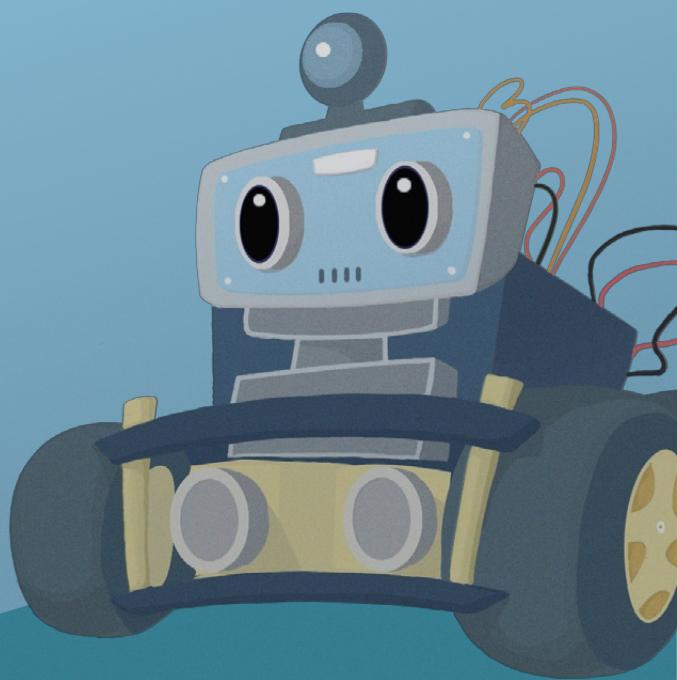


SMART ROBOT CAR V4.0 WITH CAMERA

2



Movement





Введение:

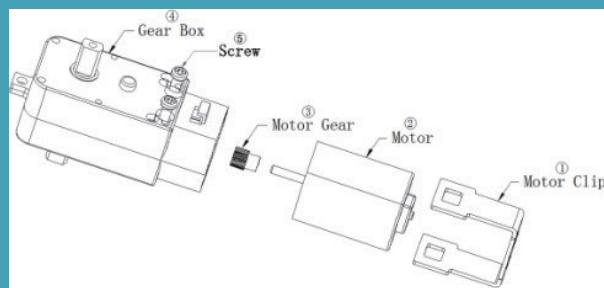
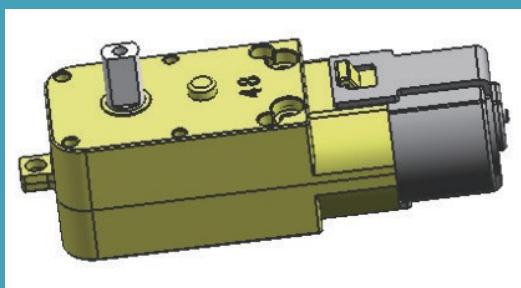
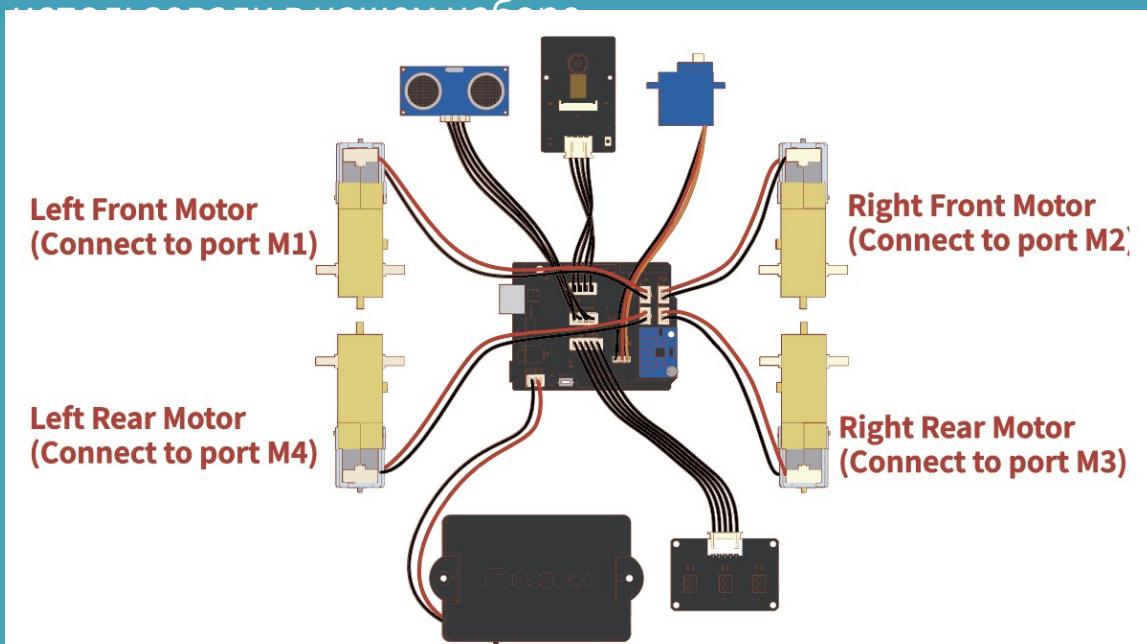
- + В этом уроке мы научим вас, как заставить автомобиль двигаться по своему желанию, управляя мотором.



Подготовка материалов:

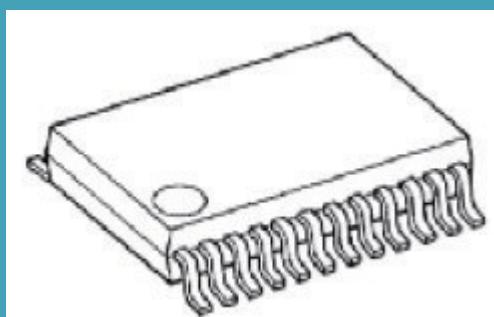
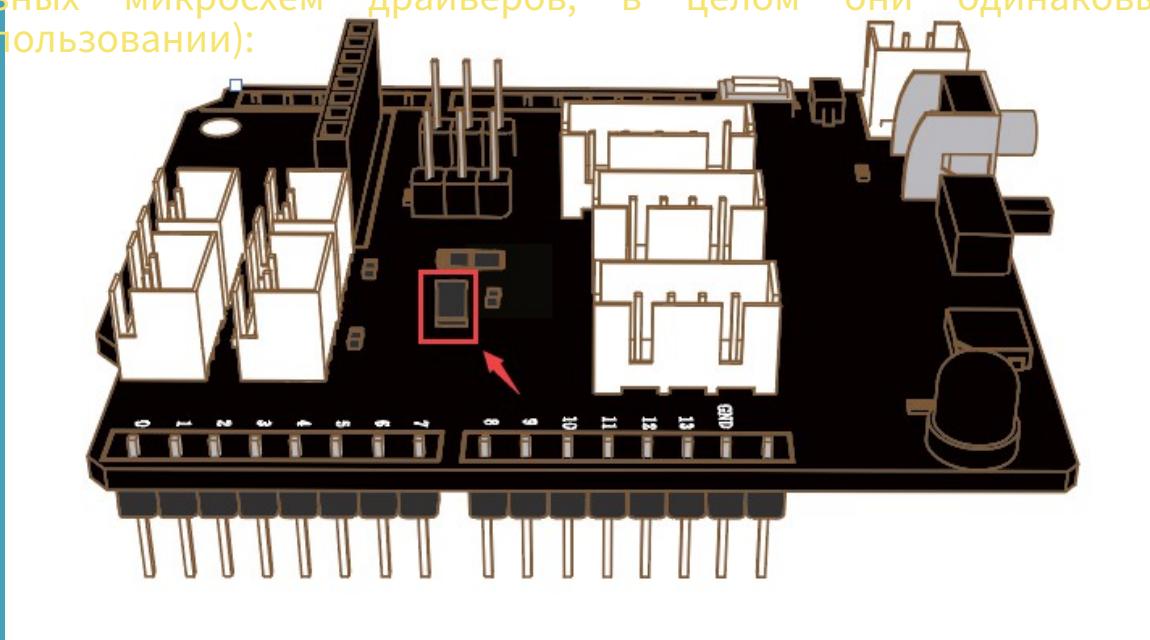
- + Умный робот-автомобиль (с аккумулятором) Кабель USB

- + Прежде всего, давайте посмотрим на 4 двигателя, которые мы будем использовать в автомобиле.



 Двигатель постоянного тока является индуктивной нагрузкой большого тока (нагрузка с параметрами индуктивности), а используемый нами порт ввода-вывода однокристального микрокомпьютера Arduino UNO имеет слабую нагрузочную способность (**способность к выходу по току**), и мощности привода не хватает для подачи тока, достаточного для вращения двигателя.

Поэтому мы решили использовать микросхему драйвера двигателя TB6612. (Хотя возможны небольшие различия в производительности разных микросхем драйверов, в целом они одинаковы в использовании):



 TB6612 - это устройство драйвера двигателя постоянного тока с сильноточной мостовой структурой MOSFET-Н и двухканальным выходом для одновременного управления двумя двигателями. Оно обладает двумя преимуществами :

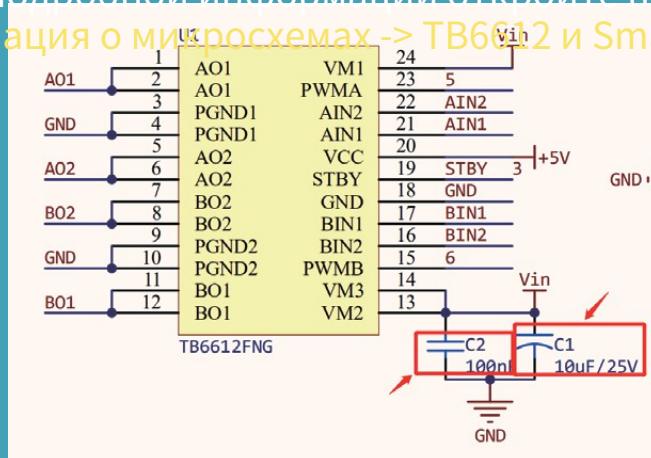
1 TB6612 обеспечивает достаточный ток для двигателя. Каждый Н-мост TB6612 может обеспечить выходной ток до 1,5 А. Он работает в диапазоне напряжений питания двигателя от 0 до 11 В и в диапазоне напряжений питания устройства от 2 до 7 В.

2 TB6612 выполняет функцию изоляции для предотвращения повреждения устройств управления импульсным током, генерируемым двигателем.

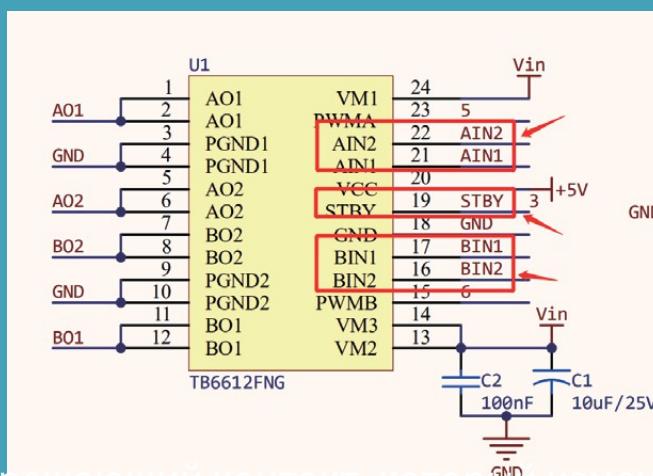
 Возможно, все больше знакомы с L298N. На самом деле, их применение в принципе одинаково. По сравнению с L298N схема теплоотвода и периферийного диода свободного хода не требует внешнего радиатора, а периферийная схема проста. Ему требуется только внешний конденсатор фильтра питания.

Прямой привод двигателя позволяет уменьшить размеры системы, а частота до 100 КГц вполне удовлетворяет большинству наших потребностей.

Для получения подробной информации откройте папку последнего уровня: Информация о микросхемах -> TB6612 и SmartCar-Shield-V1.1



Кроме того, на этой схеме видны выводы TB6612, подключенные к UNO.



EN означает разрешающий контакт, который изменяет скорость вращения двигателя путем изменения ШИМ (0~255).

РН обозначает фазовый вывод, и направление вращения (положительное/обратное) двигателя изменяется путем изменения уровня вывода (0/1).

В двигателе А PWMA подключен к выводам UNO~D5, а AIN1 - к выводу на вывод D7 UNO.

В двигателе В PWMB подключен к выводам UNO~D6, а BIN1 - к выводу UNO D8.

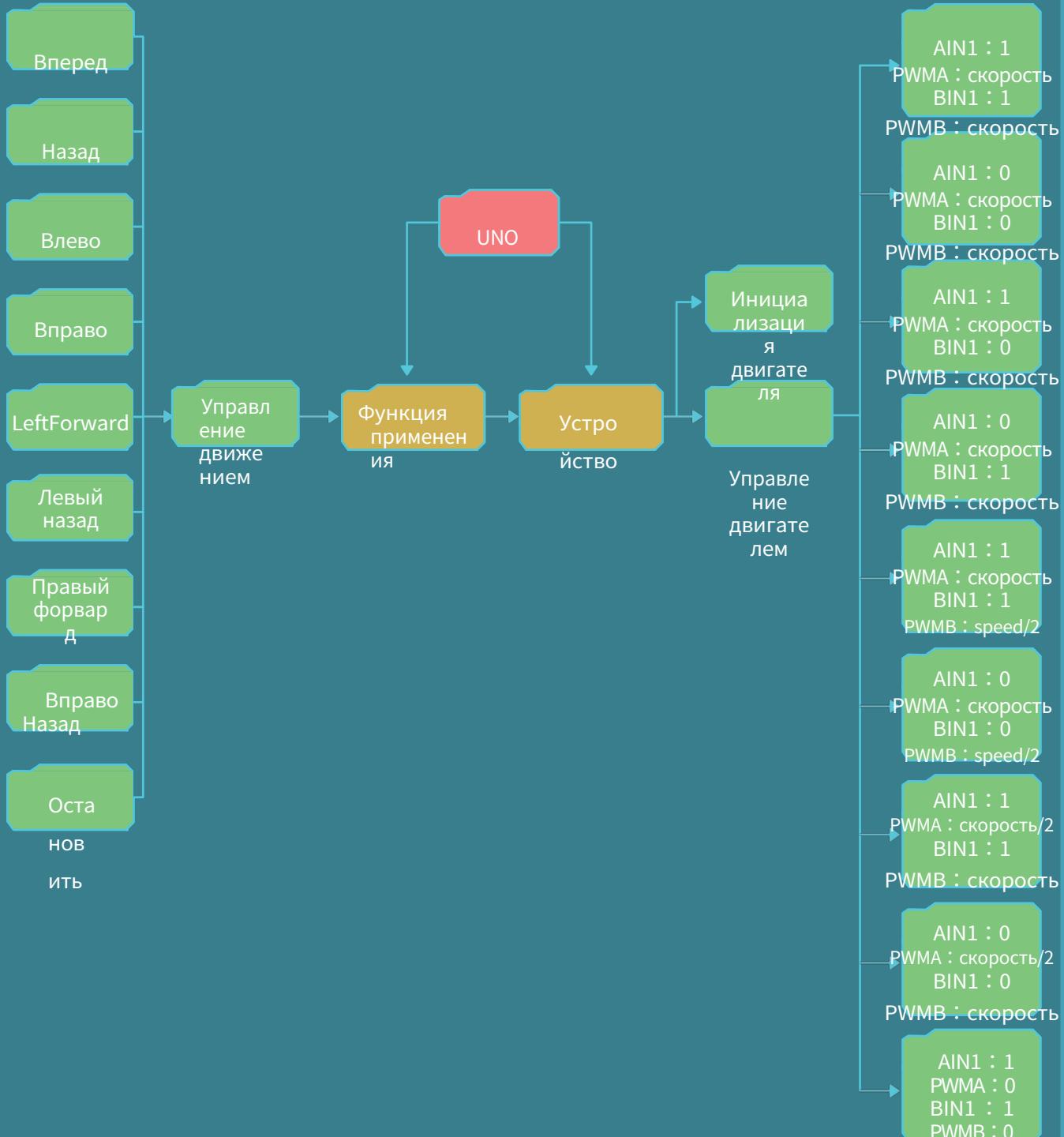
Наконец, метод управления может быть получен из информации о микросхеме Related

->TB6612.

- + После этого можно приступать к написанию программы в соответствии с приведенной ниже блок-схемой.

Откройте Demo1 в текущей папке:

Принцип реализации Move



- + Прежде всего, мы видим, что существует четыре файла.

1. Demo1 2. ApplicationFunctionSet_xxx0.cpp 3. DeviceDriverSet_xxx0.cpp 4. DeviceDriverSet_xxx0.h

+ Далее рассмотрим определение соответствующих выводов и

● переменных:

C:

// в DeviceDriverSet_xxx0.h

/*Мотор*/

класс DeviceDriverSet_Motor

{

общественность:

void DeviceDriverSet_Motor_Init(void);

#if _Test_DeviceDriverSet

void DeviceDriverSet_Motor_Test(void);

#endif

void DeviceDriverSet_Motor_control(boolean direction_A, uint8_t speed_A,

boolean direction_B, uint8_t speed_B,

логическое значение controlED

);

частный:

#define PIN_Motor_PWMA 5

#define PIN_Motor_PWMB 6

#define PIN_Motor_BIN_1 8

#define PIN_Motor_AIN_1 7

#define PIN_Motor_STBY 3

общественность:

#define speed_Max 255

#define direction_just true

#define direction_back false

#define direction_void 3

#define Duration_enable true

#define Duration_disable false

#define control_enable true

#define control_disable false

};

+ Определите максимальную скорость
вращения двигателя :

C:/

// в DeviceDriverSet_xxx0.h

#define speed_Max 255

+ Определение переменных флагов

+ Флаг направления вращения двигателя

C:/ // в DeviceDriverSet_xxx0.h

```
#define direction_just true  
#define direction_back false  
#define direction_void 3
```

+ Другие функциональные флаги

C:/ // в DeviceDriverSet_xxx0.h

```
#define Duration_enable true  
#define Duration_disable false  
#define control_enable true  
#define control_disable false
```

+ Перед началом работы эти выводы должны быть инициализированы.

C:/ // в DeviceDriverSet_xxx0.cpp

```
extern DeviceDriverSet_Motor AppMotor;  
void DeviceDriverSet_Motor::DeviceDriverSet_Motor_Init(void)  
{  
    pinMode(PIN_Motor_STBY, OUTPUT);  
    pinMode(PIN_Motor_PWMA, OUTPUT);  
    pinMode(PIN_Motor_PWMB, OUTPUT);  
    pinMode(PIN_Motor_AIN_1, OUTPUT);  
    pinMode(PIN_Motor_BIN_1, OUTPUT);  
}
```

 И назовите это
"настройкой".

```
C:/ DeviceDriverSet_Motor AppMotor;  
void setup()  
{  
    AppMotor.DeviceDriverSet_Motor_Init();  
}
```

 После инициализации выводов мы можем изменить их состояние для управления двигателем и заставить его вращаться так, как нам нужно.

1. Движение автомобиля

```
C:/ //in ApplicationFunctionSet_xxx0.cpp  
static void ApplicationFunctionSet_SmartRobotCarMotionControl  
(SmartRobotCarMotionControl direction, uint8_t is_speed)
```

параметр:

направление: Прохождение последовательности управления
направлением движения. is_speed : 0~255

```
C:/ //in ApplicationFunctionSet_xxx0.cpp
```

```
/*Последовательность управления  
направлением движения*/ enum  
SmartRobotCarMotionControl  
{  
    Вперед,      //(1)  
    Назад,      //(2)  
    Влево,      //(3)  
    Right,      //(4)  
    LeftForward, //(5)  
    LeftBackward, //(6)  
    RightForward, //(7)  
    RightBackward, //(8)  
    stop_it      //(9)  
};
```

2. Управление двигателем

```
C:/ // в DeviceDriverSet_xxx0.cpp
void DeviceDriverSet_Motor::DeviceDriverSet_Motor_control
    (boolean direction_A, uint8_t speed_A,
     //двигатель A boolean direction_B, uint8_t
     speed_B, //двигатель boolean controlED //AB
     enable true
    )
```

параметр:

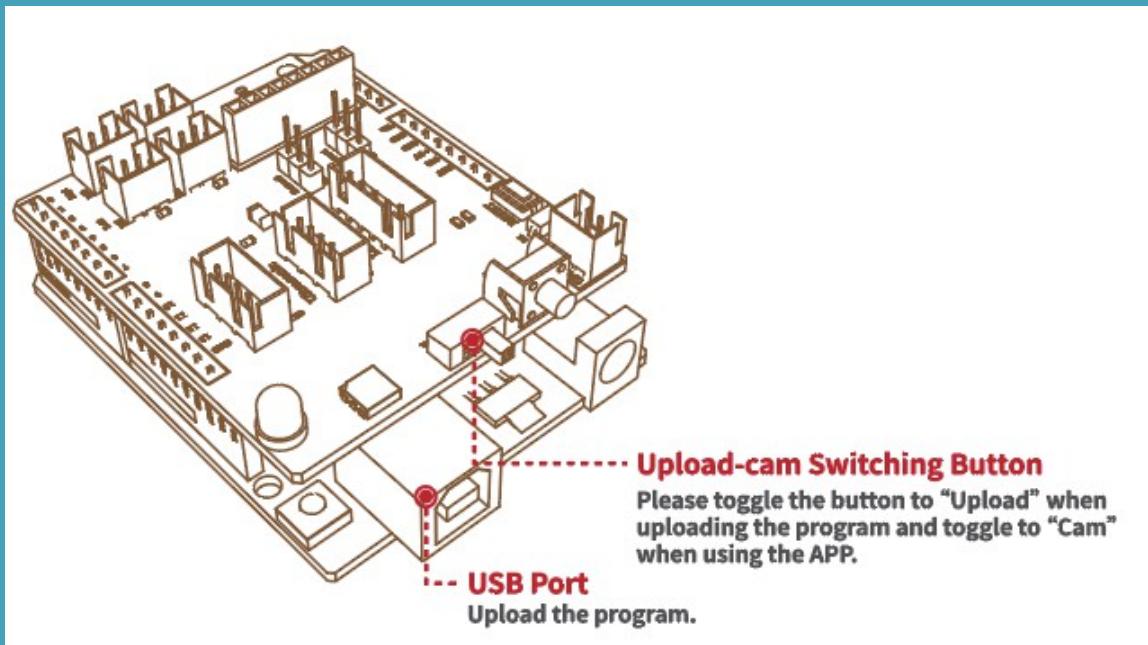
direction_A,direction_B: флаг направления вращения двигателя.
speed_A,speed_B: диапазон 0~255

```
C:/ // в DeviceDriverSet_xxx0.h
#define direction_just      true //двигатель
                                вперед #define
direction_back false //двигатель назад #define
direction_void 3
```

+ В конце концов, заставьте автомобиль двигаться в каждом направлении в течение одной секунды после 2 с ожидания.

```
C:/ void setup() {
    AppMotor.DeviceDriverSet_Motor_Init();
    delay(2000);
    for (Application_SmartRobotCarxxx0.Motion_Control = 0;
         Application_SmartRobotCarxxx0.Motion_Control < 9;
         Application_SmartRobotCarxxx0.Motion_Control =
         Application_SmartRobotCarxxx0.Motion_Control + 1)
    {
        ApplicationFunctionSet_SmartRobotCarMotionControl
        (Application_SmartRobotCarxxx0.Motion_Control /*direction*/,
         255 /*speed*/);
        delay(1000);
    }
}
```

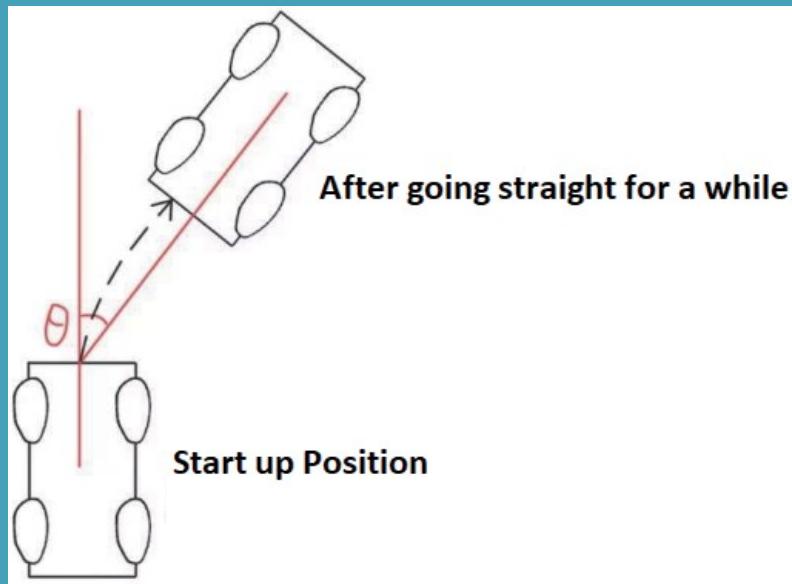
 Загрузите программу. (Пожалуйста, при загрузке программы переключите кнопку "Upload-Cam" на "Upload"). После успешной загрузки программы поставьте автомобиль на землю и включите выключатель. После этого автомобиль будет двигаться вперед, назад, поворачивать влево, вправо, вперед-влево, влево-зад, вперед-вправо, вправо-зад с интервалом в 1 с и, наконец, остановится.



После выполнения **Demo1** при достаточной внимательности можно заметить некоторые отклонения в направлении движения Smart Robot Car при движении вперед и назад. С одной стороны, разная местность имеет разное трение, с другой стороны, даже один и тот же тип двигателя может иметь небольшие различия в аппаратных характеристиках и структуре замедления, а также возмущения во время работы, такие как кратковременное проскальзывание колес, мелкие препятствия и другие факторы, которые могут привести к тому, что левое и правое колеса будут вращаться с разной скоростью при движении "прямо". Поэтому происходит небольшое отклонение от траектории.

Вы можете загрузить программу **Demo2**, чтобы продолжить наблюдение, если вы не видите ее в **Demo1**.

Загрузите программу. (При загрузке программы переключите кнопку В/У на "Upload"). Вы увидите, как автомобиль движется вперед и назад с интервалом в 1 с. В процессе движения вперед и назад маршрут движения будет иметь более заметное отклонение.



+ Из высказыванного мы знаем, что, как и при вождении автомобиля, если закрыть глаза, то даже при выпрямлении руля трудно продолжать идти прямо на большое расстояние. Поэтому приходится определять отклонение на глаз, а затем корректировать его, поворачивая руль вручную, чтобы идти по прямой. Поэтому, если мы хотим, чтобы автомобиль шел по прямой, необходимо ввести управление по замкнутому контуру.

Управление по замкнутому циклу - одно из основных понятий кибернетики. Относится к таким отношениям управления, при которых управляемый выход определенным образом возвращается к управляемому входу и оказывает управляющее воздействие на вход.

Преимущество управления с замкнутым контуром перед управлением с разомкнутым контуром заключается в том, что оно позволяет получать результаты управления, сравнивать их с желаемыми значениями и корректировать управляющее воздействие в зависимости от их ошибок.

+

Поэтому, применяя замкнутый цикл управления Smart Robot Car, мы можем выполнять коррекцию скорости по ПИД-алгоритму после интегрирования данных по оси рысканья, получив фильтр угловой скорости отклонения через MPU6050. Далее откройте [Demo3](#) в текущей папке.

 В этой программе мы сначала перенесли библиотеку "MPU6050.h".

MPU6050.cpp

MPU6050.h

-  Сначала определим связанный класс и переменную класса.

```
C:/ //in MPU6050_getdata.h
class MPU6050_getdata
{
    общественность:
    bool MPU6050_dveInit(void);
    bool MPU6050_calibration(void);
    bool MPU6050_dveGetEulerAngles(float *Yaw);
public:
    //int16_t ax, ay, az, gx, gy, gz;
    int16_t gz;
    //float pith, roll, yaw;
    unsigned long now, lastTime = 0;
    float dt;
    float agz = 0;
    long gzo = 0;
};

extern MPU6050_getdata MPU6050Getdata;
```

-  Инициализируйте MPU6050 и убедитесь, что привод работает успешно.

```
C:/ //in MPU6050_getdata.cpp
bool MPU6050_getdata::MPU6050_dveInit(void)
{
    Wire.begin();
    uint8_t chip_id = 0x00;
    uint8_t cout;
    сделать
    {
        chip_id = accelgyro.getDeviceID();
        Serial.print("MPU6050_chip_id: ");
        Serial.println(chip_id); delay(10);
        cout += 1; if
        (cout > 10)
        {
            return true;
        }
    } while (chip_id == 0X00 || chip_id == 0xFF);
    accelgyro.initialize();
    return false;
}
```

 Из-за проблемы дрейфа данных в MPU6050 лучше использовать фильтрацию средних значений при выборке, а затем получить значение угловой скорости по оси Z в момент первого размещения автомобиля.

C:/ //in MPU6050_getdata.cpp

```
bool MPU6050_getdata::MPU6050_calibration(void)
{
    unsigned short times = 100;
    for (int i = 0; i < times; i++)
    {
        gz = accelgyro.getRotationZ();
        gzo += gz;
    }
    gzo /= times;
    // gzo = accelgyro.getRotationZ();
    return false;
}
```

 Затем поместите его в вызов `setup()`.

C:/ void setup() {
 Serial.begin(9600);
 AppMPU6050getdata.MPU6050_dvelinit();
 delay(2000);
 AppMPU6050getdata.MPU6050_calibration();
}

- + Из MPU6050 известно, что коэффициент чувствительности составляет 131 LSB (count) /°/с.

Получив исходные данные датчиков, мы можем рассчитать угловую скорость, разделив исходные данные датчиков на коэффициент масштабирования их чувствительности:

угловая скорость по оси Z = (исходные данные гироскопа по оси Z /131) °/с.

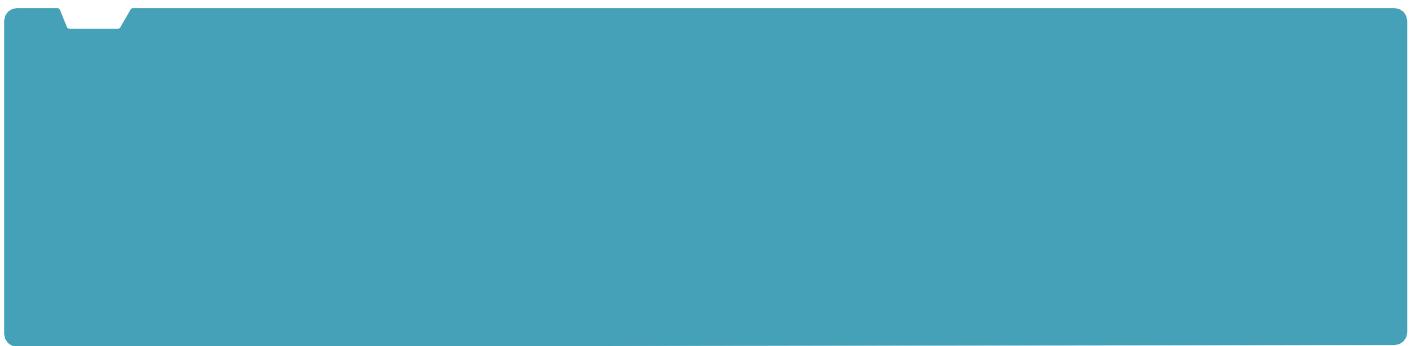
Поскольку время выполнения программы меньше 1S, необходимо умножить конечный результат на время.

Наконец, значение рысканья можно получить путем интегрирования полученной угловой скорости.

C:/

```
//in MPU6050_getdata.cpp

bool MPU6050_getdata::MPU6050_dveGetEulerAngles(float *Yaw)
{
    unsigned long now = millis();
    dt = (now - lastTime) /
    1000.0; lastTime = now;
    gz = accelgyro.getRotationZ();
    float gyroz = -(gz - gzo) / 131.0 * dt;
    if (fabs(gyroz) < 0.05)
    {
        gyroz = 0.00;
    }
    agz += gyroz;
    *Yaw = agz;
    return false;
}
```



 Перед загрузкой программы установите автомобиль в устойчивое положение. (При загрузке программы переключите кнопку "Upload-Cam" в положение "Upload"). После загрузки программы, если не двигать машину, подождать две секунды и открыть последовательный порт, угол рысканья будет колебаться около 0. Если немного подвигать машину, угол рысканья также изменится соответственно. По этому принципу запрограммирована функция прямолинейного движения автомобиля.

```
00 COM7
[|]
Send
09:24:01.317 -> Yaw:0.19
09:24:01.351 -> Yaw:0.19
09:24:01.351 -> Yaw:0.19
09:24:01.351 -> Yaw:0.19
09:24:01.384 -> Yaw:0.19
09:24:01.384 -> Yaw:0.19
09:24:01.384 -> Yaw:0.19
09:24:01.426 -> Yaw:0.19
09:24:01.426 -> Yaw:0.19
09:24:01.426 -> Yaw:0.19
09:24:01.426 -> Yaw:0.19
09:24:01.462 -> Yaw:0.19
09:24:01.462 -> Yaw:0.19
09:24:01.462 -> Yaw:0.19
09:24:01.462 -> Yaw:0.19
09:24:01.495 -> Yaw:0.19
09:24:01.495 -> Yaw:0.19
09:24:01.495 -> Yaw:0.19
```

```
00 COM7
[|]
Send
09:24:36.859 -> Yaw:6.03
09:24:36.859 -> Yaw:6.03
09:24:36.859 -> Yaw:6.03
09:24:36.859 -> Yaw:6.03
09:24:36.893 -> Yaw:6.03
09:24:36.893 -> Yaw:6.03
09:24:36.893 -> Yaw:6.03
09:24:36.927 -> Yaw:6.03
09:24:36.927 -> Yaw:6.03
09:24:36.927 -> Yaw:6.03
09:24:36.970 -> Yaw:6.03
09:24:36.970 -> Yaw:6.03
09:24:36.970 -> Yaw:6.03
09:24:36.970 -> Yaw:6.03
09:24:37.005 -> Yaw:6.03
09:24:37.005 -> Yaw:6.03
09:24:37.005 -> Yaw:6.03
```

```
00 COM7
[|]
Send
09:24:22.186 -> Yaw:-5.71
09:24:22.186 -> Yaw:-5.71
09:24:22.186 -> Yaw:-5.71
09:24:22.221 -> Yaw:-5.71
09:24:22.221 -> Yaw:-5.71
09:24:22.221 -> Yaw:-5.71
09:24:22.255 -> Yaw:-5.71
09:24:22.255 -> Yaw:-5.71
09:24:22.255 -> Yaw:-5.71
09:24:22.295 -> Yaw:-5.71
09:24:22.295 -> Yaw:-5.71
09:24:22.295 -> Yaw:-5.71
09:24:22.329 -> Yaw:-5.71
09:24:22.329 -> Yaw:-5.71
09:24:22.329 -> Yaw:-5.71
09:24:22.362 -> Yaw:-5.71
09:24:22.362 -> Yaw:-5.71
09:24:22.362 -> Yaw:-5.71
```

 После получения данных о рысканье с помощью ПИД-алгоритма можно осуществлять управление с обратной связью по замкнутому контуру для регулировки скорости вращения колес таким образом, чтобы добиться прямолинейной ходьбы. Далее откройте **Demo4** в текущей папке.

Параметр шкалы KP - это наилучшие данные, которые мы можем получить в результате многократной отладки. UpperLimit – максимальная скорость автомобиля.

C:/ //in ApplicationFunctionSet_xxx0.cpp

```
static void ApplicationFunctionSet_SmartRobotCarMotionControl(SmartRobotCarMotionControl direction, uint8_t is_speed){    Kp = 10;    UpperLimit = 255;}
```

 Данные о рысканье получают при первом запуске и обновляют их каждые десять миллиметров.

C:/ //in ApplicationFunctionSet_xxx0.cpp

```
static void ApplicationFunctionSet_SmartRobotCarLinearMotionControl(SmartRobotCarMotionControl direction, uint8_t directionRecord, uint8_t speed, uint8_t Kp, uint8_t UpperLimit){    .....    if (en != directionRecord || millis() - is_time > 10)    {        AppMotor.DeviceDriverSet_Motor_control        /*direction_A*/ direction_void, /*speed_A*/ 0,        /*direction_B*/ direction_void, /*speed_B*/ 0,        /*controlED*/ control_enable); //Управление двигателем        AppMPU6050getData.MPU6050_dveGetEulerAngles(&Yaw);        is_time = millis();    }    .....}
```


 Затем необходимо выполнить пропорциональное управление в ПИД-алгоритме по полученным входным данным.

C:/ //in ApplicationFunctionSet_xxx0.cpp

```
static void ApplicationFunctionSet_SmartRobotCarLinearMotionControl(SmartRobotCarMotionControl direction, uint8_t directionRecord, uint8_t speed, uint8_t Kp, uint8_t UpperLimit){  
....  
if(en != directionRecord )  
{  
en =  
directionRecord;  
yaw_So = Yaw;  
}  
.....  
}
```

Затем необходимо выполнить пропорциональное управление в ПИД-алгоритме по полученным входным данным.

Принцип управления Р примерно следующий: Предположим, что вы бежите к конечной точке, когда вы находитесь далеко от конечной точки, вы бежите с полной скоростью; когда вы приближаетесь к конечной точке, вы замедляетесь; и когда вы почти достигаете конечной точки, вы также почти останавливаешьесь.

 А затем не забудьте ограничить корректируемую скорость так, чтобы она не превышала максимальную.

C:/ //in ApplicationFunctionSet_xxx0.cpp

```
static void ApplicationFunctionSet_SmartRobotCarLinearMotionControl
(SmartRobotCarMotionControl direction, uint8_t directionRecord,
uint8_t speed, uint8_t Kp, uint8_t UpperLimit)
{
    .....
    int R = (Yaw - yaw_So) * Kp +
    скорость; if (R > UpperLimit)
    {
        R = UpperLimit;
    }
    else if (R < 10)
    {
        R = 10;
    }
    int L = (yaw_So - Yaw) * Kp +
    скорость; if (L > UpperLimit)
    {
        L = UpperLimit;
    }
    else if (L < 10)
    {
        L = 10;
    }
    .....
}
```

Наконец, загрузите программу. ([При загрузке программы переключите кнопку "Upload-Cam" в положение "Upload"](#)). После загрузки программы поставьте автомобиль на землю, а затем включите его. Подождите две секунды, и вы обнаружите, что автомобиль будет неоднократно двигаться вперед в течение 3 секунд, а затем двигаться назад в течение 3 секунд по относительно прямой линии.