Vesi ja vaahto[1]

Работа на всероссийский конкурс научных и инженерных проектов «Балтийский научно - инженерный конкурс» направления Робототехника «Устройство для отладки пожарных роботов компании “«ЭФЕР» - завод пожарных роботов”»

14 «января», 2020, Петрозаводск
Детский технопарк «Кванториум Сампо»
Смирнов Сергей - капитан
Диана Мелихова – соавтор
Алексей Валерьевич Панфилов – научный руководитель проекта

[1] Vesi ja vaahto – (Финск.) вода и пена

Содержание:

Оглавление

Введение:

Проблема:

на заводе пожарных роботов для отладки и тестирования роботов используется классическое приложение АБМИ.90006, открываемое на ноутбуке и через USB - RS485 подключается к общей шине робота. В рабочих условия, и (особенно) если робот находится на потолке/возвышении, то тянуть провод очень проблематично. Команде нашего кванториума нужно было разработать устройство, которое может подключаться к роботу и написать приложение на смартфон и реализовывающее часть интерфейса АБМИ.90006.

Решение:

Нашей командой было принято решение о передаче данных по Wi-Fi и использовании ESP-01 WiFi Module, который поднимает TCP – сервер, а в качестве приёмника – смартфон с ОС Android (версия не ниже 7.1)[2]

[2] На момент написания работы – Реализовано общение между Wi-Fi модулем и телефоном, отправление пакетов в систему

Физика

Внутреннее устройство системы модуля:

У модуля ESP-01 есть внутренний процессор ESP8266 с возможностью обновления прошивки, на который и устанавливается основная программа

В системе два стабилизатора питания – 3.3 и 5В – первый для питания ESP-01 и второй для питания UART TTl to RS485 module, поскольку сигнальные линии работают на 5В

Приемопередатчик на основе микросхемы MAX485 преобразует сигналы TTL в стандарт RS485 и обратно и используется для подключения устройств на основе Arduino к шине RS485.

Шестиканальный двунаправленный конвертер-преобразователь работает в диапазоне напряжений от 1.8 до 6 В. Выводы A2-A5 предназначены для включения устройств с 3.3 В логикой, а A8-A11 – с 5 В логикой. На питающий вход Vin1 следует подавать напряжение 5В, а на Vin2 – другое.

Схемы:

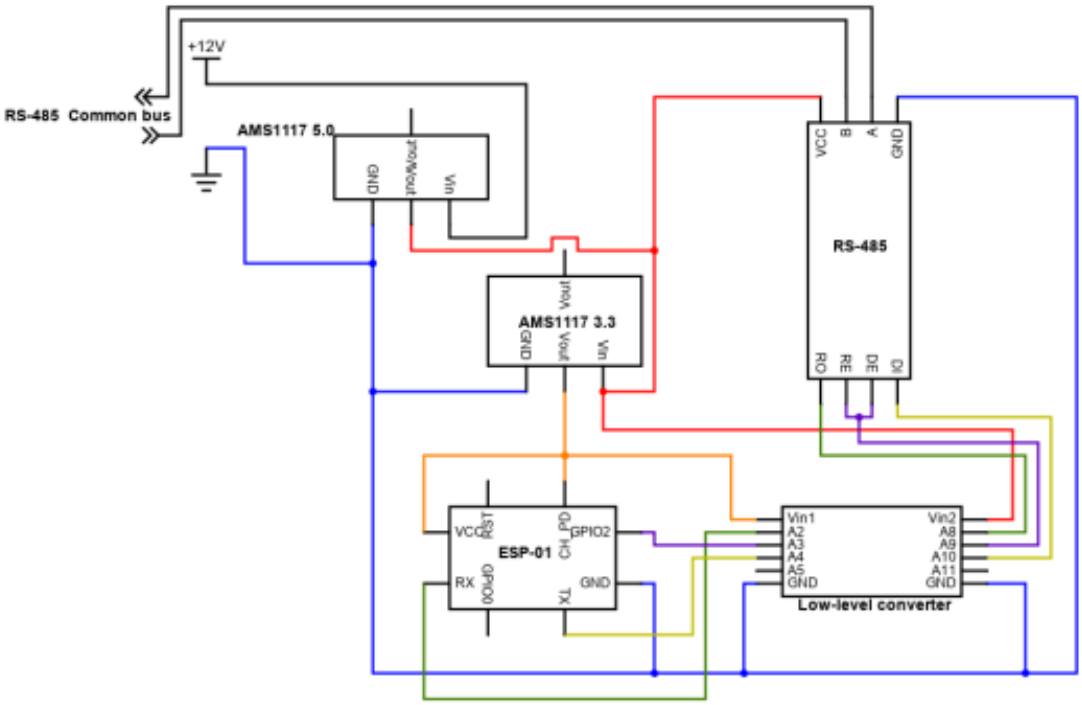


Рисунок 1. Принципиальная схема устройства – создано на Интернет-сайте Sheme – it (См. Список использованной литературы[10])

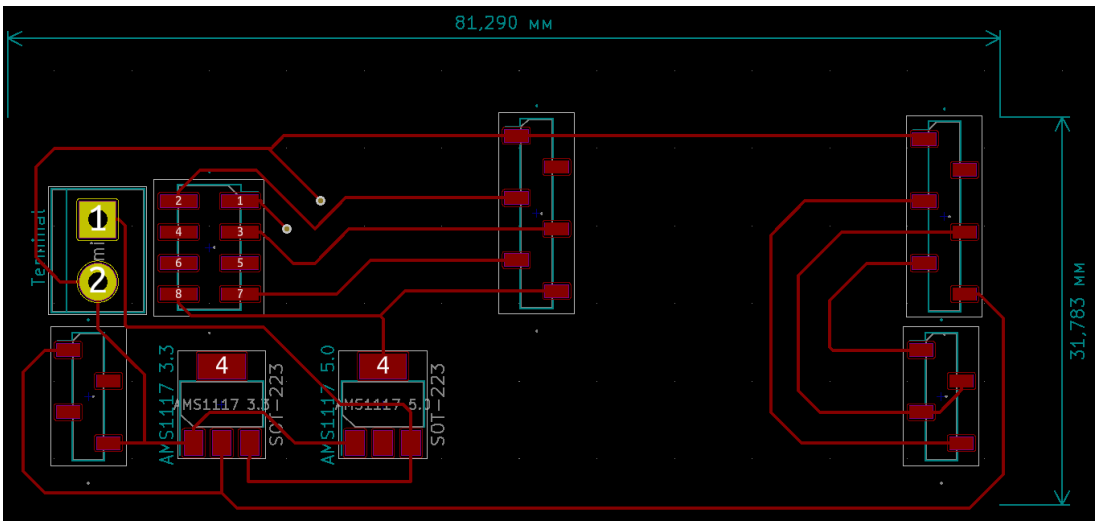


Рисунок 2. Разметка для травления выполнена в KiCadEDA(См. Список использованной литературы[11])

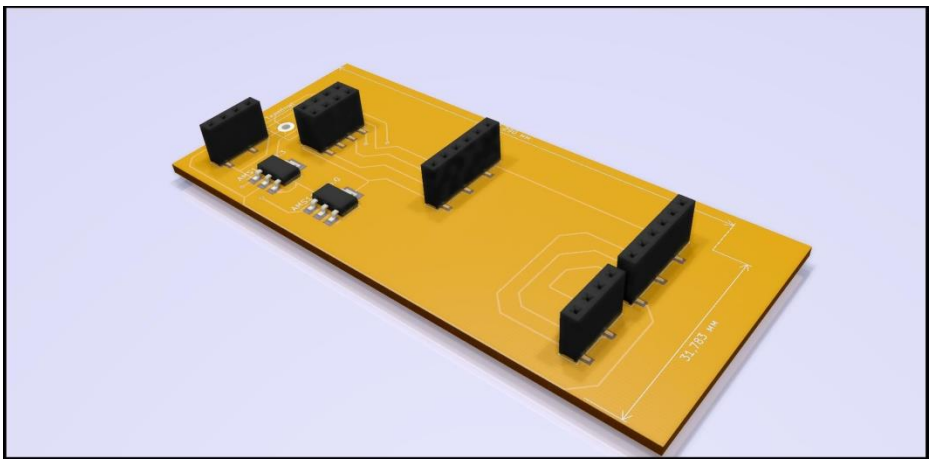


Рисунок 3. 3D-вид платы – 3D модель создана в программе KicadEDA (См. Список использованной литературы[11])

Поведение

Устройство:

Устройство должно:

- Обращаться к нужной подсистеме робота и отправлять задание на выполнение или запрашивать значения с датчиков (для чего используется протокол, используемый в самом роботе, без изменений в нём). В соответствии с протоколом, у каждого модуля в системе есть ID, наш модуль обращается к нужной подсистеме, устанавливает соединение и запрашивает/отправляет данные(См. Приложение 1 и Приложение 2)

- Поднимать TCP-сервер для общения с телефоном; используется стандартный TCP-сервер (См. Приложение 3) (См. список использованной литературы [14])

Таблица 1. Формат пакета системы "Модуль-смартфон"

Название	DST	SRC
Размер, байт	1	0..256
Назначение	Адрес получателя кадра	Данные кадра

- Для установки параметров с телефона на модуль используется Таблица 1. Формат пакета системы “Модуль – смартфон”, формат пакета такой же, как и формат пакета модуля и таким образом, обработка интерфейса взаимодействия с человеком и значений, полученных с датчиков, ложится на смартфон, что сделано с целью сэкономить малое количество оперативной памяти ESP-01

- Так как все части робота общаются по общей шине протокола RS-485 Modbus RTU, то модуль должен уметь общаться по протоколу робота(См. простейший Modbus-client на Arduino считывающий значение с кнопки и отправляющий его на master)(См. список использованной литературы [15])

Смартфон

- Должен осуществлять взаимодействие человека с интерфейсом и выводить значения, полученные с робота, для чего телефон подключается к сети Wi-Fi “FR”, и начинается передача пакетов (См. Приложение 4)

- Подключение к модулю происходит по Wi-Fi, и поэтому, используя стандартные способы передачи данных в Android Studio, применяется технология Web-sockets (См. описание технологии Web-soccets(Java)). Каждый из процессов, происходящих в Soccet, является независимым от основного потока программы (наследуется от AsincTask) и выполняется

независимо, что происходит каждый раз для любого обращения к классу ClientLoop и вызывает новый поток(См. Приложение 5) (См. список использованной литературы [5][6][7])

Отладка

- Так как отладку нужно проводить всей системы, то модуль должен выполнять все функции обычной системы в составе робота – отвечать на запросы и широковещательные пакеты, вот ответ модуля на GetStatus(стандартный запрос параметров в системе робота)(структура C++):

```
struct WiFiModule{
    uint32_t status; //Биты флагов состояния модуля в целом
    int16_t users; //Кол-во пользователей на WiFi сети
    int16_t IP;//Адресс модуля в созданной им сети
    char password;//Пароль от сети
    char ssid;//Имя сети
};
```

Таблица 2. Биты флагов status(вторые 16, специфичные для каждой системы)

Обозначение	Номер бита	Назначение
UpWiFi	16	WiFi сеть поднята

- Возможный запрос SetParam(стандартный запрос на установку параметров):

```
struct WiFiModule(){
    uint16_t id;
    char ssid;//На выбор
    char password;
    int16_t IP;
    int16_t users;
};
```

Возможные реализации:

1. HC-05/06 и другие Bluetoooth-модули - не используется по причине неприемлемой библиотеки и реализации в Android Studio, возможные реализации (См. список использованной литературы [9]):
    a. Поднимается Bluetooth-сервер и ожидается подключение телефона – работа в режиме сервера
    b. Работа в режиме BTL 4.0 - ожидается поднятия сервера на телефоне - работа в режиме клиента
2. WeMos Arduino, ArduinoWiFi rev2, NodeMCU v3, Arduino+WiFi-sheeld/ESP8266(Не используются, так как(по сравнению с ESP-модулями) используют слишком много ресурсов):
    a. Возможно использование так же в реализации как в серверном исполнении, так и в клиенте.
3. ESP8266 model 01(так же подходят и другие платы на основе ESP8266 - ESP-01-12a/12f, а так же ESP-32), возможно использование в системе, как клиент, так и в серверном исполнении (См. список использованной литературы [8]):
    a. В системе не используется как клиент из-за крайне нестабильной работы ESP в качестве клиента, плохо принимаются и отправляются пакеты, плохо находится WiFi- сеть
    b. Используется как сервер, поскольку удовлетворяет всем требованиям работы (См. список использованной литературы [14])
4. Так же разработка приложения на смартфон может быть реализована в различных источниках:
    a. Разработка под IPhone - Можно разрабатывать приложения для IPhone в Xcode на языке Swift, однако мы не стали использовать этот вариант, поскольку нужно знание Swift и компьютер с операционной системой macOS, в котором и будет производиться разработка приложения, которого у нас нет
    b. Разработка в Android Studio - Используется этот вариант, поскольку у всех в нашей команде есть телефон с ОС Android и Android Studio предоставляет все средства для разработки, в том числе и графический редактор слоёв и дизайна (См. список использованной литературы [9])
5. Так же возможна разработка на различных языках программирования:
    a. Возможна разработка ESP8266 на Lua, через ESPlorer (не используется, поскольку никто в команде не знает Lua)
    b. Возможна разработка Android-приложения на языке Kotlin или C++, но не используется из-за худшей работы, по сравнению с Java в Android Studio (См. список использованной литературы [9])

Итоги/результаты работы:

Нарисованы схемы принципиального и физического построения;

Написан простейший TCP-сервер на ESP, принимающий значения с телефона и отправляющий их на COM-порт

Написана программа, (не работает из-за проблем совместимости различных версий Android Studio) подключающаяся к модулю и отправляющая и принимающая данные

Теоретически продуманы все этапы работы, разработаны планы всех действий:

1. Создать 3D модель корпуса в Autodesk Fusion 360
2. Отладка функций приёма передачи на роботе (затрудняется высокой загруженностью компании)
3. Доразработка приложения в Android Studio (затрудняется отсутствием знаний в разработке приложений на Android)
4. Внедрение модуля в систему

Выводы

Вывод:

На данный момент, устройство не реализовано до конца и не обладает требуемым набором функций, однако, если придерживаться данных наработок, можно создать устройство, подходящее для отладки пожарных роботов, на основе этого модуля можно создать приложение для пользователя с мгновенной рассылкой о наличии возгорания

Сравнение аналогов и созданной модели:

На рынке есть несколько моделей, но основная их проблема – они всё, что считывают, все пакеты, отправляют на компьютер, что не есть хорошо при большой загруженности сети пожарного робота. Также в один из минусов данных модулей можно вынести их большой размер, по сравнению с нашим модулем, что для пожарного робота довольно важно, ведь отладку надо (на этапах разработки) вести в условиях, превышающих по жёсткости привычные для работы – и как следствие, модуль должен укладываться вовнутрь корпуса. У большинства недорогих модулей нет возможности работы со смартфоном, а те что умеют – дорогие – наш модуль можно собрать за 475₽, а цена на промышленные – от 1137₽. В минус нашей модели можно вынести относительную ненадёжность ESP8266[3].

[3]Однако, существуют промышленные системы (К примеру: © 1999-2019 Alibaba.com [Электронный ресурс]: Esp8266 Wi-Fi пульт дистанционного управления реле. 2018. URL: https://russian.alibaba.com/product-detail/esp8266-wifi-remote-control-relay-module-60540684934.html?spm=a2700.md_ru_RU.maylikeexp.9.30b6774fuv4yFr (дата обращения: 14.01.2020)) , в которых могут использоваться ESP8266

Список использованной литературы[4]

Программирование, прошивка, работа с ESP8266 в Arduino IDE на C++ и работа с RS-485 по протоколу Modbus RTU:

1. Коллективный блог [Электронный ресурс]: прошивка, программирование в Arduino IDE. 29 февраля 2016. URL: https://habr.com/ru/post/371853/ (дата обращения: 14.01.2020)
2. Иван Грохотков Arduino IDE для ESP8266 [Электронный ресурс]: описание базовых функций и основ работы. 29.03.2015. URL: https://esp8266.ru/arduino-ide-esp8266/ (дата обращения: 13.01.2020)
3. Модуль преобразователя интерфейсов UART TTL – RS-485 [Интернет-магазин]: Обзор модуля преобразователя интерфейсов TTL – RS-485, его технические характеристики, подключение к Arduino и пример использования. © 3DiY (Тридиай) - интернет-магазин комплектующих для 3D принтеров, ЧПУ станков и робототехники. 2020 URL: https://3d-diy.ru/wiki/arduino-moduli/interfeys-ttl-rs485/ (Дата обращения: 11.01.2020)
4. Коллективный блог [Электронный ресурс]: Использование протокола Modbus RTU для работы устройств с Arduino-микроконтроллерами в сетях промышленной оптимизации. 28 января 2015. URL: https://habr.com/ru/post/249043/ (дата обращения: 12.01.2020)

Sockets в Android Studio

5. Copyright © 2016-2019 Java-online.ru – всё о Java и SQL [Электронный ресурс]: Использование сокетов в Android. 2018. URL: http://java-online.ru/android-socket.xhtml (дата обращения: 13.01.2020)
6. © Google Developers Официальный сайт разработчиков Android-приложений [Электронныйресурс]: Описание всех функций в классе Sockets. 2015. URL: https://developer.android.com/reference/java/net/Socket (дата обращения: 13.01.2020)
7. Шпуряка Александр. г. Марганец. Украина. Программирование с нуля [Электронный ресурс]: Знакомство с Sockets в ОС Android. Считываем данные с ESP8266, используя AsyncTask в фоновом режиме. Остановка считывания потока по нажатию на кнопку. 15.06.2018. URL: http://www.ap-impulse.ru/svyazyvaem-android-sokety-i-esp8266-tcp-server-shag-89/ (дата обращения: 13.01.2020)

Программы для работы над проектом:

8. © 2020 Arduino Download the Arduino IDE [Электронный ресурс для скачивания]: The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. 2020. URL: https://www.arduino.cc/en/main/software (дата обращения: 12.01.2020)
9. © Google Developers Android Studio [Электронный ресурс для скачивания]: Android Studio provides the fastest tools for building apps on every type of Android device. 2020. URL: https://developer.android.com/studio (дата обращения: 12.01.2020)

[4] Так же при разработке проекта использовались документы от исследовательского центра пожарной робототехники ЭФЭР, но по корпоративной этике мы не можем представить эти документы для просмотра

10. © 1995-2020, Digi-Key Electronics. Sheme-it [Электронный ресурс]: универсальный сервер для создания принципиальных схем. 2020. URL: https://www.digikey.com/schemeit/project/ (дата обращения: 12.01.2020)
11. KiCad EDA. A Cross Platform and Open Source Electronics Design Automation Suite [Электронный ресурс для скачивания]. 2020. URL: https://kicad-pcb.org/ (дата обращения: 12.01.2020)
12. © 2020 Autodesk Inc. integrated CAD / CAE / CAM tool for industrial design and engineering. [Электронный ресурс для скачивания]: Fusion 360 free 3D CAD/CAM design software for students and educators. 2020. URL: https://www.autodesk.com/products/fusion-360/students-teachers-educators?mktvar004=668081&internalc=true (дата обращения: 12.01.2020)

Не предустановленные библиотеки для работы с WiFi модулем:

13. © 2020 Arduino esp8266 Arduino site placeholder [Электронный ресурс для скачивания]: дополнительная ссылка для Менеджера плат в настройках Arduino IDE. 2014. URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json (дата обращения: 12.01.2020)
14. © 2020 GitHub, Inc. Arduino core for ESP8266 WiFi chip [Электронный ресурс для скачивания]: ESP8266 Arduino core comes with libraries to communicate over WiFi using TCP and UDP, set up HTTP, mDNS, SSDP, and DNS servers, do OTA updates, use a file system in flash memory, and work with SD cards, servos, SPI and I2C peripherals. 14 January 2020. URL: https://github.com/esp8266/Arduino (дата обращения: 14.01.2020)
15. © 2020 GitHub, Inc. libmodbus is a library that provides a Serial Modbus implementation for Arduino. [Электронный ресурс для скачивания]: A primary goal was to enable industrial communication for the Arduino in order to link it to industrial devices such as HMIs, CNCs, PLCs, temperature regulators or speed drives. 26 May 2019. URL: https://github.com/smarmengol/Modbus-Master-Slave-for-Arduino (дата обращения: 14.01.2020)

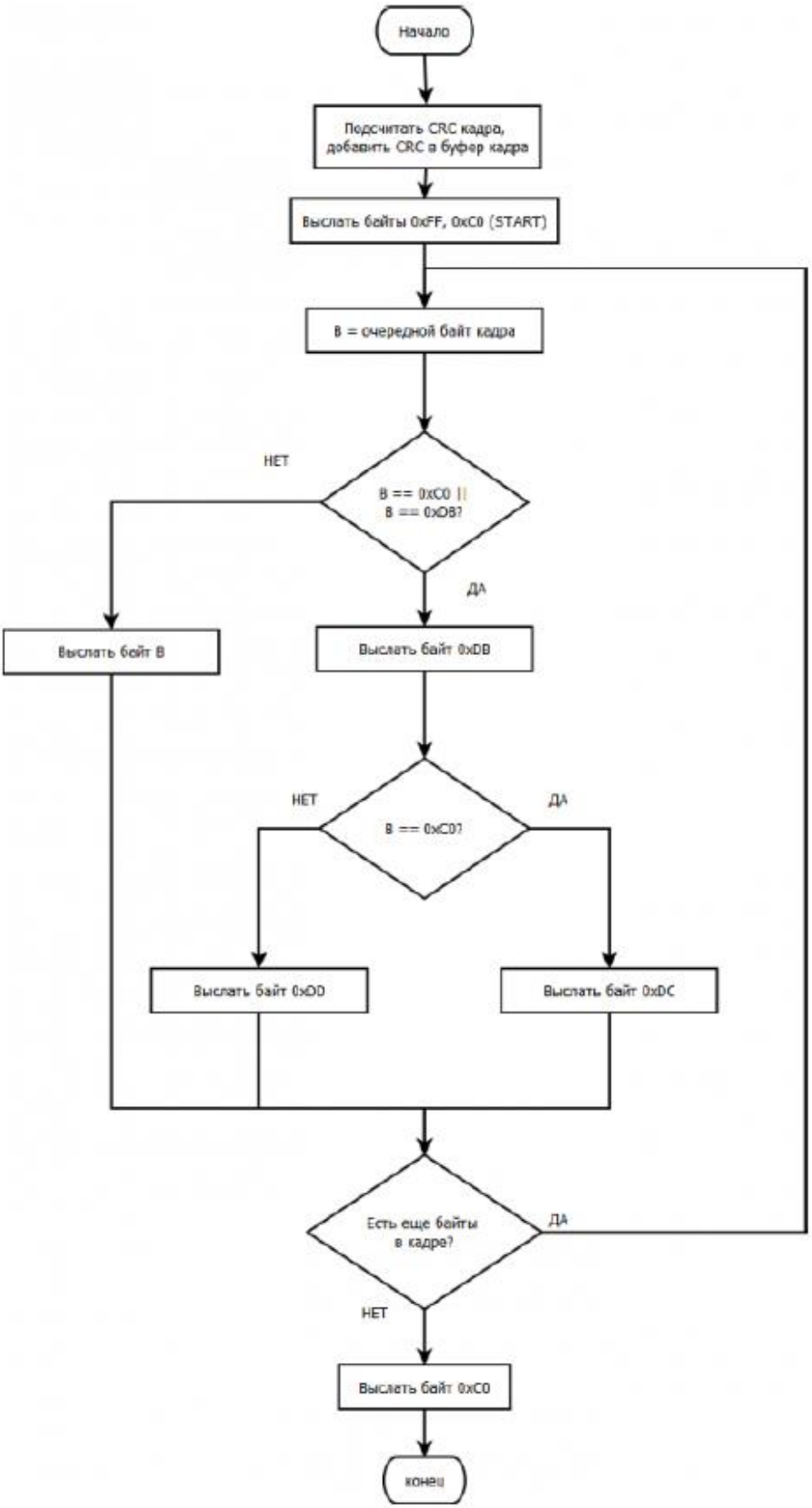
Приложения:

Приложение 1

Формат передачи кадра на ESP-01 на языке C++:

```
void WriteCadr(short size, uint8_t packet[255]){
	for(int i = 0; i < size; i++)
		CRC += packet[i]*44111;
	slave.poll(0xFF);//Высылаем байт 0xFF, как признак начала кадра
	slave.poll(0xC0);//Высылаем 0x0C
	int i = 0;
	while(i < size-1){//packet[i] - очередной байт пакета
		if(packet[i] == 0xC0 || packet[i] == 0xDB){
			slave.poll(0xDB); //Отправляем 0xDB
			if(packet[i] == 0xC0){
				slave.poll(0xDC);//Отправляем 0xDC
			} else {
				slave.poll(0xDD);//Отправляем 0xDD
			}
		} else {
			slave.poll(packet[i]);
		}
	}
	slave.poll(CRC);
}
```

Приложение 1.1



Приложение 2

Формат приёма пакета на ESP-01 (C++):

```
void ReadPacket(){
  B = slave.query();
  B = slave.query();//Считывание байтов начала
  for(int i = 0; i < 271; i++){
    B = slave.query();//B - следующий байт
    if(B == 0xDB){
      B = slave.query();
      if(B == 0xDC)
        packet[i] = 0xC0;
      if(B == 0xDD){
        packet[i] = 0xDB;
      } else {
        i = 300;
        error++;//Кадр повреждён
      }
    } else {
      if(B == 0xC0){
        CRC += packet[i-2]*44111;//Подсчёт КС
        CRCrec = slave.query();//Считывание КС
        CRCrec += slave.query();
        if(CRCrec == CRC){
          i = 300;
        } else {
          i = 300;//Под-ная КС!=Сч-ная
          error++;//Кадр повреждён
        }
      }
    }
  }
}
```

Приложение 2.1

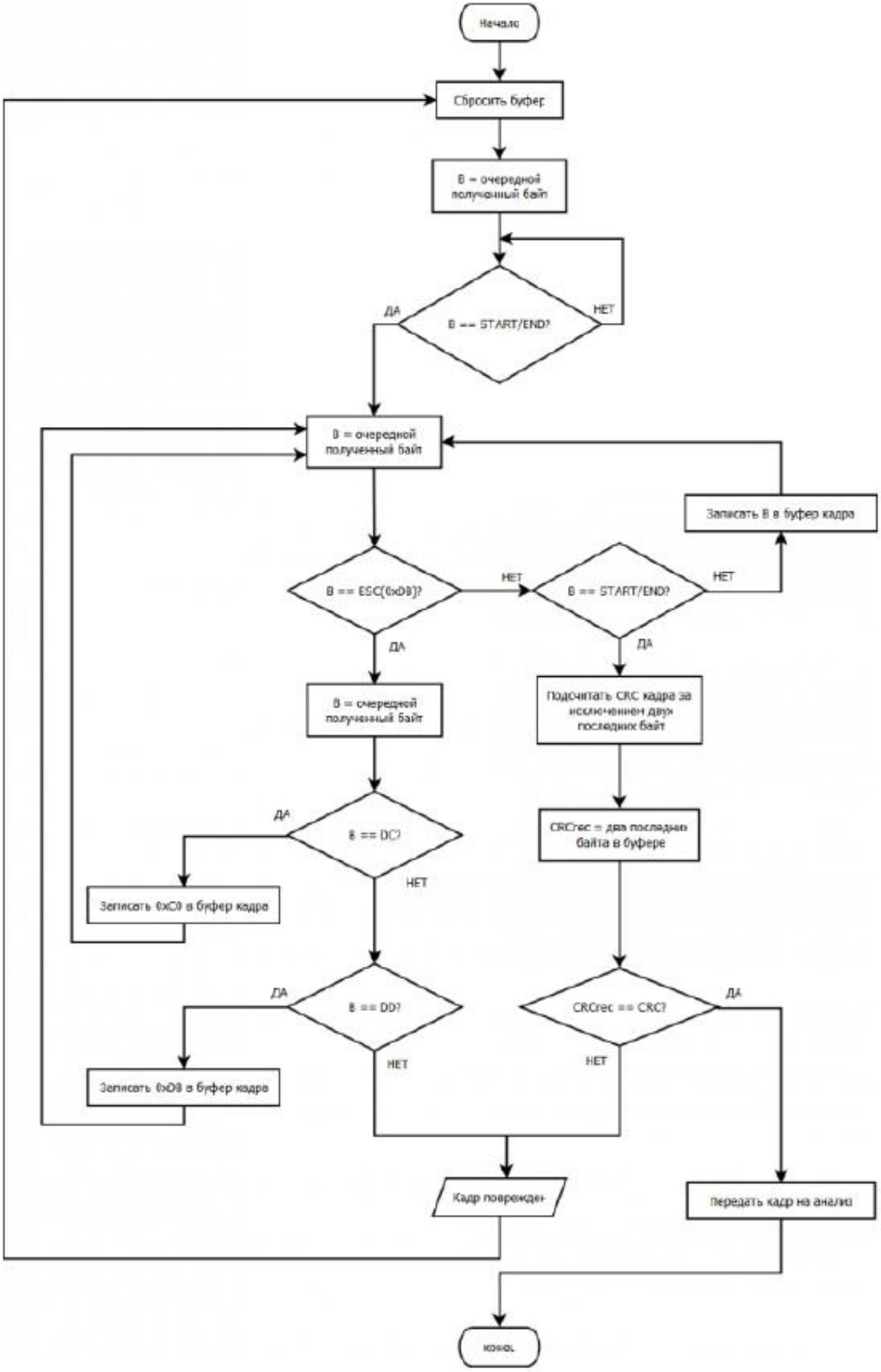


Рисунок 4. Алгоритм приёма пакета в системе

Приложение 3

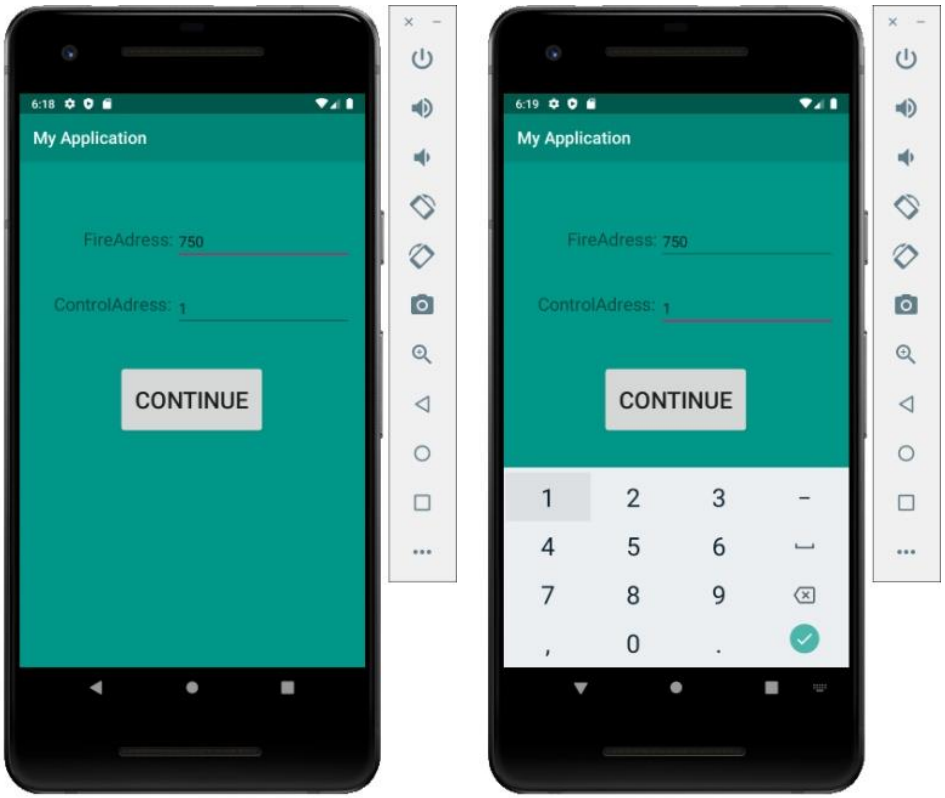
Простейший Modbus-client на Arduino считывающий значение с кнопки и отправляющий его на master:

```
#include "ModbusRtu.h"
#define ID   1      // адрес ведомого
#define btnPin  2   // номер входа, подключенный к кнопке
#define stlPin  13  // номер выхода индикатора работы
                    // расположен на плате Arduino
#define ledPin  12  // номер выхода светодиода
//Задаём ведомому адрес, последовательный порт, выход управления TX
Modbus slave(ID, 0, 4);
boolean led;
int8_t state = 0;
unsigned long tempus;
// массив данных modbus
uint16_t au16data[11];
void setup() {
  digitalWrite(stlPin, HIGH );
  digitalWrite(ledPin, LOW );
  pinMode(stlPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(btnPin, INPUT);
  // настраиваем последовательный порт ведомого
  slave.begin( 9600 );
  // зажигаем светодиод на 100 мс
  tempus = millis() + 100;
  digitalWrite(stlPin, HIGH );
}
void loop() {
  // обработка сообщений
  state = slave.poll( au16data, 11);
  // если получили пакет без ошибок - зажигаем светодиод на 50 мс
  if (state > 4) {
    tempus = millis() + 50;
    digitalWrite(stlPin, HIGH);
  }
  if (millis() > tempus) digitalWrite(stlPin, LOW );
  //обновляем данные в регистрах Modbus и в пользовательской программе
  io_poll();
}
void io_poll() {
  //Копируем Coil[1] в Discrete[0]
  au16data[0] = au16data[1];
  //Выводим значение регистра 1.3 на светодиод
  digitalWrite( ledPin, bitRead( au16data[1], 3 ));
  //Сохраняем состояние кнопки в регистр 0.3
  bitWrite( au16data[0], 3, digitalRead( btnPin ));
  //Копируем Holding[5,6,7] в Input[2,3,4]
  au16data[2] = au16data[5];
  au16data[3] = au16data[6];
  au16data[4] = au16data[7];
  //Сохраняем в регистры отладочную информацию
  au16data[8] = slave.getInCnt();
  au16data[9] = slave.getOutCnt();
  au16data[10] = slave.getErrCnt();
}
```

Приложение 4

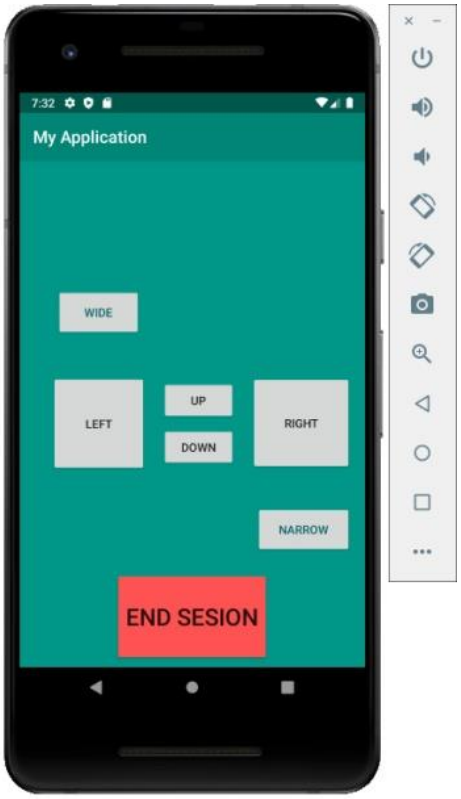
Приложение 4.1

Страница инициализации и запуска:



Приложение 4.2

Страница GamePad-управления внутренними двигателями робота:



Приложение 5

Применение технологии Web-sockets:

```
class ClentLoop extends AsyncTask<String, Byte, Void> {
    protected Void doInBackground(String... params) {
        while (isRun) {
            try {
                socket = new Socket(address, port);
                is = socket.getInputStream();
                os = socket.getOutputStream();
                while (isRun){
                    while (is.available() > 0){
                        publishProgress(Byte.valueOf((byte)
is.read()));
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return null;
    }

    protected void onProgressUpdate(Byte... values) {
        super.onProgressUpdate(values);
        updater.update(values[0]);
    }
}
```