

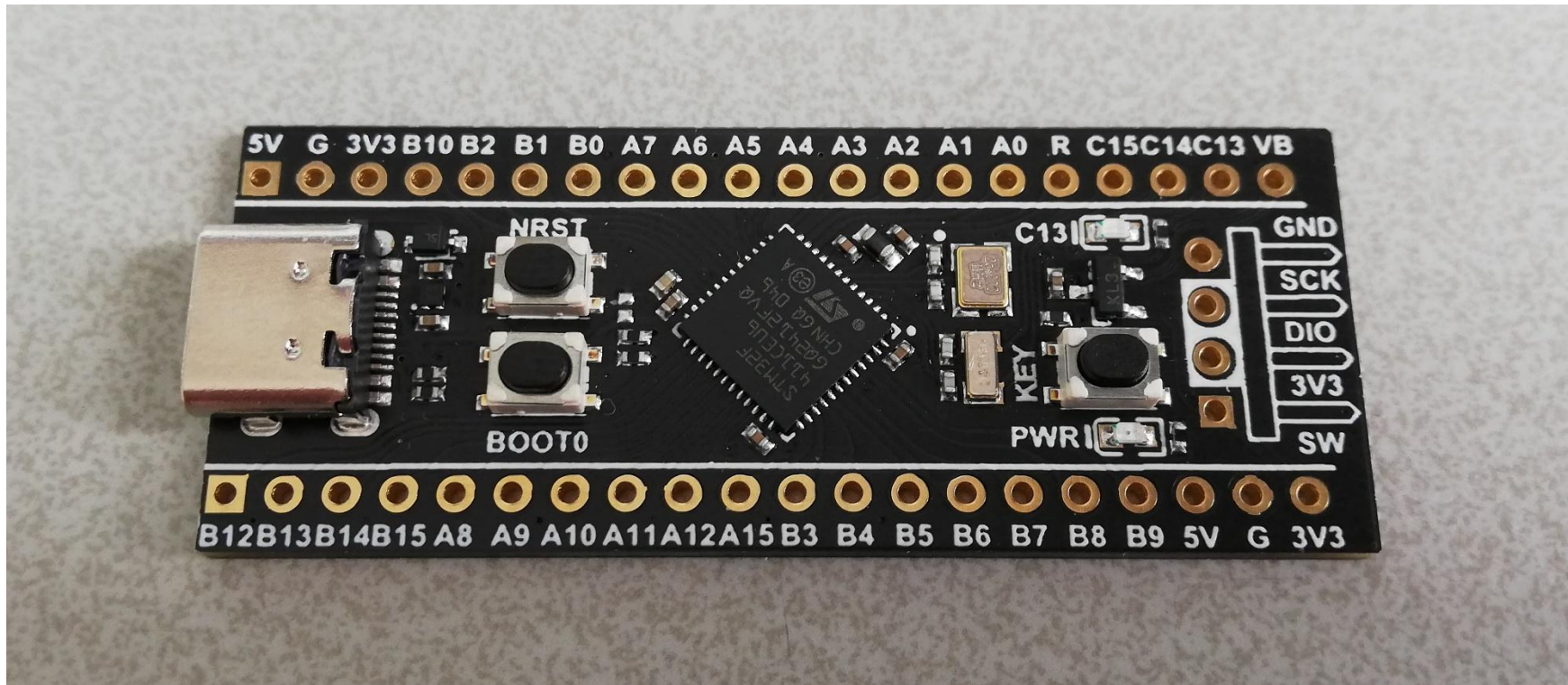
Программирование микроконтроллеров

Назаров Александр Александрович – Программист РЦР ДГТУ (2-101)

Телеграмм – @casonka

VK - <https://vk.com/casonka>

1. Что такое микроконтроллер ?

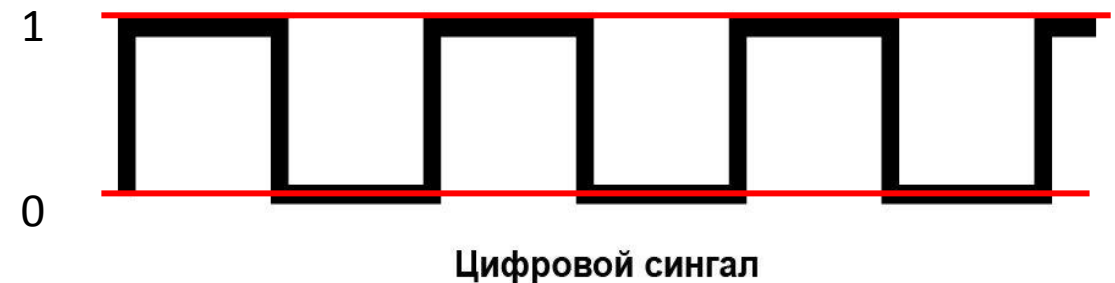


2. Обработка сигналов

Сигналы бывают двух видов: цифровые и аналоговые.

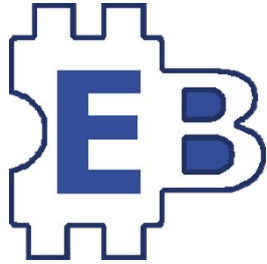
Цифровой (нижнее фото) имеет два состояния: 0 когда нет сигнала и 1 когда сигнал имеется.

Аналоговый сигнал гораздо сложнее, его значение определяется амплитудным уровнем сигнала



3. На чем писать код ?

- Embitz



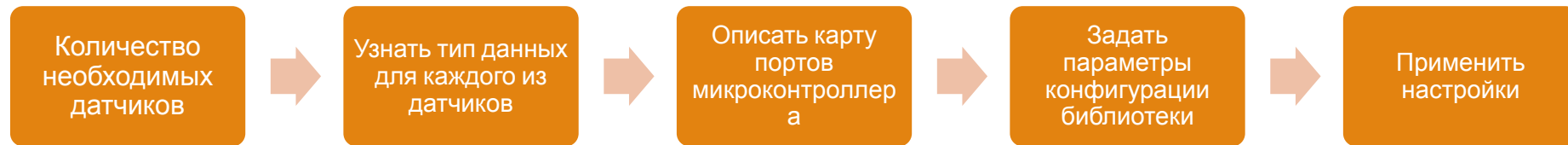
- Eclipse



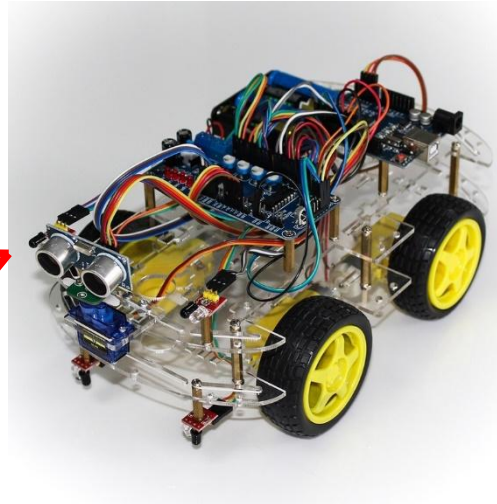
- STM32CubeIDE



4. Последовательность программирования для настройки библиотеки



5. Что потребуется для программирования обыкновенного робота ?



6. Библиотека FIL



Почему используется FIL, а не стандартная библиотека ?

- Ничего лишнего
- Удобные инструменты отладки и отслеживания
- Модульное, параметрическое программирование на языке C
- Облегченные команды для программирования микроконтроллера

6.1 Карта портов

Карта портов формирует уникальные имена для каждого пина микроконтроллера. Создается пользователем и может иметь названия связанные с :

- ❖ Название группы и номера порта (Например PA1)
- ❖ Название порта у датчика (например SDA_PIN)

```
////////////////////////////////////  
#define PA1  GPIOPinID(PORTA,1)  
  
#define Sensor_pin  GPIOPinID(PORTA,1)  
  
////////////////////////////////////  
#define BTN1_DIR_PIN  GPIOPinID(PORTB,14)  // Wheels direction  
#define BTN2_DIR_PIN  GPIOPinID(PORTB,15)  // NONE  
////////////////////////////////////  
// _____ Encoders _____ //  
////////////////////////////////////  
#define ENCODER1A_PIN  GPIOPinID(PORTA,0)  
#define ENCODER1B_PIN  GPIOPinID(PORTA,1)  
#define ENCODER2A_PIN  GPIOPinID(PORTA,6)  
#define ENCODER2B_PIN  GPIOPinID(PORTA,7)  
  
#define ENCODER1_CNT  ((uint32_t *)&(TIM3->CNT))  
#define ENCODER2_CNT  ((uint32_t *)&(TIM4->CNT))
```



```

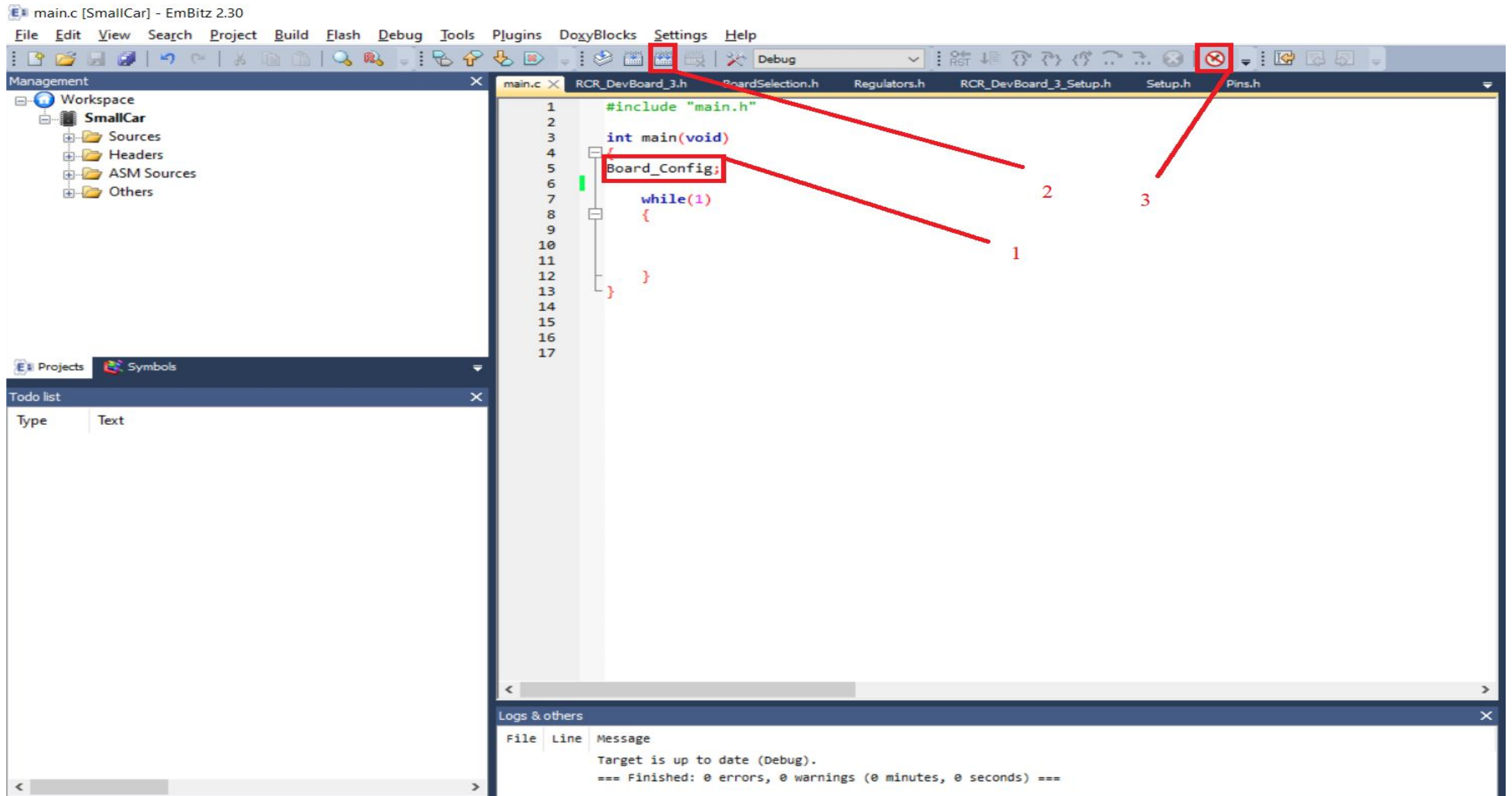
#define __configUSE_RCC          1
#define __configUSE_GPIO        1
#define __configUSE_TIM          0
#define __configUSE_USART       0
#define __configUSE_DMA          0
#define __configUSE_I2C          0
#define __configUSE_ADC          0
#define __configUSE_EXTI         0
#define __configUSE_RTC          0
#define __configUSE_FREERTOS    0

#define __configCALC_RCC         0
#define __configCALC_TIM         0
#define __configCALC_USART       0
#define __configCALC_Regulators  0
#define __configCALC_Matrix      0

#define Board_Config
    SetGPIOA;
    GPIOConfPin(PA1, ALTERNATE, PUSH_PULL, FAST_S, NO_PULL_UP);

```

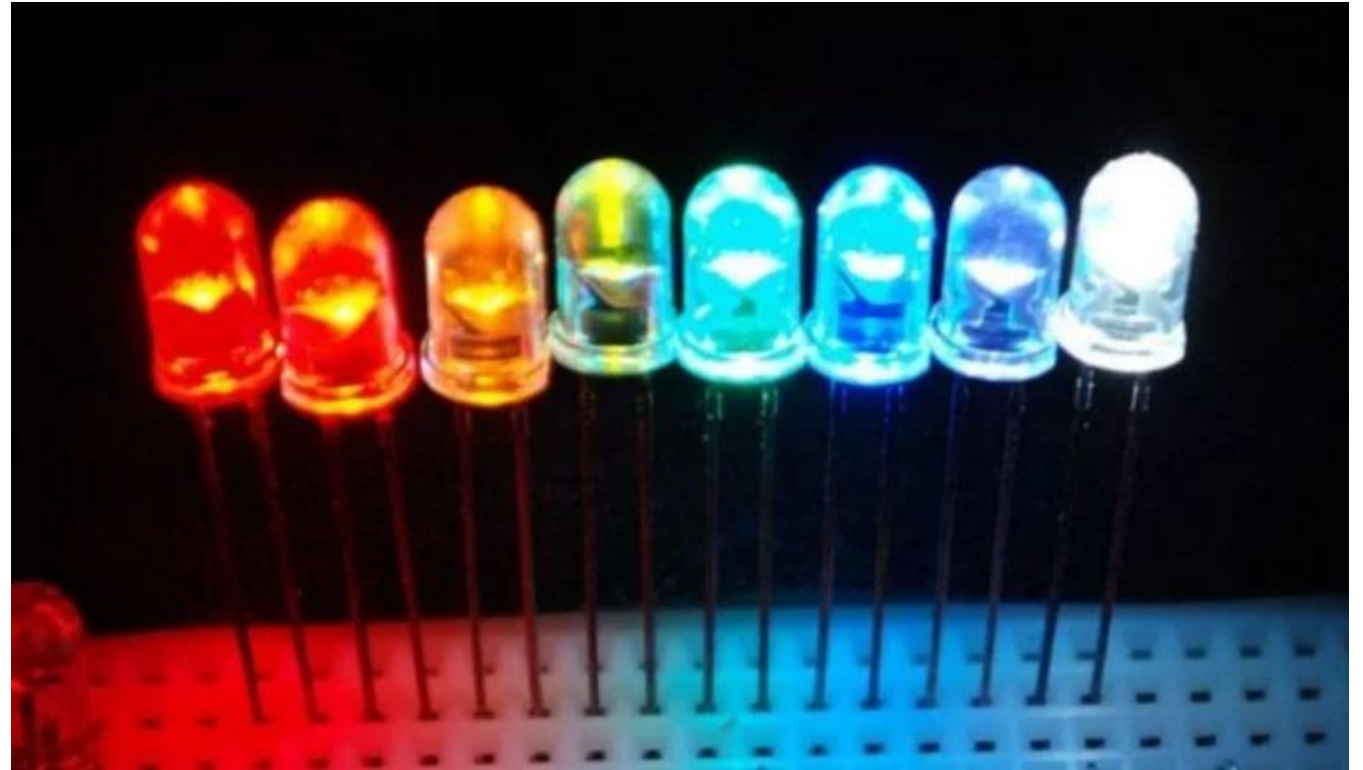
6.3 Применение настроек



7.1 Управление светодиодом

Светодиод используется в робототехнических и других системах для визуального оповещения человека о успешного/неудачного запуска того или иного участка кода.

Не является обязательным в роботах, но желателен для быстрой оценки поломки чего-либо (или плохого контакта).



```
#pragma once

#define __configUSE_RCC          1
#define __configUSE_GPIO        1
#define __configUSE_TIM          0
#define __configUSE_USART       0
#define __configUSE_DMA          0
#define __configUSE_I2C         0
#define __configUSE_ADC          0
#define __configUSE_EXTI        0
#define __configUSE_RTC         0
#define __configUSE_FREERTOS    0

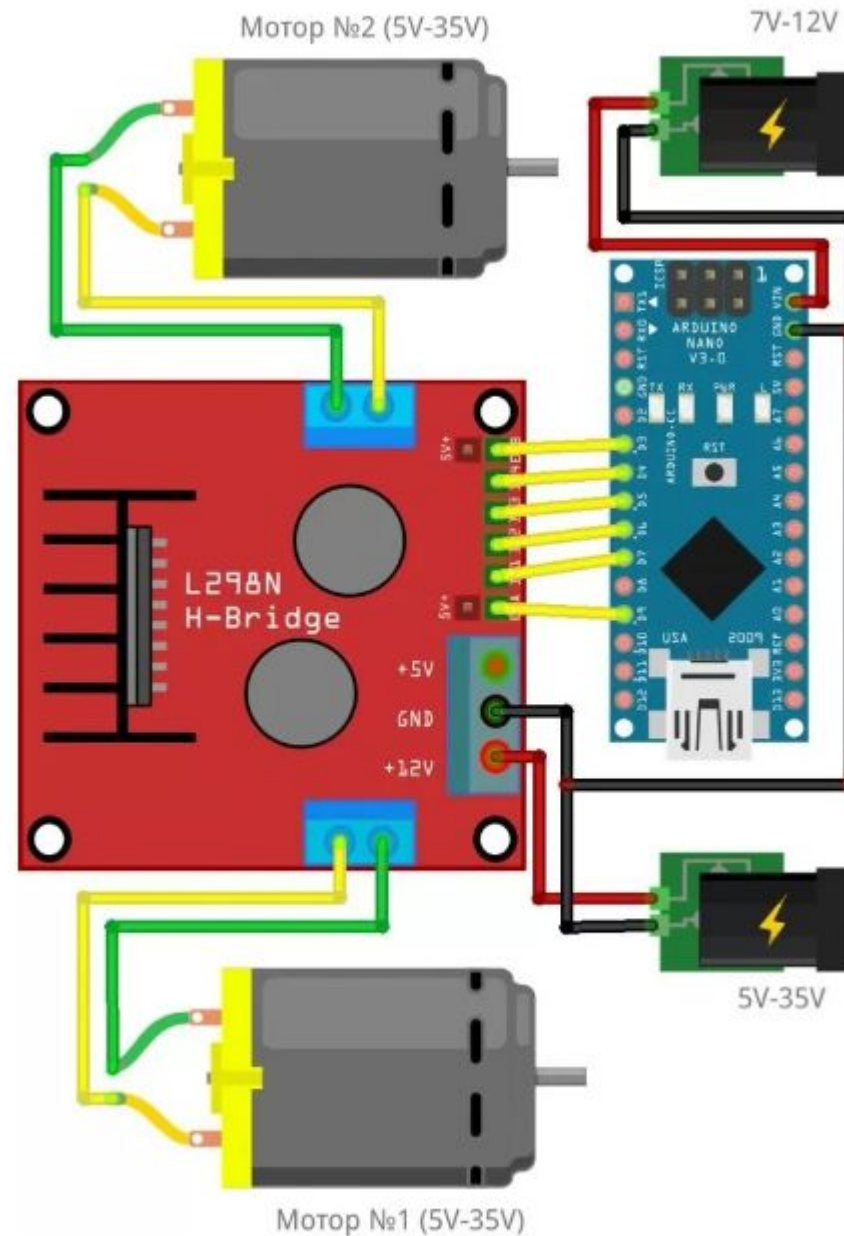
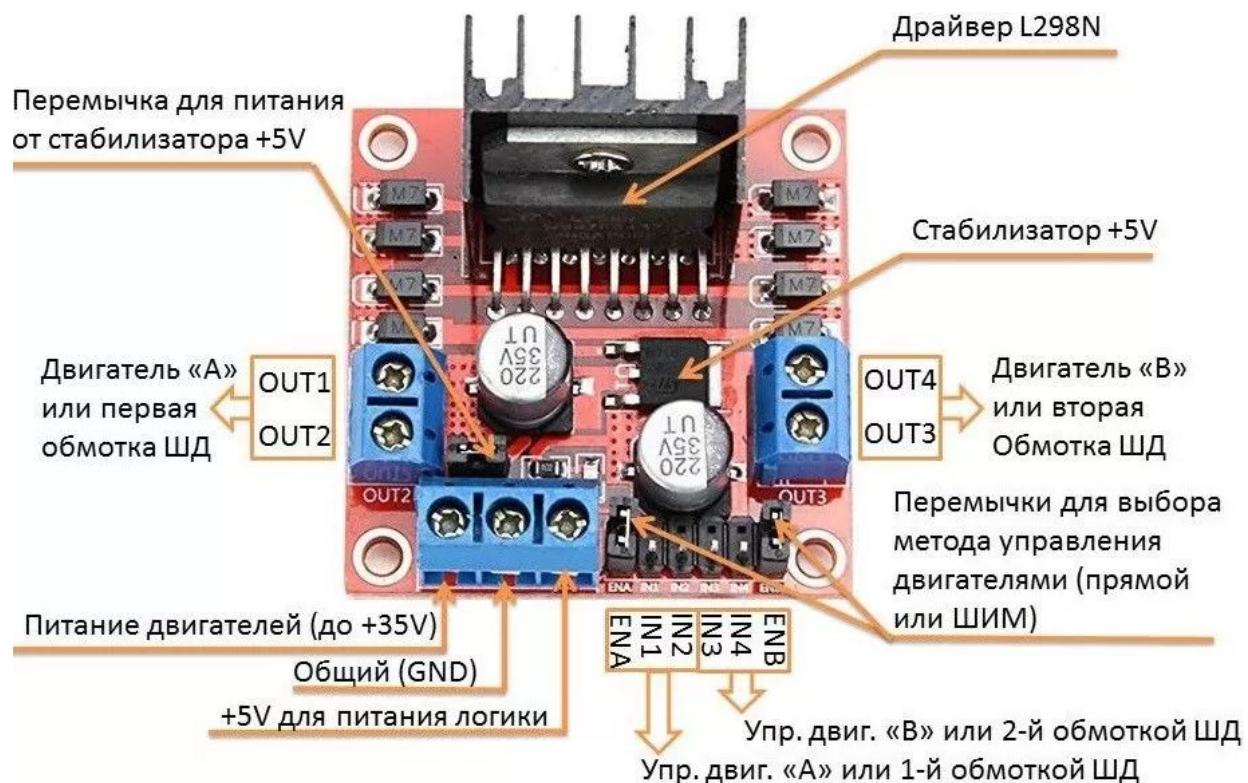
#define __configCALC_RCC        0
#define __configCALC_TIM        0
#define __configCALC_USART      0
#define __configCALC_Regulators 0
#define __configCALC_Matrix     0

#define LED_PIN  GPIOPinID(PORTA,1)

#define Board_Config
    SetGPIOA;
    GPIOConfgPin(LED_PIN, GENERAL, PUSH_PULL, FAST_S, NO_PULL_UP);\
```

```
1  #include "main.h"
2
3  int main(void)
4  {
5      Board_Config;
6
7      SetPin(LED_PIN);
8
9      ResetPin(LED_PIN);
10
11     while(1)
12     {
13
14
15
16     }
17 }
18
19
```


7.2 Управление ДПТ



```

#pragma once

#define PWM1_PIN          GPIOPinID(PORTA,8)
#define PWM2_PIN          GPIOPinID(PORTA,9)

#define PWM1_CCR          ((uint32_t *)&(TIM1->CCR1))
#define PWM2_CCR          ((uint32_t *)&(TIM1->CCR2))

#define DIR1_PIN          GPIOPinID(PORTA,10)

1  #include "main.h"
2
3  void SetVoltage(float Duty)
4  {
5      if(Duty >= 0.0)
6      {
7          SetPin(DIR1_PIN);
8          ResetPin(DIR2_PIN);
9          *PWM1_CCR = ((int32_t) (Duty * 2000));
10         return;
11     }
12     else
13     {
14         SetPin(DIR2_PIN);
15         ResetPin(DIR1_PIN);
16         *PWM1_CCR = ((int32_t)(2000 + (Duty * 2000)));
17         return;
18     }
19 }
20
21 int main(void)
22 {
23     Board_Config;
24
25     SetVoltage(0.2);
26     while(1)
27     {
28
29
30
31
32
33
34

```

```

1  #pragma once
2
3  #define __configUSE_RCC          1
4  #define __configUSE_GPIO        1
5  #define __configUSE_TIM          1
6  #define __configUSE_USART        0
7  #define __configUSE_DMA          0
8  #define __configUSE_I2C          0
9  #define __configUSE_ADC          0
10 #define __configUSE_EXTI          0
11 #define __configUSE_RTC          0
12 #define __configUSE_FREERTOS     0
13
14 #define __configCALC_RCC          0
15 #define __configCALC_TIM          0
16 #define __configCALC_USART        0
17 #define __configCALC_Regulators   0
18 #define __configCALC_Matrix      0
19
20 #define Board_Config
21     SetGPIOA;
22     SetTIM1;
23     GPIOConfPin(PWM1_PIN, ALTERNATE, PUSH_PULL, FAST_S, NO_PULL_UP);
24     GPIOConfPin(PWM2_PIN, ALTERNATE, PUSH_PULL, FAST_S, NO_PULL_UP);
25     GPIOConfAF(PWM1_PIN, AF1);
26     GPIOConfAF(PWM2_PIN, AF1);
27     GPIOConfPin(DIR1_PIN, GENERAL, PUSH_PULL, FAST_S, NO_PULL_UP);
28     GPIOConfPin(DIR2_PIN, GENERAL, PUSH_PULL, FAST_S, NO_PULL_UP);
29     TimPWMConfigure(TIM1,840,2000,1,1,0,0);
30
31

```

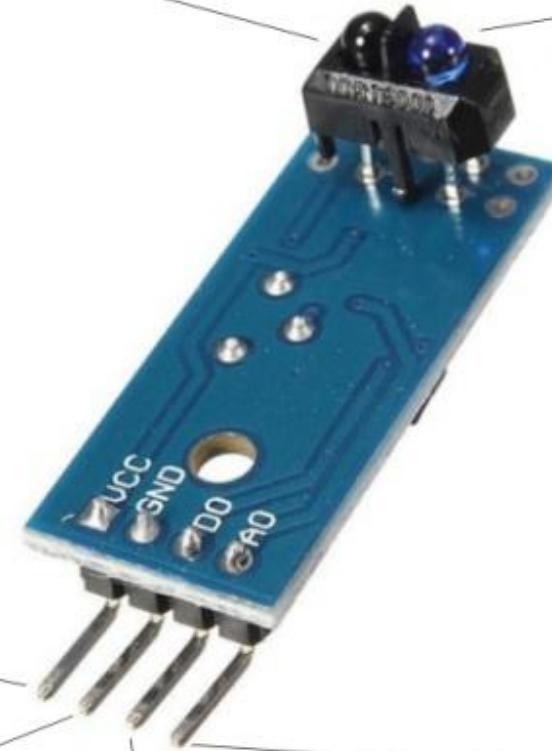

7.3 Обработка данных с датчика ЛИНИИ

Индикатор срабатывания датчика линии (горит зеленым при обнаружении черной линии)



Фототранзистор

Светодиод инфракрасного спектра (свет невидим глазом)




```

16 #define __configCALC_USART          0
17 #define __configCALC_Regulators    0
18 #define __configCALC_Matrix        0
19
20 #define __configADC_Mode            (2)
21 #define __configCONVERT_Volts      (0)
22 #define __configUSE_Battery_Charging (0)
23 #define __configUSE_Temperature_Sensor (0)
24 //-----//
25
26 #define __configUSE_SENSOR_1        (0)
27 #define __configUSE_SENSOR_2        (1)
28 #define __configUSE_SENSOR_3        (-1)
29 #define __configUSE_SENSOR_4        (-1)
30 #define __configUSE_SENSOR_5        (-1)
31 #define __configUSE_SENSOR_6        (-1)
32 #define __configUSE_SENSOR_7        (-1)
33 #define __configUSE_SENSOR_8        (-1)
34 #define __configUSE_SENSOR_9        (-1)
35 #define __configUSE_SENSOR_10       (-1)
36
37 #define __configADC_InterruptRequest (0)
38 #define __configADC_Divider          (3)
39 #define __configADC_RESOLUTION       (12) // 12-bit resolution
40 #define __configADC_CYCLES           (ADC_480_CYCLES)
41 #define __configADC_DIVIDER          (3)
42
43 #define Board_Config                  {\
44     SetGPIOA;                         \
45     SetTIM1;                           \
46     SetADC1;                           \
47     SetDMA2;                           \
48     GPIOConfPin(PWM1_PIN, ALTERNATE, PUSH_PULL, FAST_S, NO_PULL_UP); \
49     GPIOConfPin(PWM2_PIN, ALTERNATE, PUSH_PULL, FAST_S, NO_PULL_UP); \
50     GPIOConfAF(PWM1_PIN, AF1);         \
51     GPIOConfAF(PWM2_PIN, AF1);         \
52     GPIOConfPin(DIR1_PIN, GENERAL, PUSH_PULL, FAST_S, NO_PULL_UP); \
53     GPIOConfPin(DIR2_PIN, GENERAL, PUSH_PULL, FAST_S, NO_PULL_UP); \
54     GPIOConfPin(SENSOR1_PIN, ANALOG, PUSH_PULL, FAST_S, NO_PULL_UP); \
55     GPIOConfPin(SENSOR2_PIN, ANALOG, PUSH_PULL, FAST_S, NO_PULL_UP); \
56     TimPWMConfigure(TIM1, 840, 2000, 1, 1, 0, 0); \
57 }

```

```
#include "main.h"
```

```
uint16_t ADC1_Data[ADC1_NUMB];
int main(void)
```

```
{
    Board_Config;
    ADCAddRegularChannel(ADC1, 0, ADC_480_CYCLES);
    ADCAddRegularChannel(ADC1, 1, ADC_480_CYCLES);
    ConnectADCTODMA(HIGH_P, ADC1_Data, 0);
    ADCSimpleConfigure(ADC1);
    while(1)
    {
    }
}
```

```
1 #pragma once
```

```
2
3 #define LED_PIN          GPIOPinID(PORTC, 13)
```

```
4
5 #define PWM1_PIN         GPIOPinID(PORTA, 8)
```

```
6 #define PWM2_PIN         GPIOPinID(PORTA, 9)
```

```
7
8 #define PWM1_CCR         ((uint32_t *)&(TIM1->CCR1))
```

```
9 #define PWM2_CCR         ((uint32_t *)&(TIM1->CCR2))
```

```
10
11 #define DIR1_PIN         GPIOPinID(PORTA, 10)
```

```
12 #define DIR2_PIN         GPIOPinID(PORTA, 11)
```

```
13
14 #define SENSOR1_PIN      GPIOPinID(PORTA, 0)
```

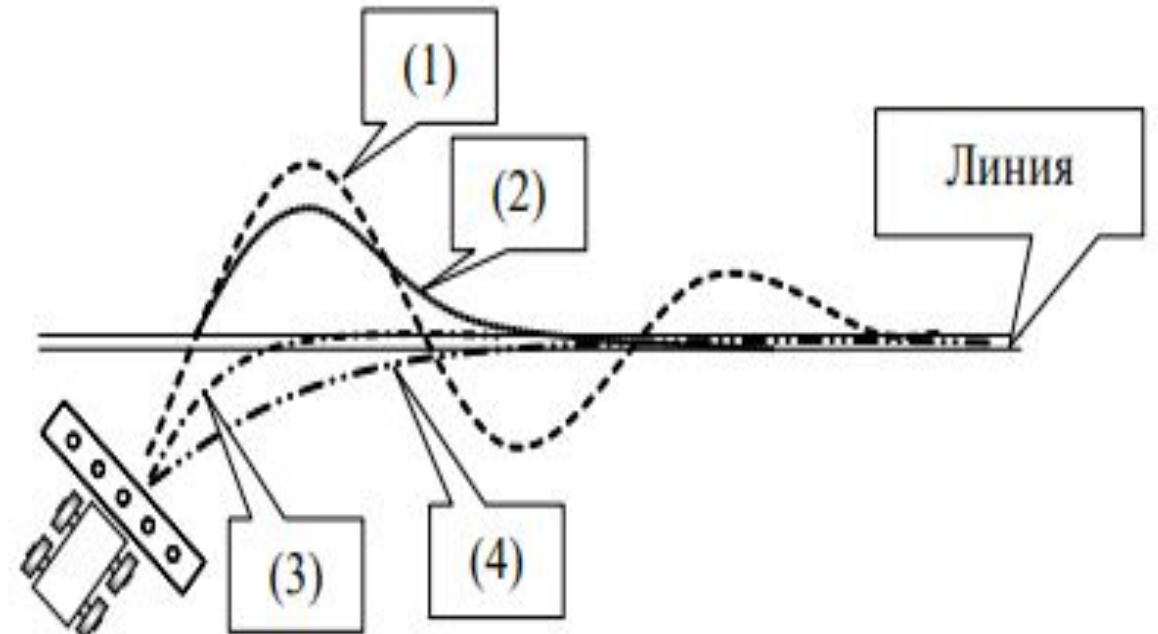
```
15 #define SENSOR2_PIN      GPIOPinID(PORTA, 1)
```

```
16
17
```

7.4 Реализация алгоритма езды по линии с регулятором

Для построения правильной с точки зрения алгоритма траектории движения робота используются регуляторы.

Регулятор позволяет корректировать сигнал управления на электродвигатели, основываясь на входной информации (в нашем примере значение с датчиков линии).



Pinout Diagram

