Работа на всероссийский конкурс научных и инженерных проектов «Балтийский научно - инженерный конкурс»

направления Робототехника

«Устройство для отладки пожарных роботов компании “«ЭФЕР» - завод пожарных роботов”»

,,Vesi ja vaahto ‘’

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).

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

Содержание:

1. Физика
2. Поведение
3. Возможные реализации
4. Итоги/результаты работы
5. Выводы

# Физика

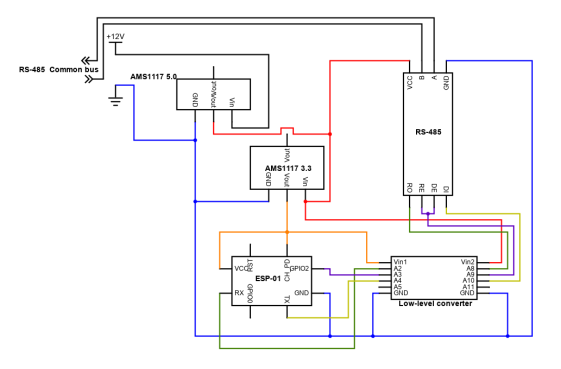
1. Основной процессор:

У модуля ESP-01 есть внутренний процессор ESP8266 с возможностью обновления прошивки, возможно так же использование HC-05/06, однако он не используется, поскольку в Android Studio сложная работа с Bluetooth, однако вполне хорошая с Wi-Fi

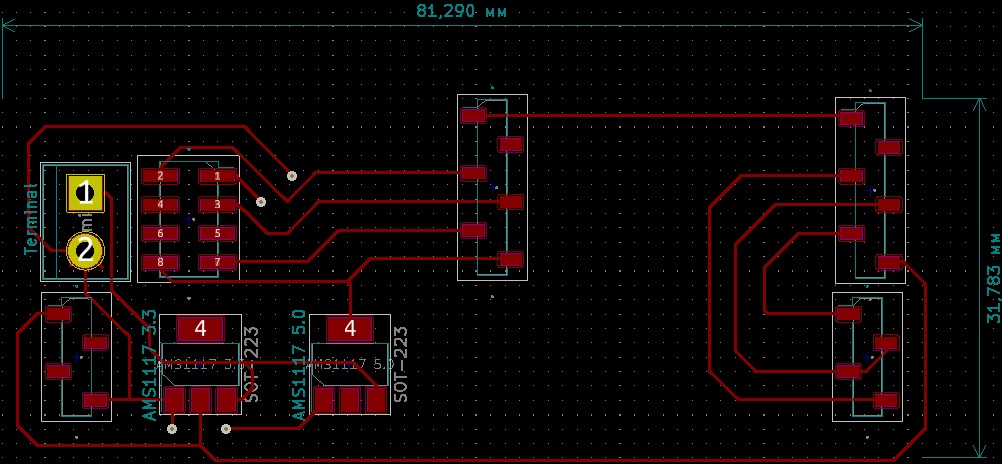
1. Стабилизаторы питания:

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

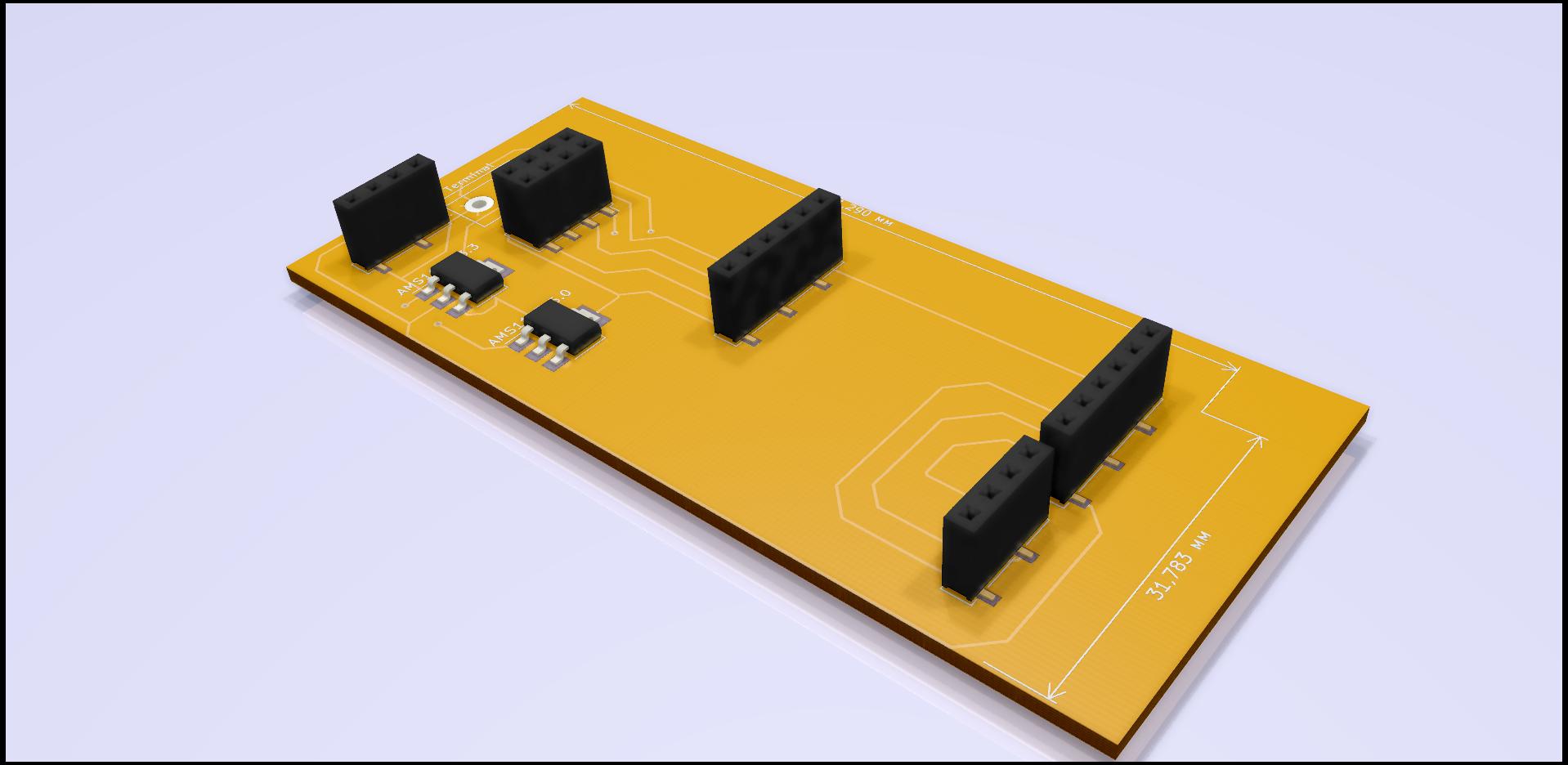
1. Схемы:
   1. Принципиальная:



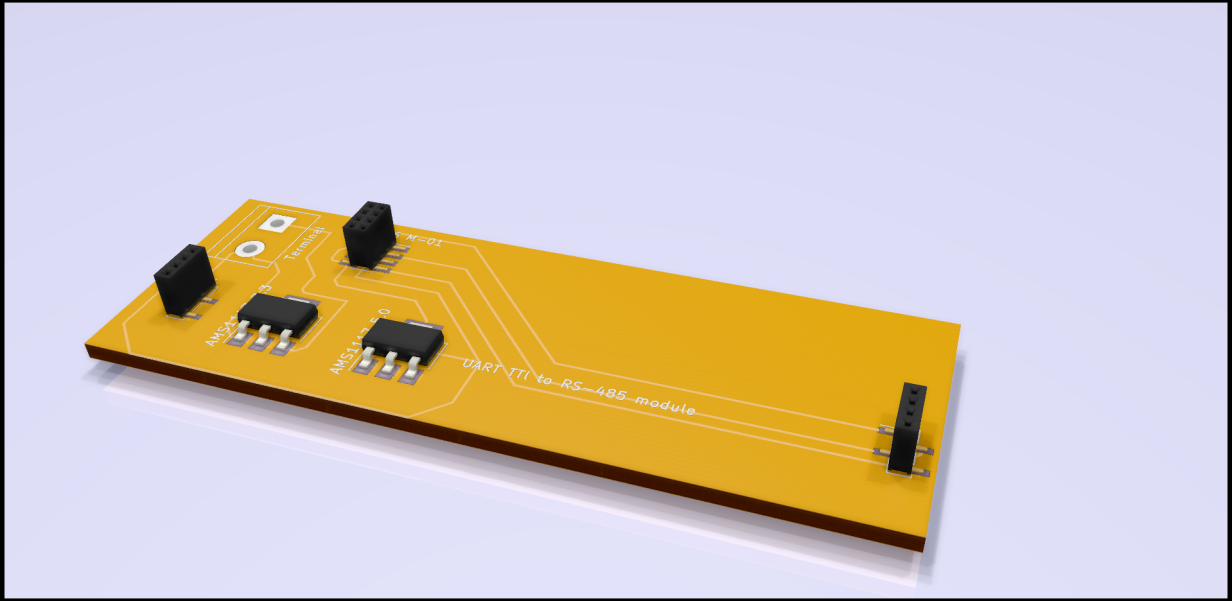
* 1. Разметка для травления:



* 1. 3D – модель:



* 1. Истинный вид:



# Поведение

## Устройство:

1. Устройство должно:
   1. обращаться к нужной подсистеме робота и отправлять ей задание на выполнение или запрашивать значения с датчиков
      1. Для этого используется протокол, используемый в самом роботе, без изменений в нём;
         1. В соответствии с протоколом, у каждого модуля в системе есть номер, наш модуль обращается к нужной подсистеме, устанавливает соединение и запрашивает/отправляет данные
         2. Формат передачи кадра на ESP-01(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

}

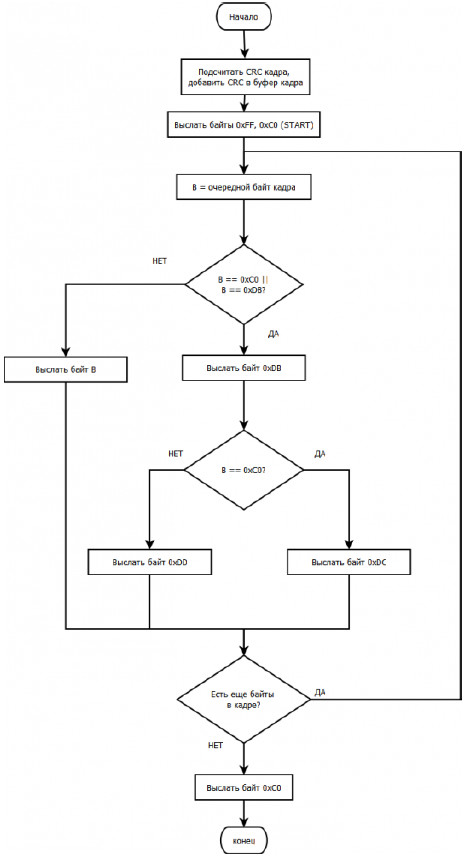
} else {

slave.poll(packet[i]);

}

}

slave.poll(СRC);



* + - 1. Формат приёма кадра(C++):

void ReadPacket(){

B = slave.query();

B = slave.query();//Cчитывание 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

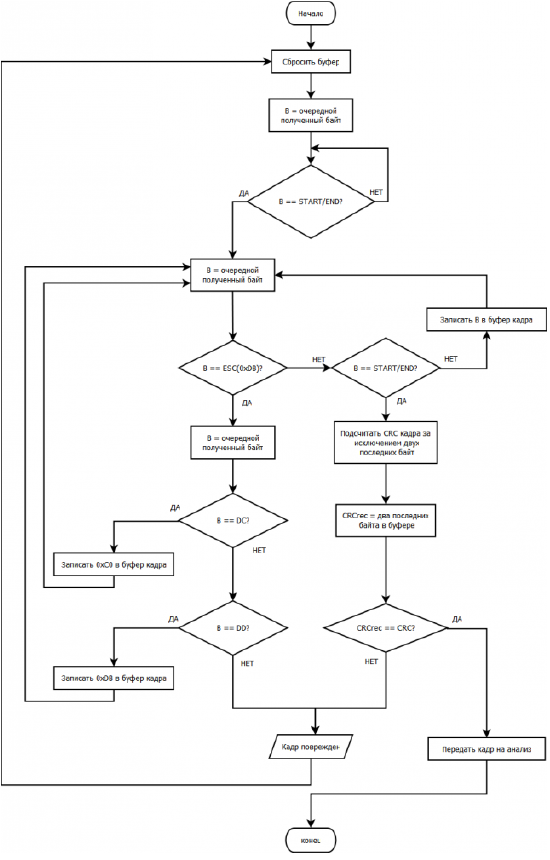
}

}

}

}

}



* 1. поднимать TCP-сервер для общения с телефоном;
     1. Используется стандартный TCP-сервер:
        1. Простейший TCP-сервер, принимает байты с телефона и отправляет их на COM-порт компьютера(C++):

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

WiFiServer server(750);

const char\* ssid = "FR";

const char\* password = "12345FR485";

**IPAddress** apIP(192, 168, 4, 1);

unsigned char RX;

void setup() {

**WiFi.**disconnect();

**WiFi**.mode(WIFI\_AP);

**WiFi**.softAPConfig(apIP, apIP, **IPAddress**(255, 255, 255, 0));

**WiFi**.softAP(ssid, password, 11, 4, false);

server.begin();

**Serial**.begin(115200);

}

void loop() {

WiFiClient client = server.available();

if (client) { //Если клиент подключён

while (client.connected()) {

if (**Serial**.available() > 0) { // Если принимаем по UART

char TX = Serial.read();

client.print(TX); //

}

if (client.available() > 0) { //Если принимаем по TCP

char RX = client.read();

**Serial**.print(RX);

}

}

}

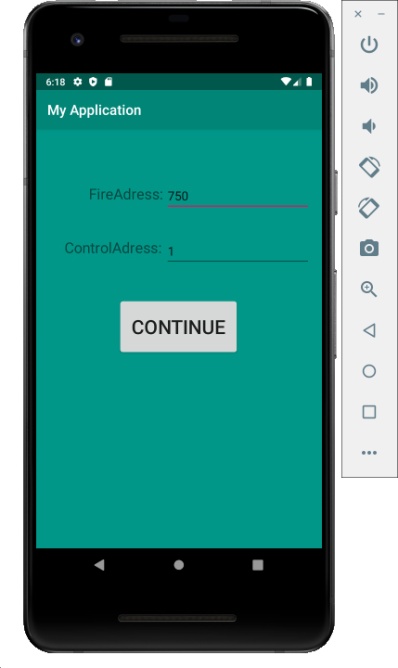
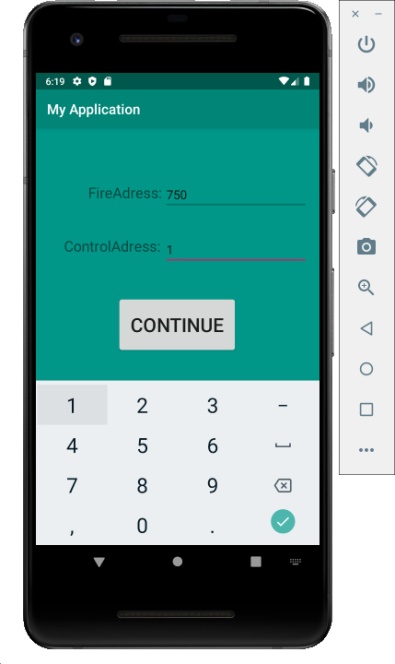
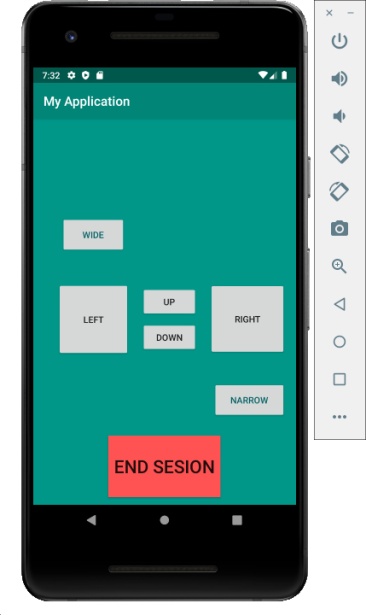
}

1. Для установки параметров с телефона на модуль используется следующая структура:
   1. Формат напоминает тот, что используется в самом роботе, что сделано для разгрузки ESP

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

таким образом, обработка интерфейса взаимодействия с человеком и значений, полученных с датчиков, ложится на смартфон

## Смартфон

1. Должен осуществлять взаимодействие человека и выводить значения, полученные с робота, для чего телефон подключается к сети Wi-Fi “FR”, и начинается передача пакетов
   1. Вот Screens с телефона:
      1. Страница инициализации и запуска:
         1.  
      2. Страница GamePad-управления внутренними двигателями робота:
         1. 
   2. Подключение к модулю происходит по Wi-Fi, и поэтому, используя стандартные способы передачи данных в Android Studio, применяется технология Web-sockets(Java):

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. Каждый из процессов, происходящих в Soccet, является независимым от основного потока программы и выполняется независимо, что происходит каждый раз для любого обращения к классу ClientLoop и вызывает новый поток

## Отладка

1. Так как отладку нужно проводить всей системы, то модуль должен выполнять все функции обычной системы в составе робота – отвечать на запросы и широковещательные пакеты:
   1. Ответ модуля на GetStatus(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

};

* + 1. Биты флагов status(вторые 16 бит, что специфичны для каждой системы):

|  |  |  |
| --- | --- | --- |
| Обозначение | Номер бита | Назначение |
| UpWiFi | 16 | WiFi сеть поднята |

* + 1. Возможный запрос SetParam:

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 на телефоне:
2. Возможно использование в различных реализациях:
   1. Поднятие сервера
      1. Поднимается Bluetooth-сервер и ожидается подключение телефона
   2. Работа в режиме клиента
      1. Работа в режиме BTL 4.0 - ожидается поднятия сервера на телефоне
3. WeMos Arduino, ArduinoWiFi rev2, NodeMCU v3, Arduino+WiFi-sheedl/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.
4. ESP8266 model 01(так же подходят и другие платы на основе ESP8266 - ESP-01-12a/12f, а так же ESP-32):
   1. Возможно использование в системе как в системе-клиенте, так и серверном исполнении:
      1. В системе не используется как клиент из-за плохой работы ESP в качестве клиента, плохо принимаются и отправляются пакеты, плохо находится WiFi- сеть
      2. Используется как сервер, поскольку удовлетворяет всем требованиям работы
5. Так же разработка приложения на смартфон может быть реализована в различных источниках:
   1. Разработка под IPhone:
      1. 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 IOS, в котором и будет производиться разработка приложения, которого у нас нет
   2. Разработка в Android Studio
      1. 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

# Итоги/результаты работы:

1. Нарисованы схемы принципиального и физического построения;
2. Написан простейший TCP-сервер на ESP, принимающий значения с телефона и отправляющий их на COM-порт
3. Написана программа, (не работает из-за проблем совместимости различных версий Android Studio) подключающаяся к модулю и отправляющая и принимающая данные
4. Теоретически продуманы все этапы работы, разработаны планы всех действий:
   1. Отладка функций приёма передачи на роботе (затрудняется высокой загруженностью компании)
   2. Разработка приложения в Android Studio (затрудняется отсутствием знаний в разработке приложений на Android)
   3. Внедрение модуля в систему

# Выводы