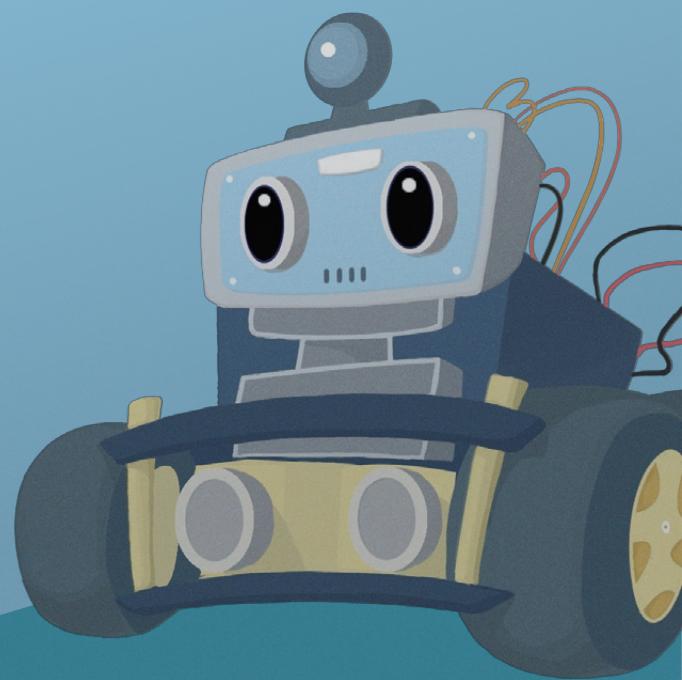
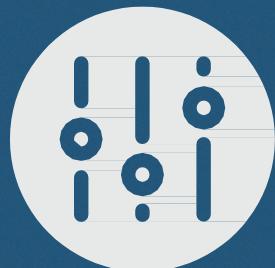


7

# SMART ROBOT CAR V4.0 WITH CAMERA



## Others





## Введение:

- + В этом уроке мы расскажем о функциях и особенностях использования всех остальных компонентов.

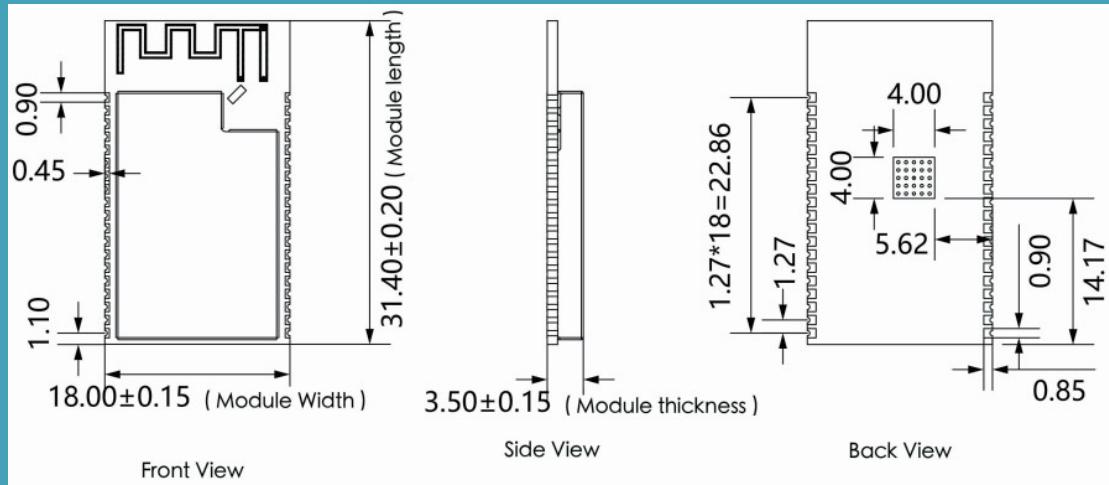


## Подготовка:

- + Умный автомобиль-робот (с аккумулятором) Провод USB



## Во-первых, использование модуля Wi-



 Wi-Fi можно понимать просто как беспроводную связь и является одной из наиболее распространенных на сегодняшний день технологий передачи данных по беспроводным сетям.

По сравнению с Bluetooth, Wi-Fi имеет более широкую зону покрытия, более высокую скорость передачи данных и способен поддерживать связь "многие-ко-многим". В нашем комплекте модуль Wi-Fi припаян к плате ESP32. Поскольку драйвер модуля Wi-Fi сложен в использовании, мы опасаемся, что у пользователей будет плохой опыт из-за неправильной эксплуатации, поэтому мы уже припаяли модуль Wi-Fi и модуль камеры к ESP32 перед отправкой с завода, и пока не будем обучать этой части ESP32.

Процесс заключается в использовании Wi-Fi модуля ESP32 для открытия точки доступа Wi-Fi и отправке информации через APP после подключения телефона к точке доступа. Информация передается через Wi-Fi модуль телефона, а затем принимается Wi-Fi модулем ESP32. Так как перед отправкой с завода мы перевели Wi-Fi модуль в сквозной режим, то полученные данные с Wi-Fi модуля ESP32 передаются на последовательный порт ESP32. А поскольку последовательный порт ESP32 соединен с последовательным портом UNO, то для получения данных из APP нам достаточно считать последовательный порт UNO.

 Что касается программы UNO, то мы получаем строку путем чтения данных, поступающих в последовательный буфер, и затем разбираем эту строку. Ключевая информация извлекается через фиксированный формат строки, в соответствии с различной ключевой информацией устанавливается бит флага, и выбираются различные функции для вызова, реализующие соответствующую функцию.

Формат строки (формат данных json) следующий: {"N": 2, "D1": 1}, причем параметры, соответствующие "N", представляют собой различные функции, а такие параметры, как "D1" и "D2", - параметры, которые необходимо изменять и настраивать при реализации различных функций.

Конкретные инструкции по обмену данными см. в разделе "Протокол обмена данными для Smart Robot Car" в папке "04 Информация о микросхемах".

 Теперь откройте **Demo1** в текущей папке:

 Для начала рассмотрим определение относительных функций чтения данных последовательного порта.

**C:/** // в ApplicationFunctionSet\_xxx0.h

```
класс ApplicationFunctionSet
{
    общественность:
        void ApplicationFunctionSet_Init(void);
        void ApplicationFunctionSet_SerialPortDataAnalysis(void);
        String CommandSerialNumber;
};
```

 Затем необходимо добавить библиотеку JSON для анализа JSON-данных, передаваемых APP.

**C:/** // в ApplicationFunctionSet\_xxx0.cpp #include  
"ArduinoJson-v6.11.1.h" //ArduinoJson

 Инициализация  
последовательного  
порта.

**C:/** // в ApplicationFunctionSet\_xxx0.cpp

```
void ApplicationFunctionSet::ApplicationFunctionSet_Init(void)
{
    Serial.begin(9600);
}
```

- ⊕ Доступ к полной инструкции из последовательного порта.

```
C:/ // в файле
ApplicationFunctionSet_xxx0.cpp
void
ApplicationFunctionSet::ApplicationFunctionSet_SerialPortDataAnalysis(void)
{
    static String SerialPortData = "";
    uint8_t c = "";
    if (Serial.available() > 0)
    {
        while (c != '}' && Serial.available() > 0)
        {
            c = Serial.read();
            SerialPortData += (char)c;
        }
        .....
    }
}
```

`Serial.available()`: определяет наличие данных в буфере последовательного порта.

`Serial.read()`: чтение по одному байту за раз.

⊕ Поскольку формат строки ([формат данных JSON](#)) следующий: `{"N": 2, "D1": 1}`, нам достаточно определить, является ли завершающим символом `"}`, чтобы понять, получили ли мы всю команду.

```
C:/ // в файле
ApplicationFunctionSet_xxx0.cpp
void
ApplicationFunctionSet::ApplicationFunctionSet_SerialPortDataAnalysis(void)
{
    .....
    if (c == '}')
    {
    }
    .....
}
```

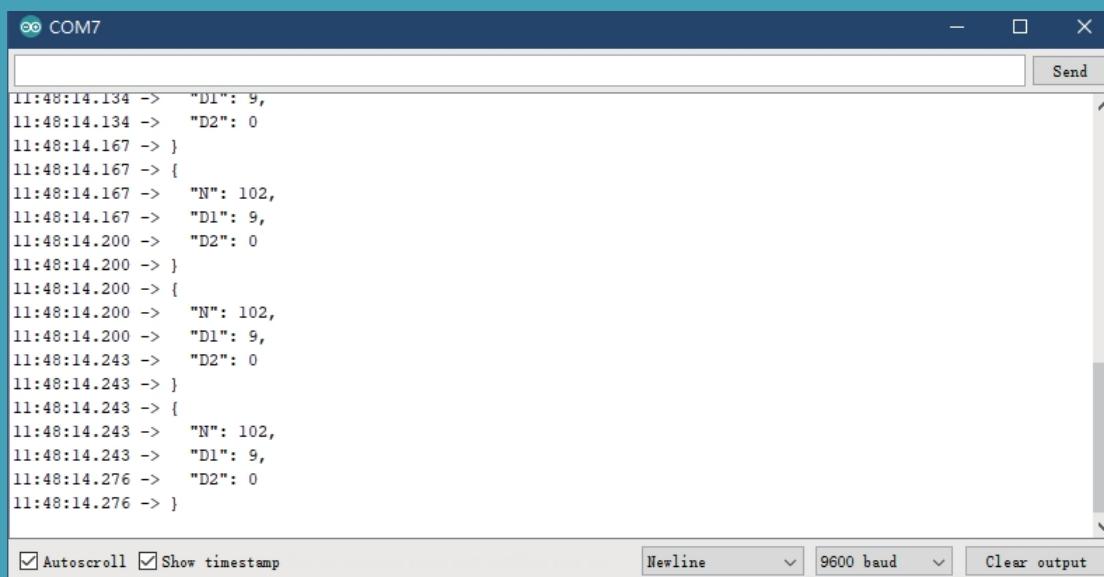
 После успешного получения команды ее необходимо разобрать.

```
● ● ● // в файле
C:/ ApplicationFunctionSet_xxx0.cpp
void ApplicationFunctionSet::ApplicationFunctionSet_SerialPortDataAnalysis(void)
{
.....
else if (!error)
{
    int control_mode_N = doc["N"];
    char buf[3];
    uint8_t temp = doc["H"];
    sprintf(buf, "%d", temp);
    CommandSerialNumber = buf;
.....
}
```

 Каждый случай представляет собой один режим, и для реализации функции переключения APP нам достаточно подставить функции, которые мы реализовали в предыдущих уроках.

```
● ● ● // в файле
C:/ ApplicationFunctionSet_xxx0.cpp
void ApplicationFunctionSet::ApplicationFunctionSet_SerialPortDataAnalysis(void)
{
.....
switch (control_mode_N)
{
    case 1:
        break;
    case 2:
.....
}
```

**+** Загрузите программу. (После успешной загрузки программы не отсоединяйте USB-кабель. Переключите кнопку "Upload-Cam" в положение "Cam", затем откройте APP мобильного устройства и подключитесь к автомобилю. После завершения подключения не спеша нажмите кнопку в APP. К этому времени откройте монитор последовательного порта в Arduino IDE, и вы обнаружите, что каждая операция, выполняемая в APP, посыпает строку в автомобиль, а автомобиль выполняет соответствующую функцию в соответствии с этими инструкциями.



The screenshot shows the Arduino Serial Monitor window titled "COM7". The text area contains a continuous stream of data, likely JSON objects, being transmitted. The data includes timestamps like "11:48:14.134", variable names like "D1", "D2", and values like "9" and "0". The bottom of the window shows checkboxes for "Autoscroll" and "Show timestamp", and buttons for "Newline", "9600 baud", and "Clear output".

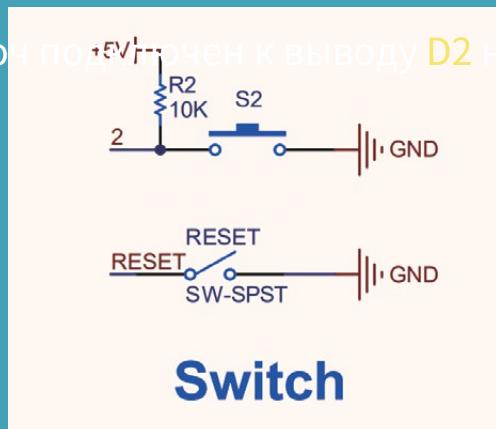


## Во-вторых, использование

**+** Мы также можем переключать функции с помощью клавиш на нашем Smart Robot Car.

Для получения более подробной информации откройте папку на предыдущем уровне: [Информация о микросхеме -> SmartRobot-Shield.pdf](#)

Как мы все знаем, ключ подключен к выводу D2 на UNO.



- + Далее откройте программу Demo2:
- + Сначала рассмотрим определение относительных выводов и переменных:

C:/ // в DeviceDriverSet\_xxx0.h

```
/*Обнаружение клавиш*/
class DeviceDriverSet_Key
{
    общественность:
        void DeviceDriverSet_Key_Init(void);
    #if _Test_DeviceDriverSet
        void DeviceDriverSet_Key_Test(void);
    #endif
        void DeviceDriverSet_key_Get(uint8_t *getKeyValue);
    public:
        #define PIN_Key 2
        #define keyValue_Max 4
    public:
        static uint8_t keyValue;
};
```

- + В последующем программировании мы будем использовать функцию прерывания.
- + Обычно наши программы выполняются последовательно, но прерывания используются, когда нужно решить что-то более срочное.
- + Теперь нам необходимо добавить соответствующую библиотеку:

C:/ // в DeviceDriverSet\_xxx0.h

```
#include "PinChangeInt.h"
```

- + Затем определите ключевой вывод и добавьте к нему функцию прерывания.

Мы установим его как триггер по падающему фронту. При нажатии клавиши уровень изменяется, и по падающему фронту срабатывает функция прерывания.

C:/

```
// в DeviceDriverSet_xxx0.cpp

void DeviceDriverSet_Key::DeviceDriverSet_Key_Init(void)
{
    pinMode(PIN_Key, INPUT_PULLUP);
    attachPinChangeInterrupt
    (PIN_Key, attachPinChangeInterrupt_GetKeyValue, FALLING);
}
```

- + И нам нужно записать, сколько раз была нажата клавиша. Мы можем записать это не более чем в четвертый раз, поскольку всего у нас 4 режима.

C:/

```
// в DeviceDriverSet_xxx0.cpp

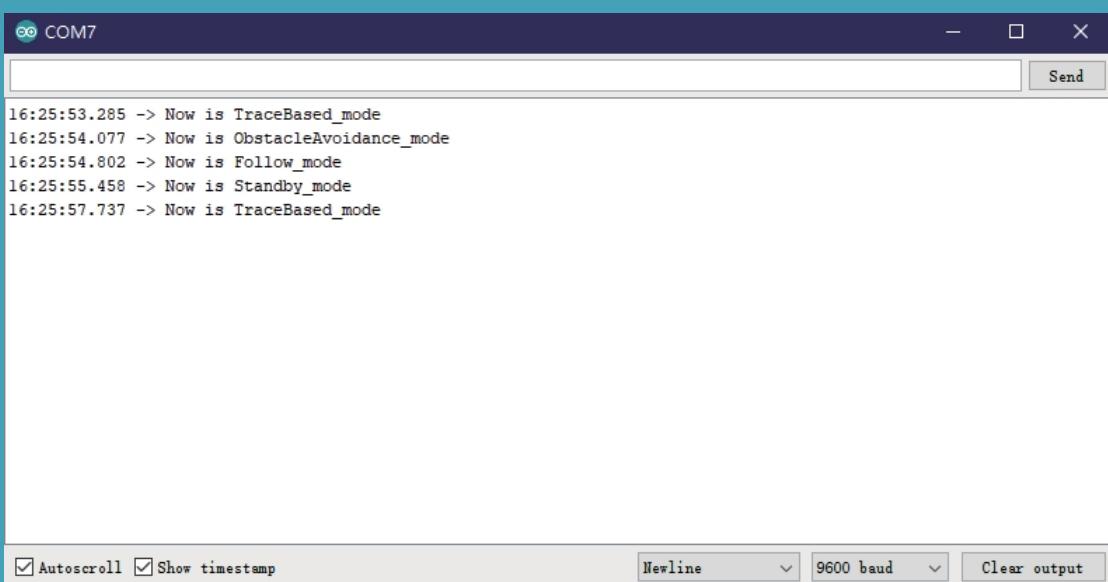
static void attachPinChangeInterrupt_GetKeyValue(void)
{
    DeviceDriverSet_Key Ключ;
    static uint32_t keyValue_time = 0;
    static uint8_t keyValue_temp = 0;
    if ((millis() - keyValue_time) > 500)
    {
        keyValue_temp++;
        keyValue_time = millis();
        if (keyValue_temp > keyValue_Max)
        {
            keyValue_temp = 0;
        }
        Key.keyValue = keyValue_temp;
    }
}
```

- ⊕ При нажатии клавиши срабатывает функция прерывания, и в зависимости от времени нажатия клавиши выбирается соответствующая функция.

C:/ // в ApplicationFunctionSet\_xxx0.cpp

```
void ApplicationFunctionSet::ApplicationFunctionSet_KeyCommand(void)
```

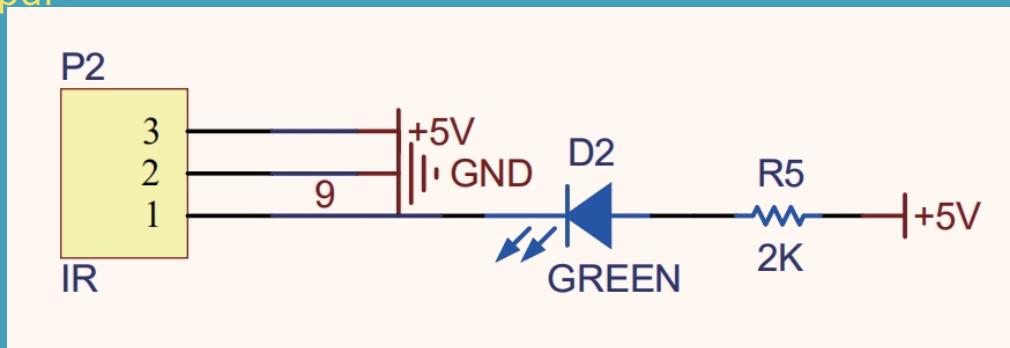
- ⊕ Загрузка программы. (После успешной загрузки программы откройте последовательный порт и нажмите клавишу, на экране появится сообщение о том, что программа переключается на различные функции. Так как в данном разделе основное внимание уделяется использованию клавиш, мы только выводим на последовательный порт сообщение о текущем режиме переключения и не реализуем соответствующую конкретную функцию, но вы можете добавить различные функции в соответствии с предыдущими уроками.



## В-третьих, использование инфракрасных приемных и передающих модуль.

- ⊕ Кроме нажатия кнопок для переключения функций, мы также можем переключать функции нашего Smart Robot Car с помощью инфракрасного пульта дистанционного управления, поскольку в нем применен инфракрасный приемный модуль.

Для получения более подробной информации откройте папку на предыдущем уровне: Информация о микросхеме -> SmartRobot-Shield.pdf



Как показано на рисунке, инфракрасный приемный модуль подключен к выводу D9 UNO, также мы подключили его к зеленому светодиодному индикатору. При нажатии кнопки пульта дистанционного управления, если инфракрасный приемный модуль получит сигнал, загорится светодиод.

Инфракрасный пульт дистанционного управления состоит в основном из инфракрасного передатчика и инфракрасного приемника.

Сигналы, передаваемые и принимаемые инфракрасным излучением, фактически представляют собой серию двоичных импульсных кодов, причем высокий и низкий уровень изменяются по определенному временному правилу для передачи соответствующей информации. Для защиты от помех со стороны других сигналов при беспроводной передаче сигнал обычно модулируется на определенной несущей частоте ([инфракрасный несущий сигнал 38К](#)) и передается через инфракрасный излучающий диод, а на приемном инфракрасном устройстве сигнал должен быть декодирован и скорректирован, а затем восстановлен в двоичный импульсный код для обработки.

 Далее откройте программу Demo3:

 Рассмотрим определение относительных выводов и переменных:

C:/ // в DeviceDriverSet\_xxx0.h class

```
DeviceDriverSet_IRecv
{
    общественность:
        void DeviceDriverSet_IRecv_Init(void);
        bool DeviceDriverSet_IRecv_Get(uint8_t *IRecv_Get /*out*/);
        void DeviceDriverSet_IRecv_Test(void);

    общественность:
        unsigned long IR_Premillis;

    частный:
        #define RECV_PIN 9
        .....
};
```

+ Затем, в соответствии с полученным сигналом, нам необходимо выделить различные ключи.

C:/ // в DeviceDriverSet\_xxx0.h

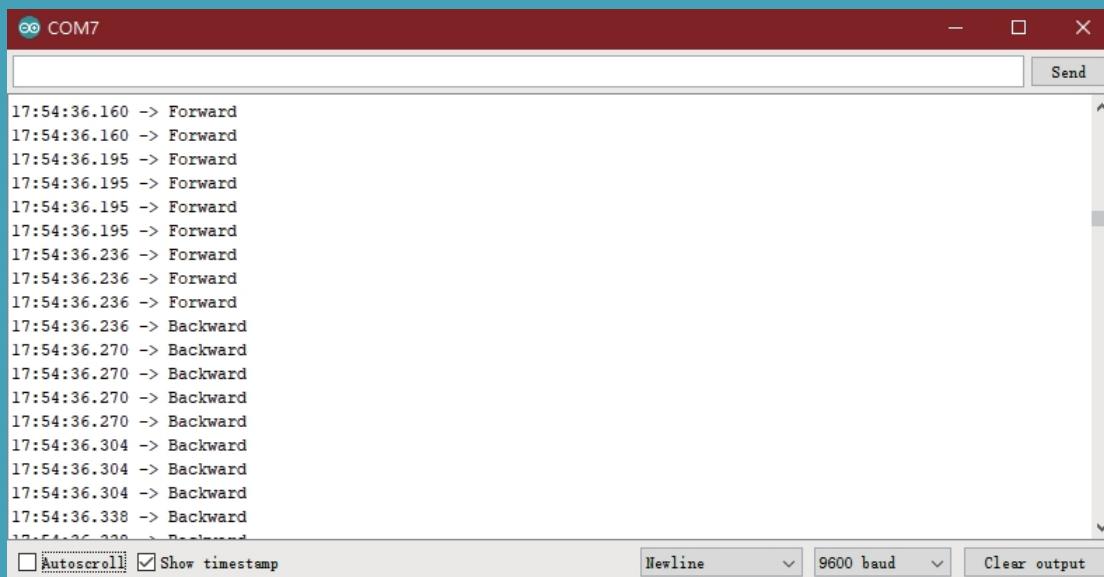
```
bool DeviceDriverSet_IRecv::DeviceDriverSet_IRecv_Get
(uint8_t *IRecv_Get /*out*/)
```

+ Наконец, реализовать различные функции в зависимости от нажатой клавиши.

C:/ // в ApplicationFunctionSet\_xxx0.cpp

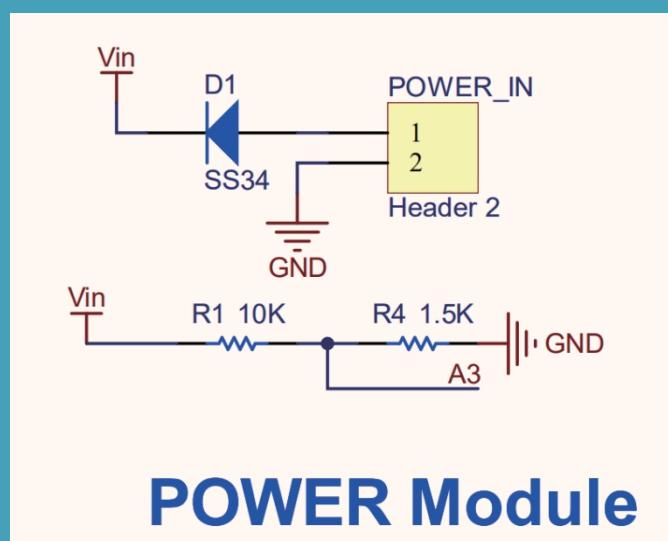
```
void ApplicationFunctionSet::ApplicationFunctionSet_IRecv(void)
```

**+** Загрузка программы. (После успешной загрузки программы откройте последовательный порт и нажмите на клавишу, в результате чего на экране появится сообщение о переключении различных функций. Так как в данном разделе основное внимание уделяется использованию клавиш, мы только выводим на последовательный порт сообщение о текущем режиме переключения и не реализуем соответствующие специфические функции, но вы можете добавить различные функции в соответствии с предыдущими уроками.



## Далее, определение напряжения.

**+** Для того чтобы предотвратить повреждение батареи в результате чрезмерного разряда, необходимо контролировать напряжение. Поэтому сейчас мы научим вас измерять напряжение батареи.



Вывод Arduino для сбора напряжения батареи обозначим как вывод A3. Поскольку внутреннее опорное напряжение микросхемы Arduino равно 5 В, то есть независимо от величины напряжения питания, значение 0~1023, сэмплируемое АЦП, соответствует значению напряжения 0~5 В. А при напряжении более 5 В значение по-прежнему равно 1023. В документе AVR также указывается, что это эталонное значение имеет некоторое отклонение. Но значение опорного напряжения 5 В слишком мало, и измеряемое напряжение должно быть меньше его. Поэтому давайте разделим напряжение, которое необходимо проверить. Здесь мы используем последовательно **с о е д и н е н н ы е** резисторы 10к и 1,5к, а затем для измерения подключаем вывод A3 к середине двух резисторов. В это время значение напряжения на одной восьмой должно быть ниже 1,1 В.

- + Теперь откройте программу **Demo4**:
- + Во-первых, нам необходимо инициализировать используемый вывод.

```
C:/ void voltagelnit()
{
    pinMode(VOL_MEASURE_PIN, INPUT);
}
```

- + Затем рассчитайте напряжение питания по формуле.

$$V_{out} = (V_{in} \times R_2) / (R_1 + R_2)$$

Затем собираем значение A3 с помощью функции **analogRead()**. Мы знаем, что контроллер Arduino имеет несколько 10-цифровых каналов модульного преобразования. Это означает, что Arduino может преобразовывать входные сигналы напряжения 0-5 В в значения 0-1023. Другими словами, мы можем разделить сигнал 5 В на 1024 равные части, при этом входному сигналу 0 В будет соответствовать значение 0, а входному сигналу 5 В - 1023. Для осуществления контроля напряжения недостаточно использовать функцию **analogRead()** для считывания значения A3. Необходимо также преобразовать считанное значение в фактическое значение напряжения, поэтому для преобразования можно воспользоваться следующей формулой:

$$\text{Напряжение батареи } V = \text{считанное значение A3} * (5.00 / 1024.00) *$$

множитель



## Подведем итоги:

Напряжение батареи  $V = \text{считанное значение A3} * (5,00 / 1024,00) * ((R1 + R2) / R2)$



Поскольку данные могут быть пропущены программой, а также возможны ошибки, если десятичная дробь при делении данных на 1024 будет слишком большой, лучше разделить данные на 1024 в конце.

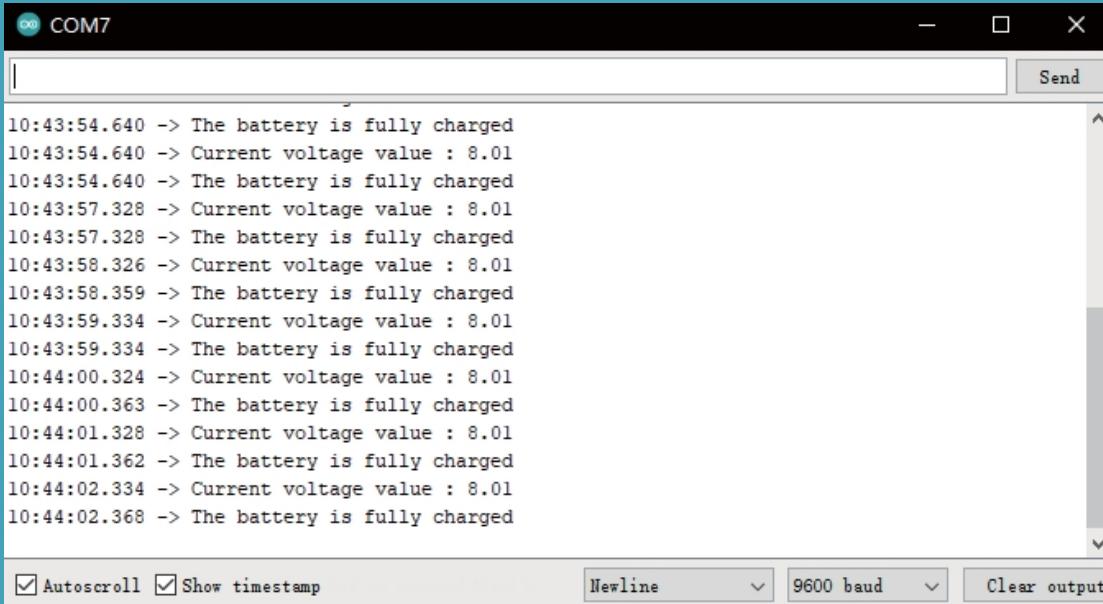
 Напряжение батареи  $V = \text{считанное значение A3} * (5.00) * ((R1 + R2) / 1024.00)$

 Наконец, необходимо исправить ошибку.

 Напряжение батареи  $V = \text{Напряжение батареи V} + (\text{Напряжение батареи V} * 0,08)$

```
void Voltage_Measure()
{
    if (millis() - vol_measure_time > 1000) //Измерения производятся каждые 1000
        миллисекунд
    {
        vol_measure_time = millis();
        float voltage =
            (analogRead(VOL_MEASURE_PIN) * 5) * ((10 + 1.5) / 1.5) / 1024;
        //Считывание значения напряжения
        //float voltage = (analogRead(VOL_MEASURE_PIN) * 0.0375);
        voltage = voltage + (voltage * 0.08);
        Serial.print("Текущее значение
                    напряжения : ");
        Serial.println(voltage); if(voltage>7.8)
            Serial.println("Аккумулятор полностью
                        заряжен"); else
            Serial.println("Разряжена батарея");
    }
}
```

 Загрузка  (После успешной загрузки программы откройте последовательный порт, и вы увидите текущее значение напряжения.



```
10:43:54.640 -> The battery is fully charged
10:43:54.640 -> Current voltage value : 8.01
10:43:54.640 -> The battery is fully charged
10:43:57.328 -> Current voltage value : 8.01
10:43:57.328 -> The battery is fully charged
10:43:58.326 -> Current voltage value : 8.01
10:43:58.359 -> The battery is fully charged
10:43:59.334 -> Current voltage value : 8.01
10:43:59.334 -> The battery is fully charged
10:44:00.324 -> Current voltage value : 8.01
10:44:00.363 -> The battery is fully charged
10:44:01.328 -> Current voltage value : 8.01
10:44:01.362 -> The battery is fully charged
10:44:02.334 -> Current voltage value : 8.01
10:44:02.368 -> The battery is fully charged
```

Autoscroll  Show timestamp

## Последнее - использование цветного RGB-света.

 Мы научим вас включать цветной свет на автомобиле, менять его яркость и цвет.

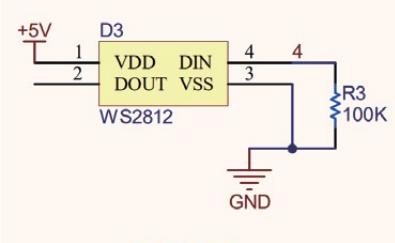
В нашем комплекте для реализации световых эффектов используется модуль WS2812B. WS2812B - это интеллектуальный светодиодный источник света с внешним управлением, объединяющий в себе схему управления и светоизлучающую схему. В качестве протокола передачи данных используется однопроводной метод обмена кодами с возвратом к нулю. После включения и сброса пикселя на DIN-клемму поступают данные, передаваемые от контроллера.

Первые переданные 24-битные данные извлекаются первым пикселем и отправляется на защелку данных внутри пикселя, оставшиеся данные формируются и усиливаются внутренней схемой обработки формирования, а затем передаются через порт DO для начала вывода на следующий каскад пикселей, причем сигнал уменьшается на 24 бита для каждого пикселя, проходящего через передачу.

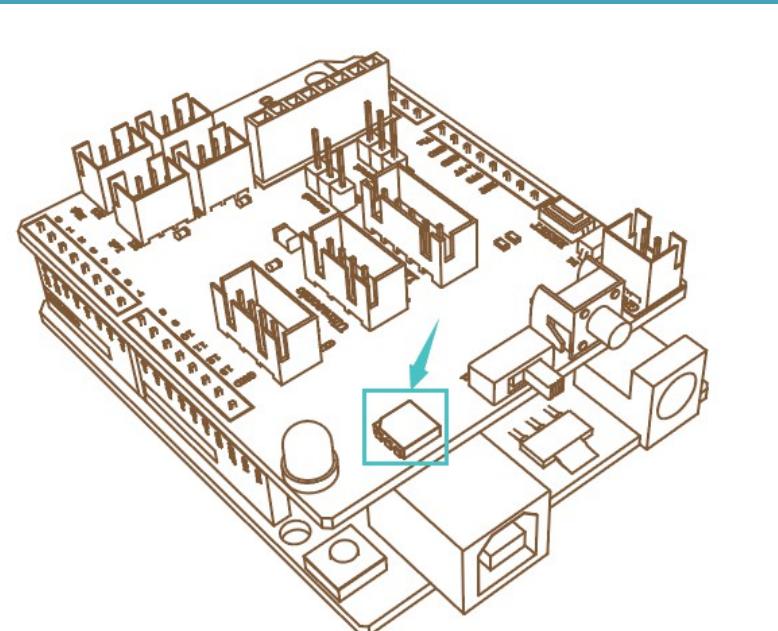
В пикселе применена технология автоматического формирования и переадресации, благодаря чему количество каскадов пикселя не ограничивается передачей сигнала, а только требованиями к скорости передачи сигнала. Светодиод обладает такими преимуществами, как низкое напряжение питания, защита окружающей среды и



Для получения более подробной информации откройте папку на предыдущем уровне: Информация о микросхеме -> SmartRobot-Shield



**RGB**



Из рисунка видно, что цветная RGB-подсветка подключена к выводу D4 UNO

- + Откройте **Demo5** в текущей папке:
- + В этом уроке мы будем использовать библиотеку **FastLED.h**.

**C:/** #include "FastLED.h"

- + Определите относительные переменные.

**C:/** #define PIN\_RGBLED 4  
#define NUM\_LEDS 1  
CRGB leds[NUM\_LEDS];

 Инициализируйте WS2812 и установите яркость свечения светодиода.

```
C:/ void setup() {  
    Serial.begin(9600);  
    FastLED.addLeds<NEOPIXEL, PIN_RGBLED>(leds, NUM_LEDS);  
    FastLED.setBrightness(20);  
}
```

 Объединяет три 1-байтовых цветовых данных в новый цвет.

```
C:/ uint32_t Color(uint8_t r, uint8_t g, uint8_t b)  
{  
    return (((uint32_t)r << 16) | ((uint32_t)g << 8) | b);  
}
```

 Наконец, измените цвет по своему усмотрению.

```
C:/ Void myColor()
```

 Загрузите программу. (При загрузке программы **переключите кнопку "Upload-Cam" на "Upload"**) После успешной загрузки программы откройте последовательный порт, и вы увидите, что цвет RGB-подсветки постепенно меняется.