МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт энергетики и электротехники
(институт)

Кафедра «Промышленная электроника»

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки, специальности)

Промышленная электроника
направленность (профиль)

БАКАЛАВРСКАЯ РАБОТА

на тему:Учебный стенд «системы умного дома»

Студент	Р.В. Савин		
	(И.О. Фамилия)		(личная подпись)
Руководитель	А.В. Прядилов		
	(И.О. Фамилия)		(личная подпись)
Консультант	О.Н. Брега		
	(И.О. Фамилия)		(личная подпись)

Заведующий кафедрой к.т.н., доцент А.А. Шевцов _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Тольятти 2017

Бланк задания на проектирование

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Тольяттинский государственный университет»

ЭНЕРГЕТИКИ И ЭЛЕКТРОТЕХНИКИ

(институт)

Промышленная электроника

(кафедра)

УТВЕРЖДАЮ

Зав. кафедрой «Промышленная электроника»

_____ А.А. Шевцов

(подпись) (И.О. Фамилия)

«____»_____20___г.

ЗАДАНИЕ

на выполнение бакалаврской работы

Студент*Савин Роман Валерьевич, ЭЛб-1301*

1. Тема *Учебный стенд «системы умного дома»*

2. Срок сдачи студентом законченной выпускной квалификационной работы *30.05.2017*

3. Исходные данные к выпускной квалификационной работе

Работа стендов в сети WI-FI.

Использование датчиков освещённости, температуры, движения.

Возможность удалённого подключения к стенду через сайт.

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов, разделов)

Аннотация

Введение

1. Состояние вопроса

1.1. Формулирование актуальности, цели и задач проекта

1.2. Анализ исходных данных и существующих решений

2. Разработка учебного стенда

2.1. Разработка схемы стенда, выбор комплектующих и оборудования

2.2. Разработка программного обеспечения для стенда

2.3. Выбор комплектующих и оборудования

3. Практическая реализация проекта

3.1. Монтаж стенда

3.2. Подключение и настройка частей стенда, отладка программной части.

4. Пособия по использованию стенда

5. Эффективность работы

Заключение

Список литературы

5.Ориентировочный перечень графического и иллюстративного материала

1. Обзорный лист

2. Схема стенда

3. Блок схема стенда

4. Плакат, иллюстрирующих работу со стендом

5. Блок-схема алгоритма программы системы.

6. Сайт управления системой

7. Дата выдачи задания «26» января 2017 г.

Руководитель выпускной квалификационной работы		А.В. Прядилов
	(подпись)	(И.О. Фамилия)
Задание принял к исполнению		Р.В. Савин
	(подпись)	(И.О. Фамилия)
Консультант		О.Н. Брега
	(подпись)	(И.О. Фамилия)

АННОТАЦИЯ

Объем 101 с., 27 рис., 1 табл., 20 источников, 2 прил.

Объектом исследования являются системы бытовой автоматизации.

Цель работы — повышение эффективности обучения учащихся принципам работы компонентов системы бытовой автоматизации и созданию управляющих программ, реализующие логику работы системы.

Задачи заключались в разработке аппаратно-программного стенда для выполнения практических работ по управлению устройствами системы бытовой автоматизации с использованием языка программирования высокого уровня Pythonи протокола передачи данных TCP/IP.

Работа состоит из пяти глав, в которых решены упомянутые задачи.

Для организации передачи данных и в качестве исполнительных устройств использованы микропроцессорные модули ESP8266-01 с интерфейсом Wi-Fi. В качестве управляющего модуля использован одноплатный компьютер Raspberry Pi 3. Для создания прошивок микроконтроллеров ESP8266 использована интегрированная среда разработки ArduinoIDE. Для создания управляющих программ работы управляющего модуля использована интегрированная среда разработки Python IDLEоперационной системы Raspbian. А также реализовано дистанционное управление стендом, через устройства на базе операционной системы Android.

В процессе работы был созданаппаратно-программный стенд для организации практического обучения учащихся навыкам написания программ на языке Python, а также основам передачи данных в локальной сети и в сети интернет. Аппаратно-программный стендпозволяет получать информацию с датчиков и удаленно управлять комплексом устройств, подключенных к сети Wi-Fi.

Степень внедрения — установка по разработанной документации является опытным образцом.

Областью применения аппаратно-программного стенда являются учебные заведения, однако возможно его использование в экспериментальных целях в организациях соответствующего рода деятельности.

Разработанные методические пособия по выполнению лабораторных работ позволят повысить эффективность обучения учащихся языку программирования высокого уровня Python, а также получить основные представления о принципах работы системы бытовой автоматизации.

Abstract

The topic of thegraduation work is “Laboratory stand for studying of home automation systems.” This work presents the development of the system’s technical and software part.

Before the development of the laboratory stand, the existing home automation systems, the principles of their operation, and programming were studied. The task was to create a training stand for students, by working with which they can learn how to develop independently such systems and how to develop software for them.

A working model consists of three parts: a block with a set of sensors and Wi-Fi modules, a block with relays and Wi-Fi modules, and control devices based on Raspberry Pi 3. For the work of the stand it was developed software accompanying with a technical guide.

The laboratory work is aimed at teaching of students of the development of home automation systems, and the development of control software. For this case, the Python programming language was chosen. There have been written 8 laboratory practices, in which various automation tasks like the following are being phased in: control of lighting, heating, air conditioning, ventilation, irrigation systems, fire safety systems and protection against pipe leaks. Software to control all the systems simultaneously and implement graphical user interface controls was also written.

In addition to the training purpose, the stand can be used as a demonstration model. For this, a program for the Android will be provided, and there will be an opportunity to demonstrate the operation of the home automation system by monitoring the stand via a smartphone.

The graduation work can be divided into several parts:
1. Creation of a home automation system, which will be implemented as a laboratory stand with the possibility of demonstration;
2. Development of software for sensors and relays, as well as the development of programs for system operation;
3. Development of a technical guide for students’ laboratory works and for studying the principles of automation, programming and control of such systems.

The technical guide contains examples of software, the code of which is provided with detailed comments.

Working with the stand, students will be able to acquire professional skills in the development of intelligent systems, expand their knowledge in the field of automation, and learn methods of wireless data transmission.

Prospects for obtaining knowledge in this area are explained by the growing demand of people for automation systems. Automation is used both in residential buildings, and in production, offices, and shops. It increases the need for professionals capable of developing such systems. And since knowledge of this kind is of great importance in our time, we can say that such equipment is very important.

Список обозначений и сокращений

API	Aapplication Programming Interface, интерфейс прикладного программирования
GPIO	General-PurposeInput/Output, интерфейс ввода/вывода общего назначения
HTML	HyperTextMarkupLanguage, язык гипертекстовой разметки
HTTP	HyperText Transfer Protocol, протоколпередачигипертекста
IDE	Integrated Development Environment, интегрированная среда разработки
TCP/IP	Transmission Control Protocol/Internet Protocol, наборсетевыхпротоколовпередачиданных
А	ампер
БД	База данных
В	вольт
ГБ	гигабайт
КБ	килобайт
кОм	килоом
МГц	мегагерц
млн.	миллион
млрд.	миллиард
см	сантиметр

Содержание

ВВЕДЕНИЕ

В последние годы все шире получает распространение концепция бытовой автоматизации – системы бытовых устройств, способных решать определенные задачи без участия человека. Наиболее известный пример бытовой автоматизации – так называемый «умный дом». «Умный дом» – это комплекс приборов, позволяющий автоматизировать включение-выключение освещения, работу отопительной системы, вентиляции и кондиционирования, работу сигнализации несанкционированного доступа в помещения, возгорания или протечки, а также многие другие операции. Кроме того, «умный дом» позволяет владельцу удаленно получать информацию о своих устройствах и удаленно ими управлять. Применение устройств бытовой автоматизации возможно не только в доме, но и в теплице, в гараже, на даче, в сауне – практически в любой области быта человека [16].

Дальнейшим развитием концепции бытовой автоматизации является «интернет вещей» - сеть физических предметов («вещей»), имеющих встроенные технологии для взаимодействия друг с другом или с внешней средой. Данная отрасль технологии с каждым годом получает все большее распространение. По оценкам аналитиков, в 2016 году количество устройств, подключенных к Интернету вещей, составило 6,4 млрд. единиц, увеличившись на 30% по сравнению с 2015 годом [14].

Непосвященному человеку, знакомому с бытовой автоматизацией лишь по ценникам на системы «умный дом» в магазинах бытовой техники, может показаться, что проектирование и управление такими системами очень сложно и недоступно для его понимания. Однако применение микропроцессорной техники в бытовой автоматизации позволяет инкапсулировать сложные процессы получения, обработки и передачи данных внутри типовых модулей. В настоящее время проектирование и построение управляющих систем типа

«умный дом» доступно любому инженеру, знакомому с основами микропроцессорной техники и программирования.

Разработанный аппаратно-программный стенд позволяет учащимся наглядно увидеть принципы работы компонентов системы бытовой автоматизации и научиться создавать управляющие программы, реализующие логику работы системы.

1 Состояние вопроса.

В продаже имеется большое количество готовых систем бытовой автоматизации, автоматизирующих типовые процессы управления в доме. Многие фирм предлагают проектирование и монтаж систем под требования конкретного покупателя, такие системы как правило имеют намного большую функциональность (и цену). Однако имея понятие о принципах работы принципах работы микропроцессорной техники и некоторые навыки программирования, вполне возможно собрать систему бытовой автоматизации с требуемой функциональностью из недорогих компонентов в домашних условиях.

В основе бытовой автоматизации лежат:

- датчики, позволяющие получить информацию о состоянии контролируемых физических параметров;

- исполнительные устройства, позволяющие включать/выключать бытовые приборы или производить другие действия;

- сеть передачи данных, позволяющая передавать информацию с датчиков в управляющий модуль и команды от управляющего модуля в исполнительные устройства;

- управляющий модуль, позволяющий автоматизировать работу системы согласно управляющей программе.

В рамках данной бакалаврской работы разработан аппаратно-программный стенд, имитирующий систему автоматизации «умный дом». Система имеет полную функциональность и вполне может применятьсяв практических условиях. Для организации передачи данных и в качестве исполнительных устройств использованы модули ESP8266-01 с интерфейсом Wi-Fi. В качестве управляющего модуля использован одноплатный компьютер Raspberry Pi 3.

Основная задача стенда рамках данной бакалаврской работы – позволить учащимся научиться создавать управляющие программы, реализующие логику

работы системы. На данном этапе обучения учащимся предлагается организовать взаимодействие управляющего модуля Raspberry Pi 3 с исполнительными модулями ESP-01 с помощью готового API, заложенного в прошивки модулей ESP-01. Для каждого модуля ESP-01 разработана собственная прошивка, реализующая чтение датчиков и управление подключенными устройствами, а также APIпередачи данных.

Взаимодействие управляющего модуля Raspberry Pi 3 с исполнительными модулями ESP-01 учащиеся выполняют с помощью программы на языке высокого уровня Python. Python — современный объектно-ориентированный мультиплатформенный язык, и он входит в состав операционной системы Raspbian.

В рамках бакалаврской работы разработана принципиальная схема подключения электронных компонентов, входящих в состав аппаратно-программного стенда, разработаны прошивки модулей ESP-01, разработаны примеры пяти управляющих программ на языке Pythonдля управляющего модуля. В практической части бакалаврской работы разработано пять методических пособий для учащихся по выполнению указанных практических занятий.

Также эта работа содержит указания на возможности изменения отдельных частей практических заданий для получения большего количества вариантов заданий.

Использование разработанного аппаратно-программного стенда не ограничивается написанием программ на языке Python. Стенд позволяет учащимся заглянуть в процесс программирования микроконтроллеров, изучить взаимодействие микроконтроллера с внешними устройствами, портами ввода-вывода, работу с подключаемыми библиотеками, аппаратными прерываниями и многое другое, что составляет увлекательный мир микроконтроллеров. Также учащиеся могут получить начальные навыки системотехники и проектирования электронных схем.

Для обеспечения возможности дальнейшего развития применения стенда в качестве учебного пособия исходный код прошивок модулей ESP-01 снабжен подробными комментариями, что позволяет при необходимости легко разработать учебный материал для обучения программированию микроконтроллеров с помощью аппаратно-программного стенда.

1.1 Формулирование актуальности, целей и задач проекта.

Перспективность данной тематики обусловлена ростом потребности людей в системах автоматизации. Автоматизация применяется, как и в жилых домах так и в производстве, офисах, магазинах. Тем самым растёт потребность в специалистах способных разработать подобные системы. Но для обучения таких специалистов требуется учебное оборудование в работе с которым студент сможет поэтапно изучить принципы создания и программирования систем автоматизации.

Целью бакалаврской работы является разработка лабораторного стенда, являющего собой систему домашней автоматизации, включающую в себя набор датчиков и исполнительных устройств, связанных в сети Wi-Fi через передатчиков, а также центральное устройство управления системой на базе RaspberryPi 3B.

Основными задачами являются:

1. Создание системы домашней автоматизации, которая будет реализована в качестве лабораторного стенда с возможностью выступать в роли демонстрационного

2. Разработка программного обеспечения для передачи и приема информации, а также разработка программ для управления системой

3.Разработка методического пособия для проведения лабораторных работ студентами по изучению принципов работы автоматизации, программирования подобных систем и их управлению.

1.2 Анализ исходных данных и существующих решений.

На данный момент существует множество устройств и систем, которые так или иначе попадают под определение «умный дом», но все они предназначены для непосредственного использования в быту. Обучающих же систем вовсе не имеется. Конечно, есть множество хорошего учебного оборудования так или иначе затрагивающего область автоматизации, но все оно не предназначено для обучения специалистов в области домашней автоматизации. Исходя из этого была поставлена, задача создать учебное оборудование, непосредственно предназначенное для обучения проектированию систем «умный дом»

2 Разработка учебного стенда.

Работа будет состоять из следующих этапов:

1. Разработка программного обеспечения для Wi-Fi модулей ESP8266-01, а также разработка управляющих программ на языке Python.

2. Сборка и программирование лабораторного стенда, проверка работы и отладка.

3. Создание методического пособия для выполнения лабораторных работ.

2.1 Используемые электронные компоненты.

В 2014 году китайский производитель Espressif представил микроконтроллер ESP8266 с интефейсом Wi-Fi. Микроконтроллер имеет до 15 портов GPIO (в зависимости от реализации конкретной модели) для подключения внешних устройств, работает на частоте 80 МГц, имеет 80 КБ оперативной памяти. Встроенный интерфейс Wi-Fi позволяет организовать беспроводное соединение для передачи данных [10].

На базе микроконтроллера ESP8266 выпущена линейка устройств, отличающихся количеством распаянных портовGPIO, габаритами и энергопотреблением. Наибольшее распространение получил модуль ESP-01 (рисунок 1). Данный модуль имеет габариты всего 1,5 х 2,5 см, низкое энергопотребление и очень низкую стоимость (менее 200 рублей), что позволяет широко использовать его в системах бытовой автоматизации.

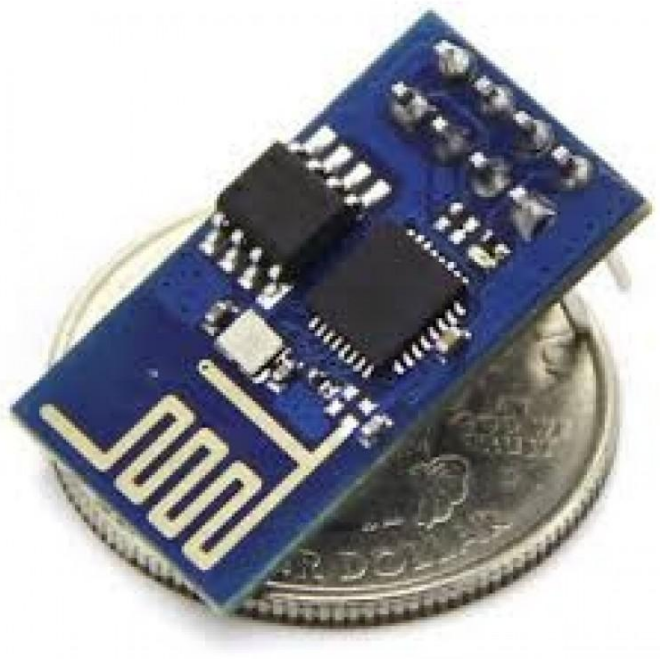

Рисунок 1 –Модуль ESP-01 на базе микроконтроллера ESP8266.

Модуль ESP-01 имеет всего 2 распаянных порта GPIO общего назначения, как правило этого достаточно для решения наиболее типичных задач – подключение 1-2 датчиков, управление 1-2 устройствами или даже управление 1 устройством на основании показаний 1 датчика с помощью внутренней управляющей программы.

Имеющиеся программные библиотеки позволяют использовать модуль ESP-01 как в качестве клиента сети, передающего данные на указанный адрес, так и в качестве сервера. Заводская прошивка микроконтроллера может быть легко заменена на собственную управляющую программу, что открывает большие возможности для творчества.

В качестве управляющего модуля использован одноплатный компьютер Raspberry Pi 3 (рисунок 2). Третья модель популярного одноплатного компьютера выпущена в 2016 году. Он оснащен 4-ядерным процессором с частотой 1200 МГц, имеет 1 ГБ оперативной памяти и встроенные беспроводные интерфейсы Wi-Fiи Bluetooth. Raspberry Pi 3 имеет небольшие габариты – размером чуть больше банковской кредитной карты.

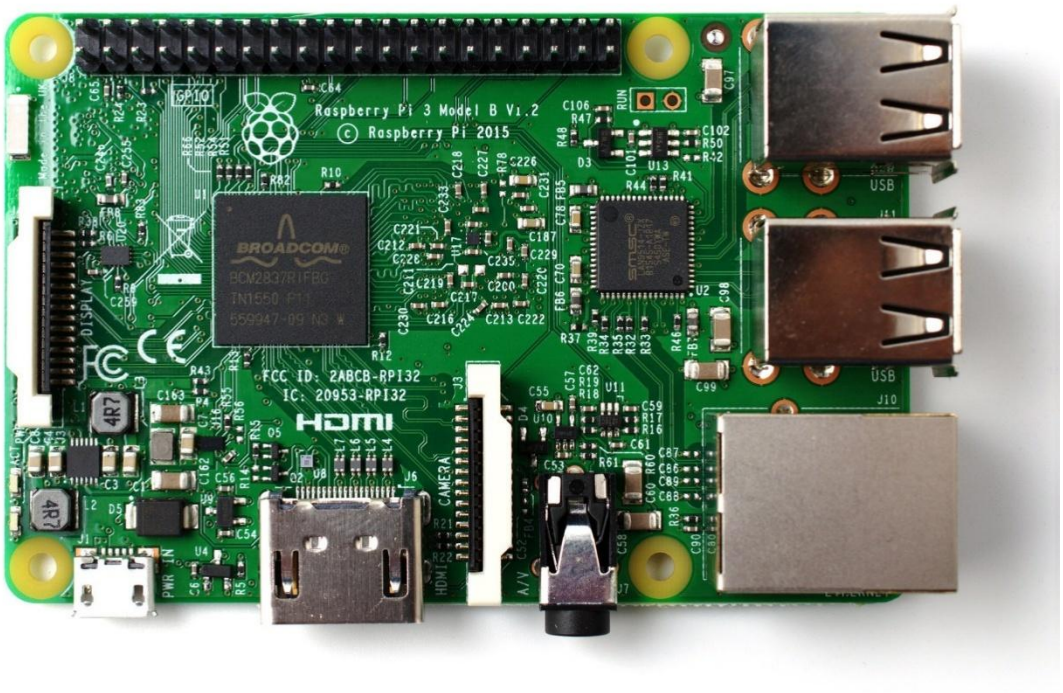

Рисунок 2 - Одноплатный компьютер Raspberry Pi 3.

Этот одноплатный компьютер изначально разработан для обучения программированию, поэтому в состав операционной системы Raspbian входит объектно-ориентированный мультиплатформенный языкPython [6].

Для получения информации о значениях контролируемых физических параметров использованы следующие датчики:

- датчик температуры и влажности воздуха DHT11 (рисунок 3). Состоит из двух компонентов — емкостного датчика влажности и термистора. Датчик позволяет определять влажность в диапазоне от 20 до 80 % и температуру в диапазоне от о до +50 градусов. Чип, находящийся внутри, выполняет аналого-цифровое преобразование и выдает цифровой сигнал, который можно считать с помощью любого микроконтроллера. Датчик DHT11 не обладают высоким быстродействием и точностью, но зато прост, недорог и отлично подходит для обучения;

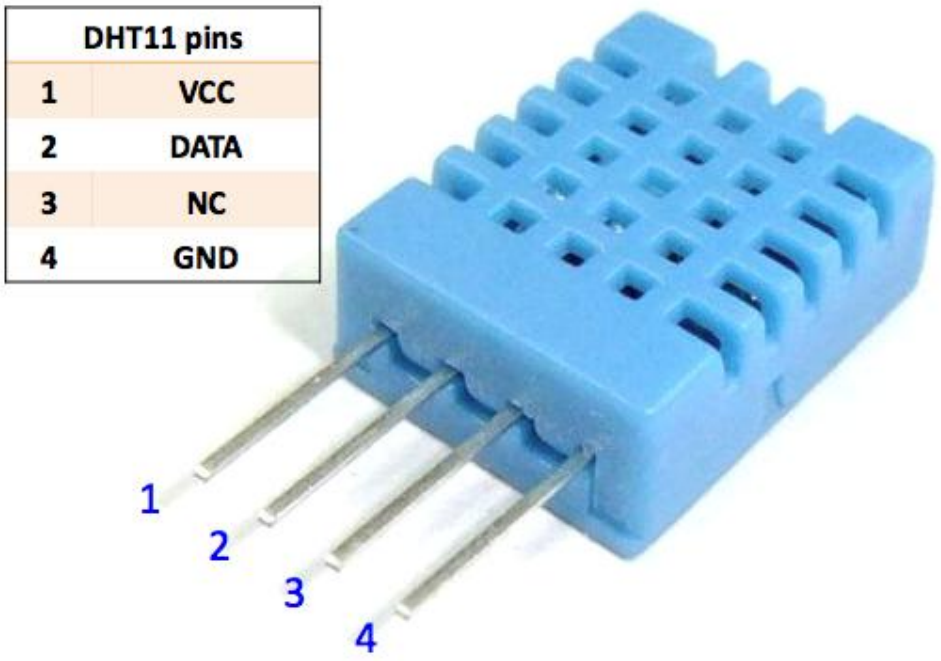

Рисунок 3 - Датчик температуры и влажности воздуха DHT11.

- датчик освещения. В качестве датчика применен фоторезистор VT90N2 (рисунок 4). Также может быть применен любой другой фоторезистор с подходящими параметрами. Поскольку модуль ESP-01 не имеет распаянного аналогового входа, фоторезистор используется в цифровом режиме (два показания – освещение достаточное либо недостаточное);

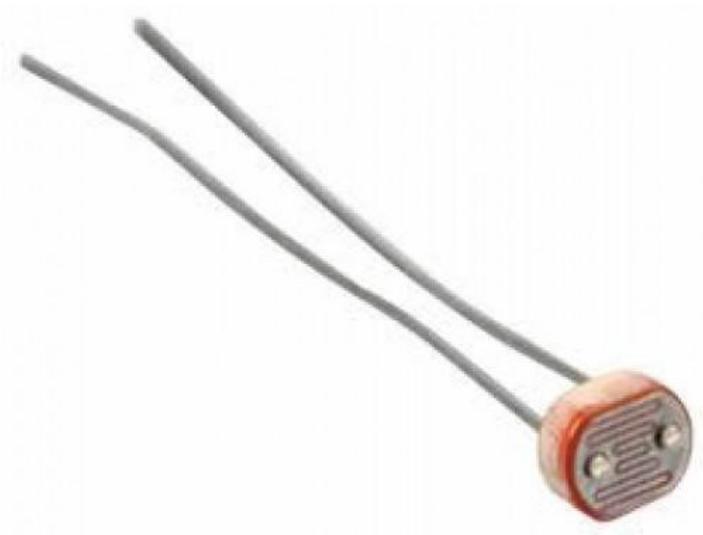

Рисунок 4 - Фоторезистор VT90N2.

- датчик влажности почвы FC-28 (рисунок 5)или аналогичный. Датчик позволяет получать показания в двух форматах – аналоговое значение влажности почвы (от 0 до 1023) или цифровое значение (сухая/влажная). Ввиду отсутствия распаянного аналогового входа на модуле ESP-01 используем цифровое значение;

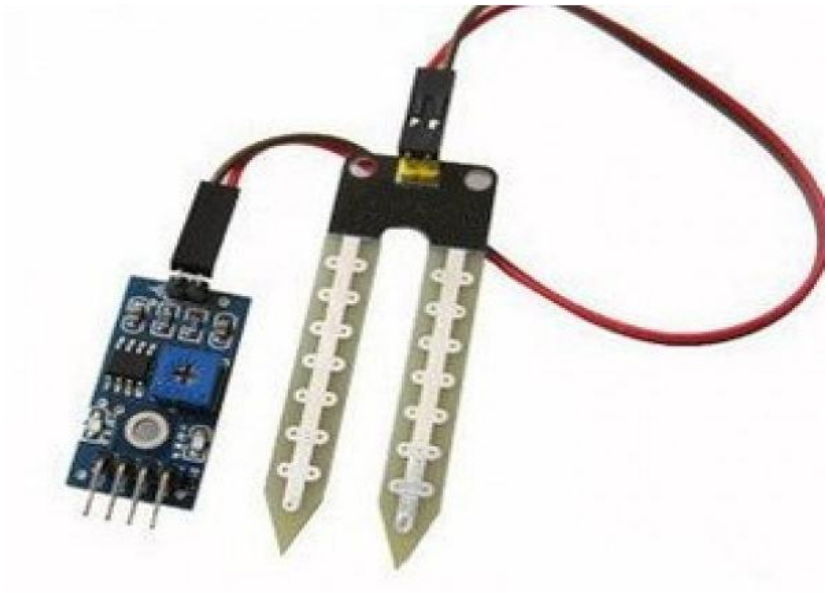

Рисунок 5 - Датчик влажности почвы FC-28.

- датчик влагиFC-37 (рисунок 6)или аналогичный. Датчик позволяет получать показания в двух форматах – аналоговое значение наличия влаги (от 0 до 1023) или цифровое значение (сухо/влажно). Ввиду отсутствия распаянного аналогового входа на модуле ESP-01 используем цифровое значение;

Рисунок 6 - Датчик влагиFC-37.

- инфракрасный датчик движенияHC-SR505 (рисунок 7) либо любой другой инфракрасный датчик с цифровым выходом. Датчик позволяет обнаруживать перемещение объектов (например, людей) в зоне действия датчика. Датчик при срабатывании выдает на выход логическую единицу, и сохраняет ее в течение 8 секунд;

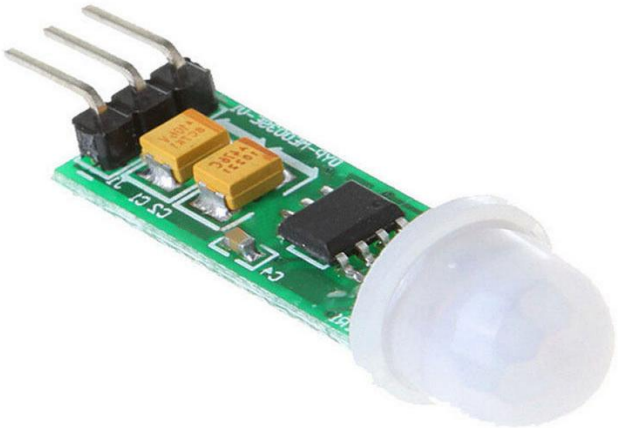

Рисунок 7 - Инфракрасный датчик движенияHC-SR505.

- датчик газа MQ-4 (бутан, пропан, метан, дым) (рисунок 8). Датчик позволяет получать показания в двух форматах – аналоговое значение концентрации газа (от 0 до 1023) или цифровое значение (есть/нет). Ввиду

отсутствия распаянного аналогового входа на модуле ESP-01 используем цифровое значение;

Рисунок 8 - Датчик газа MQ-4(бутан, пропан, метан, дым).

Датчики подключаются к модулям ESP-01 в следующей конфигурации:

- модуль ESP-01 № 1: вход GPIO 0 - датчик температуры и влажности воздуха DHT11, вход GPIO 2 - фоторезистор VT90N2;

- модуль ESP-01 № 2: вход GPIO 0 - датчик влажности почвы FC-28, вход GPIO 2 - инфракрасный датчик движенияHC-SR505;

- модуль ESP-01 № 6: вход GPIO 0 - Датчик влаги FC-37, вход GPIO 2 - Датчик газа MQ-4;

Для управления устройствами применены модули на основе электромеханического реле SDR-05VDC-SL-C (рисунок 9). Модули позволяют управлятьподключаемой нагрузкой до 10А, с напряжением постоянного тока до 30 В, или переменного тока до 250 В.

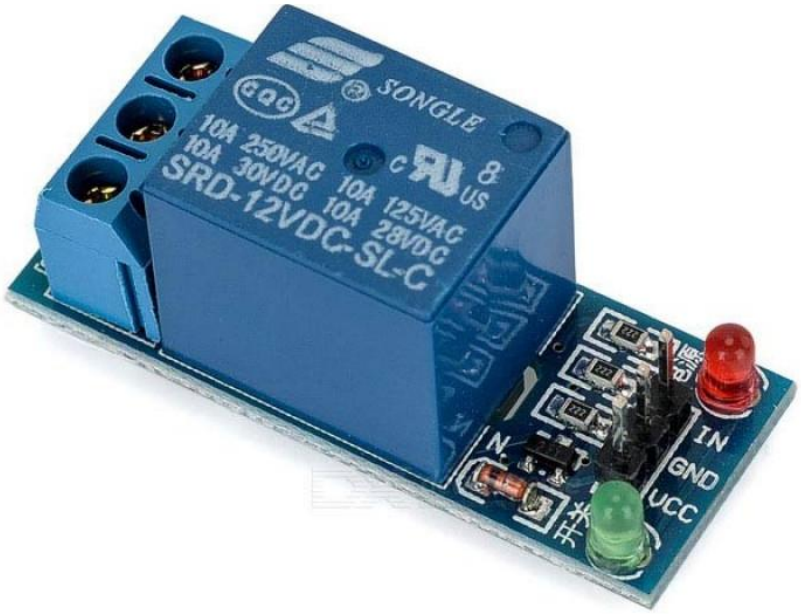

Рисунок 9 – Модуль электромеханического реле SDR-05VDC-SL-C.

Электромеханические реле подключаются к модулям ESP-01 в следующей конфигурации:

- модуль ESP-01 № 3: выход GPIO 0 – реле элекрообогревателя, выход GPIO 2 – реле кондиционера;

- модуль ESP-01 № 4: выход GPIO 0 – реле помпы полива растений, выход GPIO 2 – реле освещения;

- модуль ESP-01 № 5: выход GPIO 0 – реле увлажнителя воздуха, выход GPIO 2 – реле вытяжной вентиляции;

- модуль ESP-01 № 7: выход GPIO 0 – реле электрического клапана системы водоснабжения, выход GPIO 2 – реле системы пожаротушения;

2.2 Принципиальные схемы подключения.

Подключение датчиков и реле производится к модулям ESP-01 по следующим схемам:

- модуль ESP-01 № 1 (рисунок 10)

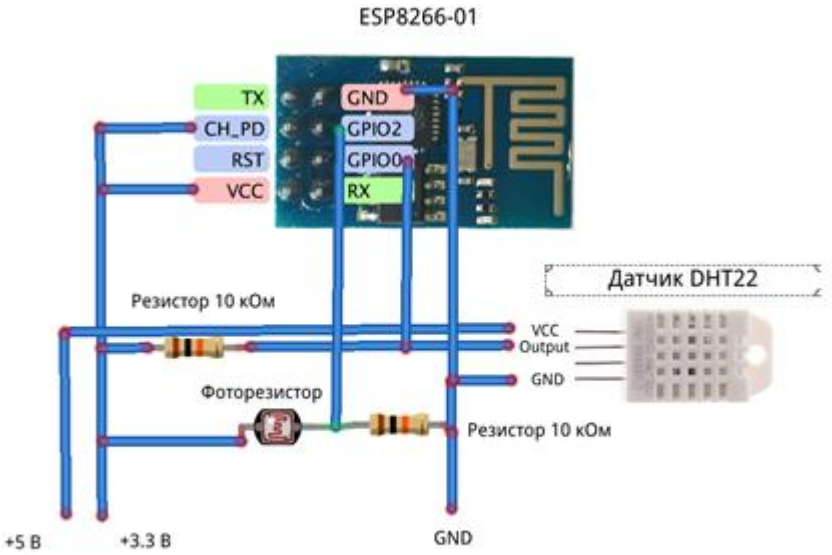

Рисунок 10 – Схема подключения модуля ESP-01 № 1.

- модуль ESP-01 № 2 (рисунок 11)

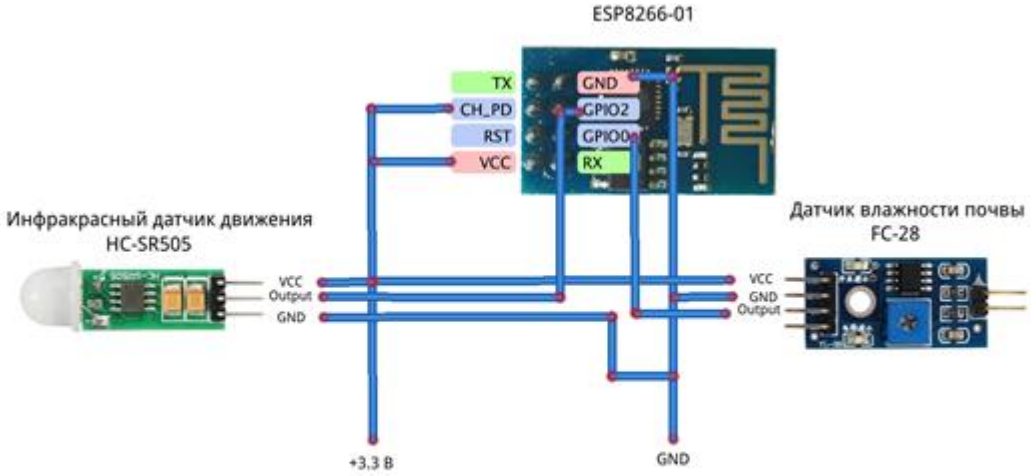

Рисунок 11 – Схема подключения модуля ESP-01 № 2.

- модуль ESP-01 № 6 (рисунок 12)

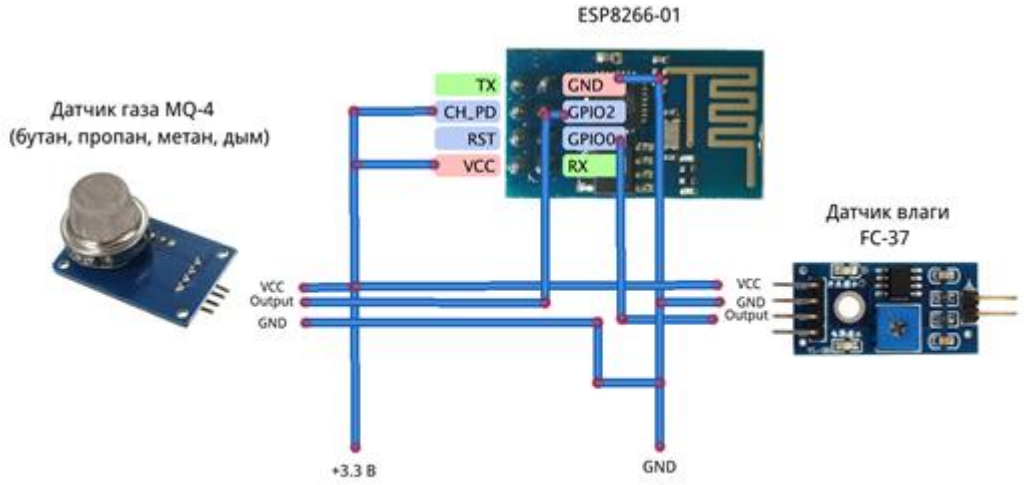

Рисунок 12 – Схема подключения модуля ESP-01 № 6.

- модуль ESP-01 № 3 (рисунок 13)

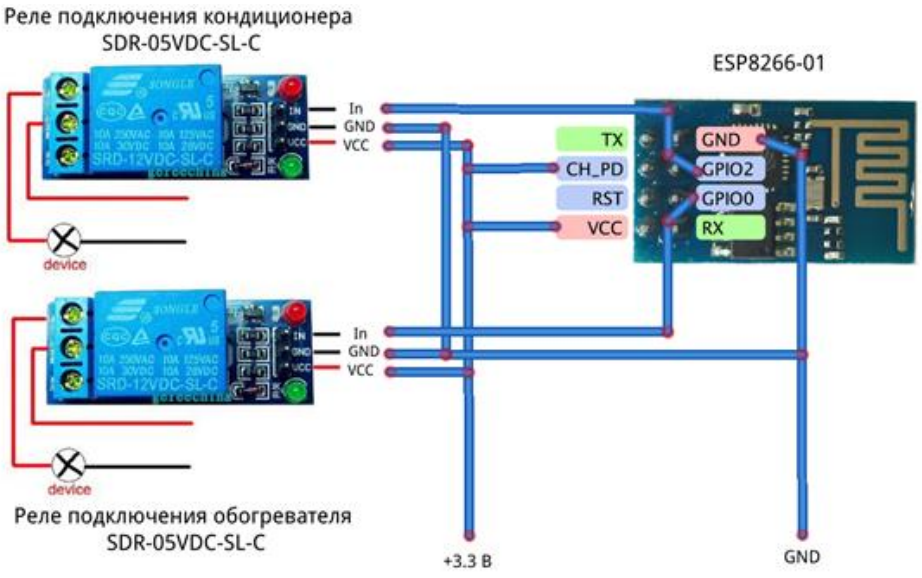

Рисунок 13 – Схема подключения модуля ESP-01 № 3.

- модуль ESP-01 № 4 (рисунок 14)

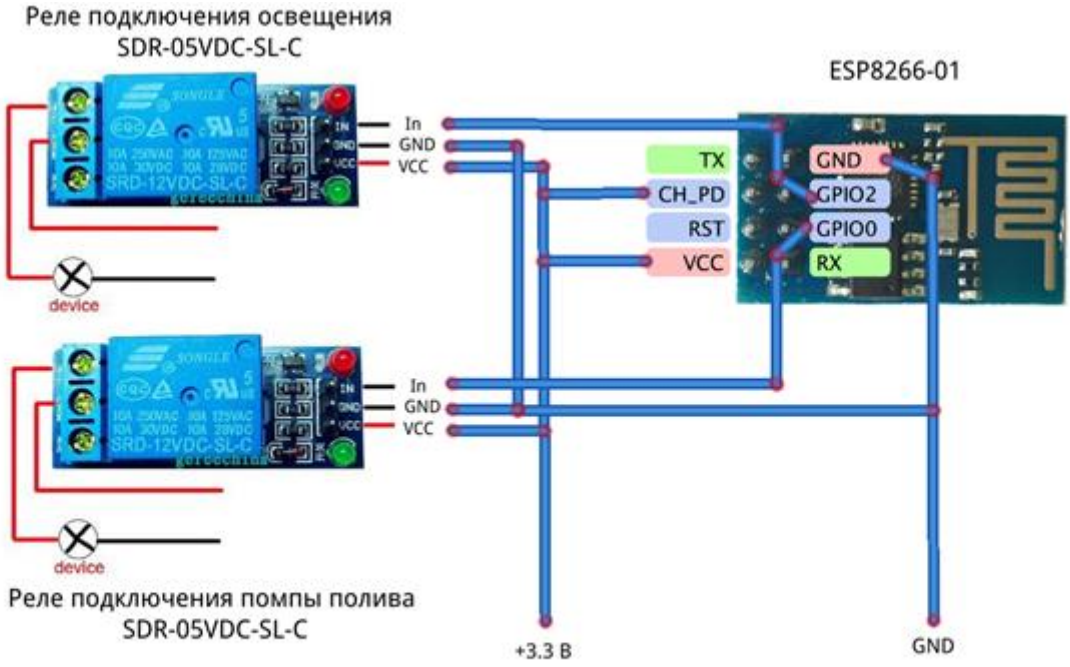

Рисунок 14 – Схема подключения модуля ESP-01 № 4.

- модуль ESP-01 № 5 (рисунок 15)

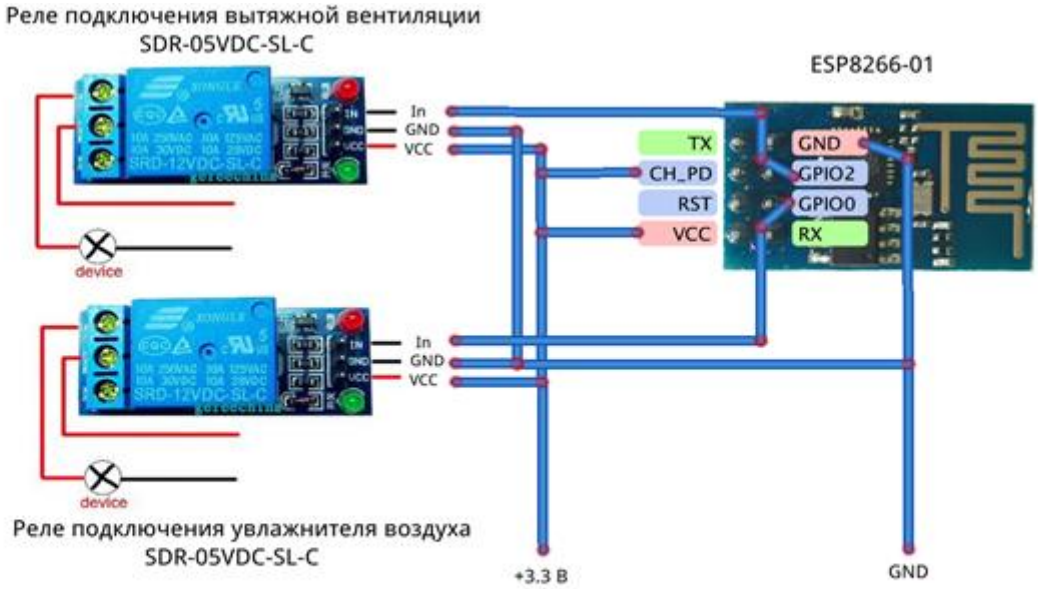

Рисунок 15 – Схема подключения модуля ESP-01 № 5.

- модуль ESP-01 № 7 (рисунок 16)

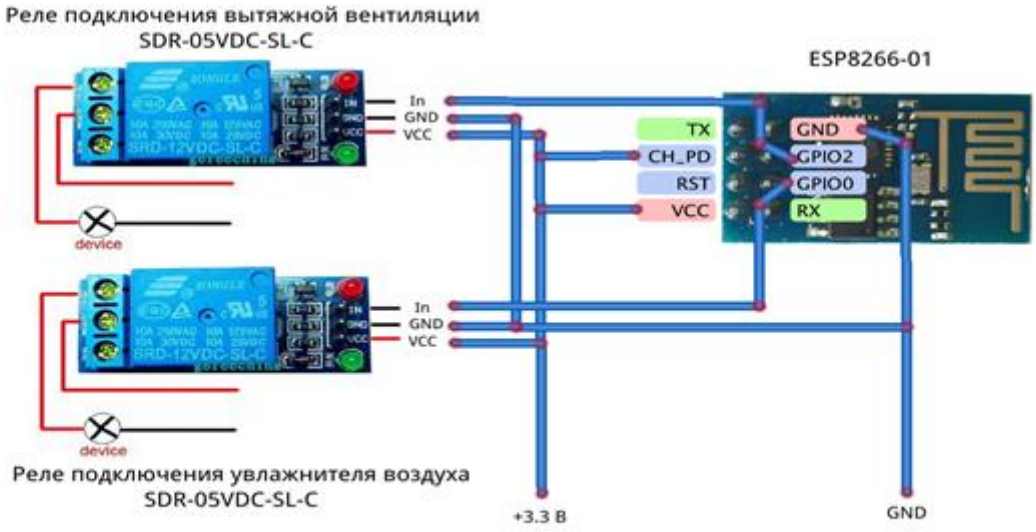

Рисунок 16 – Схема подключения модуля ESP-01 № 7.

Спецификация используемых электронных компонентов приведена в таблице 1.

Таблица 1 - Спецификация используемых электронных компонентов.

№ п/п	Наименование	Кол-во
1.	Микроконтроллер ESP8266-01	7
2.	Датчик температуры и влажности воздуха DHT11	1
3.	Фоторезистор VT90N2 или аналог	1
4.	Датчик влажности почвы FC-28	1
5.	Датчик влаги FC-37	1
6.	Инфракрасный датчик движенияHC-SR505	1
7.	Датчик газа MQ-4 (бутан, пропан, метан, дым)	1
8.	Электромеханического реле SDR-05VDC-SL-C	8
9.	Резистор 10 кОм	2

2.3 Прошивки модулей ESP-01.

Микроконтроллер ESP8266 позволяет использовать собственные прошивки для организации управления подключенными устройствами. Существует большое количество готовых прошивок, предоставляющих разные возможности. Однако для построения разработанного в рамках данной бакалаврской работы аппаратно-программного стенда эти прошивки во-первых в большой степени избыточны, во-вторых не отвечают основной задаче стенда – организация обучения. Поэтому в рамках бакалаврской работы разработаны собственные прошивки для каждого модуля ESP-01.

Разработанные прошивки позволяют получать информацию о значениях контролируемых физических параметров с подключенных датчиков и передавать управляющие сигналы на блоки электромеханических реле для управления подключенными приборами. Для взаимодействия модулей ESP-01 с управляющим модулем RaspberryPi 3 для каждого модуля ESP-01 разработан собственный API, который также реализован в прошивках.

Для разработки прошивок микроконтроллера ESP8266 имеются разные инструменты. Некоторые из них связаны с уже имеющейся прошивкой широкого назначения и реализуют программную надстройку над ней (например, LuaUploader, использующий прошивкуNodeMCU и позволяющий писать управляющие программы на скриптовом языке Lua). Другие позволяют писать прошивку на чистом языке C (либо C++), компилировать и загружать ее в память микроконтроллера для работы (среда SDK ESP8266, среда Eclipse). В рамках бакалаврской работы выбран способ разработки прошивок микроконтроллера ESP8266 с использованием среды ArduinoIDE.

Среда разработки ArduinoIDEзначительно более простая для новичков в программировании, чем SDK ESP8266 и Eclipse [5]. Эта среда, так же, как и сам микроконтроллер Arduino, разработана специально для обучения программированию и получила большое распространение за счет сочетания простоты написания программ, больших возможностей управления

микроконтроллерами и большого количества библиотек поддержки различных аппаратных модулей и датчиков.

Как указывалось ранее, разработанный аппаратно-программный стенд может использован как для обучения программированию на языке Python, так и для обучения программированию микроконтроллеров. В рамках данной бакалаврской работы подробно рассматривается организация обучения программированию на языке Python с помощью аппаратно-программного стенда. Описание функционирования прошивок лежит за рамками бакалаврской работы, однако исходный код прошивок снабжен подробными комментариями, что позволяет при необходимости легко разработать учебный материал для обучения программированию микроконтроллеров с помощью аппаратно-программного стенда. Листинги исходного кода прошивок приведены в приложении 1.

2.4 API взаимодействия с модулями ESP-01.

Микроконтроллер ESP8266 позволяет использовать собственные прошивки для организации управления подключенными устройствами. Существует большое количество готовых прошивок, предоставляющих

Для взаимодействие управляющего модуля Raspberry Pi 3 с исполнительными модулями ESP-01 для каждого модуля разработан собственный API, заложенный в прошивки модулей ESP-01. APIпозволяет управляющему модулю получать информацию с датчиков и управлять подключенными устройствами удаленно через интерфейс Wi-Fi [11].

Каждый модуль ESP-01 имеет собственный фиксированный адрес внутри локальной сети, обращение к модулям производится по этому адресу, с использованием протокола TCP/IP [9]. Адреса модулей в локальной сети:

- модуль ESP-01 № 1 – http://192.168.1.21
- модуль ESP-01 № 2 – http://192.168.1.22
- модуль ESP-01 № 3 – http://192.168.1.23

- модуль ESP-01 № 4 – http://192.168.1.24
- модуль ESP-01 № 5 – http://192.168.1.25
- модуль ESP-01 № 6 – http://192.168.1.26
- модуль ESP-01 № 7 – http://192.168.1.27

API взаимодействия с модулем ESP-01 № 1:

- http://192.168.1.21/temp - запрос температуры. Ответ: HTML-заголовок + значение температуры в формате «24.00»

- http://192.168.1.21/hum - запрос влажности. Ответ: HTML-заголовок + значение влажности в формате «56.00»

- http://192.168.1.21/lux - запрос освещения. Ответ: HTML-заголовок + логическое значение интенсивности освещения (1 - достаточное, 0 - недостаточное)

API взаимодействия с модулем ESP-01 № 2:

- http://192.168.1.22/soil - запрос влажности почвы. Ответ: HTML-заголовок + логическое значение влажности почвы(0 - сухая, 1 - влажная)

- http://192.168.1.22/move - запрос движения. Ответ: HTML-заголовок + логическое значение наличия движения (0 - движения нет, 1 - движение есть)

API взаимодействия с модулем ESP-01 № 3:

-http://192.168.1.23?pin1=1–включить обогреватель

-http://192.168.1.23?pin1=0 - выключить обогреватель

-http://192.168.1.23?pin2=1 - включить кондиционер

-http://192.168.1.23?pin2=0 - выключить кондиционер

Ответ на все команды: HTML-заголовок + логическое значение состояния обогревателя и кондиционера (00, 01, 10 или 11).

API взаимодействия с модулем ESP-01 № 4:

-http://192.168.1.24?pin1=1–включить помпу полива

-http://192.168.1.24?pin1=0 - выключить помпу полива

-http://192.168.1.24?pin2=1 - включить освещение

-http://192.168.1.24?pin2=0 - выключить освещение

Ответ на все команды: HTML-заголовок + логическое значение состояния помпы и освещения (00, 01, 10 или 11).

API взаимодействия с модулем ESP-01 № 5:

-http://192.168.1.25?pin1=1–включить увлажнитель воздуха

-http://192.168.1.25?pin1=0 - выключить увлажнитель воздуха

-http://192.168.1.25?pin2=1 - включить вытяжную вентиляцию

-http://192.168.1.25?pin2=0 - выключить вытяжную вентиляцию

Ответ на все команды: HTML-заголовок + логическое значение состояния увлажнителя и вентиляции (00, 01, 10 или 11).

API взаимодействия с модулем ESP-01 № 6:

- http://192.168.1.26/water - запрос наличия влаги. Ответ: HTML-заголовок + логическое значение наличия влаги (0 - нет, 1 - есть)

- http://192.168.1.26/smoke - запрос наличия задымления. Ответ: HTML-заголовок + логическое значение наличия задымления(0 - нет, 1 - есть)

API взаимодействия с модулем ESP-01 № 7:

-http://192.168.1.27?pin1=1–включить увлажнитель воздуха

-http://192.168.1.27?pin1=0 - выключить увлажнитель воздуха

-http://192.168.1.27?pin2=1 - включить вытяжную вентиляцию

-http://192.168.1.27?pin2=0 - выключить вытяжную вентиляцию

Ответ на все команды: HTML-заголовок + логическое значение состояния увлажнителя и вентиляции (00, 01, 10 или 11).

2.5 Организация работы управляющего модуля.

В качестве управляющего модуля аппаратно-программного стенда использован одноплатный компьютер Raspberry Pi 3. Этот одноплатный компьютер разработан специально для обучения программированию, в состав операционной системы Raspbian входит объектно-ориентированный мультиплатформенный язык программирования Python.

Язык программирования Python является высокоуровневым языкомобщего назначения. Его особенности - повышение производительности разработчика и высокая читаемость кода. В основе языка лежит очень минималистичный синтаксис, что делает его простым для освоения начинающими разработчиками. При этом стандартная библиотека языка включает довольно большой объем различных функций, делающих язык по-настоящему мощным и универсальным.

Python является интерпретируемым языком. Это означает, что команды программы интерпретируются в машинный код непосредственно в ходе выполнения программы, а не заранее при компиляции[1]. Программист может даже писать программу построчно и тут же пускать на выполнение каждую строку отдельно по мере написания. Это дает некоторое удобство программирования для новичков, но вместе с тем приводит к общему замедлению времени выполнения программы.

В состав операционной системы Raspbian входит интегрированная среда разработки Python - IDLE. Эту среду удобно использовать для написания управляющих программ в рамках выполнения учащимися лабораторных работ.

По данным статистики язык Python используется в качестве первого языка программирования в большинстве американских колледжей, потеснив на этом месте язык Java. Python используется ведущими компаниями мирового масштаба (Google, Yahoo и Nasa и другие), что делает его привлекательным и для опытных программистов. В настоящее Python время является одним из восьми самых используемых языков программирования [3].

3. Функционал и логика работы стенда.

Стенд имитирует систему домашней автоматизации «умный дом». Функционал работы стенда:

1. Управление температурой воздуха. Регулярный опрос датчика температуры, при понижении температуры ниже минимально допустимой – включение обогревателя, при повышении выше максимально допустимой – включение кондиционера.

2. Управление влажностью воздуха. Регулярный опрос датчика влажности, при понижении влажности ниже минимально допустимой – включение увлажнителя воздуха, при повышении выше максимально допустимой – включение вытяжной вентиляции.

3. Управление влажностью почвы комнатных растений. Датчик влажности почвы находится в горшке с цветком. Регулярный опрос датчика влажности почвы, при получении логического показания 0 (почва сухая) – включение помпы полива на заданное время.

4. Управление освещенностью. Датчик освещенности (фоторезистор) расположен так, чтобы свет от ламп освещения на него не попадал, улавливает только солнечный свет. Регулярный опрос датчика освещенности, при получении логического показания 0 (освещенность низкая) – включение освещения, при получении логического показания 1 (освещенность высокая) – выключение освещения.

5. Автоматическое включение освещения при входе человека в помещение. Регулярный опрос датчика движения, при получении логического показания 1 (движение зафиксировано) – опрос датчика освещенности. Если получено логическое показание 0 (освещенность низкая) – включение освещения на заданное время. При повторных срабатываниях датчика движения время обновляется. По истечении заданного времени, если больше движения нет, свет выключаем.

6. Контроль наличия влаги. Используется для определения протечек в системе водоснабжения. Датчик может быть размещен в санузлах в районе водопроводных труб, возле стиральной или посудомоечной машины для своевременного определения протечки воды. Регулярный опрос датчика влаги, при получении логического показания 1 – перекрытие электрического клапана водоснабжения. Обратное включение клапана программой не предусмотрено, производится вручную только после проведения ремонта сантехнического оборудования и устранения причин протечки.

7. Контроль наличия задымления. Используется в качестве пожарной сигнализации. Регулярный опрос датчика задымления, при получении логического показания 1 (задымление) – включение системы пожаротушения на заданное время.

4. Пособие по использованию стенда.

В рамках бакалаврской работы разработано семь вариантов лабораторных работ для учащихся по созданию управляющих программ на языке Python, на основании описанного выше функционала. Так же разработана восьмая комплексная лабораторная работа, в которой все отдельные модули, играющие ту или иную роль объединены и управляются большой общей программой

Перед выполнение данных лабораторных работ рекомендуется к прочтению следующая литература: Марк Лутц - Изучаем Python (4-е издание); Марк Саммерфилд - Программирование на Python 3. Подробное руководство; Николай Прохоренок, Владимир Дронов - Python 3. Самое необходимое.

4.1Лабораторная работа № 1. Управление температурой воздуха.

Цель работы: выработать практические навыки программирования на языке Python, научиться создавать, выполнять и исправлять простейшие программы на языке Python в режиме диалога, познакомиться с диагностическими сообщениями среды разработки об ошибках.

Дополнительные сведения: в языке Python имеется несколько библиотек для выполнения HTTP-запросов. Одна из наиболее мощных и простых в использовании – библиотека Requests. Передача запросов с использованием библиотеки Requestspпроизводится командой requests.get("adress", params=p), где adress–http-адрес, по которому производится обращение, p –параметр get-запроса в формате словаря (необязательный).

Запрос температуры с датчика на стенде производится по адресу http://192.168.1.21/temp. Формат ответа: HTML-заголовок + значение температуры в формате «24.00».

Команда на включение/выключение обогревателя – обращение по адресу http://192.168.1.23 с параметром pin1=1 (включение) или pin1=0 (выключение).

Команда на включение/выключение кондиционера – обращение по адресу http://192.168.1.23 с параметром pin2=1 (включение) или pin2=0 (выключение).

Задание: организовать поддержание температуры воздуха в пределах заданного интервала.

Этапы решения задачи:

1. Запросить у пользователя значение минимально допустимой и максимально допустимой температуры;

2. Получить показания с датчика температуры;

3. Если значение температуры ниже минимального - включить обогреватель, если выше минимального – выключить обогреватель;

4. Если значение температуры выше максимального - включить кондиционер, если ниже максимального – выключить кондиционер;

5. Повторить операции 2-4 в бесконечном цикле с регулярностью 5 секунд.

Алгоритм работы (блок-схема) (рисунок 17):

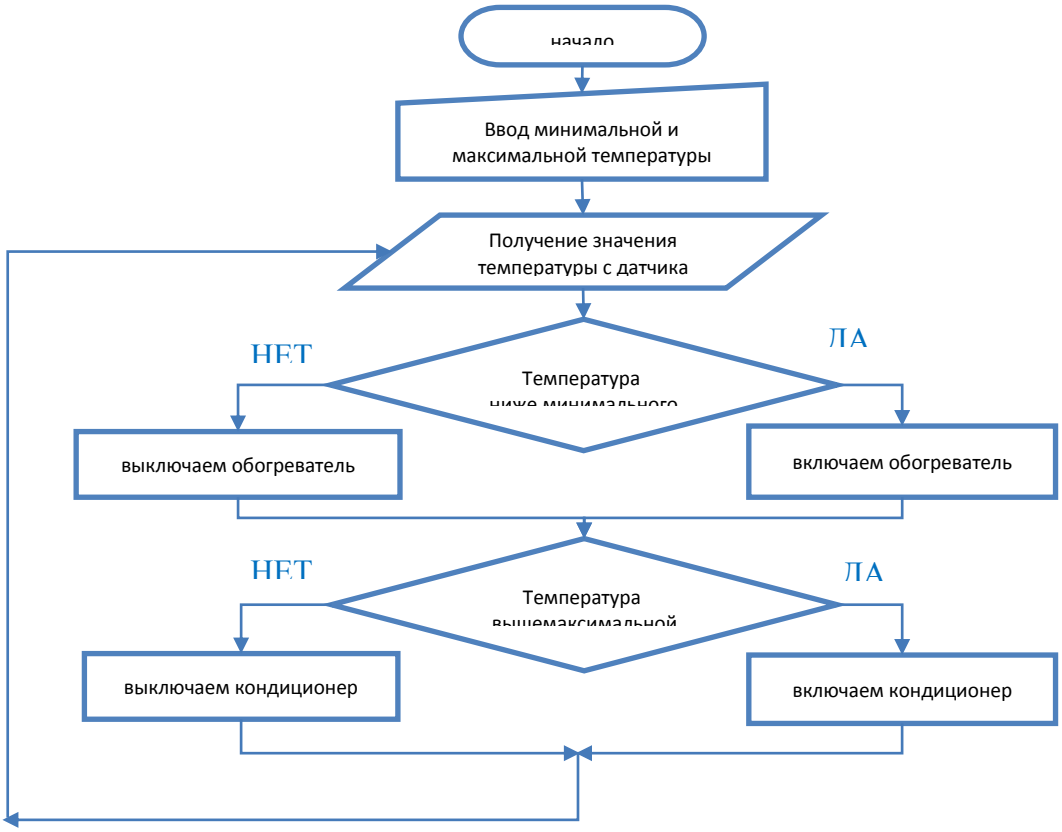

Рисунок 17 – Блок-схема программы лабораторной работы № 1.

Пример листинга программы на языке Python:

```
import requests                    # подключаем библиотеку для выполнения HTTP-запросов
import time                        # подключаем библиотеку для работы со временем
print ("Введите интервал температуры:")
sMin = input("минимум: ")          # пользователь вводит минимально допустимую температуру
sMax = input("максимум: ")         # пользователь вводит максимально допустимую температуру
tMin = int(sMin)                   # переводим текстовые переменные в числовой формат
tMax = int(sMax)
flag1 = 0                          # текущее состояние обогревателя
flag2 = 0                          # текущее состояние кондиционера
cykle = 1
while cykle == 1:                  # вечный цикл (переменная cykle всегда равна 1)
   r = requests.get("http://192.168.1.21/temp")      # запрашиваем температуру
s = r.text
   print(s)
i = len(s)                         # длина строки ответа
   s1 = s[i-5:i-3]                 # выделяем из строки ответа 3-4 символы с конца (значение температуры)
   temp = int(s1)                  # переводим строку символов в числовой формат
   print(temp)
   if temp < tMin:                 # если температура ниже минимума
      if flag1 == 0:               # если обогреватель выключен
         print("Температура ниже минимума, включаем обогреватель")
         p = {"pin1": "1"}
         flag1 = 1
         r = requests.get("http://192.168.1.23", params=p)# команда на включение обогревателя
print(r.url)
         print(r.text)             # печатаемответ
else:                              # иначе - если температура выше минимума
      if flag1 == 1:               # если обогреватель включен
```

```
            print("Выключаем обогреватель")
            flag1 = 0
            p = {"pin1": "0"}
            r = requests.get("http://192.168.1.23", params=p)# команда на выключение обогревателя
print(r.url)
            print(r.text)                    # печатаемответ
if temp > tMax:                 # если температура выше максимума
        if flag2 == 0:                  # если кондиционер выключен
            print("Температура выше максимума, включаем кондиционер")
            p = {"pin2": "1"}
            flag2 = 1
            r = requests.get("http://192.168.1.23", params=p)# команда на включение кондиционера
print(r.url)
            print(r.text)                    # печатаемответ
else:                           # иначе - если температура ниже максимума
        if flag2 == 1:                  # если кондиционер включен
            print("Выключаем кондиционер")
            flag2 = 0
            p = {"pin2": "0"}
            r = requests.get("http://192.168.1.23", params=p)# команда на выключение кондиционера
print(r.url)
            print(r.text)                    # печатаемответ
time.sleep(5)                   # задержка 5 секунд перед повторением цикла
```

Другие варианты выполнения задания:

- запросить у пользователя частоту опроса датчика (в секундах), выполнять главный цикл программы с заданной частотой;

- каждый раз при включении/выключении обогревателя или кондиционера анализировать ответ от модуля ESP8266–прошла ли команда. Если команда не прошла, делать повторную попытку;

- пользователь задает не порог, а интервал температуры для работы обогревателя и кондиционера. Например: обогреватель включается при температуре +18, выключается при температуре +22, кондиционер включается при температуре +32, выключается при температуре +28;

- кондиционер используется для охлаждения только в дневное время суток. В ночное время используется вытяжная вентиляция (ночью на улице прохладно, вытяжная вентиляция позволит эффективно охладить помещение и уменьшит расход электроэнергии). Определение ночного времени суток можно выполнять двумя способами: либо запрашивая текущее время управляющего модуля, либо запрашивая освещенность с датчика освещения (ночью темно).

4.2Лабораторная работа № 2. Управление влажностью воздуха.

Цель работы: выработать практические навыки программирования на языке Python, научиться создавать, выполнять и исправлять простейшие программы на языке Python в режиме диалога, познакомиться с диагностическими сообщениями среды разработки об ошибках.

Дополнительные сведения: в языке Python имеется несколько библиотек выполнения HTTP-запросов. Одна из наиболее мощных и простых в использовании – библиотека Requests. Передача запросов с использованием библиотеки Requests производится командой requests.get("adress", params=p), где adress–http-адрес, по которому производится обращение, p –параметр get-запроса в формате словаря (необязательный).

Запрос влажности с датчика на стенде производится по адресу http://192.168.1.21/hum Формат ответа: HTML-заголовок + значение влажности в формате «58.00».

Команда на включение/выключение увлажнителя воздуха – обращение по адресу http://192.168.1.25 с параметром pin1=1 (включение) или pin1=0 (выключение).

Команда на включение/выключение вытяжной вентиляции – обращение по адресу http://192.168.1.25 с параметром pin2=1 (включение) или pin2=0 (выключение).

Задание: организовать поддержание влажности воздуха в помещении в пределах заданного интервала.

Этапы решения задачи:

1. Запросить у пользователя значение минимально допустимой и максимально допустимой влажности;

2. Получить показания с датчика влажности воздуха;

3. Если значение влажности воздуха ниже минимального - включить увлажнитель воздуха, если выше минимального – выключить увлажнитель воздуха;

4. Если значение влажности воздуха выше максимального - включить вытяжную вентиляцию, если ниже максимального – выключить вытяжную вентиляцию;

5. Повторить операции 2-4 в бесконечном цикле с регулярностью 5 секунд.

Алгоритм работы (блок-схема) (рисунок 18):

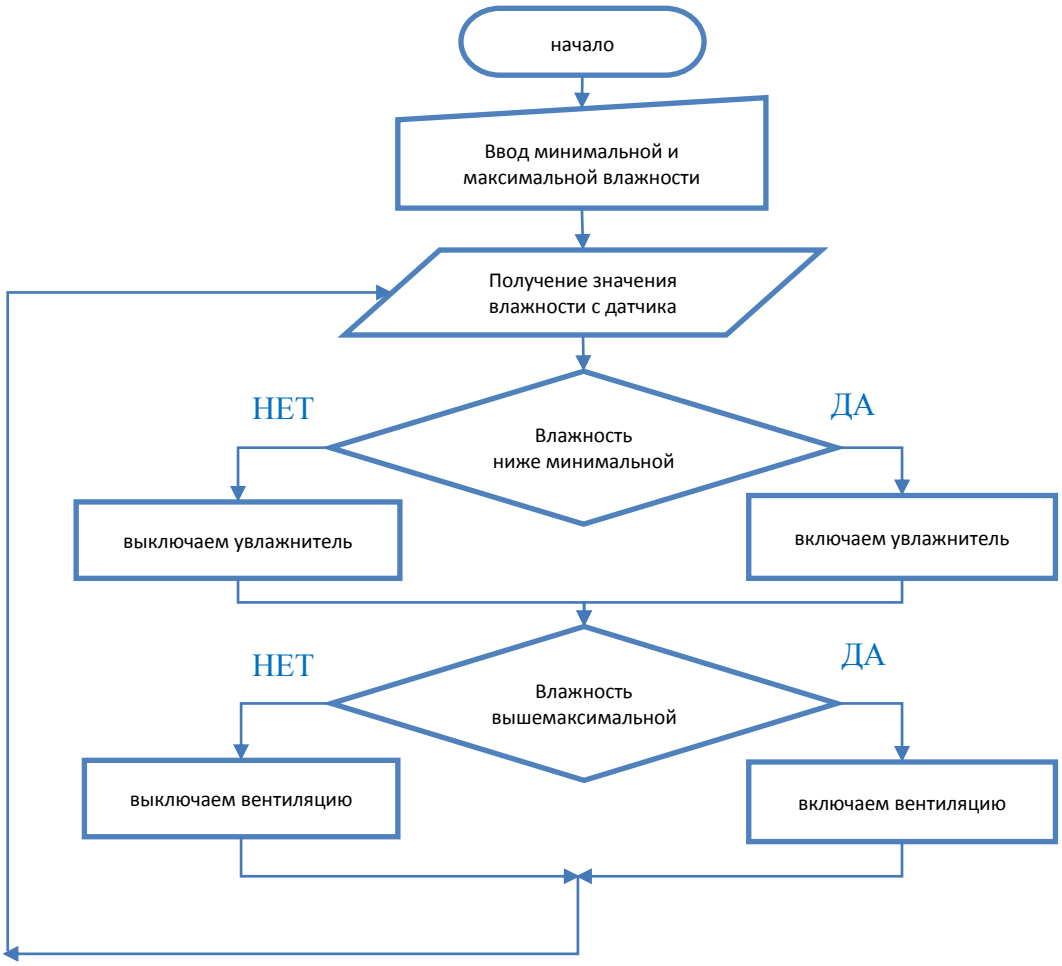

Рисунок 18 – Блок-схема программы лабораторной работы № 2.

Пример листинга программы на языке Python:

```
import requests                    # подключаем библиотеку для выполнения HTTP-запросов
import time                        # подключаем библиотеку для работы со временем
print ("Введите интервал влажности:")
sMin = input("минимум: ")          # пользователь вводит минимально допустимую влажность
```

```
sMax = input("максимум: ")                    # пользователь вводит максимально допустимую влажность
humMin = int(sMin)                            # переводим текстовые переменные в числовой формат
humMax = int(sMax)
flag1 = 0                                     # текущее состояние увлажнителя
flag2 = 0                                     # текущее состояние вытяжной вентиляции
cykle = 1
while cykle == 1:                             # вечный цикл (переменная cykle всегда равна 1)
    r = requests.get("http://192.168.1.21/hum")       # запрашиваем влажность
s = r.text
    print(s)
i = len(s)                                    # длина строки ответа
    s1 = s[i-5:i-3]                           # выделяем из строки ответа 3-4 символы с конца (значение влажности)
    hum = int(s1)                             # переводим строку символов в числовой формат
    print(hum)
    if hum < humMin:                          # если влажность ниже минимума
        if flag1 == 0:                        # если увлажнитель выключен
            print("Влажность ниже минимума, включаем увлажнитель воздуха")
            p = {"pin1": "1"}
            flag1 = 1
            r = requests.get("http://192.168.1.25", params=p)        # команда на включение увлажнителя
print(r.url)
            print(r.text)                     # печатаемответ
else:                                         # иначе - если влажность выше минимума
        if flag1 == 1:                        # если увлажнитель включен
            print("Выключаем увлажнитель воздуха")
            flag1 = 0
            p = {"pin1": "0"}
```

```
        r = requests.get("http://192.168.1.25", params=p)        # команда на выключение увлажнителя
print(r.url)
        print(r.text)                                    # печатаемответ
   ifhum>humMax:                                         # если влажность выше максимума
ifflag2 == 0:                                       # если вентиляция выключена
        print("Влажность выше максимума, включаем вытяжную вентиляцию")
        p = {"pin2": "1"}
        flag2 = 1
        r = requests.get("http://192.168.1.25", params=p)        # команда на включение вентиляции
print(r.url)
        print(r.text)                                    # печатаемответ
else:                                                   # иначе - если влажность ниже максимума
      if flag2 == 1:                                    # если вентиляция включена
        print("Выключаем вытяжную вентиляцию")
        flag2 = 0
        p = {"pin2": "0"}
        r = requests.get("http://192.168.1.25", params=p)# команда на вытяжной вентиляции
print(r.url)
        print(r.text)                                    # печатаемответ
   time.sleep(5)                                        # задержка 5 секунд перед повторением цикла
```

Другие варианты выполнения задания:

- запросить у пользователя частоту опроса датчика (в секундах), выполнять главный цикл программы с заданной частотой;

- каждый раз при включении/выключении увлажнителя или вентиляции анализировать ответ от модуля ESP8266–прошла ли команда. Если команда не прошла, делать повторную попытку;

- пользователь задает не порог, а интервал температуры для работы увлажнителя и вентиляции. Например: увлажнитель включается при влажности

30, выключается при влажности 45, вентиляция включается при влажности 80, выключается при влажности 65;

- вентиляция используется для уменьшения влажности только в ночное время суток. В дневное время используется кондиционер (днем на улице слишком жарко, вытяжная вентиляция не сможет эффективно понизить влажность и приведет к возрастанию температуры в помещении). Определение дневного времени суток можно выполнять двумя способами: либо запрашивая текущее время управляющего модуля, либо запрашивая освещенность с датчика освещения (днем светло).

4.3Лабораторная работа № 3. Управление влажностью почвы комнатных растений.

Цель работы: выработать практические навыки программирования на языке Python, научиться создавать, выполнять и исправлять простейшие программы на языке Python в режиме диалога, познакомиться с диагностическими сообщениями среды разработки об ошибках.

Дополнительные сведения: в языке Python имеется несколько библиотек выполнения HTTP-запросов. Одна из наиболее мощных и простых в использовании – библиотека Requests. Передача запросов с использованием библиотеки Requestspроизводится командой requests.get("adress", params=p), где adress–http-адрес, по которому производится обращение, p –параметр get-запроса в формате словаря (необязательный).

Запрос влажности почвы с датчика на стенде производится по адресу http://192.168.1.22/soil. Формат ответа: HTML-заголовок + логическое значение влажности почвы(0 - сухая, 1 - влажная),

Команда на включение/выключение помпы полива – обращение по адресу http://192.168.1.24 с параметром pin1=1 (включение) или pin1=0 (выключение).

Задание: обеспечить своевременный полив растений в помещении.

Этапы решения задачи:

1. Запросить у пользователя продолжительность работы помпы при поливе;

2. Получить показания с датчика влажности почвы;

3. Если получено логическое значение влажности почвы0 (сухая) – включить помпу полива на заданное время;

4. По истечении заданного времени выключить помпу полива;

5. Повторить операции 2-4 в бесконечном цикле с регулярностью 5 секунд.

Алгоритм работы (блок-схема) (рисунок 19):

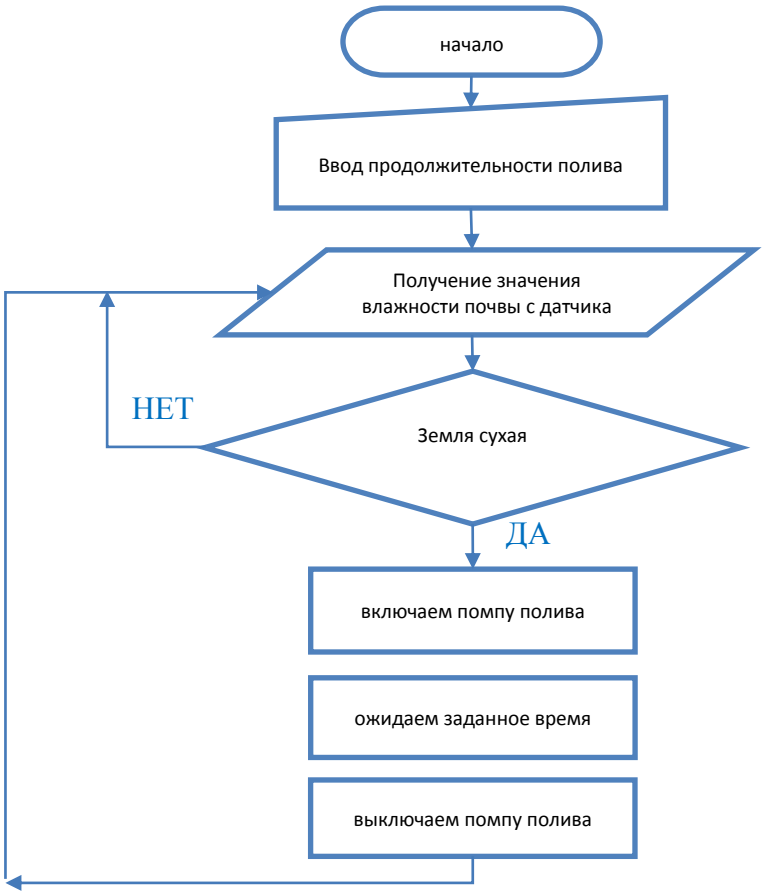

Рисунок 19 – Блок-схема программы лабораторной работы № 3.

Пример листинга программы на языке Python:

```
import requests                    # подключаем библиотеку для выполнения HTTP-запросов
import time                        # подключаем библиотеку для работы со временем
s =input ("Введите продолжительность работы помпы полива:")     # пользователь вводит продолжительность работы помпы при поливе
t = int(s)                         # переводим текстовую переменную в числовой формат
cykle = 1
while cykle == 1:                  # вечный цикл (переменная cykle всегда равна 1)
    r = requests.get("http://192.168.1.22/soil")   # запрашиваем влажность почвы
    s = r.text
    print(s)
    i = len(s)                     # длина строки ответа
    s1 = s[i-1]    # выделяем из строки ответа последний символ ( логическое значение влажности)
    soil = int(s1)                 # переводим строку символов в числовой формат
    if soil == 0:                  # если значение = 0 (земля сухая)
        print("Земля сухая, включаем полив")
        p = {"pin1": "1"}
        r = requests.get("http://192.168.1.24", params=p)      # команда на включение помпы полива
        print(r.url)
        print(r.text)              # печатаемответ
        time.sleep(t)              # пауза длительностью равна заданному времени полива
        p = {"pin1": "0"}
        r = requests.get("http://192.168.1.24", params=p)  # команда на выключение помпы полива
        print(r.url)
        print(r.text)                  # печатаемответ
    else:
        print("Земля достаточно влажная")
    time.sleep(5)                  # задержка 5 секунд перед повторением цикла
```

Другие варианты выполнения задания:

- запросить у пользователя частоту опроса датчика (в секундах), выполнять главный цикл программы с заданной частотой;

- каждый раз при включении/выключении помпы полива анализировать ответ от модуля ESP8266–прошла ли команда. Если команда не прошла, делать повторную попытку;

- для того, чтобы вода успевала правильно впитываться в землю, организовать импульсный полив – на 5 секунд включаем помпу, на 10 секунд выключаем, повторяем цикл 3 раза;

- организовать полив по часам – например, в 9-00 и в 21-00, независимо от показаний датчика влажности. Текущее время запрашивать с управляющего модуля. При этом показания датчика влажности все равно контролируем (аварийный режим).

4.4Лабораторная работа № 4. Управление освещенностью.

Цель работы: выработать практические навыки программирования на языке Python, научиться создавать, выполнять и исправлять простейшие программы на языке Python в режиме диалога, познакомиться с диагностическими сообщениями среды разработки об ошибках.

Дополнительные сведения: в языке Python имеется несколько библиотек выполнения HTTP-запросов. Одна из наиболее мощных и простых в использовании – библиотека Requests. Передача запросов с использованием библиотеки Requests производится командой requests.get("adress", params=p), где adress–http-адрес, по которому производится обращение, p –параметр get-запроса в формате словаря (необязательный).

Запрос освещенности с датчика на стенде производится по адресу http://192.168.1.21/lux. Формат ответа: HTML-заголовок + логическое значение освещенности(1 - достаточная, 0 - недостаточная).

Команда на включение/выключение освещения – обращение по адресу http://192.168.1.24 с параметром pin2=1 (включение) или pin2=0 (выключение).

Задание: организовать освещение в помещении в темное время суток.

Этапы решения задачи:

1. Получить показания с датчика освещенности;

2. Если получено логическое значение освещенности0 (недостаточная) – включить освещение. Если получено логическое значение освещенности1(достаточная) – включить освещение;

3. Повторить операции 1-2 в бесконечном цикле с регулярностью 5 секунд.

Алгоритм работы (блок-схема) (рисунок 20):

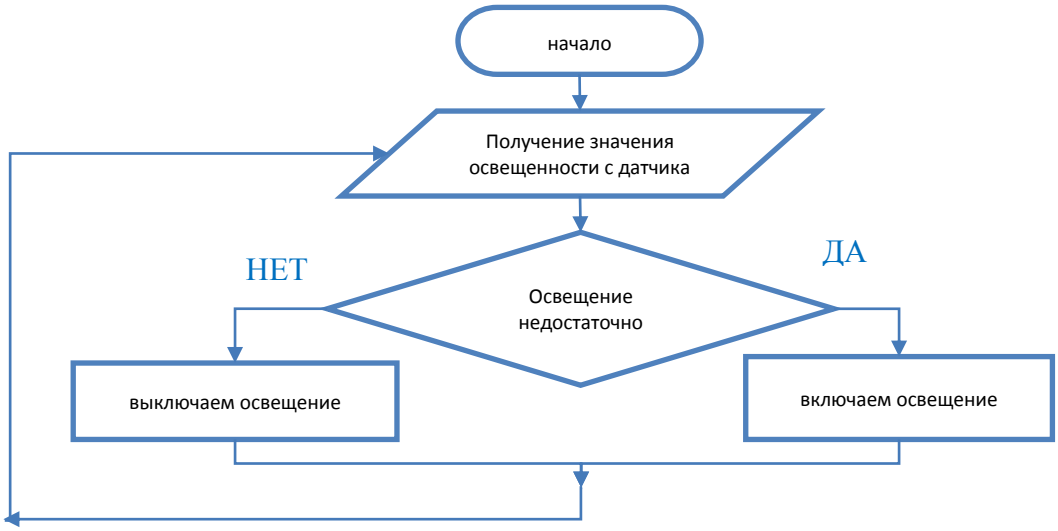

Рисунок 20 – Блок-схема программы лабораторной работы № 4.

Пример листинга программы на языке Python:

```
import requests                    # подключаем библиотеку для выполнения HTTP-запросов
import time                        # подключаем библиотеку для работы со временем
flag1 = 0                          # текущее состояние освещения
```

```
cykle = 1
while cykle == 1:                    # вечный цикл (переменная cykle всегда равна 1)
    r = requests.get("http://192.168.1.21/lux")# запрашиваем освещенность
s = r.text
    print(s)
i = len(s)                           # длина строки ответа
    s1 = s[i-1]          # выделяем из строки ответа последний символ (логическое значение освещенности)
    lux = int(s1)                    # переводим строку символов в числовой формат
    if lux == 0:                     # если значение = 0 (освещение недостаточное)
        if flag1 == 0:               # если лампа выключена
            print("Освещение недостаточно, включаем лампу")
            p = {"pin2": "1"}
            r = requests.get("http://192.168.1.24", params=p)      # команда на включение лампы
print(r.url)
            print(r.text)                        # печатаемответ
else:
        if flag1 == 1:               # если лампа включена
            print("Освещение достаточно, выключаем лампу")
            p = {"pin2": "0"}
            r = requests.get("http://192.168.1.24", params=p)      # команда на выключение лампы
print(r.url)
            print(r.text)                        # печатаемответ
time.sleep(5)                        # задержка 5 секунд перед повторением цикла
```

Другие варианты выполнения задания:

- запросить у пользователя частоту опроса датчика (в секундах), выполнять главный цикл программы с заданной частотой;

- каждый раз при включении/выключении освещения анализировать ответ от модуля ESP8266–прошла ли команда. Если команда не прошла, делать повторную попытку;

- организовать освещение по часам – например, с 22-00 по 8-00, независимо от показаний датчика освещенности. Текущее время запрашивать с управляющего модуля. В остальное время освещение по датчику освещенности;

- не освещать растения во время ночного цикла – например, с 00-00 по 6-00. Текущее время запрашивать с управляющего модуля. В остальное время освещение по датчику освещенности.

4.5Лабораторная работа № 5. Автоматическое включение освещения при входе человека в помещение.

Цель работы: выработать практические навыки программирования на языке Python, научиться создавать, выполнять и исправлять простейшие программы на языке Python в режиме диалога, познакомиться с диагностическими сообщениями среды разработки об ошибках.

Дополнительные сведения: в языке Python имеется несколько библиотек выполнения HTTP-запросов. Одна из наиболее мощных и простых в использовании – библиотека Requests. Передача запросов с использованием библиотеки Requests производится командой requests.get("adress", params=p), где adress–http-адрес, по которому производится обращение, p –параметр get-запроса в формате словаря (необязательный).

Запрос наличия движения с датчика на стенде производится по адресу http://192.168.1.22/move. Формат ответа: HTML-заголовок + логическое значение наличия движения(1–движение есть, 0–движения нет).

Команда на включение/выключение освещения – обращение по адресу http://192.168.1.24 с параметром pin2=1 (включение) или pin2=0 (выключение).

Задание: организовать автоматическое включение освещения при входе человека в помещение.

Этапы решения задачи:

1. Получить показания с датчика движения;

2. Если получено логическое значение 1(движение зафиксировано) – запросить показания с датчика освещенности;

2.1. Если получено логическое значение освещенности0 (недостаточная) – включить лампу;

3. Если получено логическое значение 0(движения нет) – проверить, включена ли лампа;

3.1. Если лампа включена – выключить лампу;

4. Повторить операции 1-3 в бесконечном цикле с регулярностью 5 секунд.

Алгоритм работы (блок-схема) (рисунок 21):

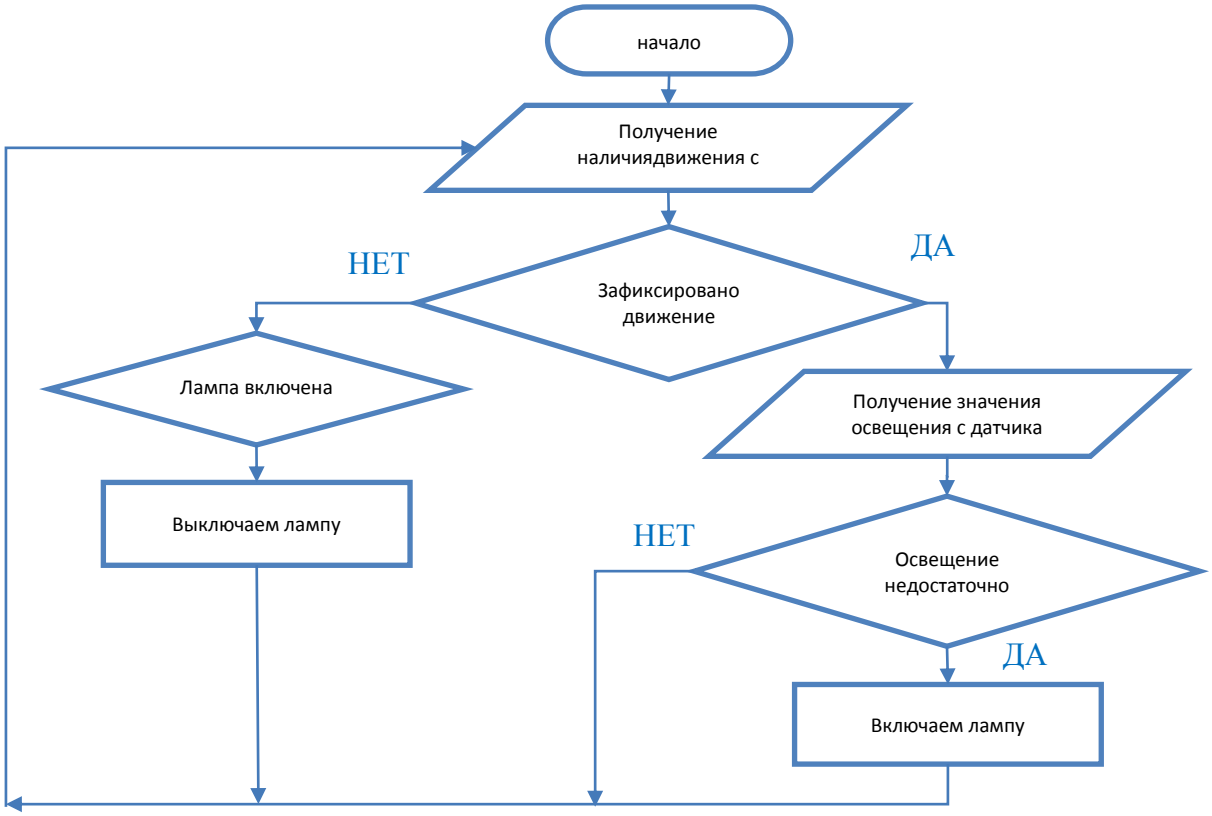

Рисунок 21 – Блок-схема программы лабораторной работы № 5.

Пример листинга программы на языке Python:

```
import requests                    # подключаем библиотеку для выполнения HTTP-запросов
import time                        # подключаем библиотеку для работы со временем
light = 0                          # текущее состояние освещения
cykle = 1
while cykle == 1:                  # вечный цикл (переменная cykle всегда равна 1)
   r = requests.get("http://192.168.1.22/move")      # запрашиваем наличие движения
s = r.text
   print(s)
i = len(s)                         # длина строки ответа
   s1 = s[i-1]                     # выделяем из строки ответа последний символ (логическое значение)
   move = int(s1)                  # переводим строку символов в числовой формат
   if move == 1:                   # если значение = 1 (зафиксировано движение)
      print("Зафиксировано движение")
      r = requests.get("http://192.168.1.21/lux")      # запрашиваем освещенность
s = r.text
      print(s)
i = len(s)                         # запрашиваем освещенность
      s1 = s[i-1]     # выделяем из строки ответа последний символ (логическое значение освещенности)
      lux = int(s1)                # переводим строку символов в числовой формат
      if lux == 0:                 # если значение = 0 (освещение недостаточное)
         print("В помещении темно, включаем свет")
         p = {"pin2": "1"}
         r = requests.get("http://192.168.1.24", params=p)# команда на включение лампы
print(r.url)
         print(r.text)             # печатаемответ
         light = 1
         time.sleep(30)            # пауза 30 секунд
   else:                           # иначе - движения нет
```

```
        print("Движение отсутствует")
        if light == 1:                                  # если лампа включена
            print("В помещении никого нет, выключаем свет")
            p = {"pin2": "0"}
            r = requests.get("http://192.168.1.24", params=p)          # команда на
выключение лампы
print(r.url)
            print(r.text)                               # печатаемответ
            light = 0
    time.sleep(5)                               # задержка 5 секунд перед повторением
цикла
```

Другие варианты выполнения задания:

- запросить у пользователя частоту опроса датчика (в секундах), выполнять главный цикл программы с заданной частотой;

- каждый раз при включении/выключении освещения анализировать ответ от модуля ESP8266–прошла ли команда. Если команда не прошла, делать повторную попытку;

- организовать режим сигнализации несанкционированного проникновения. Например, с 00-00 по 6-00 в помещении никого быть не должно. Если зафиксировано движени – значит, проник грабитель. Включаем освещение в режим стробоскопа для привлечения внимания охраны и отпугивания грабителя (режим стробоскопа – включение/выключение освещения с большой частотой). Текущее время запрашивать с управляющего модуля. В остальное время нормальный режим работы.

4.6 Лабораторная работа № 6. Контроль протечек воды.

Цель работы: выработать практические навыки программирования на языке Python, научиться создавать, выполнять и исправлять простейшие программы на языке Python в режиме диалога, познакомиться с диагностическими сообщениями среды разработки об ошибках.

Дополнительные сведения: в языке Python имеется несколько библиотек выполнения HTTP-запросов. Одна из наиболее мощных и простых в использовании – библиотека Requests. Передача запросов с использованием библиотеки Requestspпроизводится командой requests.get("adress", params=p), где adress–http-адрес, по которому производится обращение, p –параметр get-запроса в формате словаря (необязательный).

Запрос датчика влаги на стенде производится по адресу http://192.168.1.26/waterФормат ответа: HTML-заголовок + логическое значение наличия влаги(0 - сухо, 1 - влажно),

Команда на перекрытие электрического клапана системы водоснабжения – обращение по адресу http://192.168.1.27 с параметром pin1=1.

Задание: обеспечить контроль протечек воды.

Этапы решения задачи:

1. Получить показания с датчика влаги;

2. Если получено логическое значение 0 (наличие влаги) – перекрыть электрический клапан системы водоснабжения;

3. Обратное включение клапана программой не предусмотрено, производится вручную только после проведения ремонта сантехнического оборудования и устранения причин протечки;

Алгоритм работы (блок-схема) (рисунок 22):

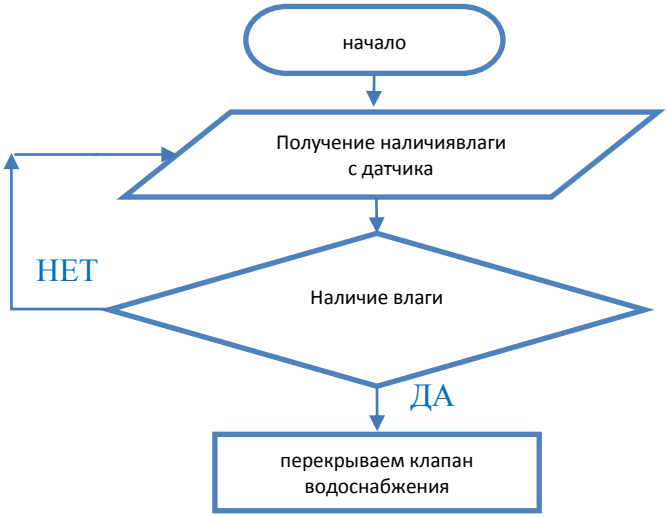

Рисунок 22 – Блок-схема программы лабораторной работы № 6.

Пример листинга программы на языке Python:

```
import requests                    # подключаем библиотеку для выполнения HTTP-запрос
import time                        # подключаем библиотеку для работы со временем
cykle = 1
while cykle == 1:                  # вечный цикл (переменная cykle всегда равна 1)
    r = requests.get("http://192.168.1.26/water")     # запрашиваем наличие влаги
s = r.text
    print(s)
i = len(s)                         # длина строки ответа
    s1 = s[i-1]                    # выделяем из строки ответа последний символ ( логическое значение влажности)
    water = int(s1)                # переводим строку символов в числовой формат
    if water == 1:                 # если значение = 1 (обнаружена влага)
        print("Обнаружена протечка воды")
        p = {"pin1": "1"}
        r = requests.get("http://192.168.1.27", params=p)      # команда на перекрытие аварийного клапана системы водоснабжения
print(r.url)
        print(r.text)              # печатаемответ
else:
        print("Протечек не обнаружено")
    time.sleep(5)                  # задержка 5 секунд перед повторением цикла
```

Другие варианты выполнения задания:

- запросить у пользователя частоту опроса датчика (в секундах), выполнять главный цикл программы с заданной частотой;

- при перекрытии клапанаводоснабжения анализировать ответ от модуля ESP8266–прошла ли команда. Если команда не прошла, делать повторную попытку;

4.7 Лабораторная работа № 7. Контроль задымления в помещении.

Цель работы: выработать практические навыки программирования на языке Python, научиться создавать, выполнять и исправлять простейшие программы на языке Python в режиме диалога, познакомиться с диагностическими сообщениями среды разработки об ошибках.

Дополнительные сведения: в языке Python имеется несколько библиотек выполнения HTTP-запросов. Одна из наиболее мощных и простых в использовании – библиотека Requests. Передача запросов с использованием библиотеки Requestspпроизводится командой requests.get("adress", params=p), где adress–http-адрес, по которому производится обращение, p –параметр get-запроса в формате словаря (необязательный).

Запрос датчика задымления на стенде производится по адресу http://192.168.1.26/smokeФормат ответа: HTML-заголовок + логическое значение наличия задымления(0 - нет, 1 - есть),

Команда на включение/выключение системы пожаротушения – обращение по адресу http://192.168.1.27с параметром pin2=1 (включение) или pin2=0 (выключение).

*Задание:*организовать пожарную сигнализацию и автоматическое пожаротушение в помещении.

Этапы решения задачи:

1. Запросить у пользователя продолжительность работы системы пожаротушения при тушении пожара;

2. Получить показания с датчика дыма;

3. Если получено логическое 1(задымление) – включить систему пожаротушения на заданное время;

4. По истечении заданного времени выключить систему пожаротушения;

5. Повторить операции 2-4 в бесконечном цикле с регулярностью 5 секунд.

Алгоритм работы (блок-схема) (рисунок 23):

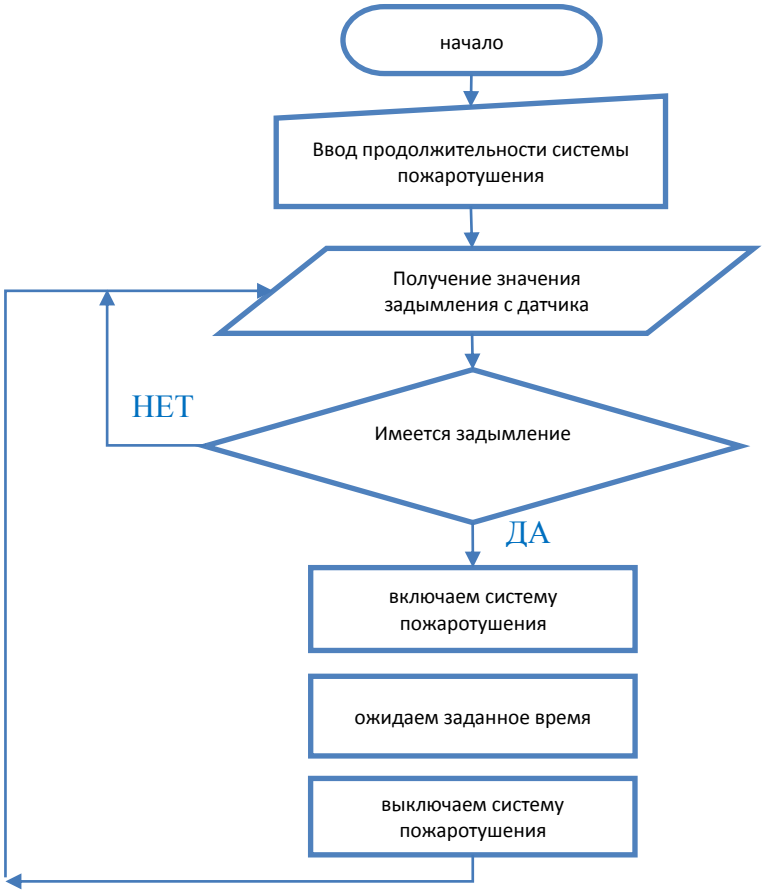

Рисунок 23 – Блок-схема программы лабораторной работы № 7.

Пример листинга программы на языке Python:

```
import requests                                    # подключаем библиотеку для выполнения HTTP-запрос
import time                                        # подключаем библиотеку для работы со временем
s =input ("Введите продолжительность работы системы пожаротушения в случае обнаружения дыма:")
t = int(s)                                         # переводим текстовую переменную в числовой формат
cykle = 1
while cykle == 1:                                  # вечный цикл (переменная cykle всегда равна 1)
    r = requests.get("http://192.168.1.26/smoke")  # запрашиваем наличие дыма
s = r.text
```

```
    print(s)
i = len(s)                                  # длина строки ответа
    s1 = s[i-1]             # выделяем из строки ответа последний символ (логическое значение влажности)
    smoke = int(s1)                         # переводим строку символов в числовой формат
    if smoke == 1:                          # если значение = 1 (обнаружено задымление)
        print("Обнаружено задымление")
        p = {"pin2": "1"}
        r = requests.get("http://192.168.1.27", params=p)       # команда на включение системы пожаротушения
print(r.url)
        print(r.text)                       # печатаемответ
time.sleep(t)                               # пауза длительностью равна заданному времени тушения
        p = {"pin2": "0"}
        r = requests.get("http://192.168.1.27", params=p)       # команда на отключение системы пожаротушения
print(r.url)
        print(r.text)                       # печатаемответ
    else:
        print("Задымления не обнаружено")
    time.sleep(5)                           # задержка 5 секунд перед повторением цикла
```

Другие варианты выполнения задания:

- запросить у пользователя частоту опроса датчика (в секундах), выполнять главный цикл программы с заданной частотой;

- каждый раз при включении/выключении системы пожаротушения анализировать ответ от модуля ESP8266–прошла ли команда. Если команда не прошла, делать повторную попытку;

- для более эффективного тушения пожара организовать импульснуюработу системы пожаротушения – на 5 секунд включаем, на 5 секунд выключаем, повторяем цикл до истечения заданного времени работы;

- при обнаружении задымления кроме включения пожаротушения принудительно отключать прочие электроприборы (освещение, нагреватель и т.д.) для исключения замыкания, а также отключать вентиляцию вентиляцию для уменьшения притока кислорода к очагу пожара.

4.8 Комплексная работа. Управление работой системы «умный дом».

Цель работы: выработать практические навыки программирования на языке Python, научиться создавать, выполнять и исправлять простейшие программы на языке Python в режиме диалога, познакомиться с диагностическими сообщениями среды разработки об ошибках.

Дополнительные сведения:

1. В языке Python имеется встроенная графическая библиотекаTkinter, позволяющая создавать программы с оконным интерфейсом.

2. В языке Python имеется несколько библиотек выполнения HTTP-запросов. Одна из наиболее мощных и простых в использовании – библиотека Requests. Передача запросов с использованием библиотеки Requests производится командой requests.get("adress", params=p), где adress–http-адрес, по которому производится обращение, p –параметр get-запроса в формате словаря (необязательный).

Запрос температуры с датчика на стенде производится по адресу http://192.168.1.21/temp. Формат ответа: HTML-заголовок + значение температуры в формате «24.00».

Команда на включение/выключение обогревателя – обращение по адресу http://192.168.1.23 с параметром pin1=1 (включение) или pin1=0 (выключение).

Команда на включение/выключение кондиционера – обращение по адресу http://192.168.1.23 с параметром pin2=1 (включение) или pin2=0 (выключение).

Запрос влажности с датчика на стенде производится по адресу http://192.168.1.21/hum Формат ответа: HTML-заголовок + значение влажности в формате «58.00».

Команда на включение/выключение увлажнителя воздуха – обращение по адресу http://192.168.1.25 с параметром pin1=1 (включение) или pin1=0 (выключение).

Команда на включение/выключение вытяжной вентиляции – обращение по адресу http://192.168.1.25 с параметром pin2=1 (включение) или pin2=0 (выключение).

Запрос влажности почвы с датчика на стенде производится по адресу http://192.168.1.22/soil. Формат ответа: HTML-заголовок + логическое значение влажности почвы(0 - сухая, 1 - влажная).

Команда на включение/выключение помпы полива – обращение по адресу http://192.168.1.24 с параметром pin1=1 (включение) или pin1=0 (выключение).

Запрос освещенности с датчика на стенде производится по адресу http://192.168.1.21/lux. Формат ответа: HTML-заголовок + логическое значение освещенности(1 - достаточная, 0 - недостаточная).

Команда на включение/выключение освещения – обращение по адресу http://192.168.1.24 с параметром pin2=1 (включение) или pin2=0 (выключение).

Запрос наличия движения с датчика на стенде производится по адресу http://192.168.1.22/move. Формат ответа: HTML-заголовок + логическое значение наличия движения(1–движение есть, 0–движения нет).

Команда на включение/выключение освещения – обращение по адресу http://192.168.1.24 с параметром pin2=1 (включение) или pin2=0 (выключение).

Запрос датчика влаги на стенде производится по адресу http://192.168.1.26/waterФормат ответа: HTML-заголовок + логическое значение наличия влаги(0 - сухо, 1 - влажно).

Команда на перекрытие электрического клапана системы водоснабжения – обращение по адресу http://192.168.1.27 с параметром pin1=1.

Запрос датчика задымления на стенде производится по адресу http://192.168.1.26/smokeФормат ответа: HTML-заголовок + логическое значение наличия задымления(0 - нет, 1 - есть).

Команда на включение/выключение системы пожаротушения – обращение по адресу http://192.168.1.27 с параметром pin2=1 (включение) или pin2=0 (выключение).

Задание: организовать поддержание температуры и влажности воздуха в помещении в пределах заданного интервала, обеспечить своевременный полив комнатных растений, организовать освещение в в темное время суток, а также автоматическое включение освещения при входе человека в помещение, обеспечить контроль протечек воды с перекрытием клапана водоснабжения в случае протечки, обеспечить пожарную сигнализацию и автоматическое пожаротушение в помещении.

Этапы решения задачи:

1. Значения минимально допустимой и максимально допустимой температуры и влажности, продолжительность работы насоса при поливе и продолжительности работы системы пожаротушения при пожаре задаются пользователем с помощью полей ввода данных в главном окне программы.

2. После запуска стенда провести обработку показаний датчиков температуры, влажности воздуха, влажности почвы, наличия влагии задымления, используя функции, рассмотренные в лабораторных работах № 1, 2, 3, 6, 7.

3. В зависимости от установок переключателя «освещение» провести обработку показаний датчика уровня освещенности или датчика наличия движения, используя функции, рассмотренные в лабораторных работах № 4 или 5.

4. Повторить операции 2-3 в бесконечном цикле с регулярностью 5 секунд.

5. Обеспечить отображение в главном окне программы текущих значений температуры и влажности воздуха, влажности почвы, уровня освещенности, наличия движения, влаги и дыма. Также обеспечить в главном окне программы текущего состояния подключенного оборудования (обогреватель, кондиционер, увлажнитель воздуха, вытяжная вентиляция, освещение, насос полива, клапан водоснабжения, система пожаротушения) – включено или выключено (рисунки 24, 25)

6. Реализовать функцию обновления настоек системы после изменения их пользователем (кнопка «Обновить»), функцию запуска/остановки работы стенда (кнопка «Пуск»/»Стоп»)

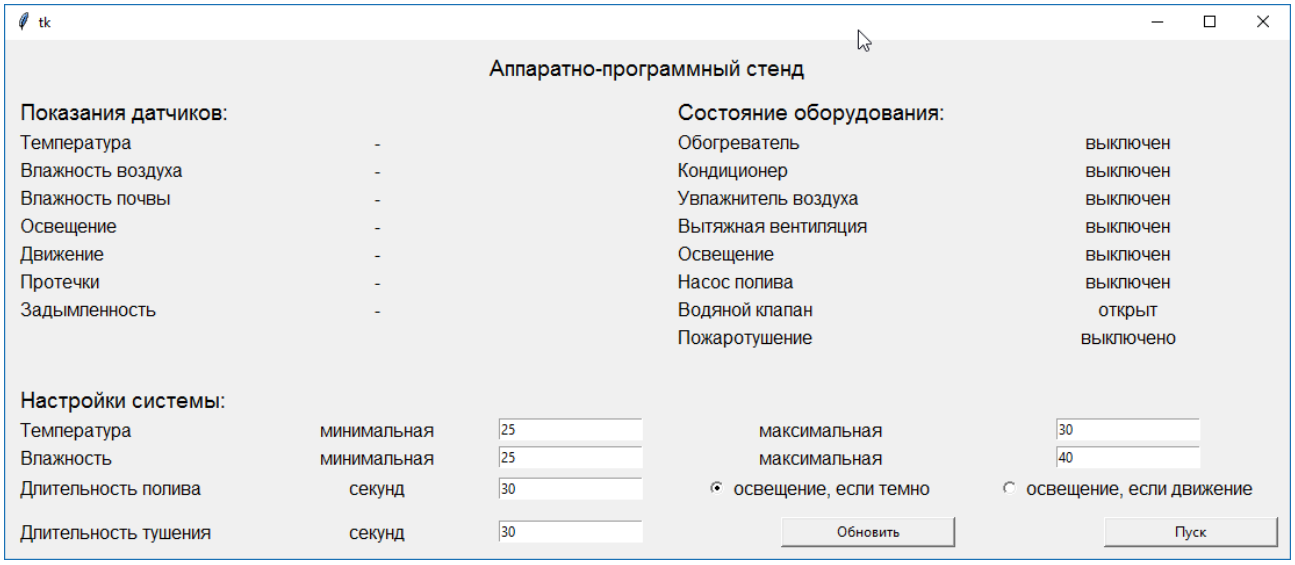

Рисунок 24 – Главное окно программы перед запуском стенда.

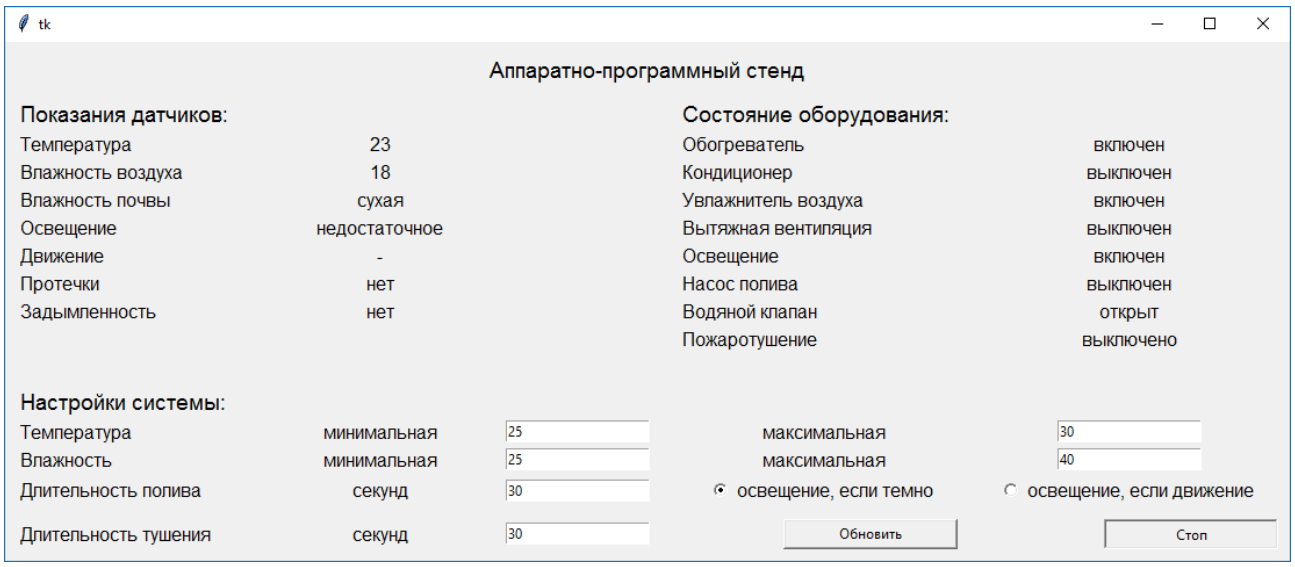

Рисунок 25 – Главное окно программы после запуска стенда.

Пример листинга программы на языке Python:

```
from tkinter import *                    # подключаем библиотеку оконного интерфейса tkinter
import requests                          # подключаем библиотеку для выполнения HTTP-запросов
import time                              # подключаем библиотеку для работы со временем

# глобальные переменные
WorkFlag = 0                             # флаг работы программы
Temp1Flag = 0                            # флаг состояния обогревателя
Temp2Flag = 0                            # флаг состояния кондиционера
Hum1Flag = 0                             # флаг состояния увлажнителя воздуха
Hum2Flag = 0                             # флаг состояния вытяжной вентиляции
LuxFlag = 0                              # флаг состояния освещения
WaterFlag = 0                            # флаг состояния водяного клапана
SmokeFlag = 0                            # флаг состояния системы пожаротушения
tMin = 25                                # минимально допустимая температура
tMax = 30                                # максимально допустимая температура
humMin = 25                              # минимально допустимая влажность
```

```
humMax = 40                                  # максимально допустимая влажность
pumpTime = 30                                # продолжительность работы насоса полива в секундах
pumpTime = 30                                # продолжительность работы насоса полива в секундах
smokeTime = 30                               # продолжительность работы системы пожаротушения в секундах

# функция обработки температуры
def Temp():
    global Temp1Flag                         # функция использует глобальные переменные Temp1Flag и Temp2Flag, изменяя их значения
    global Temp2Flag
    r = requests.get("http://192.168.1.21/temp")      # запрашиваем температуру
    s = r.text
    i = len(s)                               # длина строки ответа
    s1 = s[i-5:i-3]                          # выделяем из строки ответа 3-4 символы с конца (значение температуры)
    temp = int(s1)                           # переводим строку символов в числовой формат
    label4['text'] = s1                      # отображаем температуру в окне программы
    print(temp)
    if temp < tMin:                          # если температура ниже минимума
        if Temp1Flag == 0:                   # если обогреватель выключен
            print("Температура ниже минимума, включаем обогреватель")
            p = {"pin1": "1"}
            Temp1Flag = 1
            r = requests.get("http://192.168.1.23", params=p)# команда на включение обогревателя
            label24['text'] = 'включен'      # отображаем состояние обогревателя в окне программы
    else:                                    # иначе - если температура выше минимума
        if Temp1Flag == 1:                   # если обогреватель включен
            print("Выключаем обогреватель")
            Temp1Flag = 0
            p = {"pin1": "0"}
```

```
        r = requests.get("http://192.168.1.23", params=p)# команда на выключение обогревателя
        label24['text'] = 'выключен'                    # отображаем состояние обогревателя в окне программы
    if temp > tMax:                                     # если температура выше максимума
        if Temp2Flag == 0:                              # если кондиционер выключен
            print("Температура выше максимума, включаем кондиционер")
            p = {"pin2": "1"}
            Temp2Flag = 1
            r = requests.get("http://192.168.1.23", params=p)# команда на включение кондиционера
            label26['text'] = 'включен'                 # отображаем состояние кондиционера в окне программы
    else:                                               # иначе - если температура ниже максимума
        if Temp2Flag == 1:                              # если кондиционер включен
            print("Выключаем кондиционер")
            Temp2Flag = 0
            p = {"pin2": "0"}
            r = requests.get("http://192.168.1.23", params=p)# команда на выключение кондиционера
            label26['text'] = 'выключен'                # отображаем состояние кондиционера в окне программы

# функция обработки влажности воздуха
def Hum():
    global  Hum1Flag            # функция использует глобальные переменные Hum1Flag и Hum2Flag, изменяя их значения
    global Hum2Flag
    r = requests.get("http://192.168.1.21/hum")     # запрашиваем влажность
    s = r.text
    i = len(s)                                      # длина строки ответа
    s1 = s[i-5:i-3]                                 # выделяем из строки ответа 3-4 символы с конца (значение влажности)
    hum = int(s1)                                   # переводим строку символов в числовой формат
label6['text'] = s1
    print(hum)
```

```
if hum < humMin:                    # если влажность ниже минимума
    if Hum1Flag == 0:               # если увлажнитель выключен
        print("Влажность ниже минимума, включаем увлажнитель воздуха")
        p = {"pin1": "1"}
        Hum1Flag = 1
        r = requests.get("http://192.168.1.25", params=p)   # команда на включение увлажнителя
        label28['text'] = 'включен'     # отображаем состояние увлажнителя в окне программы
    else:                           # иначе - если влажность выше минимума
        if Hum1Flag == 1:           # если увлажнитель включен
            print("Выключаем увлажнитель воздуха")
            Hum1Flag = 0
            p = {"pin1": "0"}
            r = requests.get("http://192.168.1.25", params=p)   # команда на выключение увлажнителя
            label28['text'] = 'выключен'    # отображаем состояние увлажнителя в окне программы
    if hum > humMax:                # если влажность выше максимума
        if Hum2Flag == 0:           # если вентиляция выключена
            print("Влажность выше максимума, включаем вытяжную вентиляцию")
            p = {"pin2": "1"}
            Hum2Flag = 1
            r = requests.get("http://192.168.1.25", params=p)   # команда на включение вытяжной вентиляции
            label30['text'] = 'включен'     # отображаем состояние вентиляции в окне программы
    else:                           # иначе - если влажность ниже максимума
        if Hum2Flag == 1:           # если вентиляция включена
            print("Выключаем вытяжную вентиляцию")
            Hum2Flag = 0
            p = {"pin2": "0"}
            r = requests.get("http://192.168.1.25", params=p)   # команда на выключение вытяжной вентиляции
            label30['text'] = 'выключен'    # отображаем состояние вентиляции в окне программы
```

```
# функция обработки влажности почвы
def Soil():
    r = requests.get("http://192.168.1.22/soil")                # запрашиваем влажность почвы
    s = r.text
    i = len(s)                                                  # длина строки ответа
    s1 = s[i-1]         # выделяем из строки ответа последний символ (логическое значение влажности)
    soil = int(s1)                                              # переводим строку символов в числовой формат
    if soil == 1:
        label8['text'] = "влажная"                              # отображаем в окне программы
    else:
        label8['text'] = "сухая"
    if soil == 0:                                               # если значение = 0 (земля сухая)
        print("Земля сухая, включаем полив")
        p = {"pin1": "1"}
        r = requests.get("http://192.168.1.24", params=p)   # команда на включение помпы полива
        label34['text'] = 'включен'                             # отображаем состояние насоса в окне программы
        time.sleep(pumpTime)                                    # пауза длительностью равна заданному времени полива
        p = {"pin1": "0"}
        r = requests.get("http://192.168.1.24", params=p)           # команда на выключение помпы полива
        label34['text'] = 'выключен'                            # отображаем состояние насоса в окне программы
    else:
        print("Земля достаточно влажная")

# функция обработки уровня освещенности
def Lux():
    global LuxFlag                  # функция использует глобальные переменную LuxFlag , изменяя ее значение
    r = requests.get("http://192.168.1.21/lux")         # запрашиваем освещенность
```

```
    s = r.text
    i = len(s)                                  # длина строки ответа
    s1 = s[i-1]             # выделяем из строки ответа последний символ (логическое значение освещенности)
    lux = int(s1)                               # переводим строку символов в числовой формат
    if lux == 1:
        label10['text'] = "достаточное"                 # отображаем в окне программы
    else:
        label10['text'] = "недостаточное"
    if lux == 0:                                # если значение = 0 (освещение недостаточное)
        if LuxFlag == 0:                        # если лампа выключена
            print("Освещение недостаточно, включаем лампу")
            LuxFlag = 1
            p = {"pin2": "1"}
            r = requests.get("http://192.168.1.24", params=p)       # команда на включение лампы
            label32['text'] = 'включен'             # отображаем состояние освещения в окне программы
    else:
        if LuxFlag == 1:                        # если лампа включена
            print("Освещение достаточно, выключаем лампу")
            LuxFlag = 0
            p = {"pin2": "0"}
            r = requests.get("http://192.168.1.24", params=p)       # команда на выключение лампы
            label32['text'] = 'выключен'            # отображаем состояние освещения в окне программы

# функция обработки датчика движения
def Move():
    global LuxFlag              # функция использует глобальные переменную LuxFlag , изменяя ее значение
    r = requests.get("http://192.168.1.22/move")    # запрашиваем наличие движения
    s = r.text
```

```
    i = len(s)                                  # длина строки ответа
    s1 = s[i-1]                  # выделяем из строки ответа последний
символ (логическое значение)
    move = int(s1)                              # переводим строку символов в
числовой формат
    if move == 1:
        label12['text'] = "зафиксировано"       # отображаем в окне программы
    else:
        label12['text'] = "не зафиксировано"
    if move == 1:                               #     если     значение     =     1
(зафиксировано движение)
        print("Зафиксировано движение")
        r = requests.get("http://192.168.1.21/lux")     # запрашиваем освещенность
        s = r.text
        i = len(s)                              # запрашиваем освещенность
        s1  =  s[i-1]                # выделяем из строки ответа последний символ
(логическое значение освещенности)
        lux = int(s1)                           # переводим строку символов в
числовой формат
        if lux == 0:                            # если значение = 0 (освещение
недостаточное)
            print("В помещении темно, включаем свет")
            p = {"pin2": "1"}
            r = requests.get("http://192.168.1.24", params=p)       #     команда     на
включение лампы
            LuxFlag = 1
            label32['text'] = 'включен'         #     отображаем     состояние
освещения в окне программы
    else:                                       # иначе - движения нет
        print("Движение отсутствует")
        if LuxFlag == 1:                        # если лампа включена
            print("В помещении никого нет, выключаем свет")
            p = {"pin2": "0"}
            r = requests.get("http://192.168.1.24", params=p)       #     команда     на
выключение лампы
            LuxFlag = 0
            label32['text'] = 'выключен'        # отображаем состояние освещения в
окне программы
```

```
# функция обработки датчика влаги
def Water():
    r = requests.get("http://192.168.1.26/water")      # запрашиваем наличие влаги
    s = r.text
    i = len(s)                                         # длина строки ответа
    s1 = s[i-1]                                # выделяем из строки ответа последний символ ( логическое значение)
    water = int(s1)                                    # переводим строку символов в числовой формат
    if water == 1:
        label36['text'] = "есть"                       # отображаем в окне программы
    else:
        label36['text'] = "нет"
    if water == 1:                                     # если значение = 1 (обнаружена влага)
        print("Обнаружена протечка воды")
        p = {"pin1": "1"}
        r = requests.get("http://192.168.1.27", params=p)   # команда на перекрытие аварийного клапана системы водоснабжения
        label40['text'] = 'закрыт'                     # отображаем состояние насоса в окне программы
    else:
        print("Протечек не обнаружено")


# функция обработки задымленности
def Smoke():
    r = requests.get("http://192.168.1.26/smoke")          # запрашиваем наличие дыма
    s = r.text
    i = len(s)                                         # длина строки ответа
    s1 = s[i-1]                # выделяем из строки ответа последний символ ( логическое значение влажности)
    smoke = int(s1)                                    # переводим строку символов в числовой формат
    if smoke == 1:
        label38['text'] = "есть"                       # отображаем в окне программы
```

```
    else:
        label38['text'] = "нет"
    if smoke == 1:                                  # если значение = 1 (обнаружено задымление)
        print("Обнаружено задымление")
        p = {"pin2": "1"}
        r = requests.get("http://192.168.1.27", params=p)       # команда на включение системы пожаротушения
        label42['text'] = 'включено'                # отображаем состояние насоса в окне программы
        root.update()                               # обновляем визуальные элементы главного окна
        time.sleep(smokeTime)                       # пауза длительностью равна заданному времени тушения
        p = {"pin2": "0"}
        r = requests.get("http://192.168.1.27", params=p)       # команда на отключение системы пожаротушения
        label42['text'] = 'выключено'               # отображаем состояние насоса в окне программы
    else:
        print("Задымления не обнаружено")

# функция обновления настроек системы после их изменения (кнопка "Обновить")
def EntryUpdate(event):
    global tMin                                     # функция использует глобальные переменные, изменяя их значения
global tMax
    global humMin
    global humMax
    global pumpTime
    tMin = int(entry1.get())                        # минимальнодопустимаятемпература
    tMax = int(entry2.get())                        # максимальнодопустимаятемпература
humMin = int(entry3.get())                          # минимально допустимая влажность
    humMax = int(entry4.get())                      # максимально допустимая влажность
```

```
    pumpTime = int(entry5.get())                    # продолжительность
работы насоса полива в секундах
    smokeTime = int(entry6.get())                   # продолжительность
пожаротушения

# функция запуска/остановки работы стенда (кнопка "Пуск"/"Стоп"
def Main(event=mainloop):
    global WorkFlag             # функция использует глобальную переменную
WorkFlag, изменяя ее значение
ifWorkFlag == 0:
WorkFlag = 1                                # запускаемстенд
button1['text'] = 'Стоп'
else:
        WorkFlag = 0                                # остановка работы стенда
        button1['text'] = 'Пуск'
    MainLoop()

# главный цикл работы программы
defMainLoop():
ifWorkFlag == 1:                                    # если стенд запущен
        Temp()                                      # функция работы с
температурой
        Hum()                                       # функция работы с влажностью
воздуха
        Soil()                                      # функция работы с влажностью
почвы
        if LuxSwitch.get() == 0:                    # в зависимости от состояния
переключателя
            label12['text'] = '-'
            Lux()                                   # функция работы с уровнем
освещенности
        else:
            Move()                                  # или функция обработки
датчика движения
        root.after(5000, MainLoop)

# создаем экземпляр оконного интерфейса
root = Tk()
```

```
# определяем элементы оконного интерфейса (надписи, кнопки и поля ввода данных)
label0 = Label(root, font='arial 14')
        # надписи
label1 = Label(root, text="Аппаратно-программныйстенд", font='arial 14', width=100)
label2 = Label(root, text="Показаниядатчиков:", font='arial 14')
label3 = Label(root, text="Температура", font='arial 12')
label4 = Label(root, text="-", font='arial 12')
label5 = Label(root, text="Влажностьвоздуха", font='arial 12')
label6 = Label(root, text="-", font='arial 12')
label7 = Label(root, text="Влажностьпочвы", font='arial 12')
label8 = Label(root, text="-", font='arial 12')
label9 = Label(root, text="Освещение", font='arial 12')
label10 = Label(root, text="-", font='arial 12')
label11 = Label(root, text="Движение", font='arial 12')
label12 = Label(root, text="-", font='arial 12')
label13 = Label(root, text="Настройкисистемы:", font='arial 14')
label14 = Label(root, text="Температура", font='arial 12')
label15 = Label(root, text="минимальная", font='arial 12')
label16 = Label(root, text="максимальная", font='arial 12')
label17 = Label(root, text="Влажность", font='arial 12')
label18 = Label(root, text="минимальная", font='arial 12')
label19 = Label(root, text="максимальная", font='arial 12')
label20 = Label(root, text="Длительностьполива", font='arial 12')
label21 = Label(root, text="секунд", font='arial 12')
label22 = Label(root, text="Состояниеоборудования:", font='arial 14')
label23 = Label(root, text="Обогреватель", font='arial 12')
label24 = Label(root, text="выключен", font='arial 12')
label25 = Label(root, text="Кондиционер", font='arial 12')
label26 = Label(root, text="выключен", font='arial 12')
label27 = Label(root, text="Увлажнительвоздуха", font='arial 12')
label28 = Label(root, text="выключен", font='arial 12')
label29 = Label(root, text="Вытяжнаявентиляция", font='arial 12')
label30 = Label(root, text="выключен", font='arial 12')
label31 = Label(root, text="Освещение", font='arial 12')
label32 = Label(root, text="выключен", font='arial 12')
label33 = Label(root, text="Насосполива", font='arial 12')
```

```
label34 = Label(root, text="выключен", font='arial 12')
label35 = Label(root, text="Протечки", font='arial 12')
label36 = Label(root, text="-", font='arial 12')
label37 = Label(root, text="Задымленность", font='arial 12')
label38 = Label(root, text="-", font='arial 12')
label39 = Label(root, text="Водянойклапан", font='arial 12')
label40 = Label(root, text="открыт", font='arial 12')
label41 = Label(root, text="Пожаротушение", font='arial 12')
label42 = Label(root, text="выключено", font='arial 12')
label43 = Label(root, text="Длительностьтушения", font='arial 12')
label44 = Label(root, text="секунд", font='arial 12')
entry1 = Entry(root)                                    # полявводаданных
entry2 = Entry(root)
entry3 = Entry(root)
entry4 = Entry(root)
entry5 = Entry(root)
entry6 = Entry(root)
entry1.insert(0,"25")
entry2.insert(0,"30")
entry3.insert(0,"25")
entry4.insert(0,"40")
entry5.insert(0,"30")
entry6.insert(0,"30")
LuxSwitch=IntVar()                                      # переключатель
radio1=Radiobutton(root,text='освещение,        еслитемно',        font='arial        12',
variable=LuxSwitch,   value=0)          radio2=Radiobutton(root,text='освещение,
еслидвижение', font='arial 12', variable=LuxSwitch, value=1)
button1 = Button(root, text="Пуск", width=20)
button1.bind("<Button-1>", Main)                        # кнопки
button2 = Button(root, text="Обновить", width=20)
button2.bind("<Button-1>", EntryUpdate)
# определяем места расположения элементов оконного интерфейса
label1.grid(row=0,column=0,columnspan=5,pady=10)
label2.grid(row=1,column=0,columnspan=2,padx=10,sticky='w')
label3.grid(row=2,column=0,padx=10,sticky='w')
label4.grid(row=2,column=1)
label5.grid(row=3,column=0,padx=10,sticky='w')
label6.grid(row=3,column=1)
```

```
label7.grid(row=4,column=0,padx=10,sticky='w')
label8.grid(row=4,column=1)
label9.grid(row=5,column=0,padx=10,sticky='w')
label10.grid(row=5,column=1)
label11.grid(row=6,column=0,padx=10,sticky='w')
label12.grid(row=6,column=1)
label35.grid(row=7,column=0,padx=10,sticky='w')
label36.grid(row=7,column=1)
label37.grid(row=8,column=0,padx=10,sticky='w')
label38.grid(row=8,column=1)
label0.grid(row=10,column=0)
label13.grid(row=11,column=0,columnspan=5,padx=10,sticky='w')
label14.grid(row=12,column=0,padx=10,sticky='w')
label15.grid(row=12,column=1)
entry1.grid(row=12,column=2)
label16.grid(row=12,column=3)
entry2.grid(row=12,column=4)
label17.grid(row=13,column=0,padx=10,sticky='w')
label18.grid(row=13,column=1)
entry3.grid(row=13,column=2)
label19.grid(row=13,column=3)
entry4.grid(row=13,column=4)
label20.grid(row=14,column=0,padx=10,sticky='w')
label21.grid(row=14,column=1)
entry5.grid(row=14,column=2)
radio1.grid(row=14,column=3)
radio2.grid(row=14,column=4)
label43.grid(row=15,column=0,padx=10,sticky='w')
label44.grid(row=15,column=1)
entry6.grid(row=15,column=2)
label22.grid(row=1,column=3,columnspan=2,sticky='w')
label23.grid(row=2,column=3,sticky='w')
label24.grid(row=2,column=4)
label25.grid(row=3,column=3,sticky='w')
label26.grid(row=3,column=4)
label27.grid(row=4,column=3,sticky='w')
label28.grid(row=4,column=4)
label29.grid(row=5,column=3,sticky='w')
```

```
label30.grid(row=5,column=4)
label31.grid(row=6,column=3,sticky='w')
label32.grid(row=6,column=4)
label33.grid(row=7,column=3,sticky='w')
label34.grid(row=7,column=4)
label39.grid(row=8,column=3,sticky='w')
label40.grid(row=8,column=4)
label41.grid(row=9,column=3,sticky='w')
label42.grid(row=9,column=4)
button1.grid(row=15,column=4,padx=10,pady=10,sticky='e')
button2.grid(row=15,column=3,padx=10,pady=10,sticky='e')
# запускаем поток главного окна программы
root.mainloop()
```

4.9 Удаленное управление работой стенда с других устройств в локальной сети.

Протокол передачи данных TCP/IPпозволяет организовать управление аппаратно-программным стендом с любого устройства, имеющего подключение к локальной сети. Для демонстрации этой возможности в рамках бакалаврской работы разработана управляющая программа в среде Embarcadero RadStudio.

Среда Embarcadero RadStudio является мультиплатформенной средой разработки и позволяет создавать приложения для операционных систем Windows, Android, OSXи iOS, имеющие единый интерфейс и исходный код. Для получения приложений для разных операционных систем достаточно просто поменять опции компиляции [8].

С помощью приложения можно управлять стендом с любого устройства на базе Windows, Android, OSXи iOS через Wi-Fiили Ethernet (рисунок 26 и 27).

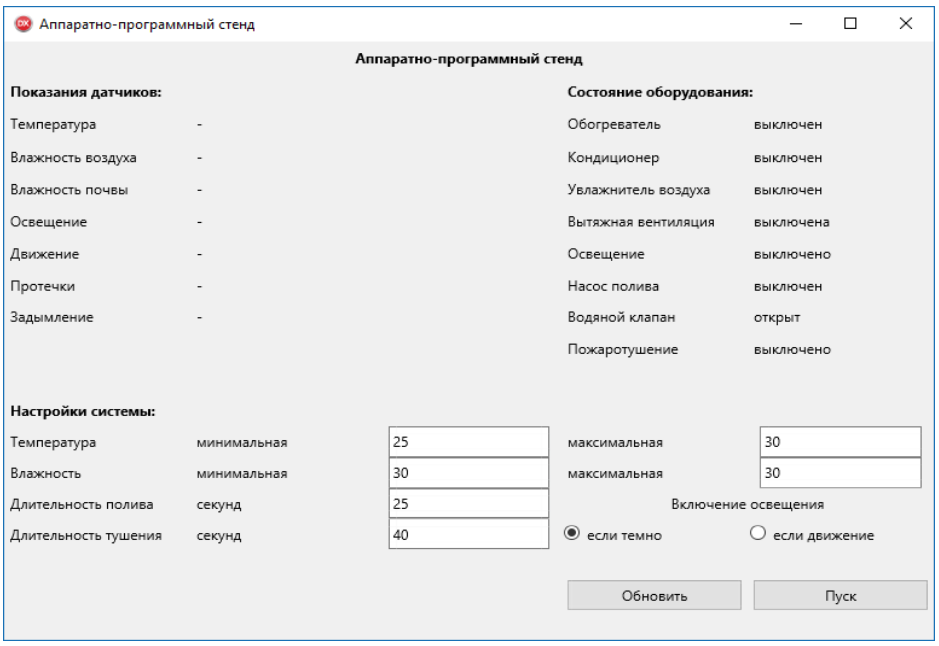

Рисунок 26 – Главное окно программы перед запуском стенда.

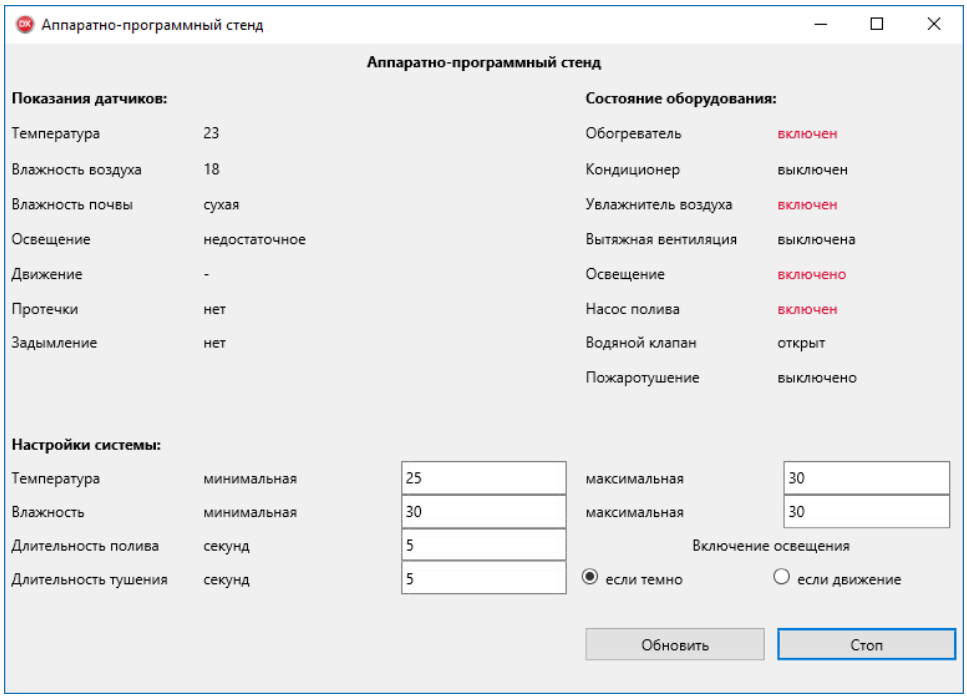

Рисунок 27 – Главное окно программы после запуска стенда.

Функционал данной управляющей программы, ее интерфейс и алгоритм работы полностью аналогичны управляющей программе на языке Python, описанной в подразделе 3.9 настоящей бакалаврской работы.

Листинг управляющей программы в среде Embarcadero RadStudioприведен в приложении 2.

5. Эффективность работы.

В ходе работы все поставленные задачи были выполнены. Опытный образец стенда собран и протестирован. Результаты бакалаврской работы могут быть применены в программах высшего образования для обучения специалистов в области бытовой автоматизации.

лабораторный стенд полностью выполняет свою роль и предназначен для обучения студентов проектированию и программированию систем домашней автоматизации, а также их управлению. К тому же данный стенд может применятся как демонстрационный на различных научных выставках и мероприятиях, так как будет иметь режим демонстрации по средствам управления через программу на смартфоне.

Возможно изготовление таких же стендов для продажи в другие учебные заведения для аналогичного использования.

ЗАКЛЮЧЕНИЕ

В ходе выполнения бакалаврской работы разработан аппаратно-программный стенд для выполнения практических работ по управлению устройствами системы бытовой автоматизации с использованием языка программирования высокого уровня Pythoni протокола передачи данных TCP/IP.

Аппаратно-программный стенд позволяет организоватьпрактическое обучение учащихся навыкам написания программ на языке Python, а также основам передачи данных в локльной сети и в сети интернет на основании разработанных методических пособий по выполнению лабораторных работ. Благодаря приведенным рекомендациям по изменению исходных задач возможно значительное увеличение вариантов практических работ.

В результате внедрения аппаратно-программного стенда планируется получить следующие результаты: получение учащимися основных навыков взаимодействия компонентов систем бытовой автоматизации, увеличение эффективности обучения программированию на языке высокого уровня Python.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1 Лутц, М. Изучаем Python (4-е издание). 2011 .— 1274 с

2 Саммерфилд, М. Программирование на Python 3. Подробное руководство. 2010.- 608 стр.

3 Прохоренок, Н.А. Python 3. Самое необходимое. 2016 .— 464 с

4 Петин, В.А. Микрокомпьютеры Raspberry Pi. Практическое руководство. 2015 .— 240 с

5 Петин, В.А. Arduino и Raspberry Pi впроектах Internet of Things . 2016. – 320 с

6 Ричардсон, М. Заводим Raspberry Pi. 2013. – 230 с

7 Грингард, С. Интернет вещей. Будущее уже здесь. 2016 г.

8 Росляков А.В. Интернет вещей. 2015 .— 136 с

9 Чеппел Лора А., Титтел Эд. TCP/IP. Учебный курс. - 2003. – 960 стр.

10 Росс, Д. Wi-Fi. Беспроводная сеть. 2007. – 320 с

11 Пролетарский А.В., Баскаков И.В., Чирков Д.Н. Беспроводные сети Wi-Fi. 2007. – 177 с.

12 Андрей Дементьев. «Умный» дом XXI века. – 2016. – 100 Страниц.

13 Othmar Kyas. How to Smart Home: A Step by Step Guide to Your Personal Internet of Things 2016. – 332 pages.

14 Christian Paetz. Z-Wave Basics: Remote Control in Smart Homes. 2014. – 265 pages.

15 Dennis C. Brewer. Home Automation Made Easy: Do It Yourself Know How Using UPB, Insteon, X10 and Z-Wave. 2013. – 400 pages.

16 Hans Fuchs. Smart home Planning Guide. 2015. – 220 pages.

17 Комер, Д. TCP/IP крупным планом. 2009 г

18 Davinder Pal., SharmaAvatar., BaldeoCassiel Phillip.Raspberry Pi based Smart Home for Deployment in the Smart Grid. – International Journal of Computer Applications (0975 – 8887) Volume 119 – No.4, June 2015

19 Arkadiusz ŚPIEWAK., Wojciech SAŁABUN.A Mobile Gas Detector with an Arduino Microcontroller. –Wojciech SALABUN et al, Int.J.Computer Technology & Applications,Vol 6 (4),636-641

20 Rongrong Zhang., Xiaoping Zou., Wenhui Huang., Qimu-Surong.A Smart Home System Design Based on GSM. –Communications and Network, 2013, 5, 25-28 doi:10.4236/cn.2013.51B007 Published Online February 2013

Приложение А. Листинг исходного кода прошивок ESP8266.

Модуль ESP-01 № 1:

```
// модульчтенияпоказанийдатчиков температуры, влажности и освещенности
// фиксированный IP-адрес внутри локальной сети 192.168.1.21
// обращение к модулю:
// http://192.168.1.21/temp - запрос температуры
// http://192.168.1.21/hum - запрос влажности
// http://192.168.1.21/lux - запрос освещения (1 - достаточное, 0 - недостаточное)
// используемые порты: gpio0 - датчик освещения (фоторезистор) 1, gpio2 - датчик температуры и влажности DHT11

// подключаемые библиотеки
#include <ESP8266WiFi.h>// библиотека работы с Wi-Fi
#include "DHT.h"                // библиотека работы с датчиком температуры и влажности DHT

// КОНСТАНТЫ, КОТОРЫЕ МОЖНО МЕНЯТЬ В РАМКАХ НАСТРОЙКИ ПРОГРАММЫ
const char* ssid = "WiFiNet";          // имясети Wi-Fi
const char* password = "MyWiFI";  // парольсети Wi-Fi
IPAddress ip(192,168,1,21);            // фиксированный IP-адрес модуля внутри локальной сети

//-----------------------------------------------------
// ДАЛЬШЕ НИЧЕГО НЕ МЕНЯЕМ
//-----------------------------------------------------

IPAddress gateway(192,168,1,1);            // адрес шлюза локальной сети
IPAddress subnet(255,255,255,0);           // маска подсети
WiFiServer server(80);                     // создаем экземпляр сервера
DHT dht(2, DHT11);     // создаем экземпляр датчика DHT11 на входе gpio2

// установочная часть программы
void setup()
{
  Serial.begin(115200);         // скорость обмена с COM-портом (вывод отладочных данных)
  dht.begin();                  // инициализируем датчик DHT
  pinMode(0, INPUT);            // инициализируем порт 0 как входной порт
Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);                 // подключаемсяксети Wi-Fi
  WiFi.config(ip, gateway, subnet);           // запрашиваядлямодуляфиксированный IP-адрес
while (WiFi.status() != WL_CONNECTED)      // пока не получим от сети ответ, что подключение состоялось - рисуем прогресс-бар из точек
{
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");                  // подключениексети Wi-Fi состоялось
  server.begin();                                   // запускаемсервер
  Serial.println("Server started");                  // серверзапущен
```

```
  Serial.println(WiFi.localIP());                    // выводимприсвоенныйсетьюIP-адрес (тот, который мы сами и запросили)
}

// главный цикл программы
void loop()
{
  WiFiClient client = server.available();       // проверяем подключение к серверу клиента
  if (!client) return;                          // если никто не подключен - сразу выход
  while(!client.available()) delay(1);          // ожидаем получения от клиента данных
  String req = client.readStringUntil('\r');    // читаем полученные данные до конца строки (символ \r - конец строки)
  client.flush();                               // отбросим все прочие данные после конца строки
String s;
  if (req.indexOf("temp") != -1)                // запрошенатемпература
{
    float temp = dht.readTemperature();                 // читаемтемпературусдатчика DHT
    s=String(temp);
  }
  else if (req.indexOf("hum") != -1)                    // запрошенавлажность
  {
    float hum = dht.readHumidity();                     // читаемвлажностьсдатчика DHT
    s=String(hum);
  }
  elseif (req.indexOf("lux") != -1)                     // запрошенаосвещенность
{
    byte lux = digitalRead(0);                          // читаем освещенность с порта 0
s=String(lux);
  }
  else
  {
    client.stop();
return;                                 // если запрос не соответствует шаблону - выход
}
  String answer = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n"+s;
  Serial.println(s);
  client.print(answer);                                 // отправляемклиенту
delay(1);
}
```

Модуль ESP-01 № 2:

```
// модуль чтения показаний датчиков влажности почвы и движения
// фиксированный IP-адрес внутри локальной сети 192.168.1.22
// обращение к модулю:
// http://192.168.1.22/soil - запрос влажности почвы (0 - сухая, 1 - влажная)
// http://192.168.1.22/move - запрос движения (0 - движения нет, 1 - движение есть)
// используемые порты: gpio0 - датчик влажности почвы, gpio2 - датчик движения
// подключаемые библиотеки
#include <ESP8266WiFi.h>  // библиотека работы с Wi-Fi
```

```
// КОНСТАНТЫ, КОТОРЫЕ МОЖНО МЕНЯТЬ В РАМКАХ НАСТРОЙКИ ПРОГРАММЫ
const char* ssid = " WiFiNet ";                 // имясети Wi-Fi
const char* password = "MyWiFi";                // парольсети Wi-Fi
IPAddress ip(192,168,1,22);                    // фиксированный IP-адрес модуля внутри локальной сети

//--------------------------------------------------
// ДАЛЬШЕ НИЧЕГО НЕ МЕНЯЕМ
//--------------------------------------------------

IPAddress gateway(192,168,1,1);                 // адрес шлюза локальной сети
IPAddress subnet(255,255,255,0);                // маска подсети
WiFiServer server(80);                          // создаем экземпляр сервера

// установочная часть программы
void setup()
{
  Serial.begin(115200);         // скорость обмена с COM-портом (вывод отладочных данных)
  pinMode(0, INPUT);            // инициализируем порты 0 и 2 как входные порты
pinMode(2, INPUT);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);                   // подключаемсяксети Wi-Fi
  WiFi.config(ip, gateway, subnet);             // запрашиваядлямодуляфиксированный IP-адрес
while (WiFi.status() != WL_CONNECTED)           // пока не получим от сети ответ, что подключение состоялось - рисуем прогресс-бар из точек
{
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");             // подключениексети Wi-Fi состоялось
  server.begin();                               // запускаемсервер
  Serial.println("Server started");             // серверзапущен
  Serial.println(WiFi.localIP());               //       выводимприсвоенныйсетьюIP-адрес       (тот, который мы сами и запросили)
}

// главный цикл программы
void loop() {
  WiFiClient client = server.available();       // проверяем подключение к серверу клиента
  if (!client) return;                          // если никто не подключен - сразу выход
  while(!client.available()) delay(1);          // ожидаем получения от клиента данных
  String req = client.readStringUntil('\r');    //  читаем полученные данные до конца строки (символ \r - конец строки)
  client.flush();                               // отбросим все прочие данные после конца строки
byte i;
  if (req.indexOf("soil") != -1)
{
i = digitalRead(0);                             // запрошена влажность почвы - читаем с порта 0
```

```
   if (i==0) i=1;                                // инвертируем значение (датчик на сухую землю выдает 1, на влажную 0 - логичнее наоборот)
elsei=0;
  }
elseif (req.indexOf("move") != -1) i = digitalRead(2); // запрошенодвижение - читаемспорта 2
else
{
   client.stop();
   return;                                       // если запрос не соответствует шаблону - выход
}
  Serial.println(i);
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n"+String(i);
  client.print(s);                              // отправляемклиенту
  delay(1);
}
```

Модуль ESP-01 № 3:

```
// модуль управления с двумя выходами (включение/выключение)
// фиксированный IP-адрес внутри локальной сети 192.168.1.23
// обращение к модулю:
// http://192.168.1.23?pin1=0 - выход 1 выключен
// http://192.168.1.23?pin1=1 - выход 1 включен
// http://192.168.1.23?pin2=0 - выход 2 выключен
// http://192.168.1.23?pin2=1 - выход 2 включен
// используемые порты: gpio0 - выход 1, gpio2 - выход 2

#include<ESP8266WiFi.h>

// КОНСТАНТЫ, КОТОРЫЕ МОЖНО МЕНЯТЬ В РАМКАХ НАСТРОЙКИ ПРОГРАММЫ
const char* ssid = "WiFiNet";                   // имя сети Wi-Fi
const char* password = "MyWiFi";                // пароль сети Wi-Fi
IPAddressip(192,168,1,23);                    // фиксированный IP-адрес модуля внутри локальной сети

//-----------------------------------------------------
// ДАЛЬШЕ НИЧЕГО НЕ МЕНЯЕМ
//-----------------------------------------------------

IPAddressgateway(192,168,1,1);                // адрес шлюза локальной сети
IPAddresssubnet(255,255,255,0);               // маска подсети
bytepin1=0, pin2=0;                           // переменные состояния выходов

WiFiServerserver(80);                         // создаем экземпляр сервера

// установочная часть программы
voidsetup()
{
Serial.begin(115200);          // скорость обмена с COM-портом (вывод отладочных данных)
pinMode(0, OUTPUT);                   // инициализируем порты 0 и 2 как выходные порты, начальное значение - выключено
```

```
digitalWrite(0, 0);
  pinMode(2, OUTPUT);
  digitalWrite(2, 0);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);                    // подключаемся к сети Wi-Fi
  WiFi.config(ip, gateway, subnet);              // запрашивая для модуля фиксированный IP-адрес
  while (WiFi.status() != WL_CONNECTED)          // пока не получим от сети ответ, что подключение состоялось - рисуем прогресс-бар из точек
{
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");              // подключение к сети Wi-Fi состоялось
  server.begin();                                // запускаем сервер
  Serial.println("Server started");              // сервер запущен
  Serial.println(WiFi.localIP());                // выводим присвоенный сетью IP-адрес (тот, который мы сами и запросили)
}

// главный цикл программы
voidloop() {
WiFiClientclient = server.available();           // проверяем подключение к серверу клиента
if (!client) return;                             // если никто не подключен - сразу выход
while(!client.available()) delay(1);             // ожидаем получения от клиента данных
Stringreq = client.readStringUntil('\r');        // читаем полученные данные до конца строки (символ \r - конец строки)
client.flush();                                  // отбросим все прочие данные после конца строки
if (req.indexOf("pin1=0") != -1) pin1=0;         // анализируем запрос и устанавливаем переменные pin1 и pin2
else if (req.indexOf("pin1=1") != -1) pin1=1;
  else if (req.indexOf("pin2=0") != -1) pin2=0;
  else if (req.indexOf("pin2=1") != -1) pin2=1;
  else
  {
client.stop();
return;                                          // если запрос не соответствует шаблону - выход
  }
Serial.println(req);
digitalWrite(0, pin1); // переключаем состояние выходов в соответствии с полученными данными
digitalWrite(2, pin2);
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n"+String(pin1)+String(pin2);
client.print(s);                                 // отправляем клиенту ответ - текущее состояние выхода 1 и 2
delay(1);
}
```

Модуль ESP-01 № 4:

```
// модуль управления с двумя выходами (включение/выключение)
// фиксированный IP-адрес внутри локальной сети 192.168.1.24
// обращение к модулю:
// http://192.168.1.24?pin1=0 - выход 1 выключен
// http://192.168.1.24?pin1=1 - выход 1 включен
// http://192.168.1.24?pin2=0 - выход 2 выключен
// http://192.168.1.24?pin2=1 - выход 2 включен
// используемые порты: gpio0 - выход 1, gpio2 - выход 2

#include<ESP8266WiFi.h>

// КОНСТАНТЫ, КОТОРЫЕ МОЖНО МЕНЯТЬ В РАМКАХ НАСТРОЙКИ ПРОГРАММЫ
const char* ssid = "WiFiNet";                 // имя сети Wi-Fi
const char* password = "MyWiFi";              // пароль сети Wi-Fi
IPAddressip(192,168,1,24);            // фиксированный IP-адрес модуля внутри локальной сети

//-----------------------------------------------------
// ДАЛЬШЕ НИЧЕГО НЕ МЕНЯЕМ
//-----------------------------------------------------

IPAddressgateway(192,168,1,1);                // адрес шлюза локальной сети
IPAddresssubnet(255,255,255,0);               // маска подсети
bytepin1=0, pin2=0;                           // переменные состояния выходов

WiFiServerserver(80);                         // создаем экземпляр сервера

// установочная часть программы
voidsetup()
{
Serial.begin(115200);// скорость обмена с COM-портом (вывод отладочных данных)
pinMode(0, OUTPUT);       // инициализируем порты 0 и 2 как выходные порты, начальное значение - выключено
digitalWrite(0, 0);
  pinMode(2, OUTPUT);
  digitalWrite(2, 0);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);                 // подключаемся к сети Wi-Fi
  WiFi.config(ip, gateway, subnet);           // запрашивая для модуля фиксированный IP-адрес
  while (WiFi.status() != WL_CONNECTED)                 // пока не получим от сети ответ, что подключение состоялось - рисуем прогресс-бар из точек
{
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");           // подключение к сети Wi-Fi состоялось
  server.begin();                       // запускаем сервер
  Serial.println("Server started");     // сервер запущен
```

```
  Serial.println(WiFi.localIP());       // выводим присвоенный сетью IP-адрес (тот, который мы сами и запросили)
}

// главный цикл программы
voidloop() {
WiFiClientclient = server.available();       // проверяем подключение к серверу клиента
if (!client) return;                         // если никто не подключен - сразу выход
while(!client.available()) delay(1);         // ожидаем получения от клиента данных
Stringreq = client.readStringUntil('\r');    // читаем полученные данные до конца строки (символ \r - конец строки)
client.flush();                              // отбросим все прочие данные после конца строки
if (req.indexOf("pin1=0") != -1) pin1=0;     // анализируем запрос и устанавливаем переменные pin1 и pin2
else if (req.indexOf("pin1=1") != -1) pin1=1;
  else if (req.indexOf("pin2=0") != -1) pin2=0;
  else if (req.indexOf("pin2=1") != -1) pin2=1;
  else
  {
client.stop();
return;                                      // если запрос не соответствует шаблону - выход
  }
Serial.println(req);
digitalWrite(0, pin1);                       // переключаем состояние выходов в соответствии с полученными данными
digitalWrite(2, pin2);
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n"+String(pin1)+String(pin2);
client.print(s);          // отправляем клиенту ответ - текущее состояние выхода 1 и 2
delay(1);
}
```

Модуль ESP-01 № 5:

```
// модуль управления с двумя выходами (включение/выключение)
// фиксированный IP-адрес внутри локальной сети 192.168.1.25
// обращение к модулю:
// http://192.168.1.25?pin1=0 - выход 1 выключен
// http://192.168.1.25?pin1=1 - выход 1 включен
// http://192.168.1.25?pin2=0 - выход 2 выключен
// http://192.168.1.25?pin2=1 - выход 2 включен
// используемые порты: gpio0 - выход 1, gpio2 - выход 2

#include<ESP8266WiFi.h>

// КОНСТАНТЫ, КОТОРЫЕ МОЖНО МЕНЯТЬ В РАМКАХ НАСТРОЙКИ ПРОГРАММЫ
const char* ssid = "WiFiNet";          // имя сети Wi-Fi
const char* password = "MyWiFi";  // пароль сети Wi-Fi
IPAddressip(192,168,1,25);             // фиксированный IP-адрес модуля внутри локальной сети

//-----------------------------------------------------
```

```
// ДАЛЬШЕ НИЧЕГО НЕ МЕНЯЕМ
//--------------------------------------------------

IPAddressgateway(192,168,1,1);              // адрес шлюза локальной сети
IPAddresssubnet(255,255,255,0);             // маска подсети
bytepin1=0, pin2=0;                    // переменные состояния выходов

WiFiServerserver(80);                       // создаем экземпляр сервера

// установочная часть программы
voidsetup()
{
Serial.begin(115200);// скорость обмена с COM-портом (вывод отладочных данных)
pinMode(0, OUTPUT);        // инициализируем порты 0 и 2 как выходные порты, начальное значение - выключено
digitalWrite(0, 0);
  pinMode(2, OUTPUT);
  digitalWrite(2, 0);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);               // подключаемся к сети Wi-Fi
  WiFi.config(ip, gateway, subnet);         // запрашивая для модуля фиксированный IP-адрес
  while (WiFi.status() != WL_CONNECTED)        //                поканеполучимотсетиответ, чтоподключениесостоялось - рисуемпрогресс-баризточек
{
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");          // подключение к сети Wi-Fi состоялось
  server.begin();                           // запускаем сервер
  Serial.println("Server started");         // сервер запущен
  Serial.println(WiFi.localIP());           //  выводим присвоенный сетью IP-адрес (тот, который мы сами и запросили)
}

// главный цикл программы
voidloop() {
WiFiClientclient = server.available();      // проверяем подключение к серверу клиента
if (!client) return;                        // если никто не подключен - сразу выход
while(!client.available()) delay(1);       // ожидаем получения от клиента данных
Stringreq = client.readStringUntil('\r');   //  читаем полученные данные до конца строки (символ \r - конец строки)
client.flush();                             // отбросим все прочие данные после конца строки
if (req.indexOf("pin1=0") != -1) pin1=0;    // анализируем запрос и устанавливаем переменные pin1 и pin2
else if (req.indexOf("pin1=1") != -1) pin1=1;
  else if (req.indexOf("pin2=0") != -1) pin2=0;
  else if (req.indexOf("pin2=1") != -1) pin2=1;
  else
```

```
  {
client.stop();
return;                    // если запрос не соответствует шаблону - выход
  }
Serial.println(req);
digitalWrite(0, pin1); // переключаем состояние выходов в соответствии с полученными данными
digitalWrite(2, pin2);
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n"+String(pin1)+String(pin2);
client.print(s);           // отправляем клиенту ответ - текущее состояние выхода 1 и 2
delay(1);
}
```

Модуль ESP-01 № 6:

```
// модуль чтения показаний датчиков влаги и дыма
// фиксированный IP-адрес внутри локальной сети 192.168.1.26
// обращение к модулю:
// http://192.168.1.26/water - запрос наличия влаги (0 - нет, 1 - есть)
// http://192.168.1.26/smoke - запрос наличия дыма (0 - нет, 1 - есть)
// используемые порты: gpio0 - датчик влаги, gpio2 - датчик дыма
// подключаемые библиотеки
#include<ESP8266WiFi.h>   // библиотека работы с Wi-Fi

// КОНСТАНТЫ, КОТОРЫЕ МОЖНО МЕНЯТЬ В РАМКАХ НАСТРОЙКИ ПРОГРАММЫ
const char* ssid = "WiFiNet";                  // имя сети Wi-Fi
const char* password = "MyWiFi";               // пароль сети Wi-Fi
IPAddressip(192,168,1,26);              // фиксированный IP-адрес модуля внутри локальной сети
//-----------------------------------------------------
// ДАЛЬШЕ НИЧЕГО НЕ МЕНЯЕМ
//-----------------------------------------------------

IPAddressgateway(192,168,1,1);                 // адрес шлюза локальной сети
IPAddresssubnet(255,255,255,0);               // маска подсети
WiFiServerserver(80);                          // создаем экземпляр сервера

// установочная часть программы
voidsetup()
{
Serial.begin(115200);// скорость обмена с COM-портом (вывод отладочных данных)
pinMode(0, INPUT);                             // инициализируем порты 0 и 2 как входные порты
pinMode(2, INPUT);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);                  // подключаемся к сети Wi-Fi
  WiFi.config(ip, gateway, subnet);            // запрашивая для модуля фиксированный IP-адрес
  while (WiFi.status() != WL_CONNECTED)        // пока не получим от сети ответ, что подключение состоялось - рисуем прогресс-бар из точек
{
```

```
   delay(500);
   Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");                // подключение к сети Wi-Fi состоялось
  server.begin();                                 // запускаем сервер
  Serial.println("Server started");               // сервер запущен
  Serial.println(WiFi.localIP());                 // выводим присвоенный сетью IP-адрес (тот, который мы сами и запросили)
}

// главный цикл программы
voidloop() {
WiFiClientclient = server.available();            // проверяем подключение к серверу клиента
if (!client) return;                              // если никто не подключен - сразу выход
while(!client.available()) delay(1);              // ожидаем получения от клиента данных
Stringreq = client.readStringUntil('\r');         // читаем полученные данные до конца строки (символ \r - конец строки)
client.flush();                                   // отбросим все прочие данные после конца строки
byte i;
  if (req.indexOf("water") != -1)
{
i = digitalRead(0);                               // запрошена влага - читаем с порта 0
if (i==0) i=1;          // инвертируем значение (датчик на сухую землю выдает 1, на влажную 0 - логичнее наоборот)
elsei=0;
  }
elseif (req.indexOf("smoke") != -1) i = digitalRead(2);       // запрошендым - читаемспорта 2
else
  {
client.stop();
return;                                           // если запрос не соответствует шаблону - выход
}
  Serial.println(i);
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n"+String(i);
  client.print(s);                                // отправляем клиенту
  delay(1);
}
```

Модуль ESP-01 № 7:

```
// модуль управления с двумя выходами (включение/выключение)
// фиксированный IP-адрес внутри локальной сети 192.168.1.27
// обращение к модулю:
// http://192.168.1.27?pin1=0 - выход 1 выключен
// http://192.168.1.27?pin1=1 - выход 1 включен
// http://192.168.1.27?pin2=0 - выход 2 выключен
// http://192.168.1.27?pin2=1 - выход 2 включен
// используемые порты: gpio0 - выход 1, gpio2 - выход 2

#include<ESP8266WiFi.h>
```

```
// КОНСТАНТЫ, КОТОРЫЕ МОЖНО МЕНЯТЬ В РАМКАХ НАСТРОЙКИ ПРОГРАММЫ
const char* ssid = "WiFiNet";                  // имя сети Wi-Fi
const char* password = "MyWiFi";               // пароль сети Wi-Fi
IPAddressip(192,168,1,27);            // фиксированный IP-адрес модуля внутри локальной сети

//-----------------------------------------------------
// ДАЛЬШЕ НИЧЕГО НЕ МЕНЯЕМ
//-----------------------------------------------------

IPAddressgateway(192,168,1,1);                 // адрес шлюза локальной сети
IPAddresssubnet(255,255,255,0);                // маска подсети
bytepin1=0, pin2=0;                            // переменные состояния выходов
WiFiServerserver(80);                          // создаем экземпляр сервера

// установочная часть программы
voidsetup()
{
Serial.begin(115200);// скорость обмена с COM-портом (вывод отладочных данных)
pinMode(0, OUTPUT);        // инициализируем порты 0 и 2 как выходные порты, начальное значение - выключено
digitalWrite(0, 0);
  pinMode(2, OUTPUT);
  digitalWrite(2, 0);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);                  // подключаемся к сети Wi-Fi
  WiFi.config(ip, gateway, subnet);            // запрашивая для модуля фиксированный IP-адрес
  while (WiFi.status() != WL_CONNECTED)        // пока не получим от сети ответ, что подключение состоялось - рисуем прогресс-бар из точек
{
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");            // подключение к сети Wi-Fi состоялось
  server.begin();                              // запускаем сервер
  Serial.println("Server started");            // сервер запущен
  Serial.println(WiFi.localIP());        // выводим присвоенный сетью IP-адрес (тот, который мы сами и запросили)
}


// главный цикл программы
voidloop() {
WiFiClientclient = server.available();         // проверяем подключение к серверу клиента
if (!client) return;                           // если никто не подключен - сразу выход
while(!client.available()) delay(1);           // ожидаем получения от клиента данных
Stringreq = client.readStringUntil('\r');      // читаем полученные данные до конца строки (символ \r - конец строки)
```

```
client.flush();                                   // отбросим все прочие данные после конца строки
if (req.indexOf("pin1=0") != -1) pin1=0;      // анализируем запрос и устанавливаем переменные pin1 и pin2
else if (req.indexOf("pin1=1") != -1) pin1=1;
  else if (req.indexOf("pin2=0") != -1) pin2=0;
  else if (req.indexOf("pin2=1") != -1) pin2=1;
  else
  {
client.stop();
return;                                           // если запрос не соответствует шаблону - выход
  }
Serial.println(req);
digitalWrite(0, pin1);                        // переключаем состояние выходов в соответствии с полученными данными
digitalWrite(2, pin2);
  String s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\n"+String(pin1)+String(pin2);
client.print(s);                          // отправляем клиенту ответ - текущее состояние выхода 1 и 2
delay(1);
}
```

Приложение Б. Листинг программы в среде Embarcadero RadStudio.

```
unit Unit1;
interface
uses                                                    // включаемыемодули
  System.SysUtils, System.Types, System.UITypes, System.Classes, System.Variants,
  FMX.Types, FMX.Controls, FMX.Forms, FMX.Graphics, FMX.Dialogs,
  FMX.Controls.Presentation, FMX.StdCtrls, FMX.Layouts, FMX.Edit, FMX.EditBox,
  FMX.NumberBox, IdBaseComponent, IdComponent, IdTCPConnection, IdTCPClient,
  IdHTTP, FMX.ScrollBox, FMX.Memo;

type
  TForm2 = class(TForm)
    GridPanelLayout1: TGridPanelLayout;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Label35: TLabel;
    Label37: TLabel;
    Label38: TLabel;
    Label42: TLabel;
    Label43: TLabel;
    Label49: TLabel;
    Label54: TLabel;
    Label55: TLabel;
```

```
    Label57: TLabel;
    Label59: TLabel;
    Label60: TLabel;
    Label62: TLabel;
    Label64: TLabel;
    Label65: TLabel;
    Label69: TLabel;
    Label70: TLabel;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    Button1: TButton;
    Button2: TButton;
    Label8: TLabel;
    StyleBook1: TStyleBook;
    NumberBox1: TNumberBox;
    NumberBox2: TNumberBox;
    NumberBox3: TNumberBox;
    NumberBox4: TNumberBox;
    NumberBox5: TNumberBox;
    NumberBox6: TNumberBox;
    IdHTTP1: TIdHTTP;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Temp;
    procedure Hum;
    procedure Soil;
    procedure Lux;
    procedure Move;
    procedure Water;
    procedure Smoke;
    procedure Work;
    procedure Pause(i:integer);
    procedure FormShow(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var                         // глобальные переменные
Form2: TForm2;
WorkFlag: integer;          // флаг работы программы
Temp1Flag: integer;         // флаг состояния обогревателя
Temp2Flag: integer;         // флаг состояния кондиционера
  Hum1Flag: integer;        // флаг состояния увлажнителя воздуха
  Hum2Flag: integer;        // флаг состояния вытяжной вентиляции
  LuxFlag: integer;         // флаг состояния освещения
  WaterFlag: integer;       // флаг состояния водяного клапана
  SmokeFlag: integer;       // флаг состояния системы пожаротушения
  tMin: integer;            // минимально допустимая температура
```

```
  tMax: integer;              // максимально допустимая температура
  humMin: integer;            // минимально допустимая влажность
  humMax: integer;            // максимально допустимая влажность
  pumpTime: integer;          // продолжительность работы насоса полива в секундах
  smokeTime: integer;         // продолжительность работы системы пожаротушения в
секундах

implementation
{$R *.fmx}

procedure TForm2.Temp;
// функция обработки температуры
vars:string;
i,temp:integer;
begin
  s:=IdHTTP1.Get('http://192.168.1.21/temp');       // запрашиваем температуру
  i:=s.Length;                                      // длина строки ответа
  s:=Copy(s,i-4,2);                                 // выделяем из строки ответа 3-4 символы с
конца (значение температуры)
  Label5.Text:=s;
  temp:=StrToInt(s);                                // переводим строку в число
  if (temp < tMin) then                             // если температура ниже минимума
  begin
    if (Temp1Flag = 0) then                         // если обогреватель выключен
    begin
      IdHTTP1.Get('http://192.168.1.23/?pin1=1');   // команда на включение обогревателя
      Temp1Flag:= 1;
      Label7.Text:= 'включен';          // отображаем состояние обогревателя в окне программы
Label7.TextSettings.FontColor:=TAlphaColorRec.Crimson;
end
  end
  else                                              // если температура выше минимума
  begin
    if (Temp1Flag = 1) then                         // если обогреватель включен
    begin
      IdHTTP1.Get('http://192.168.1.23/?pin1=0');   // команда на выключение обогревателя
      Temp1Flag:=0;
      Label7.Text:= 'выключен';         // отображаем состояние обогревателя в окне программы
Label7.TextSettings.FontColor:=TAlphaColorRec.Black;
end;
  end;
  if (temp > tMax) then                             // если температура выше максимума
  begin
    if (Temp2Flag = 0) then                         // если кондиционер выключен
    begin
      IdHTTP1.Get('http://192.168.1.23/?pin2=1');   // команда на включение кондиционера
      Temp2Flag:= 1;
      Label13.Text:= 'включен';         // отображаем состояние кондиционера в окне программы
Label13.TextSettings.FontColor:=TAlphaColorRec.Crimson;
end
  end
```

```
  else                                                  // если температура ниже максимума
  begin
    if (Temp2Flag = 1) then                             // если кондиционер включен
    begin
      IdHTTP1.Get('http://192.168.1.23/?pin2=0');     // команда на выключение кондиционера
      Temp2Flag:=0;
      Label13.Text:= 'выключен';       // отображаем состояние кондиционера в окне программы
Label13.TextSettings.FontColor:=TAlphaColorRec.Black;
    end;
  end;
  Application.ProcessMessages;                          // обработкасобытийформы
end;

procedure TForm2.Hum;
// функция обработки влажности воздуха
var s:string;
    i,hum:integer;
begin
  s:=IdHTTP1.Get('http://192.168.1.21/hum');          // запрашиваем влажность воздуха
  i:=s.Length;                                          // длина строки ответа
  s:=Copy(s,i-4,2);      // выделяем из строки ответа 3-4 символы с конца (значение влажности)
  Label10.Text:=s;
  hum:=StrToInt(s);                                     // переводим строку в число
  if (hum < humMin) then                                // если влажность ниже минимума
  begin
    if (Hum1Flag = 0) then                              // если увлажнитель выключен
    begin
      IdHTTP1.Get('http://192.168.1.25/?pin1=1');     // команда на включение увлажнителя
      Hum1Flag:= 1;
      Label18.Text:= 'включен';         // отображаем состояние увлажнителя в окне программы
Label18.TextSettings.FontColor:=TAlphaColorRec.Crimson;
end
  end
  else                                                  // если влажность выше минимума
  begin
    if (Hum1Flag = 1) then                              // если увлажнитель включен
    begin
      IdHTTP1.Get('http://192.168.1.25/?pin1=0');     // команда на выключение увлажнителя
      Hum1Flag:=0;
      Label18.Text:= 'выключен';       // отображаем состояние увлажнителя в окне программы
Label18.TextSettings.FontColor:=TAlphaColorRec.Black;
    end;
  end;
  if (hum > humMax) then                                // есливлажностьвышемаксимума
  begin
    if (Hum2Flag = 0) then                              // есливентиляциявыключена
begin
      IdHTTP1.Get('http://192.168.1.25/?pin2=1');     // команда на включение вентиляции
      Hum2Flag:= 1;
      Label23.Text:= 'включен';         // отображаем состояние вентиляции в окне программы
Label23.TextSettings.FontColor:=TAlphaColorRec.Crimson;
```

```
end
  end
  else                                                  // если влажность ниже максимума
begin
if (Hum2Flag = 1) then                  // есливентиляциявключена
begin
IdHTTP1.Get('http://192.168.1.25/?pin2=0');          // команда на выключение вентиляции
      Hum2Flag:=0;
      Label23.Text:= 'выключен';       // отображаем состояние вентиляции в окне программы
Label23.TextSettings.FontColor:=TAlphaColorRec.Crimson;
    end;
  end;
  Application.ProcessMessages;                          // обработкасобытийформы
end;

procedure TForm2.Soil;
// функция обработки влажности почвы
var s:string;
    i,soil:integer;
begin
  s:=IdHTTP1.Get('http://192.168.1.22/soil');          // запрашиваем влажность почвы
  i:=s.Length;                                          // длина строки ответа
  s:=Copy(s,i,1);        // выделяем из строки ответа последний символ с конца (логическое
значение влажности)
  soil:=StrToInt(s);                                    // переводим строку в число
  if (soil=0) then                                      // если значение = 0 (земля сухая)
  begin
    IdHTTP1.Get('http://192.168.1.24/?pin1=1');       // команда на включение насоса полива
    Label15.Text:= 'сухая';                             // отображаем влажность в окне программы
    Label33.Text:= 'включен';                 // отображаем состояние насоса в окне программы
Label33.TextSettings.FontColor:=TAlphaColorRec.Crimson;
Pause(pumpTime);                                        // задержка времени на полив
    IdHTTP1.Get('http://192.168.1.24/?pin1=0');       // команда на выключение насоса полива
    Label33.Text:= 'выключен';                // отображаем состояние насоса в окне программы
Label33.TextSettings.FontColor:=TAlphaColorRec.Black;
end
  else Label15.Text:= 'влажная';                        // отображаем влажность в окне программы
Application.ProcessMessages;                            // обработкасобытийформы
end;

procedure TForm2.Lux;
// функция обработки уровня освещенности
var s:string;
    i,lux:integer;
begin
  s:=IdHTTP1.Get('http://192.168.1.21/lux');          // запрашиваем освещенность
  i:=s.Length;                                          // длина строки ответа
  s:=Copy(s,i,1);        // выделяем из строки ответа последний символ с конца (логическое
значение освещенности)
  lux:=StrToInt(s);                                     // переводим строку в число
  if (lux=0) then                                 // если значение = 0 (освещение недостаточное)
```

```
begin
   if (LuxFlag=0) then                          // еслилампавыключена
begin
     LuxFlag:=1;
     IdHTTP1.Get('http://192.168.1.24/?pin2=1');    // команда на включение лампы
     Label20.Text:= 'недостаточное';            // отображаем освещенность в окне программы
     Label28.Text:= 'включено';                 // отображаем состояние лампы в окне программы
Label28.TextSettings.FontColor:=TAlphaColorRec.Crimson;
   end;
  end
  else
  begin
   if (LuxFlag=1) then                  // еслилампавключена
begin
     LuxFlag:=0;
     IdHTTP1.Get('http://192.168.1.24/?pin2=0');    // команда на bsключение лампы
     Label20.Text:= 'достаточное';              // отображаем освещенность в окне программы
     Label28.Text:= 'выключено';                // отображаем состояние лампы в окне программы
Label28.TextSettings.FontColor:=TAlphaColorRec.Black;
   end;
  end;
  Application.ProcessMessages;                          // обработкасобытийформы
end;

procedure TForm2.Move;
// функция обработки датчика движения
var s:string;
   i,move,lux:integer;
begin
  s:=IdHTTP1.Get('http://192.168.1.22/move');       // запрашиваем движение
  i:=s.Length;                                      // длина строки ответа
  s:=Copy(s,i,1); // выделяем из строки ответа последний символ с конца (логическое
значение)
  move:=StrToInt(s);                                // переводим строку в число
  if (move=1) then                          // если значение = 1 (зафиксировано движение)
  begin
   Label25.Text:= 'зафиксировано';                  // отображаем движение в окне программы
   s:=IdHTTP1.Get('http://192.168.1.21/lux');       // запрашиваем освещенность
   i:=s.Length;                                     // длина строки ответа
   s:=Copy(s,i,1); // выделяем из строки ответа последний символ с конца (логическое
значение)
   lux:=StrToInt(s);                                // переводим строку в число
   if (lux=0) then                          // если значение = 0 (освещенность недостаточная)
   begin
     LuxFlag:=1;
     IdHTTP1.Get('http://192.168.1.24/?pin2=1');    // команда на включение лампы
     Label28.Text:= 'включено';                 // отображаем состояние лампы в окне программы
Label28.TextSettings.FontColor:=TAlphaColorRec.Crimson;
end;
  end
  else
```

```
  begin
    Label25.Text:= 'не зафиксировано';                  // отображаем движение в окне программы
    if (LuxFlag=1) then                                 // если лампа включена
    begin
      LuxFlag:=0;
      IdHTTP1.Get('http://192.168.1.24/?pin2=0');     // команда на выключение лампы
      Label28.Text:= 'выключено';                 // отображаем состояние лампы в окне программы
Label28.TextSettings.FontColor:=TAlphaColorRec.Black;
    end
  end;
  Application.ProcessMessages;                          // обработкасобытийформы
end;

procedure TForm2.Water;
// функцияобработкидатчикавлаги
var s:string;
    i,water:integer;
begin
  s:=IdHTTP1.Get('http://192.168.1.26/water');        // запрашиваем протечки
  i:=s.Length;                                          // длина строки ответа
  s:=Copy(s,i,1); // выделяем из строки ответа последний символ с конца (логическое
значение)
  water:=StrToInt(s);                                   // переводим строку в число
  if (water=1) then                                     // если значение = 1 (обнаружена влага)
  begin
    Label30.Text:= 'есть';                              // отображаем в окне программы
      IdHTTP1.Get('http://192.168.1.27/?pin1=1');     // команда на перекрытие электрического
клапана системы водоснабжения
      Label38.Text:= 'закрыт';                  // отображаем состояние клапана в окне программы
Label38.TextSettings.FontColor:=TAlphaColorRec.Crimson;
end
  else Label30.Text:= 'нет';                            // отображаем в окне программы
Application.ProcessMessages;                            // обработкасобытийформы
end;

procedure TForm2.Smoke;
// функция обработки задымленности
var s:string;
    i,smoke:integer;
begin
  s:=IdHTTP1.Get('http://192.168.1.26/smoke');        // запрашиваем задымленность
  i:=s.Length;                                          // длина строки ответа
  s:=Copy(s,i,1); // выделяем из строки ответа последний символ с конца (логическое
значение)
  smoke:=StrToInt(s);                                   // переводим строку в число
  if (smoke=1) then                             // если значение = 1 (обнаружено задымление)
  begin
    IdHTTP1.Get('http://192.168.1.27/?pin2=1')      // команда на включение системы
пожаротушения
    Label35.Text:= 'есть';                              // отображаем в окне программы
```

```
    Label43.Text:= 'включено';                // отображаем состояние системы пожаротушения в окне программы
Label43.TextSettings.FontColor:=TAlphaColorRec.Crimson;
Pause(smokeTime);                                    // задержка времени на тушение
    IdHTTP1.Get('http://192.168.1.27/?pin2=0');      // команда на выключение системы пожаротушения
    Label43.Text:= 'выключено';       // отображаем состояние системы пожаротушения в окне программы
Label43.TextSettings.FontColor:=TAlphaColorRec.Black;
end
  else Label35.Text:= 'нет';                         // отображаем в окне программы
Application.ProcessMessages;                         // обработкасобытийформы
end;

procedure TForm2.Button1Click(Sender: TObject);
// функция обновления настроек системы после их изменения (кнопка "Обновить")
begin
  tMin:= Round(NumberBox1.Value);                    // минимально допустимая температура
  tMax:= Round(NumberBox5.Value);                    // максимально допустимая температура
  humMin:= Round(NumberBox2.Value);                  // минимально допустимая влажность
  humMax:= Round(NumberBox6.Value);                  // максимально допустимая влажность
  pumpTime:= Round(NumberBox3.Value);          // продолжительность работы насоса полива в секундах
  smokeTime:= Round(NumberBox4.Value);         // продолжительность пожаротушения
end;

procedure TForm2.Pause(i:integer);
// функция запуска/остановки работы стенда (кнопка "Пуск"/"Стоп")
var j:integer;
begin
  for j:= 1 to i*10 do
begin
Sleep(100);
    Application.ProcessMessages;      // обрабатываем события (чтобы программа не замерзла)
end;
end;
procedure TForm2.Button2Click(Sender: TObject);
// функция запуска/остановки работы стенда (кнопка "Пуск"/"Стоп")
begin
  if (WorkFlag=0) then
  begin
    WorkFlag:=1;                                     // запускаемстенд
    Button2.Text:='Стоп';
  end
  else
  begin
WorkFlag:=0;                                         // остановкаработыстенда
Button2.Text:='Пуск'
end;
end;
```

```
procedure TForm2.Work;
vari: Integer;
// обработкавсехдатчиковиустройств
begin
  if (WorkFlag=1) then                  // еслистендзапущен
begin
    Temp;                                          // функция работы с температурой
    Hum;                                           // функция работы с влажностью воздуха
    Soil;                                          // функция работы с влажностью почвы
    Water;                                         // функция работы с датчиком влаги
    Smoke;                                         // функция работы с задымлением
    if (RadioButton1.IsChecked=true) then          // в зависимости от состояния переключателя
begin
Label25.Text:= '-';
Lux;                                               // функция работы с уровнем освещенности
    end
    else Move;                                     // или функция обработки датчика движения
end;
  Pause(5);                                        // пауза 5 секунд
end;
procedure TForm2.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
// закрыть программу
begin
  Halt;
end;
procedure TForm2.FormShow(Sender: TObject);
// при запуске программы
var Loop:boolean;
begin
  NumberBox1.Value:=25;                            // начальныеустановки
  NumberBox2.Value:=30;
NumberBox3.Value:=25;
  NumberBox4.Value:=40;
  NumberBox5.Value:=30;
  NumberBox6.Value:=30;
  WorkFlag:=0;                                     // флагработыпрограммы
Temp1Flag:=0;                                      // флаг состояния обогревателя
  Temp2Flag:=0;                                    // флаг состояния кондиционера
  Hum1Flag:=0;                                     // флаг состояния увлажнителя воздуха
  Hum2Flag:=0;                                     // флаг состояния вытяжной вентиляции
  LuxFlag:=0;                                      // флаг состояния освещения
  WaterFlag:=0;                                    // флаг состояния водяного клапана
SmokeFlag:=0;                                      // флаг состояния системы пожаротушения
RadioButton1.IsChecked:=true;
Button1Click(Sender);
  Loop:=true;
  while Loop do Work;                              // бесконечныйцикл
end;
end.
```