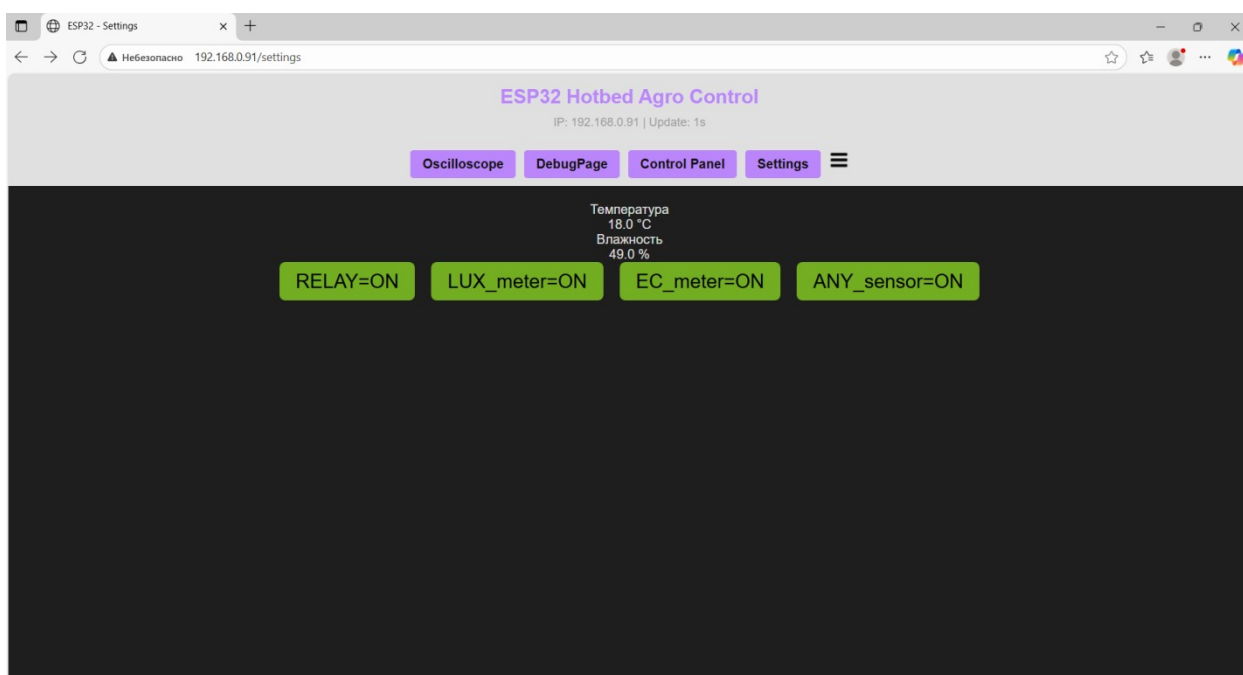
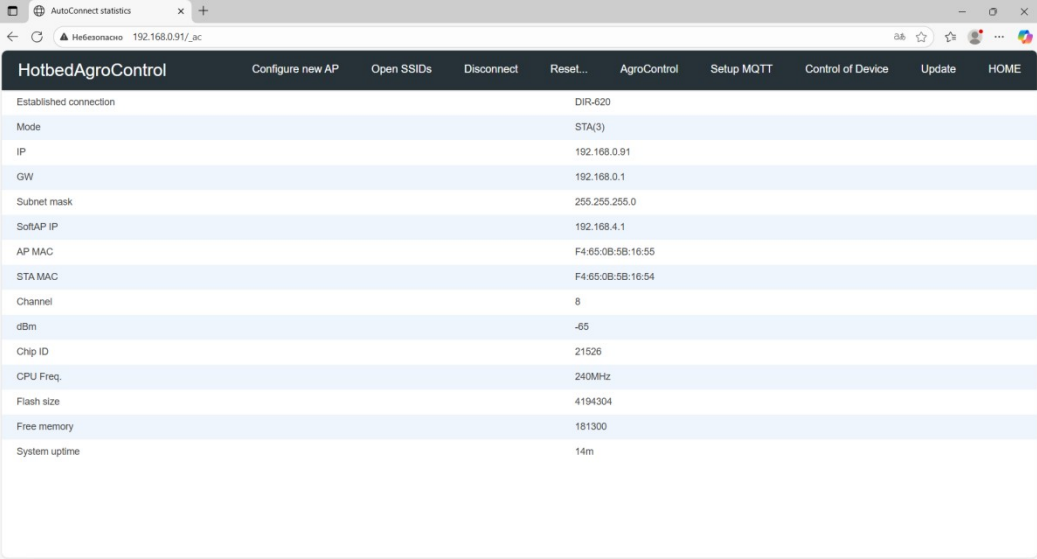


Рис. 1. Вид web-страницы устройства

На рис.1 представлен общий вид страницы с открытой закладкой Setting, которая будет отображаться по умолчанию при открытии страницы. Данная закладка отображает текущую температуру и влажность воздуха. Также на ней же расположены кнопки включения/отключения тех или иных компонент устройства. Здесь можно отключить управление реле – кнопка RELAY, измерение освещенности – LUX_meter, измерение электропроводности – EC_meter и общая кнопка отключения остальных датчиков – ANY_sensors.

Кнопки Setting, Control Panel, DebugPage и Oscilloscope включают отображение соответствующих закладок web-страницы. Правее кнопки Setting находится элемент включения web-страницы AutoConnect. Ее внешний вид при открытии представлен на рис. 2. Здесь, нажав кнопку HOME? Можно вернуться на главную страницу устройства (см. рис. 1).



Established connection	DIR-620
Mode	STA(3)
IP	192.168.0.91
GW	192.168.0.1
Subnet mask	255.255.255.0
SoftAP IP	192.168.4.1
AP MAC	F4:65:0B:5B:16:55
STA MAC	F4:65:0B:5B:16:54
Channel	8
dBm	-65
Chip ID	21526
CPU Freq.	240MHz
Flash size	4194304
Free memory	181300
System uptime	14m

Рис. 2. Вид web-страницы AutoConnect

Control Panel

Закладка Control Panel имеет вид, показанный на рис. 3.

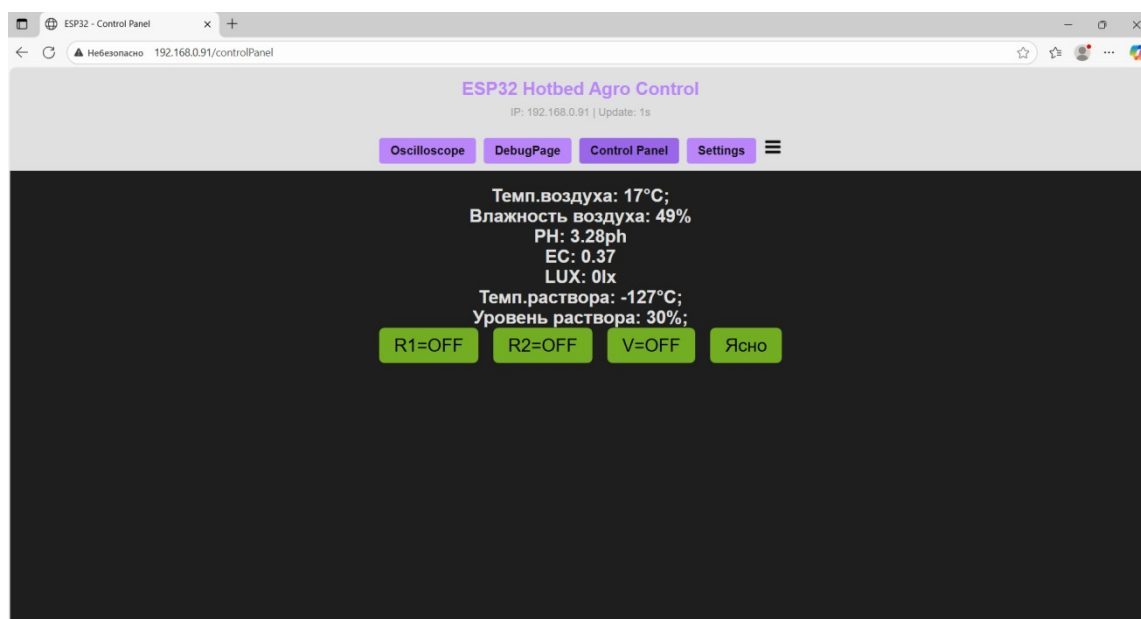


Рис. 3. Закладка Control Panel

Здесь отображаются текущие значения датчиков устройства, а также имеются кнопки управления реле и кнопка установки режима измерения освещенности.

Debug Page

Вид закладки Debug Page показан на рис. 4. Она создана для проведения экспериментов с настройкой датчиков измерения электропроводности и PH.

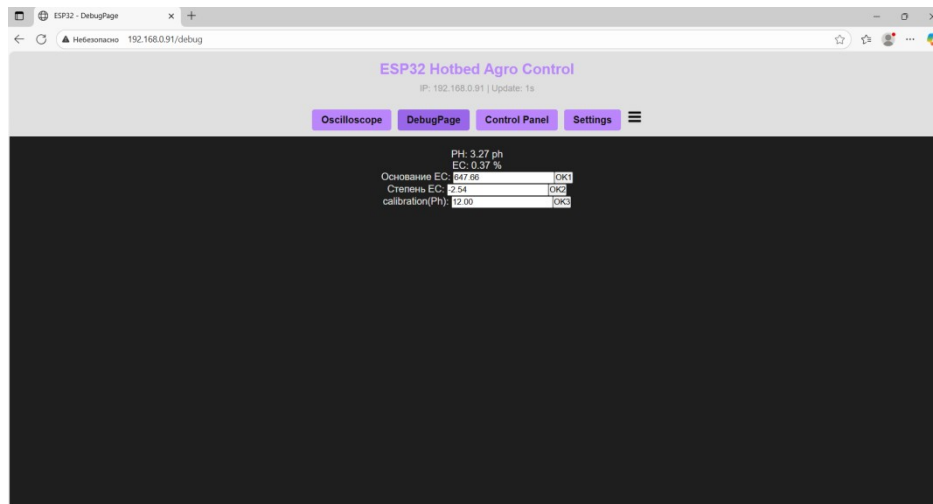


Рис.4. Закладка DebugPage

Данную закладку можно исключить из кода ПО, если закомментировать переменную PAGE_DEBUG в файле Config.h.

Oscilloscope

Данная закладка имеет вид, показанный на рис. 5. Здесь в графическом виде отображаются текущие значения температуры и влажности. При необходимости, проведя соответствующие изменения в коде приложения, можно отображать в графическом виде также текущие значения PH и электропроводности – EC.

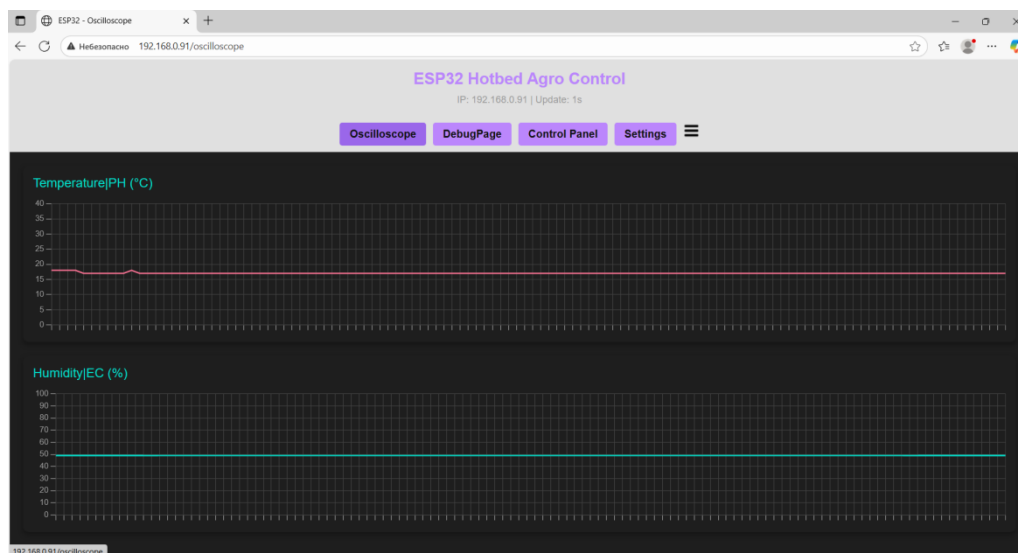


Рис. 5. Закладка Oscilloscope.

Web-страница AutoConnect

Внешний вид страницы AutoConnect был представлен выше на рис. 2, где также описано, как ее открыть на закладке Setting основного окна приложения.

Данная страница, кроме стандартных закладок AutoConnect, включает дополнительные закладки. К ним относятся: Control of Device, Setup MQTT, Agro Control. Закладка Control of Device аналогична по выполняемым функциям закладкам Control Panel и DebugPage основного окна приложения. С помощью закладки Setup MQTT настраиваются параметры подключения по MQTT к сайту Home Assistant. Закладка Agro Control отображает статус подключения по протоколу MQTT. Вид данных закладок показан на рис. 6.

The screenshot displays the AutoConnect web interface with three tabs: "Control of Device", "Setup MQTT", and "AgroControl".

- Control of Device:** Contains checkboxes for "Relay", "EC", and "PH", all of which are checked. Below these are input fields for "Delay(EC)" (0), "Cycles(EC)" (16), "Температура воздуха" (17.00), and "Влажность воздуха" (49.00). A green "Применить" button is at the bottom.
- Setup MQTT:** Features a checked "MQTT" checkbox. Below it are input fields for "user" (mq_umki), "pwd" (54321), "сервер" (80.237.33.118), "порт" (12883), "топик" (HB), "имя устройства" (AgroControl), and "интервал, сек" (10). A green "Задать" button is at the bottom.
- AgroControl:** Shows the status "Статус OT: MQTT connected" and a green "Обновить" button.

Рис. 6. Закладки AutoConnect

Web.cpp

Данный модуль содержит объекты-шаблоны для создания закладок на web-странице AutoConnect. Если в качестве примера взять закладку Control of Device, то это будут следующие объекты кода:

1. Создание элементов закладки

```
AutoConnectCheckbox CtrlChbUseRelay("ChbUseRelay", "7", "Relay", false, AC_Behind, AC_Tag_DIV);
AutoConnectCheckbox CtrlChbUseEC("ChbUseEC", "7", "EC", false, AC_Behind, AC_Tag_DIV);
AutoConnectCheckbox CtrlChbUsePH("ChbUsePH", "7", "PH", false, AC_Behind, AC_Tag_DIV);
ACInput(Set_DelayEC, "", "Delay(EC)", "", "", AC_Tag_BR, AC_Input_Number, STYLE_WIDTH);
ACInput(Set_CyclesEC, "", "Cycles(EC)", "", "", AC_Tag_BR, AC_Input_Number, STYLE_WIDTH);
ACInput(Set_Tair, "", "Температура воздуха");
ACInput(Set_Hair, "", "Влажность воздуха");
```

2. Объект закладки

```
AutoConnectAux Control_Page(CONTROL_URI, "Control of Device", true, { CtrlChbUseRelay, CtrlChbUseEC, CtrlChbUsePH,
Set_DelayEC, Set_CyclesEC, Set_Tair, Set_Hair, ApplyChCntrl});
```

3. Подключение закладки к странице AutoConnect

```
portal.join({InfoPage, Setup_Page, SetParPage, Control_Page});
```

4. Создать обработчик событий от закладки

```
extern bool bIfRelay;
extern bool bIfEcMeter;
extern bool bIfPhMeter;
String on_Control(AutoConnectAux& aux, PageArgument& args)
{
    if( CtrlChbUseRelay.checked) bIfRelay = true;
    else bIfRelay = false;
    if( CtrlChbUseEC.checked) bIfEcMeter = true;
    else bIfEcMeter = false;
    if( CtrlChbUsePH.checked) bIfPhMeter = true;
    else bIfPhMeter = false;

    int v = Set_DelayEC.value.toInt();
    if((unsigned int)v != EC_Meter.nDelayEC )
    {
        EC_Meter.nDelayEC = v;
    }
    v = Set_CyclesEC.value.toInt();
    if((unsigned int)v != EC_Meter.nCyclesEC )
    {
        EC_Meter.nCyclesEC = v;
    }

    char str[40];
    sprintf(str, "%.2f", Amt1001.T_Air);
    Set_Tair.value = str;
    sprintf(str, "%.2f", Amt1001.H_Air);
    Set_Hair.value = str;
    return String();
}
```

5. Подключить обработчик в функции setup_web_common()

```
Control_Page.on(on_Control);
```

Сайт Home Assistant

Внешний вид страницы сайта Home Assistant с подключенными элементами устройства показан на рис. 7.

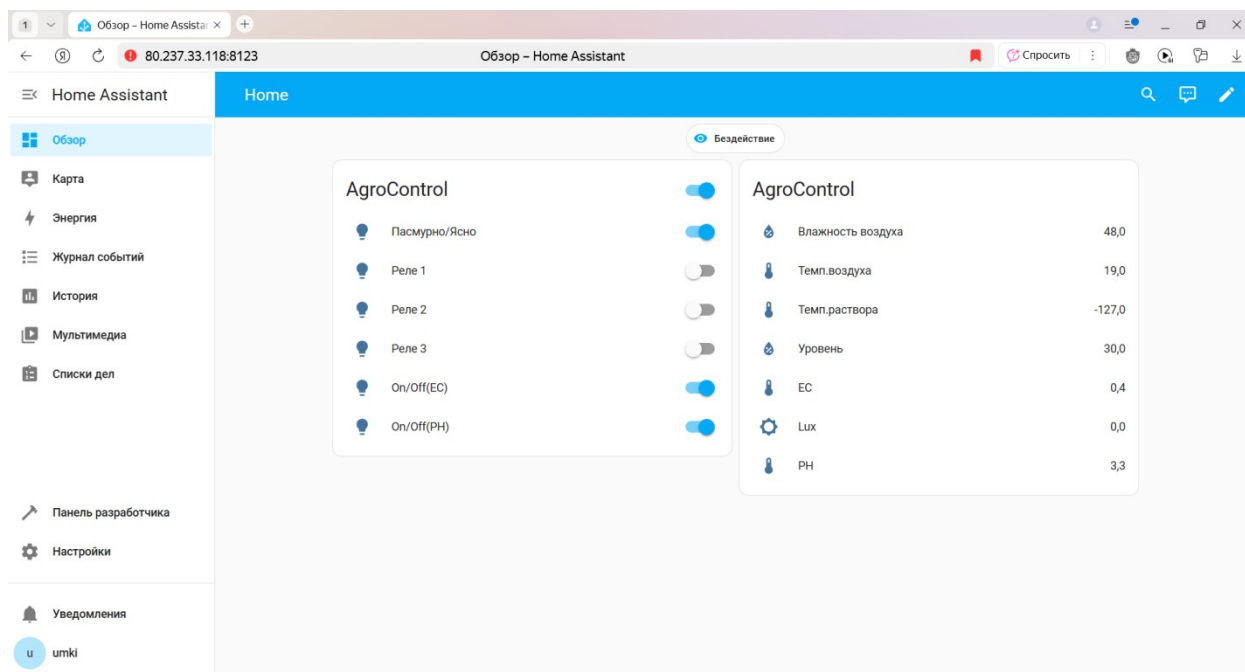


Рис. 7. Общий вид сайта Home Assistant

Чтобы сформировать подобный вид нужно открыть сайт в браузере:

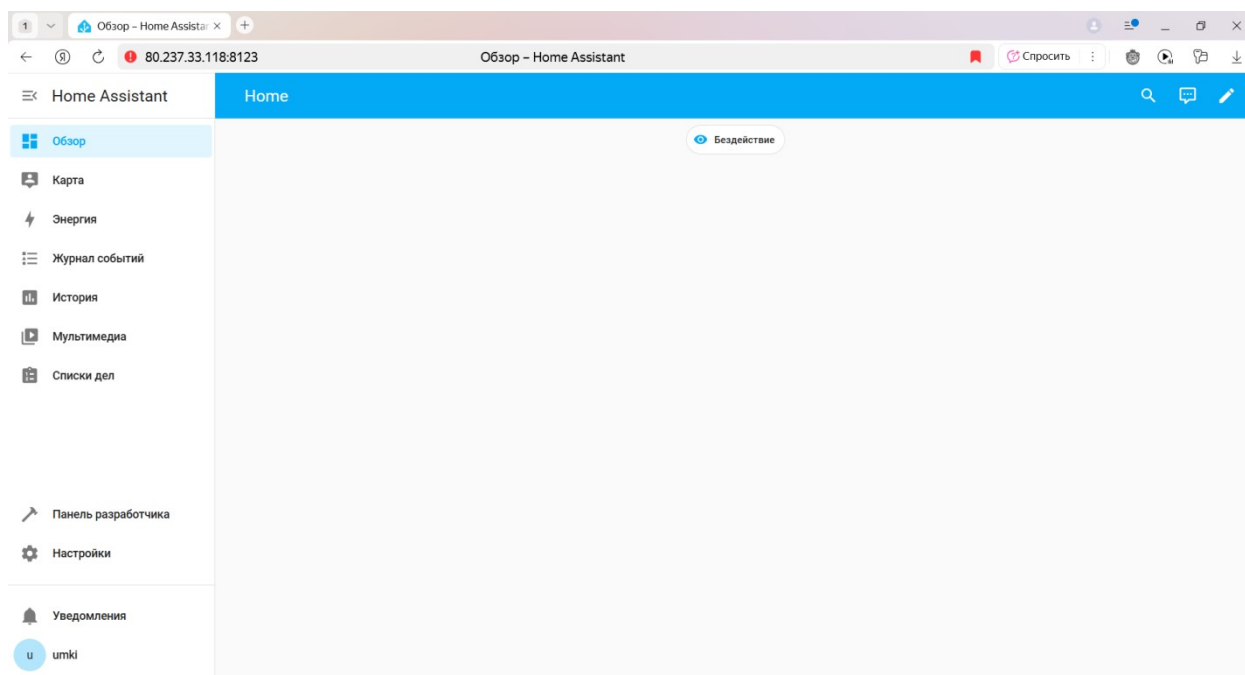


Рис.8. Вход на сайт

Зайти в Настройки, выбрать Устройства и службы, открыть закладку Устройства, выбрать в ней Добавить устройства и в открывшемся диалоге задать MQTT.

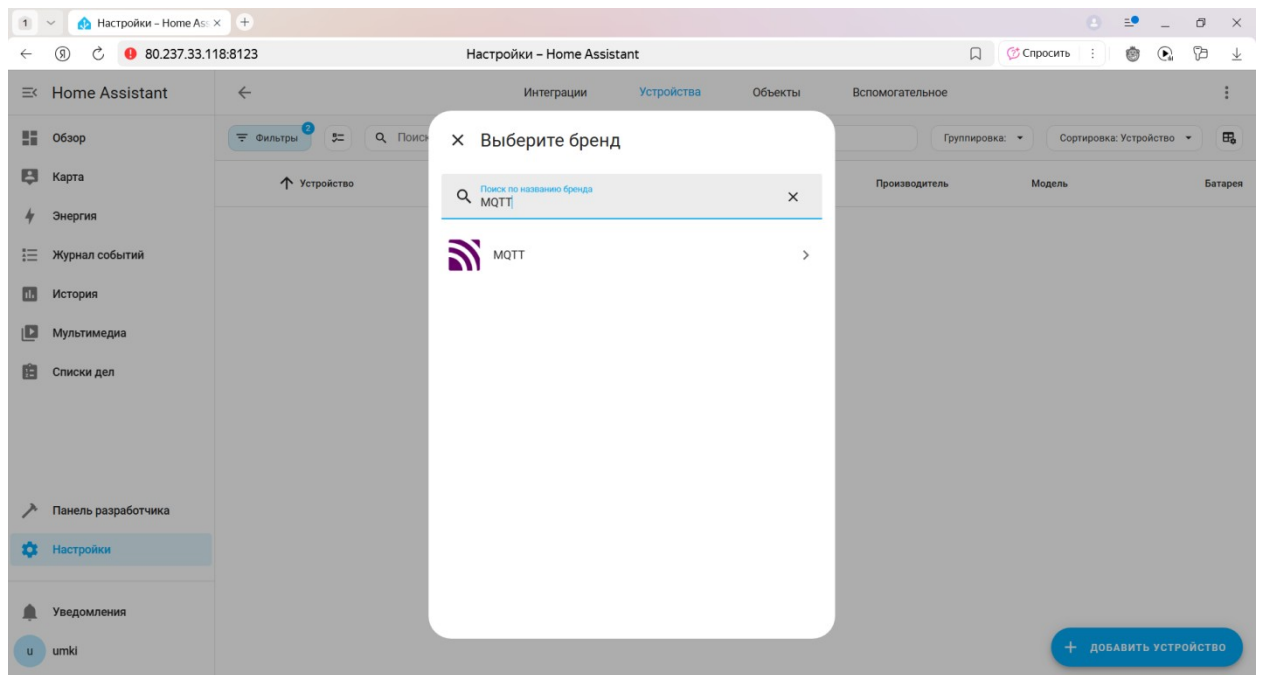


Рис. 9. Выбор устройства MQTT

Выбрать MQTT и задать настройки MQTT из закладки Setup MQTT странички AutoConnect.

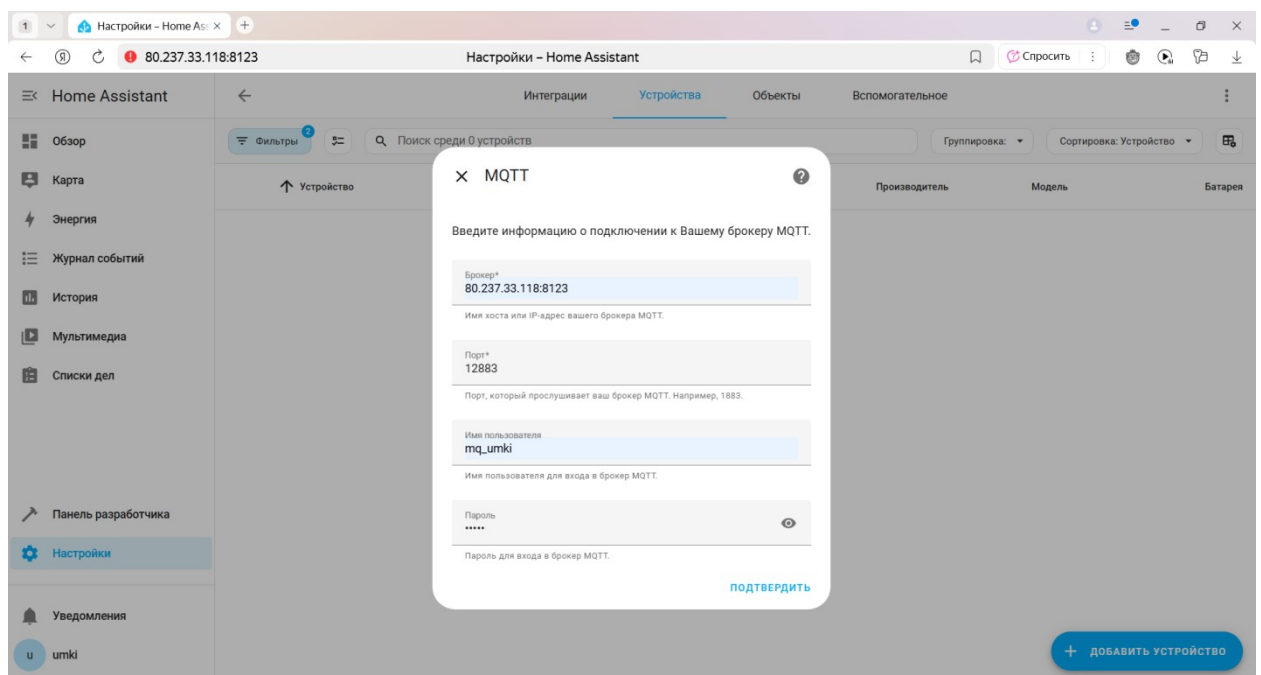


Рис.10. Вход на сайт

Выбрать - ПОДТВЕРДИТЬ.

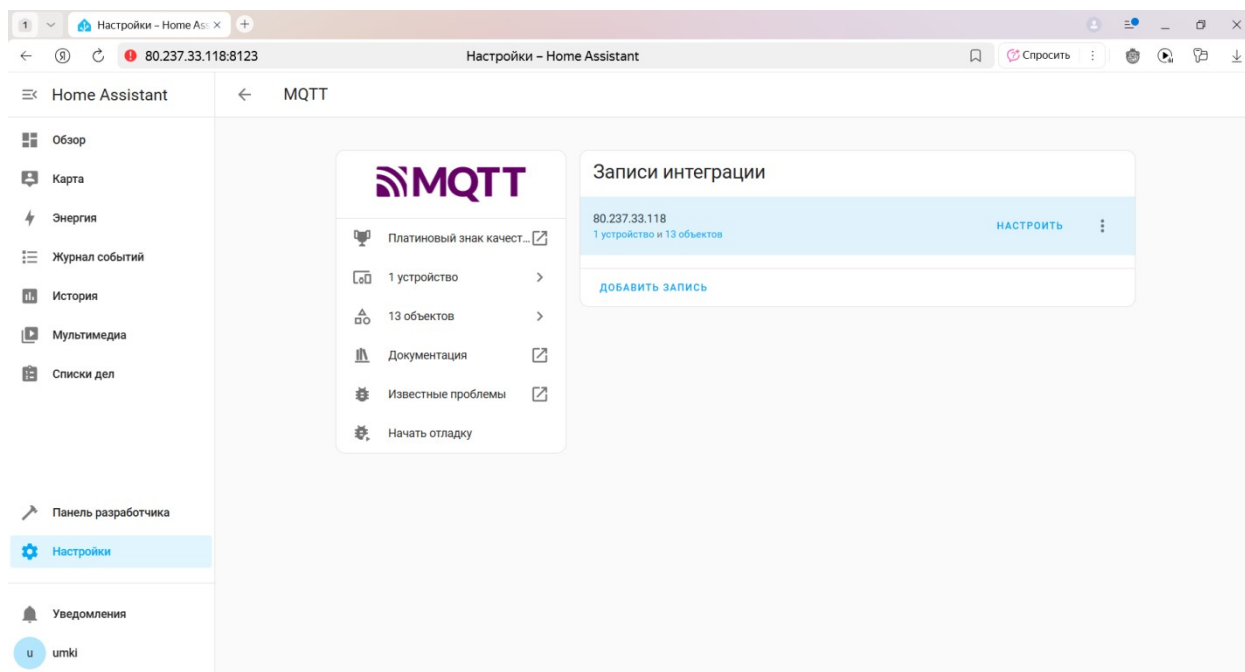


Рис.11. Подключенное устройство

Открыть устройство, где будут отображены сгруппированные компоненты страницы управления устройством:

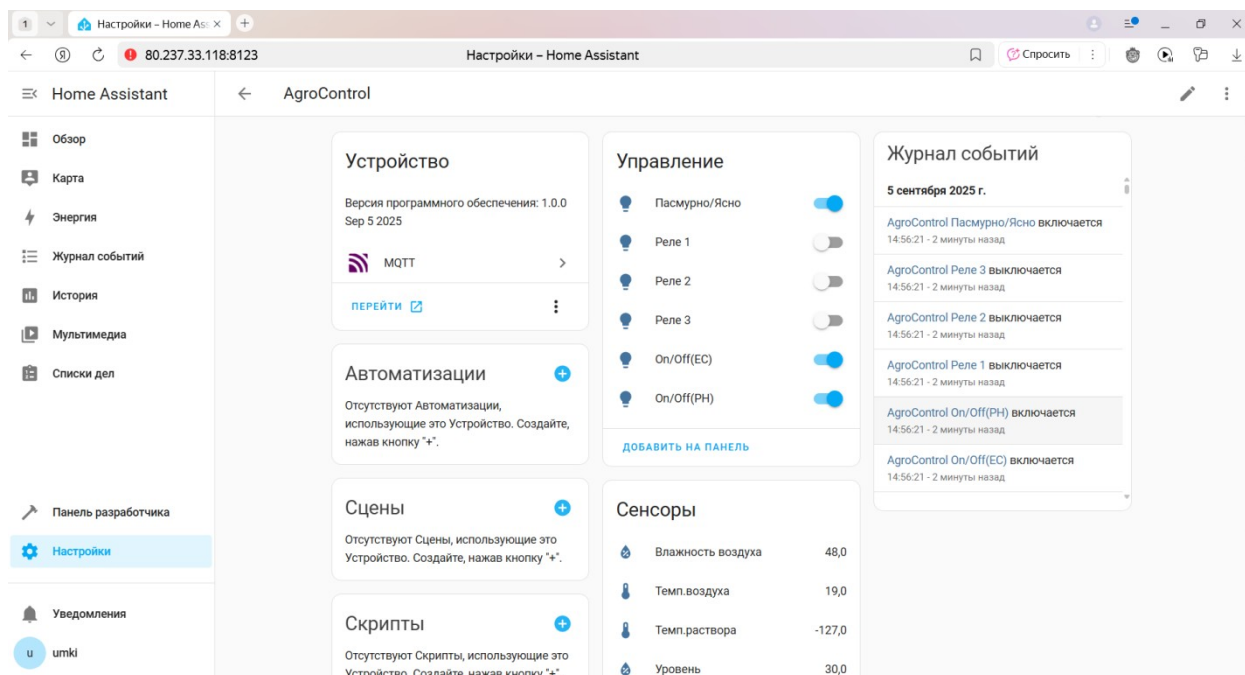


Рис. 12. Перечень объектов устройства

Для удобства работы с ними их нужно добавить на панель, нажав соответствующую кнопку.

После этого выбрать Обзор на странице сайта, который примет при этом следующий вид:

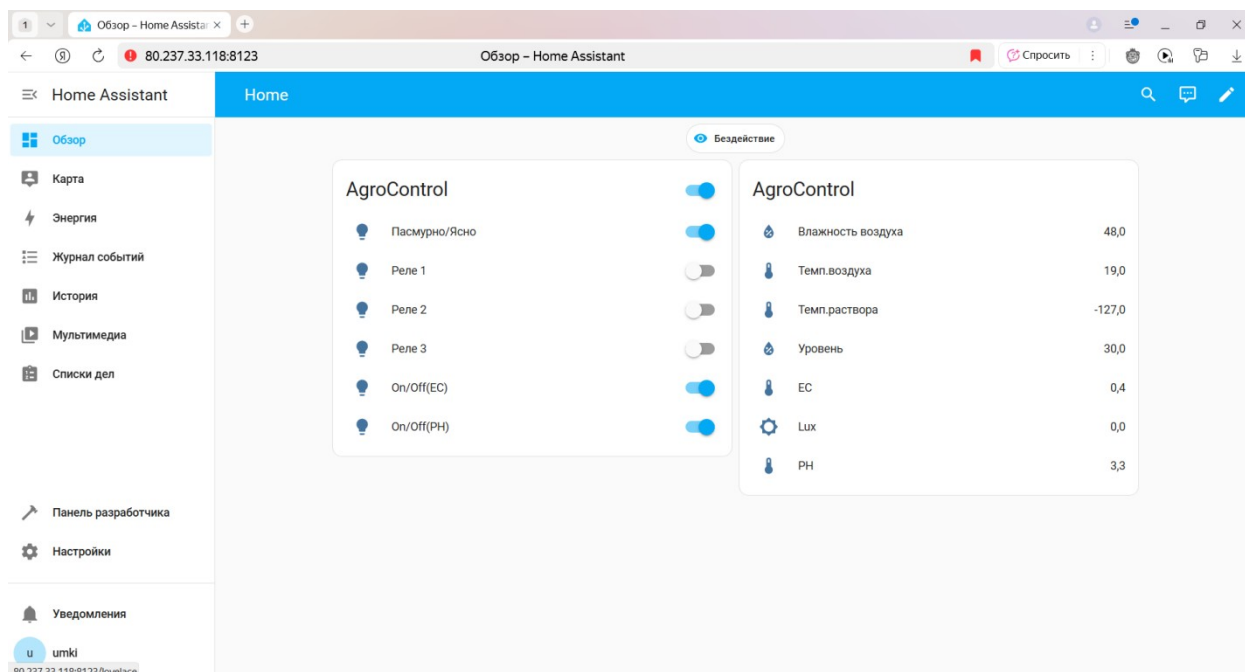


Рис.13.

Теперь, выбирая соответствующие переключатели, можно управлять устройством, а также видеть текущие значения датчиков. Эти же данные можно выводить в графическом виде, визуализируя историю их изменения (см. рис. 14).

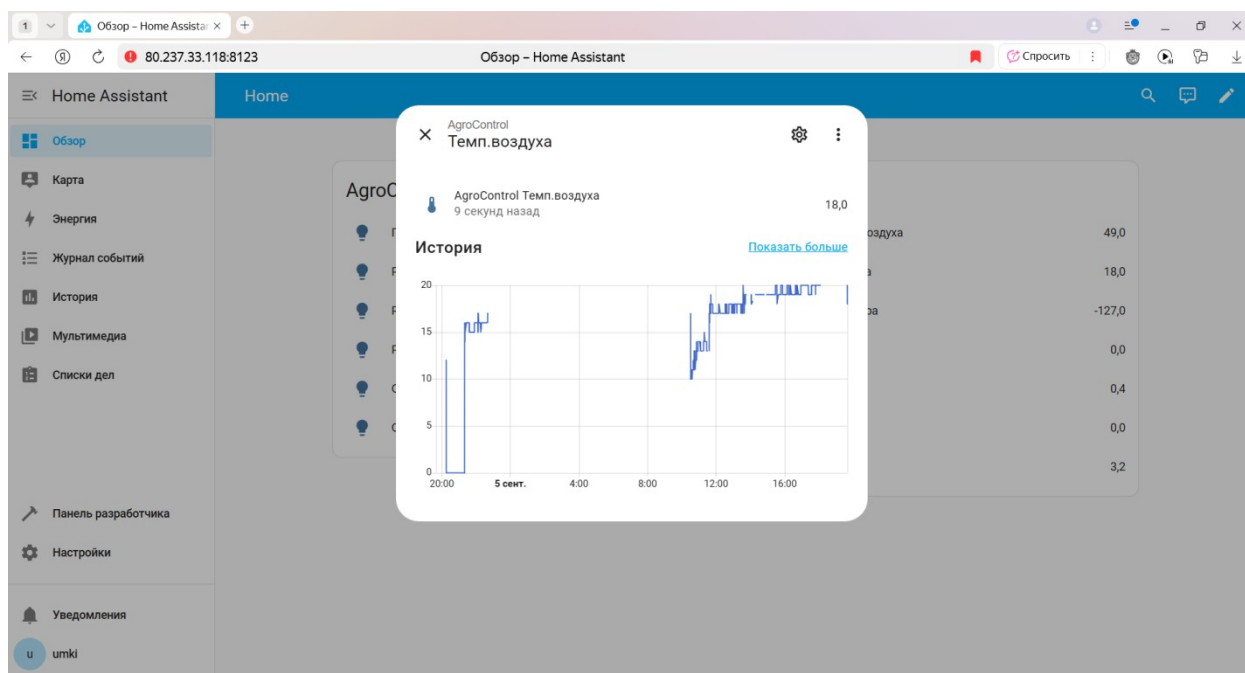


Рис. 14. Вывод истории изменения температуры в графической форме

SD_MQTT.cpp

Формирование элементов сайта Home Assistant реализует модуль SD_MQTT.cpp. Данный модуль формирует внешний вид страницы Home Assistant. Для этого необходимо организовать подключение к сайту. Делается это следующим образом:

```
WiFiClient espClient;  
PubSubClient client(espClient);  
HADevice device;  
HAMqtt mqtt(espClient, device, 27);
```

Листинг 1.

Затем создаются объекты для, например, отображения какого-то значения. Возьмем например, температуру.

1. Создается объект для датчика температуры

```
const char * temperature_str = "temperature";  
#if defined (IF_AMT1001)  
HASensor sensorAMT_T(NULL, HANumber::PrecisionP3);  
#endif
```

Листинг 2.

2. Далее в функции mqtt_setup производится его настройка

```
#if defined (IF_AMT1001)  
sensorAMT_T.setAvailability(true);  
sensorAMT_T.setNameUniqueIdStr(SmOT.MQTT_topic, "Темп.воздуха", "agr_t");  
sensorAMT_T.setDeviceClass(temperature_str);  
#endif
```

Листинг 3.

3. Затем в функции mqtt_loop() организуется вывод значения

```
#if defined(IF_AMT1001)  
float ddd;  
ddd = Amt1001.T_Air;  
sprintf(str, "%.1f", ddd);  
sensorAMT_T.setValue(str);  
#endif
```

Листинг 4.

Если это элемент управления, то его создание и управление им покажем на примере переключателя режима измерения освещенности.

1. Создаем обработчик

```
#if defined (sens_Lux)  
HASwitch ClearCloudy("ClearCloudy");  
void onSwitchCommandClearCloudy(bool state, HASwitch* sender)  
{  
    Serial.printf("onSwitchCommandClearCloudy - %d\n", state);  
#if defined (IF_LUX)  
    if (state) {  
        Lux_meter.FStatus1 = true;  
        Lux_meter.FStatus2 = true;  
    }  
    else {  
        Lux_meter.FStatus1 = true;  
        Lux_meter.FStatus2 = false;  
    }  
#endif  
}
```

```

}
sender->setState(state); // report state back to the Home Assistant
#endif
}
#endif

```

Листинг 5.

2. Настройка в mqtt_setup

```

#if defined (sens_Lux)
  ClearCloudy.setIcon("mdi:lightbulb");
  ClearCloudy.setName("Пасмурно/Ясно");
  ClearCloudy.onCommand(onSwitchCommandClearCloudy);
  ClearCloudy.setNameUniqueIdStr(SmOT.MQTT_topic,"ClearCloudy", "agr_ClearCloudy");
#endif

```

Листинг 6.

3. Организуем управление от переключателя в рамках функции mqtt_loop

```

static bool bIfInitSwith=false;
if (!bIfInitSwith) {
  IFEC.setState(bIfLuxMeter, true);
  IFPH.setState(bIfPhMeter, true);
  ClearCloudy.setState(true, true);
  Lux_meter.FStatus1 = true;
  Lux_meter.FStatus2 = true;
  bIfInitSwith = true;
}

```

Листинг 7.

Работа с датчиками и оборудованием микроконтроллера

Логика работы с датчиками и оборудованием типа реле сосредоточена в объектах, созданных классами, представленными кодом модулей каталога SWITCH проекта. Это работа с датчиком АМТ1001 – класс АМТ1001, измерение электропроводности ЕС – класс ЕС_meter, измерение освещенности – класс LUX_meter, измерение PH и температуры – класс PhAndTemperature, работа с датчиком измерения температуры DS18B20 – представлено классом SensorDS18D20, а измерение уровня жидкости - объектом класса SensorLevel. Управление реле представлено функциями модуля RelayControl.cpp.

Реализация автоматов

Важным и отличительным свойством проекта является представления отдельных алгоритмов в форме модели конечного автомата, а их параллельную работу – в форме сетей автоматов. Алгоритмическая модель в форме модели конечного автомата – это база для технологии программирования, названной, соответственно автоматной. Но модель автомата не поддерживается ни аппаратно, ни программными конструкциями современных языков программирования, а потому необходимы приемы их программной реализации. Но это требует определенных ресурсов, среди которых, пожалуй, основными является объем оперативной и стековой памяти. В случае рассматриваемого нами устройства мы имеет достаточно ограниченный объем оперативной памяти, а потому пришлось прибегнуть к наиболее «экономному» варианту программной реализации автоматов.

В нашем случае нет какой-либо библиотеки для реализации технологии автоматного программирования. И в этом смысле нет трат ресурса оперативной памяти, которую часто по большей части занимаю используемые в проекте библиотеки. В нашем случае этот объем, занимаемый ими, доходит до 85%. Это библиотеки поддерживающие: подключение по WiFi, работу с датчиками проекта, реализацию функции автоконнекта, взаимодействие с сайтом Home Assistant и т.д. и т.п. Поэтому на реализацию прикладных алгоритмов, собственной web-странички устройства и т.п. остается достаточно малый объем памяти. Именно по этой причине пришлось отказаться от специализированной библиотеки, поддерживающей реализацию автоматной модели программирования.

Для реализации автоматов был создан простой базовый класс, названный LFsaAppl. Он формирует единый «автоматный протокол» для алгоритмов. Заголовок данного класса и его реализация показан соответственно на листингах 8 и 9.

```
#ifndef LFSAAPPL_H
#define LFSAAPPL_H
#include <Arduino.h>
class LFsaAppl {
public:
    void virtual run();
    LFsaAppl(String StrNam);
    virtual ~LFsaAppl() {};
    virtual String FGetState();
    bool FGoToState(String nam);
    String FGetNameFsa();
protected:
    int nState{0};
    int nTmpState{0};
    String StrName;
};
#endif // LFSAAPPL_H
```

Листинг 8. Заголовок базового автоматного класса.

```
#include "LFsaAppl.h"
LFsaAppl::LFsaAppl(String StrNam)
{
    StrName = StrNam;
    nState = nTmpState = 0;
}

void LFsaAppl::run() {
}

String LFsaAppl::FGetState() { return String(nState); }

bool LFsaAppl::FGoToState(String nam) { nState = nam.toInt(); return true; }

String LFsaAppl::FGetNameFsa() { return StrName; }
```

Листинг 9. Реализация автоматного класса.

Используя данный объект, мы определяем имя объекта – свойство StrName, которое можем узнать/вернуть, используя метод FGetNameFsa(). Для реализации логики поведения подобного автоматного класса необходимо переопределить метод run(), в котором и будет реализована модель поведения автомата. При этом свойство nState определяет текущее состояние автомата, а

nTmpState – следующее состояние, которое становится новым текущим, если, конечно, сработал хотя бы один переход автомата.

Пример подхода к реализации автоматной модели разберем на примере автомата, показанного на рис. 15, демонстрирует листинг 10. Здесь текущее состояние автомата принимает целые значения, где кружком с двойными границами обозначается начальное состояние (состояние «0» на рис. 15). При этом, если возникает условие перехода из одного состояния в другое, т.е. оператор if принимает истинное значение, то выполняется тело оператора if, включающее действия, совершаемые на данном переходе и задается номер следующего состояния.

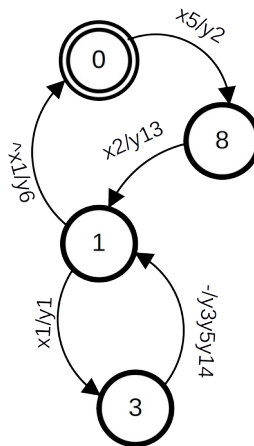


Рис. 15. Пример графа автомата

```
void EC_meter::run() {  
    if (nState==0 && x5()) { nTmpState=8; y2(); }  
    if (nState==8 && x2()) { y13(); nTmpState=1; }  
    if (nState==1 && x1()) { y1(); nTmpState=3; }  
    if (nState==1 && !x1()) { y6(); nTmpState=0; }  
    if (nState==3) { y3(); y5(); y14(); nTmpState=1; }  
    if (nTmpState != nState) { nState = nTmpState; }  
}
```

Листинг 10. Реализации автоматов на операторах if

Листинг 10 демонстрирует реализацию автоматов на базе управляющего оператора if. Аналогичную реализацию можно сделать и на основе других управляющих операторов языка или даже их смеси. Например, известен подход под названием SWICH-технология, который использует для программной реализации автоматов оператор switch.

В результате реализация поведения автоматных классов становится однотипной, что упрощает и само кодирование, которое реализуется по определенному шаблону. Упрощается и процесс отладки за счет наблюдаемости состояния автомата. Легко организовать/симулировать параллельный режим работы объектов автоматного класса, если не зацикливать процесс работы автомата в методе run.

Описанный выше подход к реализации автоматной модели вычислений фактически реализует так называемую кооперативную многозадачность или, по-другому, режим сопрограммной работы, когда управление процессу/автомату передается лишь на время работы одного цикла работы метода run. Для этого достаточно поместить в основной программный цикл loop вызов методов

run все автоматных объектов приложения. Здесь же, кстати, можно использовать информацию о текущих состояниях объектов в целях отладки или взаимной синхронизации работы процессов.

Класс AMT1001

Логика работы класса представлена графом автомата на рис. 15. Работа данного класса заключается в циклическом запуске одного из двух методов класса, реализующих опрос датчика и приведение значения к реальному виду. Код одного из данных методов представлен на листинге.



Краткое описание алгоритма

В цикле происходит вызов метода readAMT1001Data(T_Air, H_Air), который возвращает температуру - T_Air и влажность - H_Air. Последние являются свойствами класса AMT1001. Подробнее смотреть – заголовок и реализацию класса.

Рис. 15.

```
bool AMT1001::readAMT1001DataV(float &temperature, float &humidity) {
    // Получаем аналоговое значение (для ESP32, 12-bit ADC, 0-4095)
    float sav = temperature;
    float temp, hum;

    float t_a = t_air * (3.3 / 4095.0); //Преобразуем значение от АЦП ESP32 в Вольты
    double Y = ((5-(1.785* t_a))/(1.785* t_a))*10; // пересчитываем вольты в КОмы, по формуле с коэффициентами от
    Сереги

    temp =((-23.336)*log(Y))+79.581; // Пересчитываем КОМ в градусы Цельсия по таблице из DataSheet на AMT1001,
    натуральный логарифм взят из LibreOffice
    if (temp>T_AirMin && temp<T_AirMax) { temperature = temp - 2; } // отсекаем лишнее
    else { temperature = sav; }

    double volt = (double)h_air * (3.3 / 4095.0);
    humidity = amt1001_gethumidity(volt); // вычисляем влажность в %
    return true;
}
```

Класс EC_meter

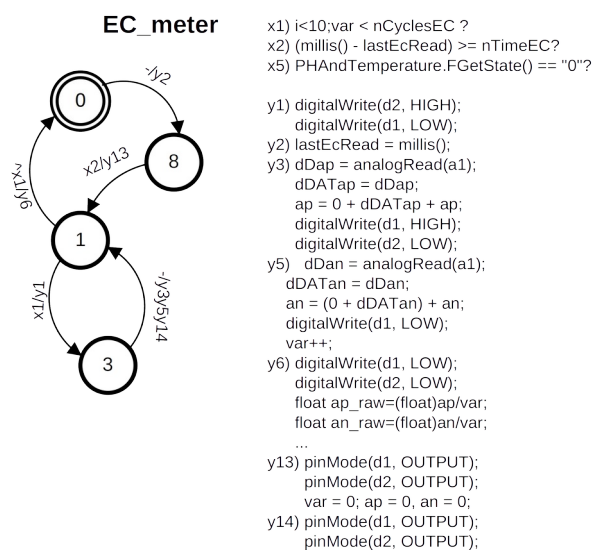
Логика работы класса представлена графом автомата на рис. 16. Объект, представленный данным классом, начинает свою работу только в случае, когда не активен процесс, представленный классом PhAndTemperature. Таким механизмом обеспечивается более надежная и устойчивая работа аналоговых входов микроконтроллера.

После запуска в работу выдерживается пауза перед первым измерением электропроводности. Ее значение определяется действием y2. На переходе в состояние 1 выполняется инициализация/сброс параметров – y13. Далее переходы между состояниями «1» и «3» реализуют измерение электропроводности. Для этого сначала действие y1 включает каналы в нужной комбинации. Затем следует последовательный запуск действий y3, y5, y14. Так

определяется один цикл измерений, который повторяется заданное число раз. За определение этого момента отвечает предикат x1.

Выполнив заданное число циклов, на переходе из состояния «1» в состояние «0» в действии у6 выполняется усреднение измеренных значений, и текущее значение электропроводности присваивается соответствующему свойству класса - EC_асс. Данное значение, как правило, поступает на вход цифрового фильтра, который формирует окончательное значение ЕС, которое также принадлежит объекту EC_meter.

Примечание 1. Если необходимо отключить фильтрацию значения электропроводности, то необходимо напрямую присвоить ЕС значение EC_асс. Аналогичным образом отключение цифровых фильтров будет реализовано, как правило, в общем случае, т.е. и в случае других объектов.



Краткое описание алгоритма

Алгоритм начинает работу, когда завершил свою работу алгоритм измерения PH (x5) (измерения PH и температуры). Затем выполняется заданное число циклов измерения ЕС. Каждый цикл измерения включает два измерения электропроводности в прямом и обратном направлении. В заключении в рамках действия у6 Выполняется истинное значение электропроводности.

Замечание. В рамках начального запуска (см. setup()) запускается метод setup_Ec_meter, реализующий функции начальной установки для измерения ЕС.

Рис. 16.

Класс LUX_meter

Логика работы класса представлена графом автомата на рис. 16. Работа данного класса сводится к циклическому запуску метода у1, который непосредственно выполняет чтение аналогового входа, к которому подключен датчик освещенности. Здесь же выполняется приведение введенного значения к значению освещенности.

```

void LUX_meter::y1()
{
  float sensorValue = analogRead(sensLUX); // Чтение аналогового значения
  float voltage = sensorValue * (3.3 / 4095.0); // Преобразование в напряжение (ESP32 имеет 12-битный АЦП)
  int bDegree=1; // степень - OFF(2) ON(4)
  // OFF - яркая улица
}
  
```

```

if(FCommand1 && FCommand2) { bDegree = 4; }
// ON - темная комната
if(FCommand1 && !FCommand2) { bDegree = 2; }
float x = voltage; float Y;
if (bDegree == 2) {
    // ступень 2 прямая
    Y = 6053.183 * x - 4927.1739; // подобраны калибровочные коэф. Для 2-й ступени
    // ступень 2 степенная
    Y = 1279.753 * pow(x, 2.377);
}
else if (bDegree == 4) {
    // ступень 4 степенная
    Y = 14936.076 * pow(x, 2.035);
}
else {
    Y=9999;
}
L = Y;
// Вывод считанных данных в лог
#ifdef MY_DEBUG
    Serial.print("LUX Value: ");
    Serial.println(LUX);
#endif
}

```

Листинг.

В целях повышения точности определения освещенности можно задавать режим работы датчику, что делается в рамках метода LUXMeterControl (см. листинг). Выбор режима работы – «пасмурно» или «ясно» можно сделать или на закладке Setting web-страницы устройства или на сайте Home Assistant.

```

// Управление LUX_meter
void LUX_meter::LUXMeterControl()
{
    // Новое состояние реле 1 отличается от текущего, требуется переключение
    if (FCommand1 != FStatus1) {
        FCommand1 = FStatus1;
        digitalWrite(gpioF1, FStatus1);
    };
    // Новое состояние реле 2 отличается от текущего, требуется переключение
    if (FCommand2 != FStatus2) {
        FCommand2 = FStatus2;
        digitalWrite(gpioF2, FStatus2);
    };
}

```

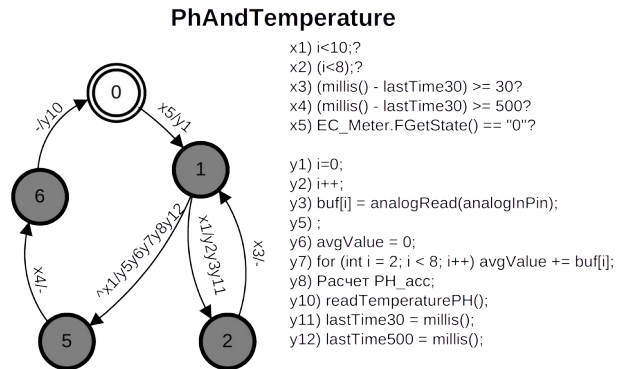
Класс PhAndTemperature

Логика работы класса представлена графом автомата на рис. 17. Объект, представленный данным классом, начинает свою работу только в случае, когда не активен процесс EC_meter (см. выше описание данного класса). Такая связь между данными процессами обеспечивается более надежная и устойчивая работа аналоговых входов микроконтроллера.

После запуска действие y1 сбрасывает счетчик циклов. Далее переходы в состояниях «1» и «2» реализуют один цикл измерений. При этом между одиночными измерениями Ph создается пауза – 30 мсек.

Сделав заданное число циклов на переходе в состояние «5» выполняется усреднение данных. Затем создается пауза в 500 мсек и измеряется температура.

На этом, переходом в состояние «0», текущий цикл измерения Ph и температуры завершается. Далее все повторяется



Краткое описание алгоритма

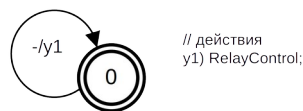
Алгоритм начинает работу, когда завершил свою работу алгоритм измерения ЕС (измерения электропроводности). Затем выполняется заданное число циклов измерения PH. Измеренные значения помещаются в массив измерений. Далее последовательно выполняется ряд действий: массив сортируется - действие y5; Отбрасывается минимальное и максимальное значения и оставшиеся суммируются – y7; значения усредняются приводятся к заданной форме – y8. В заключение реализуется задержка между измерениями – переход 5->6->0.

Рис. 17

RelayControl

Логика работы класса представлена графом автомата на рис. 18. В цикле текущие состояния каждого из реле запоминаются в соответствующих переменных. Для каждого из реле это будут переменные relayStatus1, relayStatus2, relayStatus3. Новые состояния реле должны соответствовать переменным – led1State, led2State, led3State. Эти переменные устанавливаются или на закладке Control Panel – кнопки R1, R2 и V (см. функцию handleGPIO() в модуле SendHTML.cpp), или переключателями на сайте Home Assistant. В последнем случае обрабатывают обработчики событий от переключателей в модуле SD_MQTT.cpp. Они задают новое состояние реле и обновляют закладки сайта устройства, чтобы обновить текущее состояние кнопок, отражающих текущее положение реле.

RelayControl



Краткое описание алгоритма

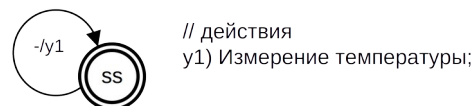
Управление реле сводится к циклическому запуску функции RelayControl(), которая сравнивает сохраненное состояние реле с необходимым значением и затем приводится в соответствии с тем, что задано.

Рис. 18

SensorDS18B20

Логика работы класса представлена графом автомата на рис. 19. В цикле выдается запрос к библиотеке на получение температуры. Данную процедуру отражает код листинга.

SensorDS18B20



Краткое описание алгоритма

В цикле выполняются замеры температуры, реализуемые в рамках внешней библиотеки работы с датчиком DS18B20.

Рис. 19

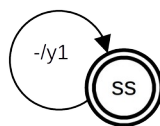
```
void SensorDS18B20::y1()
{
    // отправляем запрос на измерение температуры
    sensor.requestTemperatures();
    // считываем данные из регистра датчика
    sensDS18B20 = sensor.getTempCByIndex(0);
    // выводим температуру в Serial-порт
    #if defined (MY_DEBUG)
    Serial.print("Temp(DS18B20) C: ");
    Serial.println(DS18B20);
    #endif
}
```

Листинг.

SensorLevel

Логика работы класса представлена графом автомата на рис. 20. В цикле сначала считывается текущее значение входов микроконтроллера, к которым подключены датчики уровня. Затем анализируется их состояние, которой приводится к значению, отражающему в процентном соотношении заполнение бака жидкостью. Все эти процедуры отражает код листинга.

SensorLevel



// действия
у1)Измерение значения
уровня жидкости;

Краткое описание алгоритма

В рамках действия у1 происходит опрос трех цифровых входов отражающий состояния текущего уровня жидкости. Их текущая комбинация соситояния преобразуется в уровень жидкости, выраженный процентами наполнения бака.

Рис.

```
void SensorLevel::y1() {  
    levelStatus1 = digitalRead(gpioLevel1);  
    levelStatus2 = digitalRead(gpioLevel2);  
    levelStatus3 = digitalRead(gpioLevel3);  
    if (!levelStatus1&&levelStatus2&&levelStatus3) fLevel=30;  
    else if (levelStatus1&&!levelStatus2) fLevel=60;  
    else if (!levelStatus3) fLevel=90;  
    else if (levelStatus1&&levelStatus2&&levelStatus3) fLevel=0;  
    else fLevel=-1;  
}
```