Bluetooth Car

Ляо Ихун

Го Цзыхань

May 2022

Рефрат

Это документация согласна GOST 7.32-2001. Этот проект является развитием автомобиля управления bluetooth. Конкретная реализация кода доступна по ссылке:`https://github.com/spbgzh/BluetoothCar`.

Ключевые слова:
ANDROID, SWIFT, JAVA, BlUETOOTH, STM32, ДВИГАТЕЛЬНЫЙ МОТОРНЫЙ ПРИВОД, IDE, USART, IIC, PWM

Содержание

Введение

Разработать автомобиль, которым можно управлять по bluetooth на макетной плате stm32. В то же время пульт дистанционного управления Bluetooth был разработан для Android и Ipad соответственно. Пульт дистанционного управления должен иметь возможность управлять скоростью, направлением и т. д. автомобиля. Обратить внимание, что автомобиль не является водонепроницаемым и должен использоваться в сухой среде.В то же время, поскольку приложение Android не используем ble, при использовании автомобильного аккумулятора соединение Bluetooth будет очень нестабильным. Поэтому, если вы хотите использовать приложение Android для подключения Bluetooth, вам необходимо убедиться, что аккумулятор полностью заряжен.

1 Анализ проекта

1.Реализации взаимодействие Bluetooth между мобильным телефоном и stm32

2.Дизайн интерфейса пользователя мобильного приложения

3.stm32 управляет скоростью и направлением движения тележки

4.Создание тележки с IDE экраном, на котором показывает графику когда подключается с батареей, своими руками

2 Проектирование проекта

1.Для Android Studio инструмента разработки Andriod языком разработки является java. Язык разработки IOS swift. Каждый использует существующий API для обеспечения Bluetooth-соединения и взаимодействия. В то же время выполняйте быстрый дизайн пользовательского интерфейса с помощью соответствующих инструментов разработки.

2.Силовыми колесами тележки являются два задних колеса. Управляйте скоростью, контролируя уровень выходного сигнала, чтобы контролировать скорость двигателя тележки. Затем управление направлением движения осуществляется путем управления разницей скоростей между двумя колесами. Мы выбираем протокол PWM для управления уровнем напряжения.

3.Мы проектируем собственные автомобили. В составе кузова автомобиля можно использовать технологию 3D печати или купить готовый кузов самостоятельно.

4.Встроенная разработка с помощью STM32cube. Язык разработки — C. Используются протолколы PWM - контроль уровня напражения, USART - передача данных, IIC - ввод и вывод данных одновременно.

3 Реализация проекта

3.1 Схема

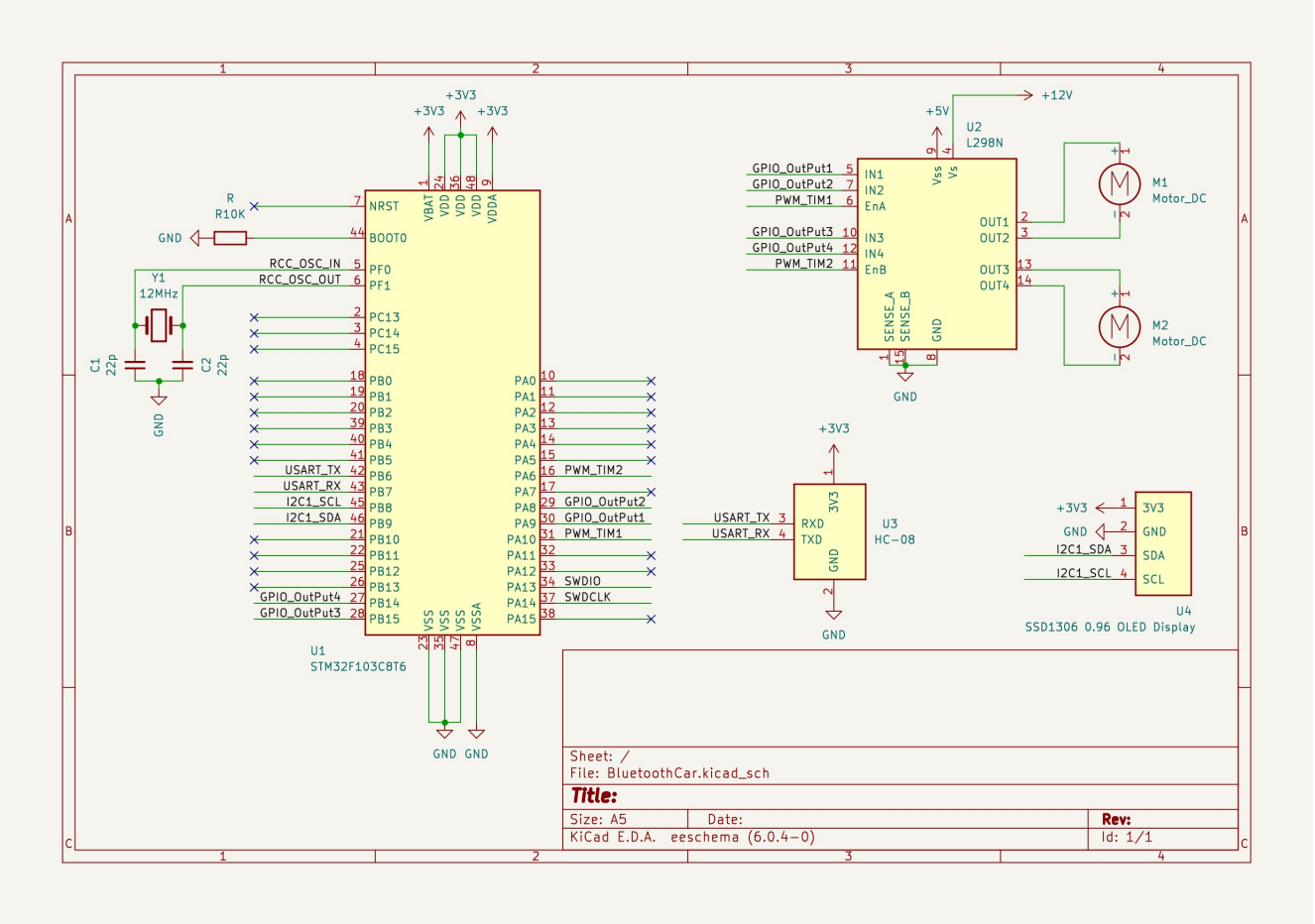


Рис. 1:

3.2 Реализация приложения Bluetooth на Android

3.2.1 Структура приложения

1.UI дизайн:
Main:/res/layout/activity_main.xml
Дистанционное управление:/res/layout/activity_control.xml

2.Главная страница:
/java/com.example.car/MainActivity.java
В главной старнице начинаем сканирование сканирование окружающих bluetooth оборудования. В результате у нас получается список всех оборудования. Мы создаем соединение между телефоном и оборудованием при клике соответстующее имя оборудования.

3.Дистанционное управление:
/java/com.example.car/ControlActivity.java
Отправить сигнал в автомобиль через bluetooth для управления автомобилем.Настройки сигнала согласно таблице 3.4.2.

3.2.2 Реализация

1.Сканирование окружающих bluetooth оборудования:

```
public void scan(){
    Log.i( tag: "Scanner", msg: "Scanner running and trying to find nearby bluetooth");
    //判断设备是否存在蓝牙
    if(bluetoothAdapter == null){
        Log.e( tag: "Bluetooth", msg: "No bluetooth on device");
        return;
    }
    //如果蓝牙没有打开，那么打开蓝牙
    //需要CONNECT权限
    if(!bluetoothAdapter.isEnabled()){
        Log.i( tag: "Bluetooth", msg: "Bluetooth not opened, now try to open it");
        if(bluetoothAdapter.enable()){
            Log.i( tag: "Bluetooth", msg: "Success");
        }else {
            Log.i( tag: "Bluetooth", msg: "Failed");
        }
    }else {
        Log.i( tag: "Bluetooth",  msg: "Bluetooth is opened");
    }
    //已配对
    if(matched) {
        pairedDevices = bluetoothAdapter.getBondedDevices();
        if (pairedDevices.size() > 0) {
            for (BluetoothDevice bd : pairedDevices) {
                String a = bd.getName();
                String b = bd.getAddress();
                String r = a + " " + b;
                boundString.add(r);
            }
            arrayAdapter2 = new ArrayAdapter<String>( context: MainActivity.this,
                    android.R.layout.simple_list_item_1, boundString);
            matched_bluetooth_list.setAdapter(arrayAdapter2);
        }
        matched = false;
    }
    //发现新设备
    bluetoothAdapter.startDiscovery();
}
```

2.Создание соединения с bluetooth оборудованием:

```
private class ConnectThread extends Thread{
    public ConnectThread(){
        BluetoothSocket tmp = null;
        try{
            tmp = car.createRfcommSocketToServiceRecord(MY_UUID);
        }catch (IOException i){
            Log.e( tag: "Socket_err", msg: "Failed to create a socket");
        }
        btSocket = tmp;
    }

    public void run(){
        while(true) {
            try{
                btSocket = car.createRfcommSocketToServiceRecord(MY_UUID);
            }catch (IOException i){
                Log.e( tag: "Socket_err", msg: "Failed to create a socket");
            }
            try {
                // Connect to the remote device through the socket. This call blocks
                // until it succeeds or throws an exception.
                btSocket.connect();
            } catch (IOException connectException) {

            }
            if(btSocket!=null&&btSocket.isConnected()){
                break;
            }
            try {
                Thread.sleep( millis: 700);
            }catch (InterruptedException i){
                Log.e( tag: "Sleeping", msg: "Can't sleep");
            }
        }
        try {
            outputStream = btSocket.getOutputStream();
        }catch (IOException e){
            Log.e( tag: "Socket_err", msg: "Could not open stream");
        }
    }
}
```

3.Отправление сигнала(пример напрваления right,дргие ситуации также):

```
public void write(char bytes) {
    outputStream = MainActivity.outputStream;
    try {
        outputStream.write(bytes);
        Log.e( tag: "Sending",String.valueOf(bytes));
    } catch (IOException e) {
        Log.e( tag: "Conversation err",  msg: "Error occurred when sending data", e);
    }
}
```

```
case R.id.right:
    if (motionEvent.getAction() == MotionEvent.ACTION_DOWN){
        right.setBackgroundColor(Color.parseColor(deep_purple));
        turn_right = true;
        while(turn_right) {
            if (go_ahead) {
                if (is_sp_1) {
                    write(FRONT_RIGHT_1);
                } else if (is_sp_2) {
                    write(FRONT_RIGHT_2);
                } else if (is_sp_3) {
                    write(FRONT_RIGHT_3);
                }
            } else if (go_back) {
                if (is_sp_1) {
                    write(BACK_RIGHT_1);
                } else if (is_sp_2) {
                    write(BACK_RIGHT_2);
                } else if (is_sp_3) {
                    write(BACK_RIGHT_3);
                }
            }
        }
    }
    if(motionEvent.getAction() == MotionEvent.ACTION_UP){
        right.setBackgroundColor(Color.parseColor(normal_purple));
        turn_right = false;
    }
    try {
        Thread.sleep( millis: 700);
    }catch (InterruptedException e){
        e.printStackTrace();
    }
    break;
```

3.2.3 Результат

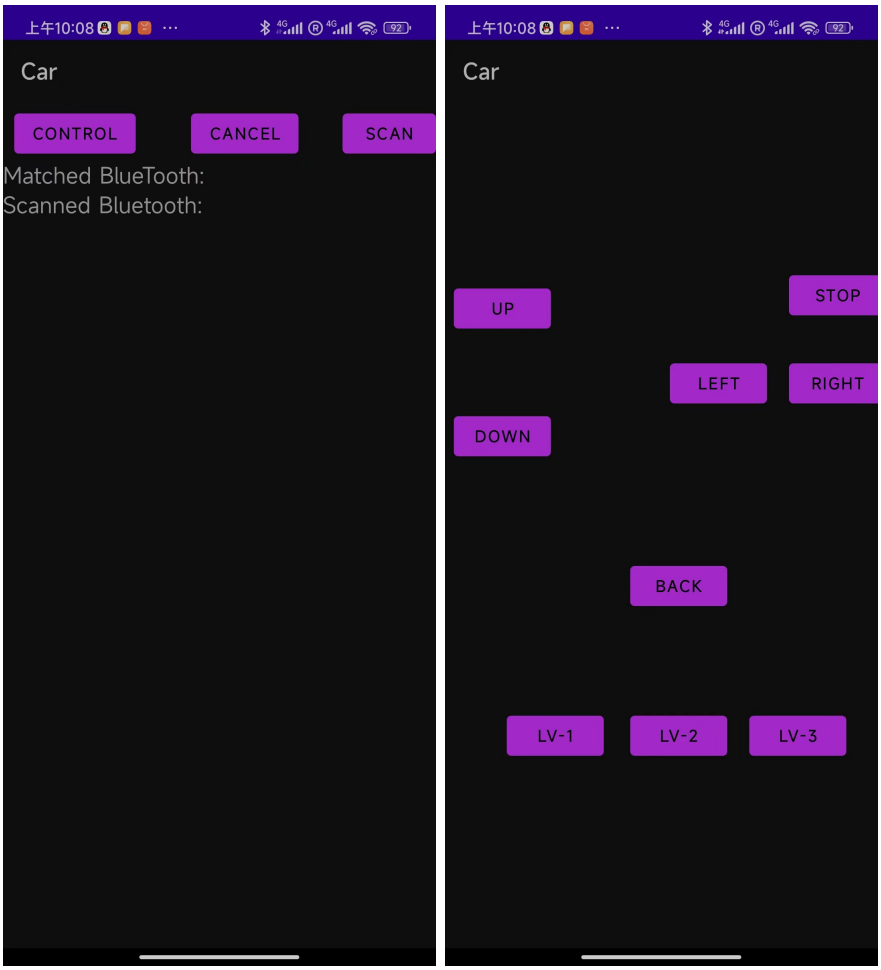


Рис. 2:

3.3 Реализация приложения на IOS

3.3.1 Реализация

1.Сканирование окружающих bluetooth оборудования:

```
// MARK: 中心管理器扫描到了设备
func centralManager(_ central: CBCentralManager, didDiscover peripheral: CBPeripheral, advertisementData: [String :
    Any], rssi RSSI: NSNumber) {
    guard peripheral.name == "CarControler",!aPeArray.contains(peripheral), let deviceName = peripheral.name,
        deviceName.count > 0 else {
        return
    }
    self.doConnect(peripheral: peripheral)
    aPeArray.append(peripheral)
    //传出去实时刷新
    if let backPeripheralsBlock = backPeripheralsBlock {
        backPeripheralsBlock(aPeArray)
    }
}
```

2.Создание соединения с bluetooth оборудованием

```
    // MARK: 连接外设成功，开始发现服务
    func centralManager(_ central: CBCentralManager, didConnect peripheral: CBPeripheral) {
        print("\(#function)连接外设成功。\ncentral:\(central),peripheral:\(peripheral)\n")
        // 设置代理
        peripheral.delegate = self
        // 开始发现服务
        peripheral.discoverServices(nil)
    }
```

```
extension BleHelper: CBPeripheralDelegate {
    //MARK: 匹配对应服务UUID
    func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
        if let error = error  {
            print("\(#function)搜索到服务-出错\n设备(peripheral): \(String(describing: peripheral.name))
                搜索服务(Services)失败: \(error)\n")
            return
        } else {
            print("\(#function)搜索到服务\n设备(peripheral): \(String(describing: peripheral.name))\n")
        }
        for service in peripheral.services ?? [] {
            peripheral.discoverCharacteristics(nil, for: service)
        }
    }

    //MARK: 服务下的特征
    func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor service: CBService, error: Error?) {
        if let _ = error {
            print("\(#function)发现特征\n设备(peripheral): \(String(describing:
                peripheral.name))\n服务(service): \(String(describing:
                service))\n扫描特征(Characteristics)失败: \(String(describing: error))\n")
            return
        } else {
            print("\(#function)发现特征\n设备(peripheral): \(String(describing:
                peripheral.name))\n服务(service): \(String(describing: service))\n服务下的特征: \(service.characteristics ??
                [])\n")
        }

        for characteristic in service.characteristics ?? [] {
            if characteristic.uuid.uuidString.lowercased().isEqual(BLE_WRITE_UUID) {
                pe = peripheral
                writeCh = characteristic

                if let block = backConnectedBlock {
                    block(peripheral,characteristic)
                }
            }
            let dataSender = sender()
            queue.async {
                while true{
                    if self.pe != nil && self.writeCh != nil{
                        dataSender.sentData(ble: self)
                    }
                }
            }
            //此处代表连接成功
        }
    }
```

3.Отправление сигнала(пример напрваления right,дргие ситуации также)

```
private func sent(ble:BleHelper,str:String){
    ble.sendPacketWithPieces(data: str.data(using: .ascii) ?? "0".data(using: .ascii)!,peripheral: ble.pe ??
        CBPeripheral.value(forKey: "0") as! CBPeripheral , characteristic: ble.writeCh ??
        CBCharacteristic.value(forKey: "0") as! CBCharacteristic)
    print(str)
    Thread.sleep(forTimeInterval: 0.7)
}
```

3.3.2 Результат

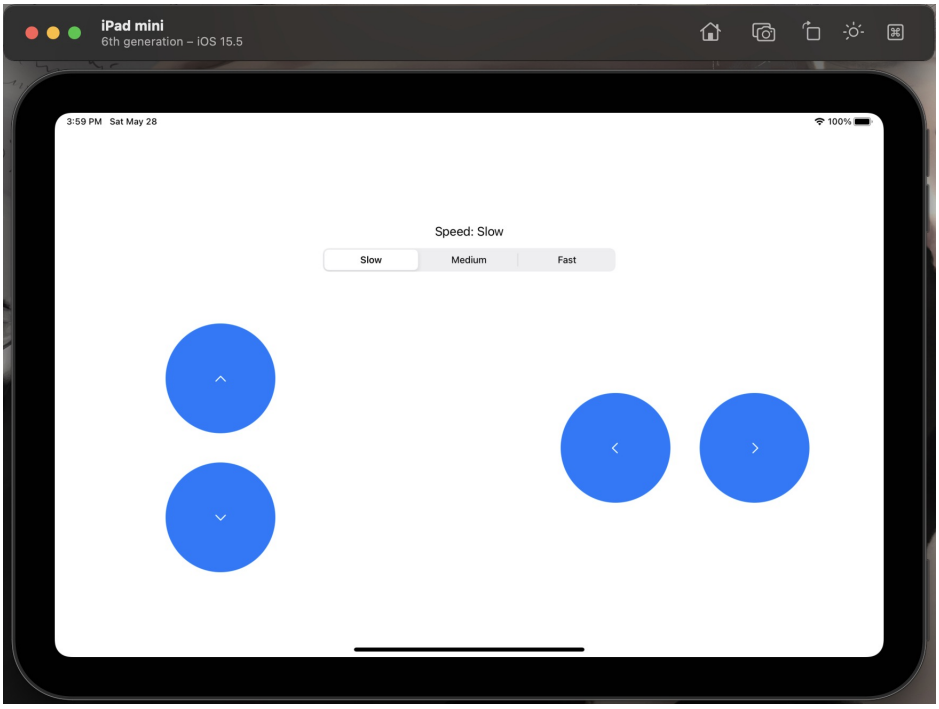


Рис. 3:

3.4 Реализация контроля тележки на STM32

3.4.1 Архитектура

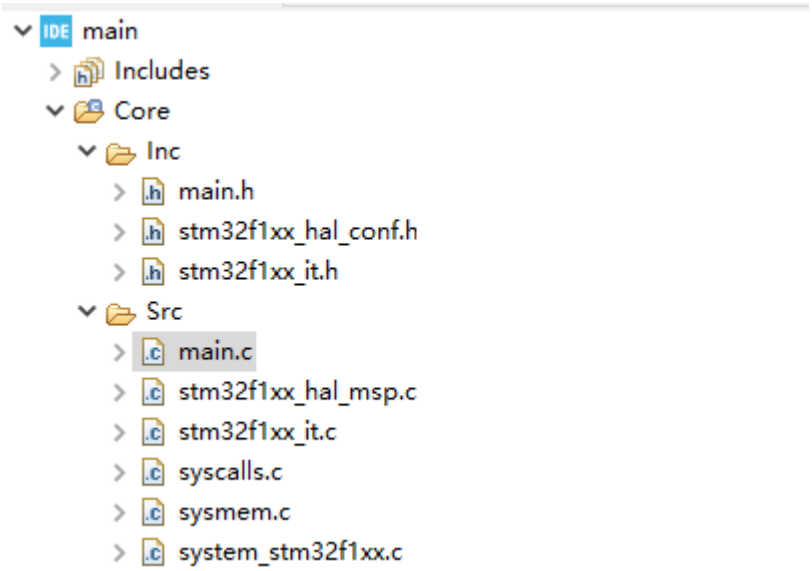


Рис. 4:

Настройки сигнала согласно таблице 3.4.2.

3.4.2 Реализация

1.Схема

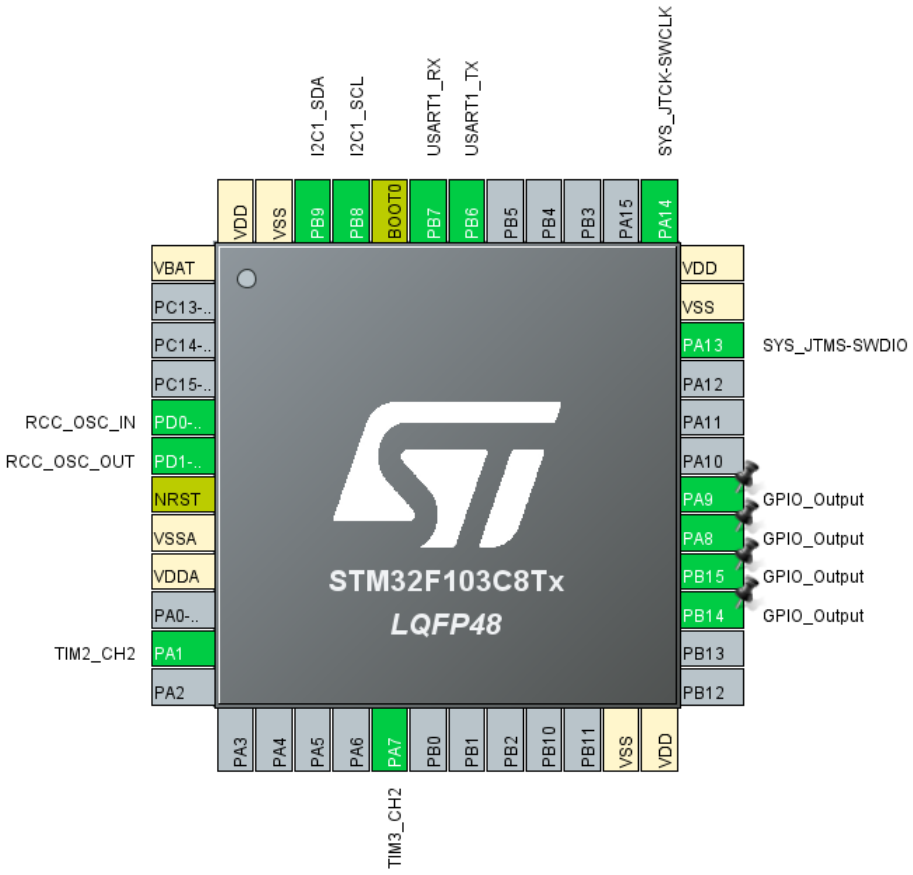


Рис. 5:

2.Управление скоростью:

```
void SpeedController()
{
    if(leftDirction == 1){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, 0);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 1);
    }
    else if(leftDirction == 0){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, 1);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 0);
    }
    else{
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, 0);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 0);
    }
    if(rightDirction == 1){
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, 0);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, 1);
    }
    else if(rightDirction == 0){
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, 1);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, 0);
    }
    else{
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_15, 0);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_14, 0);
    }
    __HAL_TIM_SET_COMPARE(&htim1,TIM_CHANNEL_3, leftSpeed);
    __HAL_TIM_SET_COMPARE(&htim3,TIM_CHANNEL_1, rightSpeed);
    HAL_Delay(50);
}
```

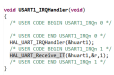
3.Котенок анимация:

```
while (1)
{
    SpeedController();
  /* USER CODE END WHILE */
    SSD1306_Clear();
    SSD1306_DrawBitmap(0,15,photo1,128,40,1);
    SSD1306_UpdateScreen();
    SSD1306_Clear();
    SSD1306_DrawBitmap(0,15,photo2,128,40,1);
    SSD1306_UpdateScreen();
    SSD1306_Clear();
    SSD1306_DrawBitmap(0,15,photo3,128,40,1);
    SSD1306_UpdateScreen();
    SSD1306_Clear();
    SSD1306_DrawBitmap(0,15,photo4,128,40,1);
    SSD1306_UpdateScreen();
    SSD1306_Clear();
    SSD1306_DrawBitmap(0,15,photo5,128,40,1);
    SSD1306_UpdateScreen();
    SSD1306_Clear();
    SSD1306_DrawBitmap(0,15,photo6,128,40,1);
    SSD1306_UpdateScreen();
    SSD1306_Clear();
    SSD1306_DrawBitmap(0,15,photo7,128,40,1);
    SSD1306_UpdateScreen();

  /* USER CODE BEGIN 3 */
}
```

4.Получение сингнала:

5.Обработка сингнала:



В функции HAL_UART_RxCpltCallback в main.c, здесь покажу одну ситуацию.

```
switch(r){
    case 'A': { //3
        leftSpeed = fast;
        rightSpeed = fast;
        leftDirction = 1;
        rightDirction = 1;
        break;
    }
```

Заключение

В результате у нас получаются автомобиль, который может управлять скоростью и направлением, и соответствующий пульт дистанционного управления для Android и Ipad. На Android пользуется обычный bluetooth, а на Ipad пользуется ble.

Приложение А Картики и таблицы

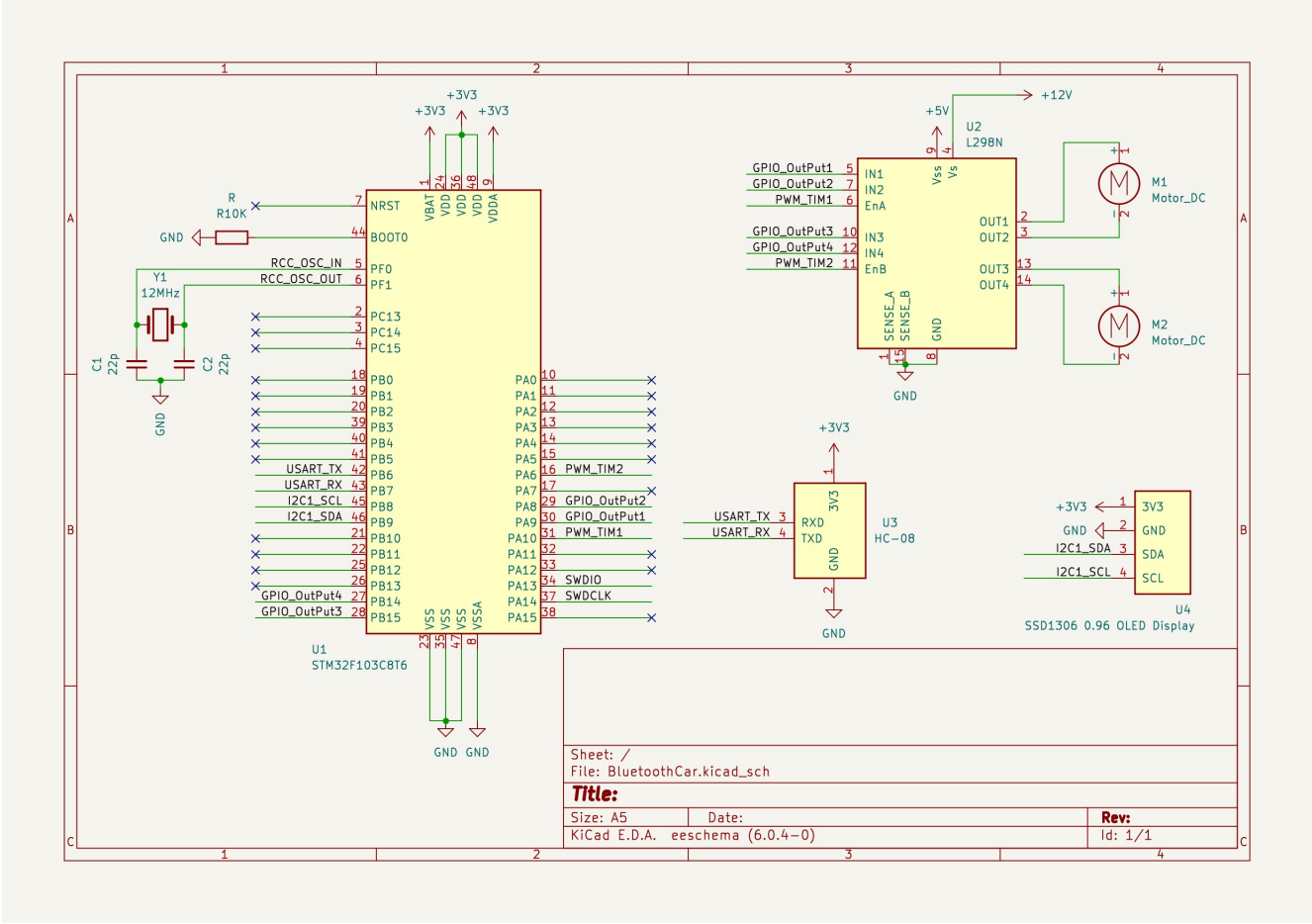


Рис. 6:

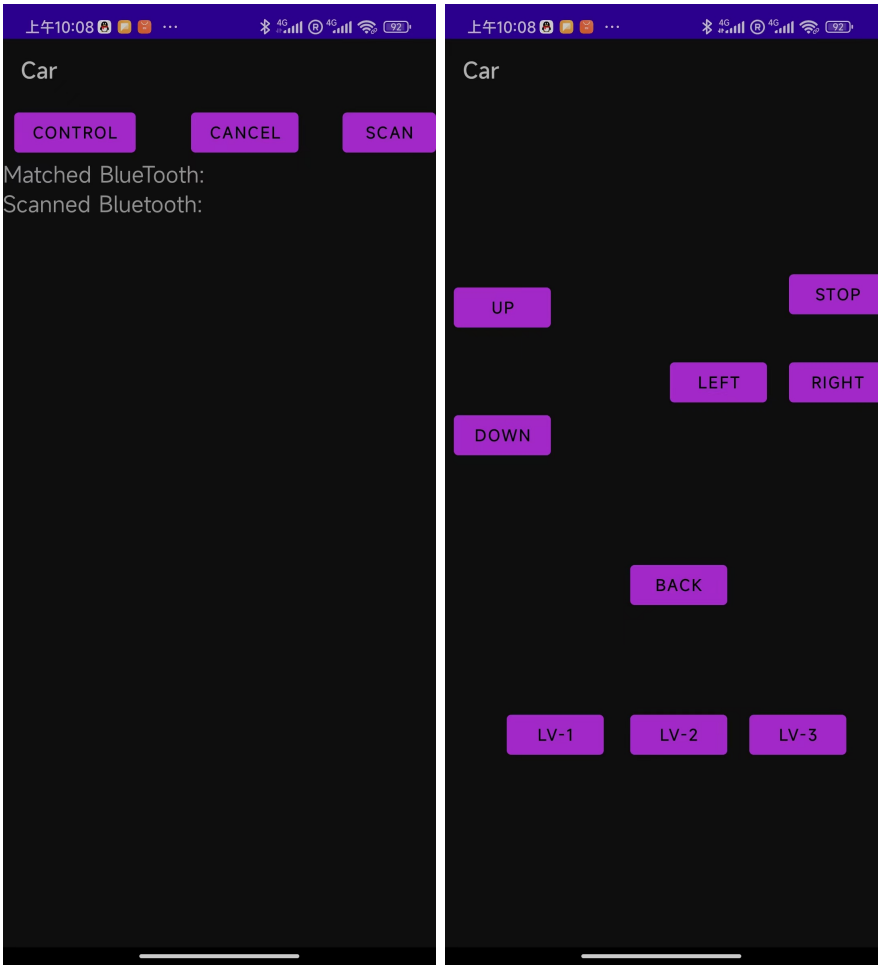

上午10:08
Car
CONTROL
CANCEL
SCAN
Matched BlueTooth:
Scanned Bluetooth:
上午10:08
Car
UP
STOP
LEFT
RIGHT
DOWN
BACK
LV-1
LV-2
LV-3

Рис. 7:

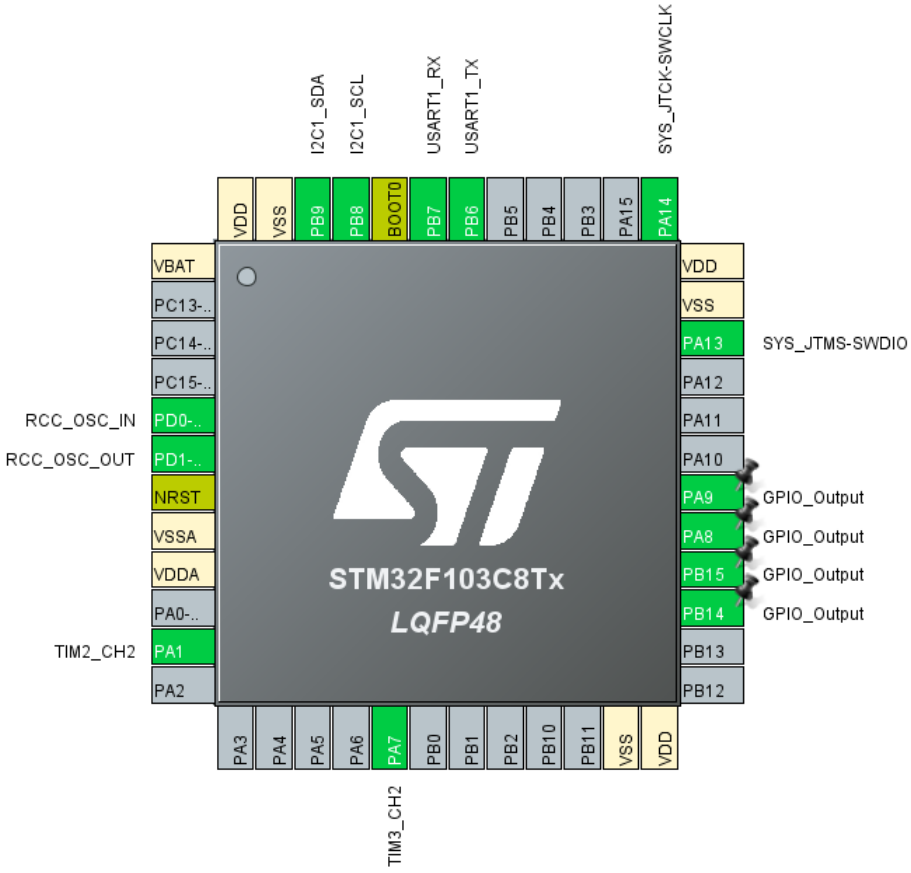

I2C1_SDA
I2C1_SCL
USART1_RX
USART1_TX
SYS_JTCK-SWCLK
VDD
VSS
PB9
PB8
BOOT0
PB7
PB6
PB5
PB4
PB3
PA15
PA14
VBAT
PC13-..
PC14-..
PC15-..
RCC_OSC_IN
PD0-..
RCC_OSC_OUT
PD1-..
NRST
VSSA
VDDA
PA0-..
TIM2_CH2
PA1
PA2
VDD
VSS
PA13
SYS_JTMS-SWDIO
PA12
PA11
PA10
PA9
GPIO_Output
PA8
GPIO_Output
PB15
GPIO_Output
PB14
GPIO_Output
PB13
PB12
STM32F103C8Tx
LQFP48
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB10
PB11
VSS
VDD
TIM3_CH2

Рис. 8:

Направление-Скорость	char
right-back-1	S
right-front-1	O
left-back-1	L
left-front-1	I
front-1	C
back-1	F
right-back-2	Q
right-front-2	N
left-back-2	K
left-front-2	H
front-2	B
back-2	E
right-back-3	P
right-front-3	M
left-back-3	J
left-front-3	G
front-3	A
back-3	D
stop	X

Приложение В список Детали

плата stm32
Bluetooth-модуль
Bluetooth-модуль hc-08(ble)
Моторный привод
Мотор*2
oled
электропровод