УДК: 004.432.2 DOI: 10.18324/2224-1833-2024-3-21-29

Некоторые аспекты совершенствования применения современных гаджетов в промышленности

А.Н. Бильданов[a], Д.Б. Горохов[b], А.М. Патрусова[c]

Братский государственный университет, ул. Макаренко, 40, Братск, Россия
[a] fossadoe@yandex.ru, [b] denis_gorohov@mail.ru, [c] patrusova@mail.ru
[a] https://orcid.org/0009-0004-7081-0145, [b] https://orcid.org/0000-0001-7271-350X,
[c] https://orcid.org/0000-0001-5433-9614
Статья поступила 09.09.2024, принята 04.10.2024

В статье показан пример использования языка программирования Python для создания смарт-часов. Целью статьи является рассмотрение примера использования языка Python для программирования микроконтроллеров. Был описан язык программирования Python с точки зрения работы с аппаратным уровнем. В статье также приводится сравнение сильных сторон языка программирования Python в сравнении с другими популярными языками для микроконтроллеров. Были описаны основные функции и методы, демонстрирующие функциональность языка. В статье приводятся пользовательские функции как «низкого», так и «высокого» уровня, такие как вывод информации на дисплей, считывание сигналов с внешних модулей, удаленное подключение к серверам, парсинг данных. Пользовательские функции дополняют функционал смарт-часов, делая их использование более удобным. Помимо стандартного вывода времени на дисплей смарт-часов, в статье продемонстрированы функции по выводу даты в удобном для пользователя формате, вывода прогноза погоды и вывода информации о текущем курсе валют. Программирование на «низком» уровне демонстрируется в статье при программировании кнопки для свободного переключения различных дисплеев смарт-часов. В статье описано программирование на микроконтроллере ESP32, который имеет встроенный wi-fi модуль, используемый в различных функциях (в статье также демонстрируется процесс подключения микроконтроллера к Интернету с помощью wi-fi модуля). Функция подключения написана на языке Python. Основные особенности и спецификация микроконтроллера ESP32 также описаны в статье, проведено его сравнение с другими популярными моделями микроконтроллеров.

Ключевые слова: программируемый микроконтроллер; язык *Python*; смарт-часы; создание пользовательских функций.

Some aspects of improving the use of modern gadgets in industry

A.N. Bildanov[a], D.B. Gorohov[b], A.M. Patrusova[c]

Bratsk State University; 40, Makarenko St., Bratsk, Russia
[a] fossadoe@yandex.ru, [b] denis_gorohov@mail.ru, [c] patrusova@mail.ru
[a] https://orcid.org/0009-0004-7081-0145, [b] https://orcid.org/0000-0001-7271-350X,
[c] https://orcid.org/0000-0001-5433-9614
Received 09.09.2024, accepted 04.10.2024

The article shows an example of using the Python programming language to create a smart watch. The purpose of the article is to consider an example of using the Python language for programming microcontrollers. The Python programming language is characterized from the point of view of working with the hardware level. The article also compares the strengths of the Python programming language compared to other popular microcontroller languages. The main functions and methods are described to demonstrate the functionality of the language. The article also defines user functions of both "low" and "high" levels, such as displaying information on the display, reading signals from external modules, remotely connecting to servers, and data parsing. Custom features complement the functionality of smartwatches, making them more convenient to use. In addition to the standard display of time on the display of a smart watch, the article demonstrates functions for displaying the date in a user-friendly format, displaying a weather forecast, and displaying information about the current exchange rate. Low-level programming is demonstrated in the article by programming a button to freely switch between different smartwatch displays. The article presents programming on the ESP32 microcontroller, which has a built-in Wi-Fi module used in various functions. As a result, the article also demonstrates the process of connecting the ESP32 microcontroller to the Internet using a Wi-Fi module. The connection function is written in Python. The main features and specifications of the ESP32 microcontroller are also defined in the article, as well as a comparison with other popular microcontroller models.

Keywords: programmable microcontroller; Python language; smart watches; creating custom functions.

Введение. Формирование цифрового тренда в развитии производства и функционировании предприятий свидетельствует об изменении требований со стороны потребителя и стремительном внедрении инфокоммуникационных технологий во все сферы жизнедеятельности человека. Расширение спектра функциональных возможностей современных электронных устройств обусловлено высоким спросом на такие гаджеты, как ноутбуки, мониторы, системные блоки, смартфоны, фитнес-браслеты, умные колонки, беспроводные наушники, планшеты и др. Для обработки цифровой информации и выполнения программ в данном случае используют микроконтроллеры [1].

Компактный размер является одной из главных особенностей микроконтроллера, из-за чего долгое время микроконтроллеры обладали низкой мощностью [2].

Слабые характеристики микроконтроллеров требовали использования языка С и ограничивали функционал, который можно было реализовать на микроконтроллере [3].

На данный момент рынок микроконтроллеров насчитывает большое количество плат от разных производителей с различными характеристиками и спецификациями [4].

Рост характеристик микроконтроллеров позволяет использовать не только С в качестве языка программирования, но и *Python*, чей функционал больше [5].

При применении микроконтроллера для создания «умных вещей» от языка программирования требуется:

1. Прием/передача данных по *Bluetooth*.
2. Прием/передача данных по *Wi-Fi*.
3. Обмен данными по API-протоколу.
4. Ввод/вывод данных.
5. Отображение данных на OLED-дисплее.

В качестве реализуемого объекта были выбраны смарт-часы из-за из актуальности в текущее время. Смарт-часы требуют реализации всего вышеперечисленного функционала при их создании.

В отличие от языка программирования С, который хорошо задокументирован при работе с микроконтроллерами, *Python* активно применятся с недавнего времени и практически не имеет документации в этой области [6].

Целью статьи является демонстрация применения языка *Python* для реализации функционала, требуемого для создания смарт-часов.

В работе используется микроконтроллер ESP32, обладающий высокой производительностью, но низким объемом памяти. Поэтому язык выбирался исходя из возможностей самого микроконтроллера. *Python* является наилучшим выбором среди прочих языков из-за простого синтаксиса и малого объема готового кода [7].

Данная статья содержит материал, демонстрирующий возможности языка *Python* в ограниченных средах.

Главной задачей является использования языка P*ython* для программирования микроконтроллера ESP32, что демонстрирует конкурентоспособность *Python* в сравнении с языком программирования С в области работы с микроконтроллерами.

Python упрощает разработку программного обеспечения и адаптирован для свободного применения с микроконтроллерами, которые являются основой «умной техники» [8].

Структура работы представлена четырьмя разделами. Во втором разделе кратко описан язык программирования *Python*, используемый при реализации проекта. В третьем разделе показан процесс программирования смарт-часов на языке *Python*.

1. Описание языка программирования и популярные модели микроконтроллеров. *Python* — высокоуровневый язык программирования с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика и читаемости кода [9]. Большинство библиотек и модулей расширения доступны по лицензиям MIT или аналогичным лицензиям [10]. Лицензия MIT (англ. *Massachusetts Institute of Technology*) — лицензия свободного программного обеспечения, разработанная Массачусетским технологическим институтом [11]. Это значит, что *Python* можно свободно использовать и адаптировать для личного использования, в сфере образования и в коммерческих продуктах, а исходный код доступен для загрузки на странице *GitHub*, *GitLab* — наиболее популярных системах контроля версий программного обеспечения [12]. Главными преимуществами языка являются компактность и простота разработки, например:

1. *Python* занимает всего 256 Кб свободного места на внутренней памяти микроконтроллера ESP32 и требует для работы 16 Кб оперативной памяти.

2. «Высокоуровневый» синтаксис делает код понятным и читаемым, что ускоряет разработку проектов различной сложности.

Скорость работы *Python* достаточно высока. Время запуска от включения до загрузки первого скрипта (файла boot.py) составляет 150 микросекунд [13]. Несмотря на преимущества, *Python* уступает конкурирующим языкам программирования по скорости из-за принципа работы, показанного на рис. 1.

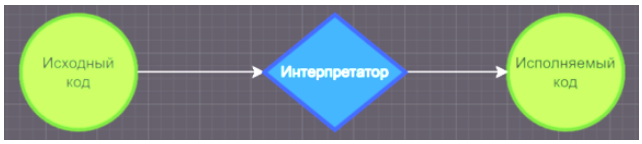

Рис. 1. Компиляция *Python*

Python загружает на микроконтроллер исходный код, а встроенный интерпретатор преобразовывает его в машинный код в режиме реального времени [14]. Однако разница в скорости работы несущественна, что делает *Python* оптимальным языком для реализации большинства проектов.

Наиболее популярными моделями микроконтроллеров являются ESP32, ESP8266, *Raspberry Pi*, *Arduino Nano*, *PyBoard*.

2. Реализация проекта. Применение языка *Python* будет продемонстрировано при программировании микроконтроллера ESP32 для дальнейшего использования в смарт-часах.

ESP32 — высокоинтегрированный чип с крайне малым энергопотреблением. Он был выбран в данной работе из-за своих малых размеров и высокой производительности. Одним из главных преимуществ данного чипа является наличие встроенных модулей *Wi-Fi* и *Bluetooth*.

ESP32 был создан как замена предшествующему ESP8266 и имеет производительность в 4 раза выше. Наличие двух ядер, работающих на частоте 160 МГц, значительно упрощает оптимизацию кода под выбранный микроконтроллер.

Единственным недостатком ESP32 является малый объем памяти — как постоянной, так и оперативной. Объем оперативной памяти составляет 520 Кб, а постоянной — всего 448 Кб.

Перед программной реализацией необходимо определиться с необходимыми модулями, которое будут использоваться в работе. В моем случае это OLED-дисплей для вывода получаемой информации и кнопка для переключения дисплеев.

У распиновки ESP32 есть особенность — большинство пинов являются комбинированными пинами типа GPIO (англ. *General-Purpose Input/Output*). Это означает, что один и тот же контакт может быть использован в качестве входа или выхода.

Это значительно упрощает подключение модулей и уменьшает шанс ошибки при подключении.

Пины OLED-дисплея подключаются к соответствующим пинам микроконтроллера ESP32 согласно следующей схеме:

1. GND – GND
2. VCC – 3V3
3. SCL – D14
4. SDA – D13

Пины кнопки подключаются к пинам микроконтроллера GND и D4 в любом порядке.

Готовая схема проекта показана на рис. 2.

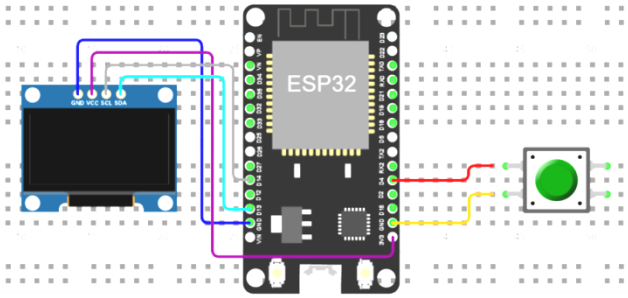

Рис. 2. Схема проекта

Определившись с будущим функционалом, необходимо сначала загрузить все необходимые библиотеки: *ssd1306* и *uresqests*. Первая библиотека используется для взаимодействия с OLED-дисплеем, а вторая — для серверных запросов. Скачанная библиотека представляет собой файл с расширением «ру», который в дальнейшем необходимо вручную загрузить на ESP32 [15]. При реализации проекта необходимо использовать строго требуемое количество библиотек, чтобы не упереться в потолок памяти. Не используемые библиотеки необходимо не только не объявлять в коде, но и полностью удалять из внутренней памяти микроконтроллера.

Установка библиотек на микроконтроллер значительно отличается от установки тех же библиотек в обычной версии *Python*. Отсутствие терминала делает невозможным ввести команду «*pip install …*» для установки необходимой библиотеки. Поэтому они загружаются вручную.

Файл с расширение «ру» переносится в память микроконтроллера через среду разработки. После чего библиотеку можно вызывать стандартным «*import*» в коде.

После установки всего необходимого программного обеспечения можно приступать к программной реализации проекта. Для начала необходимо подключить будущие смарт-часы к *Wi-Fi*. Например, такие функции, как получение времени, даты и погоды, осуществляются через подключение к RTC-серверу и получение необходимых данных. RTC-сервер (англ. *Real-Time Clock*) представляет собой сервер, содержащий данные о реальном текущем времени. Для этого микроконтроллер должен быть подключен к интернету. На рис. 3 представлена функция, вызов которой отвечает за подключение ESP32 к *Wi-Fi*.

```
def ConnectNet():
    try:
        wifi = network.WLAN(network.STA_IF)
        wifi.active(True)
        print(wifi.scan())
        if not wifi.isconnected():
            wifi.connect(SSID, PASSWORD)
            print('start to connect wifi')
            oled.text("WiFi connect", 0,  0)
            oled.show()
            for i in range(30):
                print('try to connect wifi in {}s'.format(i))
                oled.text(".", 12*8+i*4,  0)
                oled.show()
                time.sleep(1)
                if wifi.isconnected():
                    oled.fill(0)
                    break
        if wifi.isconnected():
            print('WiFi connection OK!')
            print('Network Config=',wifi.ifconfig())
            oled.text("WiFi connected!", 0,  0)
            oled.text("IP:{}".format(wifi.ifconfig()[0]), 0,  8)
            oled.show()
            time.sleep_ms(4000)
        else:
            print('WiFi connection Error')
    except Exception as e: print(e)
```

Рис. 3. Функция подключения к *Wi-Fi*

Во время выполнения функции на OLED-дисплее показывается статус подключения. Метод *oled*, импортируемый из библиотеки *ssd1306*, позволяет отображать на дисплеях любые изображения и текстовые сообщения [16].

Библиотека *ssd1306* работает на «низком» уровне и поддерживает вывод только английского текста, цифр и некоторых символов. Большинство выводимой информации приходится отображать вручную, конкретно указывая в коде, какой пиксель необходимо подсветить.

Во время подключения пользователь видит следующее сообщение, показанное на рис. 4, а после — сообщение об успешном подключении и полученный ip-адрес.

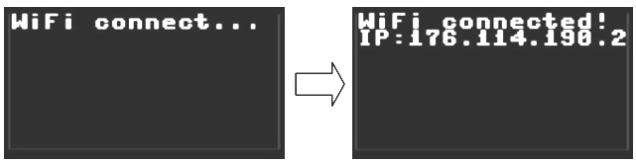

Рис. 4. Вывод об успешном подключении

Смарт-часы настраивают время автоматически при подключении к *Wi-Fi*. Для этого смарт-часы подключаются к RTC-серверу, используя URL. После чего в переменную *requests* записываются полученные данные, содержащие информацию о текущей дате и текущем времени. Проводится парсинг полученных данных с целью получения нужных [17–20]. Код, отвечающий за подключение к RTC-серверу, представлен на рис. 5.

```
def net_rtc():
    try:
        global check_con
        oled.fill(0)
        oled.text("Try to connect", 0,  0)
        oled.text("to RTC server", 0,  8)
        for i in range(3):
            time.sleep_ms(1)
            oled.text(".", 14*8+i*4, 8)
            oled.show()
        requests = urequests.get("http://worldtimeapi.org/api/tim
        print(requests.text)
        parsed = requests.json()
        datetime_str = str(parsed["datetime"])
        year = int(datetime_str[0:4])
        month = int(datetime_str[5:7])
        day = int(datetime_str[8:10])
        hour = int(datetime_str[11:13])
        minute = int(datetime_str[14:16])
        second = int(datetime_str[17:19])
        subsecond = int(round(int(datetime_str[20:26]) / 10000))

        rtc.datetime((year, month, day, 0, hour, minute, second,
        print(rtc.datetime())
        print("RTC updated\n")
        check_con = True
    except Exception as e: print(e)
```

Рис. 5. Функция подключения к RTC-серверу

После получения данных необходимо вывести их на дисплей с помощью библиотеки *ssd1306* и метода *oled* в любом удобном для пользователя виде. Функция, отвечающая за вывод времени на дисплей, представлена на рис. 6.

```
def display():
    datetime=rtc.datetime()
    oled.fill(0)
    oled.text("{0}/{1:0>2d}/{2:0>2d}<{3}>".format(datetime[0],
    oled.text("{0:0>2d}:{1:0>2d}:{2:0>2d}".format(datetime[4],
    oled.show()
```

Рис. 6. Функция вывода времени на дисплей

Примеры реализованных дисплеев показаны на рис. 7.

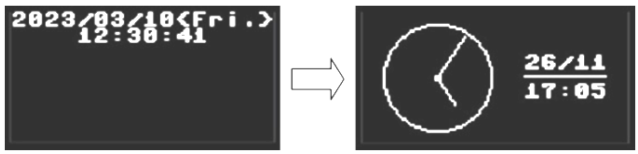

Рис. 7. Дисплеи

Стрелки аналоговых часов являются динамичными объектами и изменяют свое положение согласно реальному времени.

Для этого необходимо написать функции для отрисовки каждого положения стрелки, как показано на рис. 8, 9.

Необходимы также функции, которые будут отображать текущее положение стрелок и скрывать предыдущее. Код с функциями приведен на рис. 10.

Помимо времени, на смарт-часах, реализована функция показа погоды.

Текущая информация запрашивается через веб-сервис *WeatherAPI*. Для этого используется библиотека *urequests*, «облегченная» версия библиотеки *requeses*, созданная для работы на микроконтроллерах. Код представлен на рис. 11

```
def hand_11(x, l, k):
    for i in range(l):
        oled.pixel(65+x-i*2, 30-i*3, k)
        oled.pixel(64+x-i*2, 29-i*3, k)
        oled.pixel(64+x-i*2, 28-i*3, k)
    if l != 4:
        oled.pixel(53+x, 12, k)
    oled.show()

def hand_2(x, l, k):
    for i in range(l):
        oled.pixel(68+x+i*3, 31-i*2, k)
        oled.pixel(69+x+i*3, 30-i*2, k)
        oled.pixel(70+x+i*3, 30-i*2, k)
    if l != 4:
        oled.pixel(86+x, 19, k)
    oled.show()
```

Рис. 8. Пример отрисовки нескольких больших стрелок

```
def hand_0(x, l, k):
    if l ==6:
        l = 0
    for i in range(8+l*2.5, 31):
        oled.pixel(66 + x, int(i), k)
    oled.show()

def hand_3(x, l, k):
    if l ==6:
        l = 0
    for i in range(68, 91-l*2.5):
        oled.pixel(int(i) + x, 32, k)
    oled.show()

def hand_6(x, l, k):
    if l ==6:
        l = 0
    for i in range(34, 56-l*2.5):
        oled.pixel(66 + x, int(i), k)
    oled.show()

def hand_9(x, l, k):
    if l ==6:
        l = 0
    for i in range(42+l*2.5, 65):
        oled.pixel(int(i) + x, 32, k)
    oled.show()
```

Рис. 9. Пример отрисовки нескольких маленьких стрелок

```
def hand(x, l, k, t):
    oled.invert(0)
    if t == 0:
        hand_0(x, l, k)
    elif t == 1:
        hand_1(x, l, k)
    elif t == 2:
        hand_2(x, l, k)
    elif t == 3:
        hand_3(x, l, k)
    elif t == 4:
        hand_4(x, l, k)
    elif t == 5:
        hand_5(x, l, k)
    elif t == 6:
        hand_6(x, l, k)
    elif t == 7:
        hand_7(x, l, k)
    elif t == 8:
        hand_8(x, l, k)
    elif t == 9:
        hand_9(x, l, k)
    elif t == 10:
        hand_10(x, l, k)
    elif t == 11:
        hand_11(x, l, k)

def date_time(d1, d2, d3, d4, d5):
    oled.invert(0)
    for i in range(78, 118):
        for j in range(21, 44):
            oled.pixel(i, j, 0)
        if d5 == True:
            oled.pixel(i, 31, 1)
    if d5 == False:
        oled.text(" x5", 82, 35)
        oled.text("Test", 82, 21)
    else:
        oled.text("{0:0>2d}:{1:0>2d}".format(d3, d4)
        oled.text("{0:0>2d}/{1:0>2d}".format(d1, d2)
    oled.show()
```

Рис. 10. Функции динамического отображения стрелок

```
s_city = "Bratsk,RU"
city_id = 0
appid = "119bdd048752fae4e2eae76eeea06b55"
data = 0

def con_to_weather():
    global data
    try:
        oled.fill(0)
        oled.text("Connecting to", 0,  0)
        oled.text("WeatherAPI.com", 0,  8)
        for i in range(3):
            time.sleep(1)
            oled.text(".", 14*8+i*4, 8)
            oled.invert(0)
            oled.show()
        res = urequests.get(
            "http://api.openweathermap.org/data/2.5/find?q=Bratsk,RU&type=l
        data = res.json()
        cities = ["{} ({})".format(d['name'], d['sys']['country'])
                  for d in data['list']]
        print("city:", cities)
        city_id = data['list'][0]['id']
        print('city_id=', city_id)
        print(data['list'][0]['weather'][0]['description'])
        print(int(float(data['list'][0]['main']['temp']) - 273.15))
        print(int(float(data['list'][0]['main']['feels_like']) - 273.15))
    except Exception as e:
        print("Exception (find):", e)
        pass
```

Рис. 11. Функция получения информации о погоде

После получения данных на OLED-дисплей выводится температура, как она ощущается, и информация об осадках. Код программы представлен на рис. 12.

```
def deg_symbol(i, j):
    oled.pixel(i+1, j, 1)
    oled.pixel(i+2, j, 1)
    oled.pixel(i, j+1, 1)
    oled.pixel(i+3, j+1, 1)
    oled.pixel(i, j+2, 1)
    oled.pixel(i+3, j+2, 1)
    oled.pixel(i+1, j+3, 1)
    oled.pixel(i+2, j+3, 1)
    oled.text("C", i+5, j)

def display_weather():
    oled.fill(0)
    oled.text("tempreture:", 2, 8)
    oled.text(str(int(float(data['list'][0]['main']['temp'])-273.1
    deg_symbol(13*8, 8)
    oled.text("feels like:", 2, 24)
    oled.text(str(int(float(data['list'][0]['main']['feels_like'])
    deg_symbol(13*8, 24)
    oled.text(data['list'][0]['weather'][0]['description'],2,40)
    oled.show()
```

Рис. 12. Функция получения информации о погоде

Помимо данных о погоде, пользователь видит статус подключения к web-серверу. После успешного подключения на дисплее отобразится полученная информация, как показано на рис. 13.

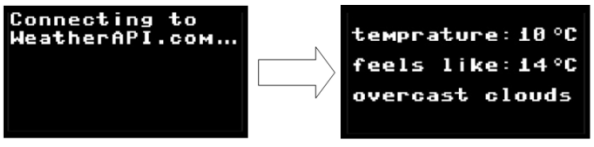

Рис. 13. Получение и вывод информации о погоде

Для получения нужной информации о стоимости каждой валюты необходимо подключиться к базе данных ЦБ РФ, представленной на специальном сайте. Функция, представленная на рис. 14, получает данные от сервера и проводит парсинг данных для получения нужной информации. В нашем случае это информация о трех основных валютах.

```
def show_currency():
    data = urequests.get('https://www.cbr-xml-daily.ru/daily_json.js')
    oled.fill(0)
    date = str(data['Date'])
    oled.text('Date:'+date[8:10]+"/"+date[5:7]+"/"+date[0:4], 0,  3)

    oled.text('1',     4, 19)
    oled.text('EUR',  16, 19)
    oled.text('=',    44, 19)
    oled.text(str(data['Valute']['EUR']['Value'])[0:4], 56, 19)
    oled.text('RUB',  92, 19)

    oled.text('1',     4, 35)
    oled.text('USD',  16, 35)
    oled.text('=',    44, 35)
    oled.text(str(data['Valute']['USD']['Value'])[0:4], 56, 35)
    oled.text('RUB',  92, 35)

    oled.text('1',     4, 51)
    oled.text('CNY',  16, 51)
    oled.text('=',    44, 51)
    oled.text(str(data['Valute']['CNY']['Value'])[0:4], 56, 51)
    oled.text('RUB',  92, 51)
    oled.show()
```

Рис. 14. Получение и вывод информации о валютах

Прежде чем вывести необходимую информацию, пользователь увидит сообщение о подключении к серверу. На рис. 15 представлена функция, отвечающая за вывод сообщения о подключении к серверу, а впоследствии — вывод текущей информации о валютах.

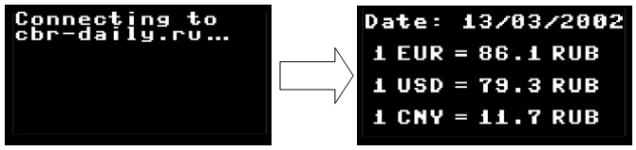

Рис. 15. Функция получения информации о валютах

При запуске часов первым компилируется файл main.py. Этот же файл вызывает все последующие функции. В цикле «*WhileTrue:*» написан код, отвечающий за отображение дисплеев. При нажатии на кнопку в переменную *logic_state* записывается значение 1, в ином случае — значение 0. В таком случае есть возможность вызывать только 2 дисплея, поэтому был написан код, отвечающий за переключение любого количества дисплеев, используя всего одну кнопку. Код представлен на рис. 16.

```
while True:
    logic_state = push_button.value()
    if logic_state == True and check == True:
        show_display += 1
        check = False
    elif logic_state == False and check == False:
        check = True
```

Рис. 16. Код, отвечающий за переключение дисплеев

Переменная *show_display* содержит значение номера дисплея, который должен быть отображен, и увеличивается на единицу при нажатии кнопки.

Вывод каждого из дисплеев представляет собой условие «*If*» в зависимости от значения переменной *show_display*. Код программы продемонстрирован на рис. 17.

```
# Первый дисплей
if show_display % n == 0:
    check_weather = False
    Clock.display()

# Второй дисплей
elif show_display % n == 1:
    Clock.display1()

# Третий дисплей
elif show_display % n == 2:
    if check_weather == False:
        check_weather = True
        Weather.con_to_weather()
    Weather.display_weather()

# Четвертый дисплей
elif show_display % n == 3:
    if check_analog == False:
        check_analog = True
        Analog.draw_clock(pos)
    Clock.rtc_update()
    if prev_h != (Clock.datetime[4]%12):
        Analog.hand(pos, 4, 0, prev_h)
        Analog.hand(pos, 4, 1, (Clock.datetime[4]))
        prev_h = Clock.datetime[4]
    if prev_m != (Clock.datetime[5]//5):
        Analog.date_time(Clock.datetime[2], Clock.date
        Analog.hand(pos, 6, 0, prev_m)
        Analog.hand(pos, 6, 1, (Clock.datetime[5]//5))
        Analog.hand(pos, 4, 1, (Clock.datetime[4]%12))
        prev_m = Clock.datetime[5]//5
# Пятый дисплей
elif show_display % n == 4:
    currency_check = False
    if test_analog == False:
        test_analog = True
        Analog.draw_clock(pos)
        Analog.hand(pos, 6, 0, prev_m)
        Analog.hand(pos, 4, 0, prev_h)
    test_b += 1
    if test_b == 12:
        test_a += 1
        test_b = 0
    if test_a == 12:
        test_a = 0
    if prev_h != test_a:
        Analog.hand(pos, 4, 0, prev_h)
        Analog.hand(pos, 4, 1, test_a)
        Analog.date_time(Clock.datetime[2], Clock.datet
        prev_h = test_a
    if prev_m != test_b:
        Analog.hand(pos, 6, 0, prev_m)
        Analog.hand(pos, 6, 1, test_b)
        Analog.hand(pos, 4, 1, test_a)
        prev_m = test_b

# Шестой дисплей
elif show_display % n == 5:
    if currency_check == False:
        currency_check = True
        Currency.start_currency()
    Currency.show_currency()
time.sleep_ms(200)
```

Рис. 17. Условия отображения всех дисплеев

Кнопка опрашивается с частотой каждые 200 мс. Чтобы избежать хаотичного переключения дисплеев при зажатии кнопки, была добавлена переменная *check*. Поэтому даже если зажать кнопку, то переключится только один дисплей. Чтобы продолжить, необходимо отпустить кнопку и нажать ее еще раз. Это предотвращает случайные переключения дисплеев.

На рис. 18 показано, какие файлы с расширением «ру» были загружены в память микроконтроллера ESP32, а именно:

1. Файлы библиотек;
   1.1. urequests
   1.2. ssd1306
2. Системные файлы, загружающиеся вместе с установкой *Python*;
   2.1. main
   2.2. boot
3. Пользовательские функции;
   3.1. Analog
   3.2. Clock
   3.3. Currency
   3.4. Weather

Крайне важно учитывать размер готового проекта из-за ограничения в памяти ESP32, которое составляет 448 КБ.

Рис. 18. Загруженные на микроконтроллер файлы

Реализованный проект занимает 286 Кб памяти из 448 доступных, а именно:

1. *Python* занимает 256 Кб.
2. Пользовательские файлы и библиотеки занимают 30 Кб.

Свободную память можно использовать для добавления дополнительного функционала смарт-часов.

Заключение. Разработанный проект показывает, что язык программирования *Python* может быть использован для программирования «низкоуровневых» проектов, например, при создании смарт-часов.

Численность микроконтроллеров активно растет, а вместе с ней и потребность в их программировании. *Python* упрощает разработку программного обеспечения и адаптирован для работы с различными микроконтроллерами.

Современные смарт-часы представляют собой небольшую плату с чипом, на котором реализованы пользовательские функции. Благодаря своим компактным размерам выбранный микроконтроллер ESP32 может быть применен практически в любой сфере, в том числе и при разработке смарт-часов.

В статье приведены основные функции и методы, демонстрирующие функциональность языка. Также были написаны пользовательские функции, расширяющие удобство использования рассматриваемого гаджета:

1. На «низком» уровне:
   – вывод информации на дисплей;
   – считывание сигналов с внешних модулей.
2. На «высоком» уровне:
   – удаленное подключение к серверам;
   – подключение к *Wi-Fi*;
   – парсинг данных.

Литература

1. Рудова И.Е., Гришин Р.В., Селютина А.А. Программирование микроконтроллеров в среде разработки LabVIEW // Интерэкспо Гео-Сибирь. 2018. Т. 2, № 8. С. 10-13.
2. Дунайцев П.С. Программно-технические средства программирования и отладки микропроцессорной РЗА на основе микроконтроллеров STM32 // Электроэнергетика. Энергия-2019: материалы четырнадцатой Всерос. науч.-технической конф. студентов, аспирантов и молодых ученых (2-4 апр. 2019 г.). Иваново, 2019. С. 53.
3. Исаев О.В., Изъюров Д.Е. Разработка и программирование альтернативного мобильного контрольного устройства СЭМПЛ на примере универсального микроконтроллера // Техника и безопасность объектов уголовно-исполнительной системы: сб. материалов Междунар. науч.-практической конф. (17-18 мая 2023 г.). Воронеж, 2023. С. 74-77.
4. Викулов Ю.Н., Габрусев В.В., Крыжко С.М., Кузякин Г.М., Ярошик А.С. Программа для программирования встроенной флеш-памяти микроконтроллера и памяти программируемой логической интегральной схемы: свид. о регистрации программы для ЭВМ RU 2020619190, 13.08.2020; заявл. 03.08.2020 № 2020618528.
5. Уварова Н.А. Программа для визуального программирования аппаратно-программного комплекса на базе микроконтроллера K1986BE92QI: свид. о регистрации программы для ЭВМ RU 2021665256, 22.09.2021; заявл. 08.09.2021 № 2021664042.
6. Архипов А.В., Табаков А.А., Тягун Н.В. Функциональное программное обеспечение для программирования микроконтроллеров 1986BE1T и 1888TX018, ПЛИС1 И ПЛИС2: свид. о регистрации программы для ЭВМ RU 2022615264, 30.03.2022; заявл. 21.03.2022 № 2022614232.
7. Палкин Г.А., Хазагаров А.А., Черных Д.А. Программирование микроконтроллера для визуализации программы «Электронные часы» // Кулагинские чтения: техника и технологии производственных

процессов: сб. ст. XVI Междунар. науч.-практической конф. (28-30 нояб. 2016 г.). Чита, 2016. С. 130-132.
8. Тюрин С.Ф., Ковыляев Д.А., Данилова Е.Ю., Городилов А.Ю. Изучение программирования микроконтроллеров в САПР PROTEUS // Вестн. Пермского ун-та. Математика. Механика. Информатика. 2021. № 2 (53). С. 69-74.
9. Родин В.В. Разработка средств программирования и отладки микроконтроллеров // Проблемы и перспективы развития отечественной светотехники, электротехники и энергетики: материалы XII Всерос. науч.-технической конф. с междунар. участием (28-29 мая 2015 г.). Саранск, 2015. С. 467-470.
10. Петров Д.А., Землянухин П.А., Сердюков П.С., Забабурина Т.Н., Бедикян Д.Р. Создание беспроводного модуля управления системы «Умный дом» на основе микроконтроллера ESP32 // Современные технологии: проблемы инновационного развития: сб. ст. Междунар. науч.-практической конф. (4 дек. 2019 г.). Петрозаводск, 2019. С. 171-180.
11. Okonkwo C.W., Ade-Ibijola A. Python-bot: a chatbot for teaching python programming // Engineering Letters. 2021. V. 29, № 1. P. 25-34.
12. Жорняк А.Г., Морозова Т.А. Специализированный дистрибутив Python (x,y) языка программирования Python для научных и инженерных вычислений // Науч.-технический вестн. Поволжья. 2022. № 7. С. 39-42.
13. Ликсина Е.В., Колосов Е.Д. C Sharp или Python? // Сборники конф. НИЦ Социосфера. 2020. № 11. С. 173-175.
14. Салихова Г.Л., Потапова О.Н. Язык программирования Python для решения технических задач в нефтегазовом вузе // Науч.-технический вестн. Поволжья. 2023. № 8. С. 97-100.
15. Кондратюк А.Д., Ерохов П.О. Совмещенная система дистанционного мониторинга параметров помещений на базе микрокотроллера ESP32 и протокола MQTT // Информационные системы и технологии: сб. тезисов докл. 59-й науч. конф. аспирантов, магистрантов и студентов (17-23 апр. 2023 г.). Минск, 2023. С. 16-18.
16. Пирматов А.З., Камалов С.С., Абдукадыр К.А., Сүйөркул К.Н. Объектно-ориентированное программирование на языке PYTHON // Вестн. Жалал-Абадского гос. ун-та. 2022. № 4 (53). С. 22-28.
17. Мазур Е.В., Бабешко В.Н. Парсинг как источник получения данных для анализа // Перспективное развитие науки, техники и технологий: сб. науч. ст. 9-ой Междунар. науч.-практической конф. (1 нояб. 2019 г.). Курск, 2019. С. 161-164.
18. Байгушкина Е.А. Исследование алгоритма парсинга структур баз данных // Энигма. 2021. № 29-2. С. 132-136.
19. Боковиков С.А. Извлечение данных о погоде с помощью парсинга веб-страниц в Python // Современные науч. исследования и инновации: электрон. науч.-практический журнал. 2024. № 1 (153). URL: https://web.snauka.ru/issues/2024/01/101382 (дата обращения: 08.09.2024).
20. Ефремова Д.В. Поправки к федеральному закону «О персональных данных»: запрет парсинга и ликвидация понятия «общедоступные данные» // Сравнительно-правовые аспекты правоотношений гражданского оборота в современном мире: сб. ст. Междунар. науч. юридического форума памяти проф. В.К. Пучинского (15 окт. 2021 г.). М., 2021. С. 431-435.

References

1. Rudova I.E., Grishin R.V., Selyutina A.A. Programming microcontrollers in the LabVIEW development environment // Interekspo Geo-Sibir'. 2018. V. 2, № 8. P. 10-13.
2. Dunajcev P.S. Software and hardware programming and debugging tools for microprocessor-based RPA based on STM32 microcontrollers // Elektroenergetika. Energiya-2019: materialy chetyrnadcatoj Vseros. nauch.-tekhnicheskoj konf. studentov, aspirantov i molodyh uchenyh (2-4 apr. 2019 g.). Ivanovo, 2019. P. 53.
3. Isaev O.V., Iz"yurov D.E. Development and programming of an alternative mobile control device SAMPLE using the example of a universal microcontroller // Tekhnika i bezopasnost' ob"ektov ugolovno-ispolnitel'noj sistemy: sb. materialov Mezhdunar. nauch.-prakticheskoj konf. (17-18 maya 2023 g.). Voronezh, 2023. P. 74-77.
4. Vikulov Yu.N., Gabrusev V.V., Kryzhko S.M., Kuzyakin G.M., Yaroshik A.S. A program for programming the built-in flash memory of a microcontroller and the memory of a programmable logic integrated circuit: svid. o registracii programmy dlya EVM RU 2020619190, 13.08.2020; zayavl. 03.08.2020 № 2020618528.
5. Uvarova N.A. A program for visual programming of a hardware and software complex based on the K1986BE92QI microcontroller: svid. o registracii programmy dlya EVM RU 2021665256, 22.09.2021; zayavl. 08.09.2021 № 2021664042.
6. Arhipov A.V., Tabakov A.A., Tyagun N.V. Functional software for programming microcontrollers 1986VE1T and 1888TH018, PLIZ 1 and PLIS2: svid. o registracii programmy dlya EVM RU 2022615264, 30.03.2022; zayavl. 21.03.2022 № 2022614232.
7. Palkin G.A., Hazagarov A.A., Chernyh D.A. Programming of the microcontroller for visualization of the Electronic Clock program // Kulaginskie chteniya: tekhnika i tekhnologii proizvodstvennyh processov: sb. st. XVI Mezhdunar. nauch.-prakticheskoj konf. (28-30 noyab. 2016 g.). Chita, 2016. P. 130-132.
8. Tyurin S.F., Kovylyaev D.A., Danilova E.Yu., Gorodilov A.Yu. Learning how to program microcontrollers in PROTEUS CAD // Bulletin of Perm state university. Mathematics. Mechanics. Information science. 2021. № 2 (53). P. 69-74.
9. Rodin V.V. Development of microcontroller programming and debugging tools // Problemy i perspektivy razvitiya otechestvennoj svetotekhniki, elektrotekhniki i energetiki: materialy XII Vseros. nauch.-tekhnicheskoj konf. s mezhdunar. uchastiem (28-29 maya 2015 g.). Saransk, 2015. P. 467-470.
10. Petrov D.A., Zemlyanuhin P.A., Serdyukov P.S., Zababurina T.N., Bedikyan D.R. Creation of a wireless control module of the Smart Home system based on the ESP32 microcontroller // Sovremennye tekhnologii: problemy innovacionnogo razvitiya: sb. st. Mezhdunar.

nauch.-prakticheskoj konf. (4 dek. 2019 g.). Petrozavodsk, 2019. P. 171-180.
11. Okonkwo C.W., Ade-Ibijola A. Python-bot: a chatbot for teaching python programming // Engineering Letters. 2021. V. 29, № 1. P. 25-34.
12. Zhornyak A.G., Morozova T.A. Specialized Python (x, y) distribution of the Python programming language for scientific and engineering computing // Scientific and Technical Volga region Bulletin. 2022. № 7. P. 39-42.
13. Liksina E.V., Kolosov E.D. With Sharp or Python? // Sborniki konf. NIC Sociosfera. 2020. № 11. P. 173-175.
14. Salihova G.L., Potapova O.N. Python programming language for solving technical problems in an oil and gas university // Scientific and Technical Volga region Bulletin. 2023. № 8. P. 97-100.
15. Kondratyuk A.D., Erohov P.O. A combined system for remote monitoring of room parameters based on the ESP 32 microcontroller and the MQTT protocol // Informacionnye sistemy i tekhnologii: sb. tezisov dokl. 59-j nauch. konf. aspirantov, magistrantov i studentov (17-23 apr. 2023 g.). Minsk, 2023. P. 16-18.
16. Pirmatov A.Z., Kamalov S.S., Abdukadyr K.A., Syjərkul K.N. Object-oriented programming in PYTHON // Bulletin Jalal-Abad state university. 2022. № 4 (53). P. 22-28.
17. Mazur E.V., Babeshko V.N. Parsing as a source of data for analysis // Perspektivnoe razvitie nauki, tekhniki i tekhnologij: sb. nauch. st. 9-oj Mezhdunar. nauch.-prakticheskoj konf. (1 noyab. 2019 g.). Kursk, 2019. P. 161-164.
18. Bajgushkina E.A. Investigation of the algorithm for parsing database structures // Enigma. 2021. № 29-2. P. 132-136.
19. Bokovikov S.A. Extracting weather data using web page parsing in Python // Modern scientific researches and innovations. 2024. № 1 (153). URL: https://web.snauka.ru/issues/2024/01/101382 (data obrashcheniya: 08.09.2024).
20. Efremova D.V. Amendments to the Federal law "On Personal Data": prohibition of parsing and elimination of the concept of "publicly available data" // Sravnitel'no-pravovye aspekty pravootnoshenij grazhdanskogo oborota v sovremennom mire: sb. st. Mezhdunar. nauch. yuridicheskogo foruma pamyati prof. V.K. Puchinskogo (15 okt. 2021 g.). M., 2021. P. 431-435.