Проект 2.3

Telegram bot

Теоретическая часть:

API – Application Programming Interface, что значит программный интерфейс приложения. В контексте API слово «приложение» относится к любому ПО с определенной функцией. Интерфейс можно рассматривать как сервисный контракт между двумя приложениями. Этот контракт определяет, как они взаимодействуют друг с другом, используя запросы и ответы. Документация API содержит информацию о том, как разработчики должны структурировать эти запросы и ответы.

Как работают API?

Архитектура API обычно объясняется с точки зрения клиента и сервера. Приложение, отправляющее запрос, называется клиентом, а приложение, отправляющее ответ, называется сервером. Итак, в примере с погодой база данных службы – это сервер, а мобильное приложение – это клиент.

Существует четыре различных способа работы API в зависимости от того, когда и почему они были созданы.

Telegram API vs Telegram Bot API

Телеграм использует собственный протокол шифрования MTProto. MTProto API (он же Telegram API) — это API, через который ваше приложение Телеграм связывается с сервером. Telegram API полностью открыт, так что любой разработчик может написать свой клиент мессенджера.

Для написания ботов был создан Telegram Bot API — надстройка над Telegram API. Перевод с официального сайта:

Чтобы использовать Bot API, вам не нужно ничего знать о том, как работает протокол шифрования MTProto — наш вспомогательный сервер будет сам обрабатывать все шифрование и связь с Telegram API. Вы соединяетесь с сервером через простой HTTPS-интерфейс, который предоставляет простую версию Telegram API.

Среди упрощений Bot API: работа через вебхуки, упрощенная разметка сообщений и прочее. Почему-то мало кто знает о том, что Телеграм боты могут работать напрямую через Telegram API. Более того, таким образом можно даже обойти некоторые ограничения, которые даёт Bot API. Об авторизации ботов через Telegram API в официальной документации. Вся информация ниже будет по умолчанию относиться и к Bot API, и к Telegram

API. О различиях я буду упоминать. От некоторых ограничений Bot API можно избавиться с помощью локального сервера, об этом в конце статьи.

На чём пишут Телеграм-ботоы

Бот должен уметь отправлять запросы Телеграм-серверу и получать от него апдейты (updates, обновления).

Как получать апдейты в Bot API

Получать апдейты можно одним из двух способов:

- Поллинг — просто регулярно отправлять запрос к серверу Телеграма для получения обновлений,
- Вебхук — сделать так, чтобы Телеграм сам отправлял запросы по нужному URL.

Конечно, удобнее использовать библиотеки, чем делать http-запросы «руками».

Если вы попробуете загуглить, как написать Телеграм-бота на Python, вам предложат воспользоваться библиотеками python-telegram-bot и telebot. Но не стоит.

Ну, если вы только хотите познакомиться с разработкой ботов и написать своего hello-world-бота, то можете, конечно использовать и их. Но эти библиотеки могут далеко не всё. Среди разработчиков ботов лучшей библиотекой для ботов на Python считается aiogram. Она асинхронная, использует декораторы и содержит удобные инструменты для разработки. Ещё был хороший Rocketgram, но он давно не обновлялся.

Также ботов часто пишут на JavaScript, для этого обычно используется Telegraf. Библиотеки есть и для многих других языков, но используют их реже.

Если же вы хотите использовать Telegram API, то можете воспользоваться Python’овскими Telethon и Pyrogram.

На чём пишут Телеграм-ботов

Бот должен уметь отправлять запросы Телеграм-серверу и получать от него апдейты (updates, обновления).

Практическая часть:

Программа (шаблон для будущего бота):

```
#define WIFI_SSID "**********"
#define WIFI_PASS "**********"
#define BOT_TOKEN "**********************************************"

#include <FastBot.h>
FastBot bot(BOT_TOKEN);

void setup() {
connectWiFi();
bot.attach(newMsg);
}

void newMsg(FB_msg& msg) {
Serial.print(msg.chatID); // ID чата
Serial.print(", ");
Serial.print(msg.username); // логин
Serial.print(", ");
Serial.println(msg.text); // текст
}

void loop() {
bot.tick();
}

void connectWiFi() {
delay(2000);
Serial.begin(115200);
Serial.println();
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
if (millis() > 15000) ESP.restart();
}
Serial.println("Connected");
}
```

В данном блоке кода находятся данные: Wi-Fi, пароль от Wi-Fi и токен от telegram bot(a)

```
#define WIFI_SSID "**********"
#define WIFI_PASS "**********"
#define BOT_TOKEN "**********"
```

```
#include <FastBot.h>
FastBot bot(BOT_TOKEN);
```

Далее в void setup() идёт подключение к Wi-Fi с помощью функции "connectWiFi()", а также инициализация бота

```
void setup() {
connectWiFi();
bot.attach(newMsg);
}
```

Функция вывода информации в последовательный COM-порт о новом сообщении

```
void newMsg(FB_msg& msg) {
Serial.print(msg.chatID); // ID чата
Serial.print(", ");
Serial.print(msg.username); // логин
Serial.print(", ");
Serial.println(msg.text); // текст
}
```

"Цикл" loop (бесконечный цикл) (в нём как раз и выполняется основная программа)

```
void loop() {
bot.tick();
}
```

И конечно же функция подключения к Wi-Fi

```
void connectWiFi() {
delay(2000);
Serial.begin(115200);
Serial.println();
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
if (millis() > 15000) ESP.restart();
}
Serial.println("Connected");
}
```

Создание telegram bot(a) в Telegram

А теперь мы подошли к самой интересной части в работе с telegram - это создание самого бота, т.е. чата, куда будут поступать наши команды, а ESP32 (с уже загруженной программой) будет их анализировать.
Своего бота я создавал с помощью другого telegram bot(a) "@BotFather". Если вы создаёте (или уже создали) бота в telegram другим способом, то ничего страшного, главное - это токен бота, который надо будет вписать в программу.

Создание бота:

- Заходим в Telegram, находим через поиск бота @BotFather и открываем чат с ним, нажав внизу экрана кнопку *Start*.

- Запускаем процедуру регистрации нового бота, для чего набираем в чате с @BotFather команду */newbot*

- Далее, следуя инструкциям, вводим имя бота. Это просто его название, то, как он будет отображаться в списке контактов. Это имя потом можно будет изменить с помощью команды /setname в чате с BotFather.

- Теперь нужно ввести username. Это уже строковый идентификатор, используемый для ссылок на бота. Его нельзя менять, он должен быть уникальным и должен обязательно заканчиваться на bot. Если вы введёте некорректный username, или такой, который уже есть в системе, BotFather сообщит Вам об этом и предложит ввести другой.

- Если всё пройдёт успешно, BotFather напишет: «Done! Congratulations on your new bot…», и далее сообщит ссылку на аккаунт вашего нового бота t.me/*anynamebot*, а также токен для авторизации оператора: «Use this token to access the HTTP API: *anynamebottoken*«.

Работа с telegram bot(ом) и ESP32:

- После того как вы получили токен бота, Вы должны вписать его в поле "BOT_TOKEN" в скетче (программе) в Arduino IDE

- Теперь загружаем скетч (программу в ESP32)(Если вы никогда не загружали скетч в ESP32, то скоро по этой теме выйдет моя статья)

- Далее, в telegram находим своего бота, пишем "/start" и отправляем любое сообщение. В COM-порт должны прийти данные о сообщении и пользователе, который использует вашего бота.

Информация от бота: "ID чата, логин, текст сообщения”
Разберём пример скетча для ответа на слово "Hello"!

Программа для ответа на слово "Hello":

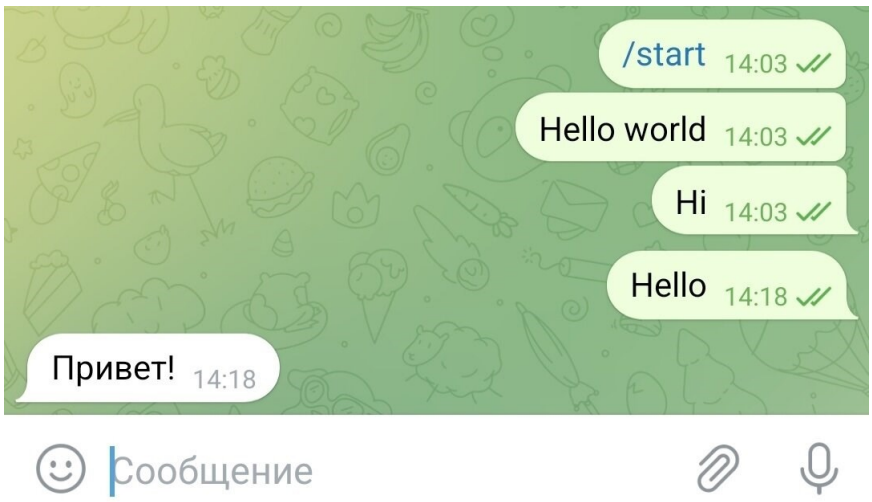
```
#define WIFI_SSID "**********"
#define WIFI_PASS "**********"
#define BOT_TOKEN "**********************************************"

#include <FastBot.h>
FastBot bot(BOT_TOKEN);

void setup() {
connectWiFi();
bot.attach(newMsg);
}

void newMsg(FB_msg& msg) {
String tem = "("+msg.chatID+", "+msg.username+", "+msg.text+")";
if(msg.text == "Hello"){
String otp = "Привет!";
bot.sendMessage(otp, msg.chatID);
Serial.println(tem+otp);
}
Serial.println(tem);
}

void loop() {
bot.tick();
}

void connectWiFi() {
delay(2000);
```

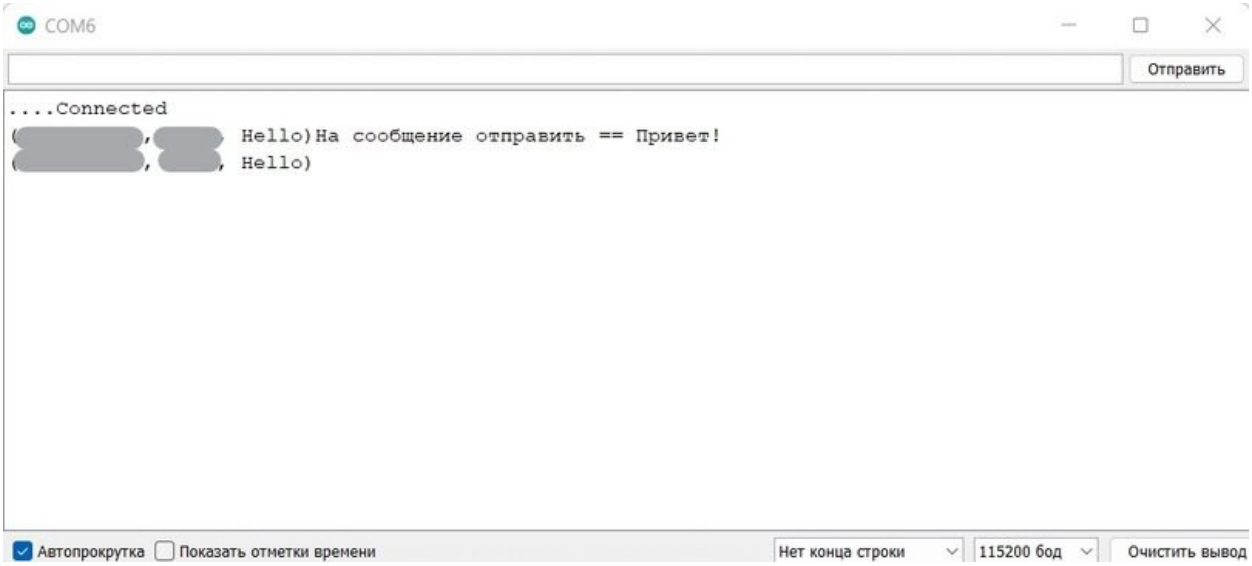
```
Serial.begin(115200);
Serial.println();
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
if (millis() > 15000) ESP.restart();
}
Serial.println("Connected");
}
```

В данном коде изменилась функция "void newMsg(FB_msg& msg)":

```
void newMsg(FB_msg& msg) {
String tem = "("+msg.chatID+", "+msg.username+", "+msg.text+")";
if(msg.text == "Hello"){
String otp = "Привет!";
bot.sendMessage(otp, msg.chatID);
otp = "На сообщение отправить == "+otp;
Serial.println(tem+otp);
}
Serial.println(tem);
}
```

Теперь бот отвечает на слово "Hello" словом "Привет!"





Усложняем задачу:

1) Добавить возможность ответа на сообщения «Как дела?» и другие формы приветствия. Бот должен рандомно выбирать фразы из списка заявленных.

2) Добавить возможность управления ботом по кнопкам бота.

3) Добавить в схему датчик звука ky – 038 и датчик света ky -018.

4) По кнопке звук вывести уровень звука, по кнопке свет вывести уровень освещенности

**

5) Создание интерфейса напоминаний. Пользователь должен вводить текст напоминания. Пользователь должен вводить частоту напоминания. Программа каждый промежуток времени должна выводить сообщение пользователю.