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

|  |
| --- |
| Смирнов Сергей - капитан  Диана Мелихова – участник  Алексей Панфилов – научный руководитель проекта  14 «января», 2020 Петрозаводск |

Содержание:

Оглавление

[Введение: 3](#_Toc29914330)

[Проблема: 3](#_Toc29914331)

[Решение: 3](#_Toc29914332)

[Физика 4](#_Toc29914333)

[Внутреннее устройство системы модуля: 4](#_Toc29914334)

[Схемы: 4](#_Toc29914335)

[Поведение 6](#_Toc29914336)

[Устройство: 6](#_Toc29914337)

[Смартфон 6](#_Toc29914338)

[Отладка 7](#_Toc29914339)

[Возможные реализации: 8](#_Toc29914340)

[Итоги/результаты работы: 9](#_Toc29914341)

[Выводы 10](#_Toc29914342)

[Список представленной литературы 11](#_Toc29914343)

# 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:

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)[[1]](#footnote-2)

# 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 А8-А11 – с 5 В логикой. На питающий вход Vin1 следует подавать напряжение 5В, а на Vin2 – другое.

## Схемы:

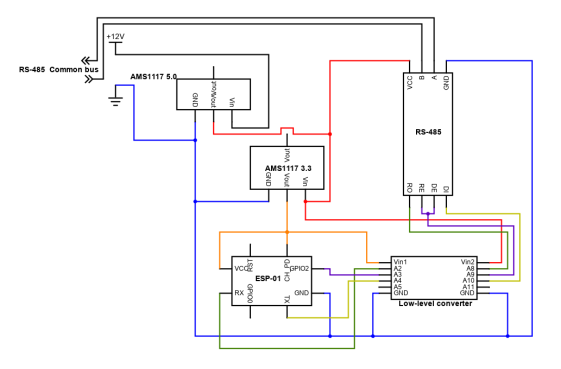


Рисунок . Принципиальная схема устройства

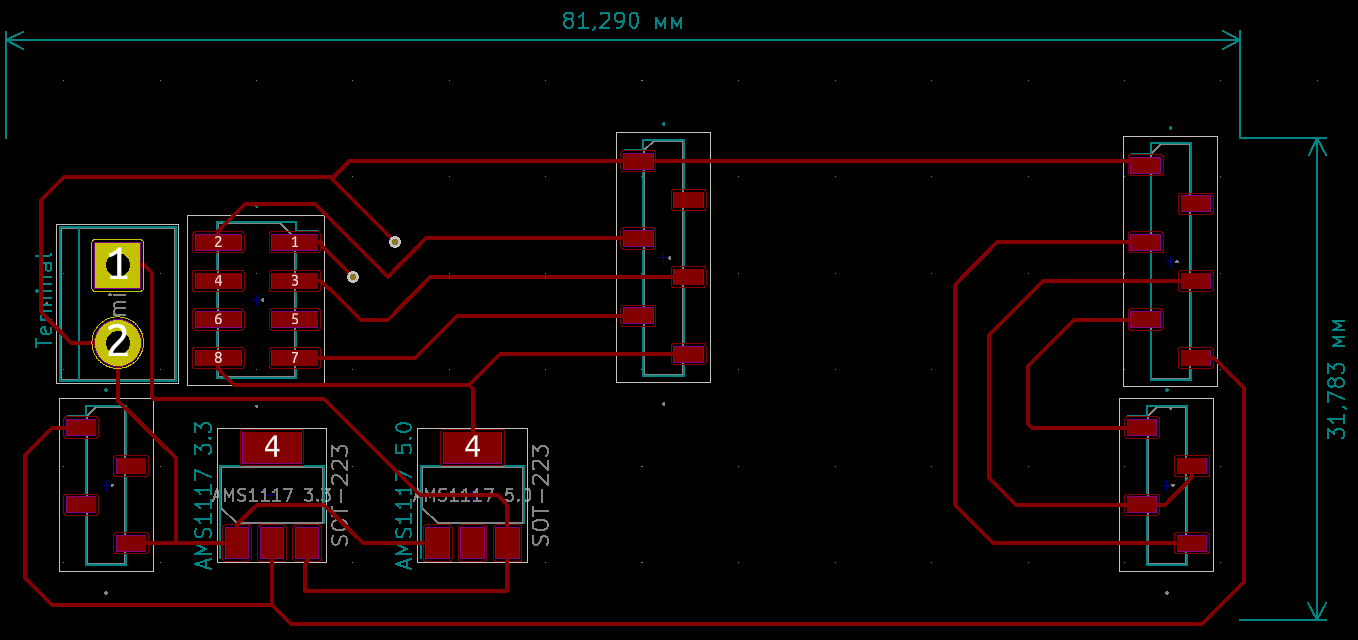


Рисунок . Разметка для травления

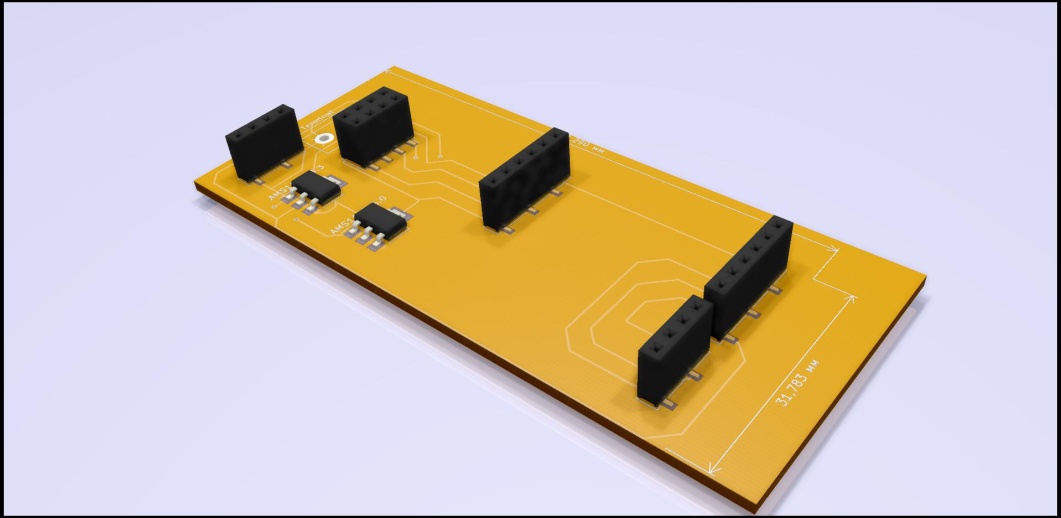


Рисунок . 3D-вид платы

# 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 номер, наш модуль обращается к нужной подсистеме, устанавливает соединение и запрашивает/отправляет данные(Формат передачи-приёма пакета (См. Формат передачи кадра на ESP-01 на языке C++ и См. Формат передачи кадра на ESP-01(С++))
* поднимать TCP-сервер для общения с телефоном; используется стандартный TCP-сервер (См. простейший TCP-сервер, принимает байты с телефона и отправляет их на COM-порт компьютера(C++))

Таблица . 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 эта структура, 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 столь 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)

## 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. Screens с телефона - Страница инициализации и запуска и Страница GamePad-управления внутренними двигателями робота)
* 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

## 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 . 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 праметров):

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:
   1. Поднимается Bluetooth-сервер и ожидается подключение телефона – работа в режиме сервера
   2. 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):
   1. 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:
   1. В системе не используется как клиент из-за крайне нестабильной работы ESP в качестве клиента, плохо принимаются и отправляются пакеты, плохо находится WiFi- сеть
   2. Используется как сервер, поскольку удовлетворяет всем требованиям работы
4. Так же разработка приложения на смартфон может быть реализована в различных источниках:
   1. 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
   2. 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
5. Так же возможна разработка на различных языках программирования:
   1. Возможна разработка ESP8266 на Lua, через ESPlorer (не используется, поскольку никто в команде не знает Lua)
   2. Возможна разработка Android-приложения на языке Kotlin или C++, но не используется из-за худшей работы, по сравнению с Java в Android Studio

# 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

Если придерживаться данных наработок, можно создать устройство, подходящее для отладки пожарных роботов

# Список представленной литературы

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

* 1. ESP8266 прошивка, программирование в Arduino IDE [Электронный ресурс]

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(СRC);

}

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();

}

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]);  
 }  
}

1. На момент написания работы – Реализовано общение между Wi-Fi модулем и телефоном, отправление пакетов в систему [↑](#footnote-ref-2)