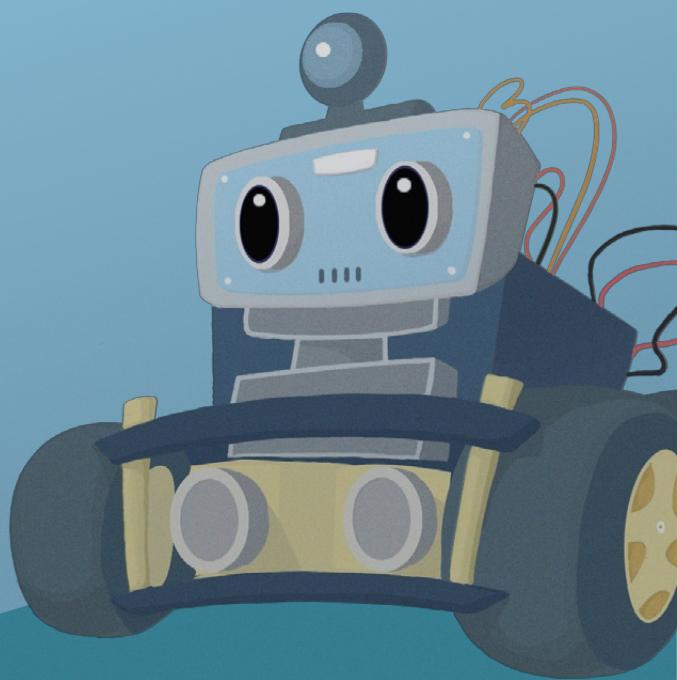


SMART ROBOT CAR V4.0 WITH CAMERA

5



Obstacle-avoidance
Mode





Введение:

+ В этом уроке мы расскажем, как добиться режима уклонения от препятствий в Smart Robot Car и заставить его менять направление движения при столкновении с препятствиями.

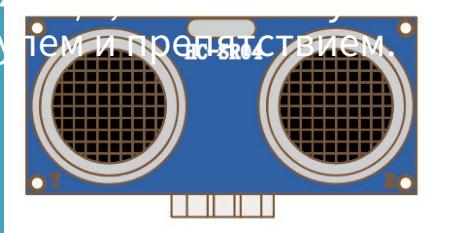


Подготовка:

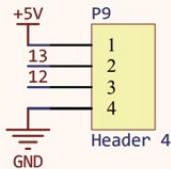
+ Умный автомобиль-робот (с аккумулятором) Провод USB

+ В нашем комплекте режим уклонения от препятствий реализован с помощью ультразвукового модуля. Принцип ультразвукового измерения расстояния заключается в том, что излучатель испускает ультразвуковые волны в определенном направлении и начинает отсчет времени в момент запуска. Ультразвуковые волны распространяются по воздуху и при встрече с препятствиями немедленно возвращаются обратно.

Затем, получив отраженные волны, ультразвуковой приемник сразу же останавливает отсчет времени. По зафиксированному времени и известному условию, что скорость распространения ультразвука в воздухе составляет 340 м/с , вычисляется расстояние между ультразвуковым модулем и препятствием.



 Для получения подробной информации откройте последнюю папку: Информация о микросхеме -> SmartRobot-Shield



UltraIsonic

Из рисунка видно, что ультразвуковой модуль подключен к D12 и D13 на плате UNO.

Затем откройте программу Demo1:

 Далее рассмотрим определение относительных штифтов и переменных УЗК.

C:/ // в DeviceDriverSet_xxx0.h

класс DeviceDriverSet_ULTRASONIC

{

общественность:

```
void DeviceDriverSet_ULTRASONIC_Init(void);
void DeviceDriverSet_ULTRASONIC_Test(void);
void DeviceDriverSet_ULTRASONIC_Get
(uint16_t *ULTRASONIC_Get /*out*/);
```

частный:

```
#define TRIG_PIN 13
// Вывод Arduino привязан к выводу триггера
ультразвукового датчика. #define ECHO_PIN 12
// Вывод Arduino связан с выводом echo на
ультразвуковом датчике. #define MAX_DISTANCE 200
// Максимальное расстояние, на которое мы хотим пинговать (в
сантиметрах).
//Максимальное расстояние до датчика составляет 400-500 см.
};
```

 Затем инициализируйте штифты УЗК.

C:/ // в DeviceDriverSet_xxx0.cpp

```
void DeviceDriverSet_ULTRASONIC::DeviceDriverSet_ULTRASONIC_Init(void)
{
    pinMode(ECHO_PIN, INPUT); //Инициализация ультразвукового модуля
    pinMode(TRIG_PIN, OUTPUT);
}
```

 А затем вызвать его в функции Setup():

C:/ // в файле

Demo1.ino void

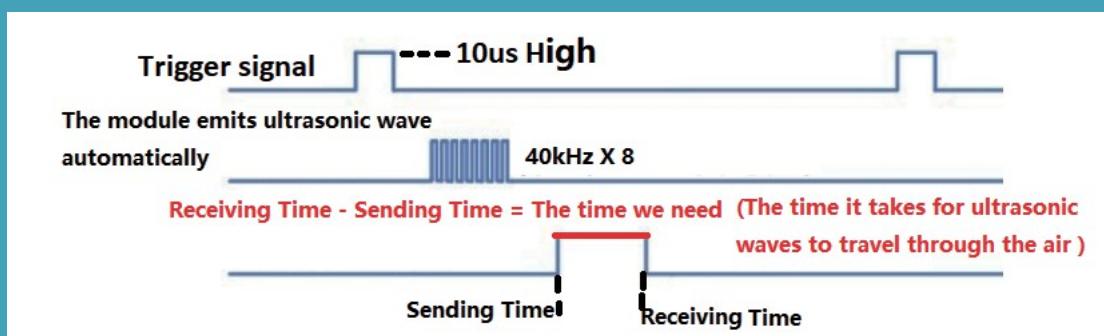
```
setup()
{
    myUltrasonic.DeviceDriverSet_ULTRASONIC_Init();
}
```

 Далее рассмотрим процесс получения данных измерений с помощью ультразвукового модуля.

1. Используя Arduino, подайте сигнал высокого уровня на вывод Trig модуля SR04 не менее чем за 10 мкс, чтобы запустить функцию измерения расстояния модуля SR04.

2. Срабатывания модуль автоматически посылает восемь ультразвуковых импульсов частотой 40 КГц и определяет, есть ли возврат сигнала. Этот шаг выполняется модулем автоматически.

3. Если сигнал возвращается, то на вывод Echo будет подан высокий уровень, а длительность высокого уровня - это время от запуска до возвращения ультразвука. В этот момент мы можем использовать функцию `pulseIn()` для получения данных измерения расстояния и вычисления фактического расстояния до измеряемого объекта.



 В процессе программирования мы будем использовать функцию `pulseIn(pin,value)`. `pulseIn(pin, value)`: Ширина импульса, используемого для определения высокого и низкого уровня на выходе пина, т.е. время, которое нам необходимо.

Pin: вывод, используемый для получения импульса. Значение: Тип импульса, HIGH или LOW

+ Поскольку единицей измерения функции `pulseIn()` по умолчанию является "us", а скорость распространения ультразвуковых волн в воздухе составляет около 340 м/с, необходимо произвести преобразование единиц измерения.

Потому что

расстояние, пройденное ультразвуковой волной, = расстояние от точки запуска до конца + расстояние от конца обратно до точки запуска.

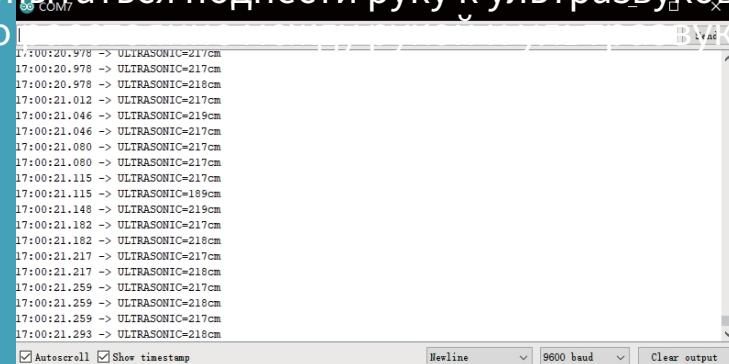
Поэтому,

расстояние, пройденное ультразвуковой волной = `pulseIn() / 29,15 / 2` ≈

```
① ② ③  
C:/  
// в DeviceDriverSet_xxx0.cpp  
  
void DeviceDriverSet_ULTRASONIC::DeviceDriverSet_ULTRASONIC_Test(void)  
{  
  
    unsigned int tempda = 0;  
    digitalWrite(TRIG_PIN, LOW);  
    delayMicroseconds(2);  
    digitalWrite(TRIG_PIN, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(TRIG_PIN, LOW);  
    tempda = ((unsigned int)pulseIn(ECHO_PIN, HIGH) / 58);  
  
    Serial.print("ULTRASONIC=");  
    Serial.print(tempda); // Перевести время пинга в расстояние и вывести  
    // результат  
    // (0 = вне установленного диапазона расстояний, ping-эхо отсутствует)  
    Serial.println("cm");  
}
```



Загрузите программу. (При загрузке программы **переключите** кнопку "Upload-Cam" на "Upload"). После успешной загрузки программы откройте последовательный порт. Если перед камерой нет препятствий, ультразвуковое расстояние будет отображаться как более 200. Если же попытаться поднести руку к ультразвуковому модулю, то будет показано



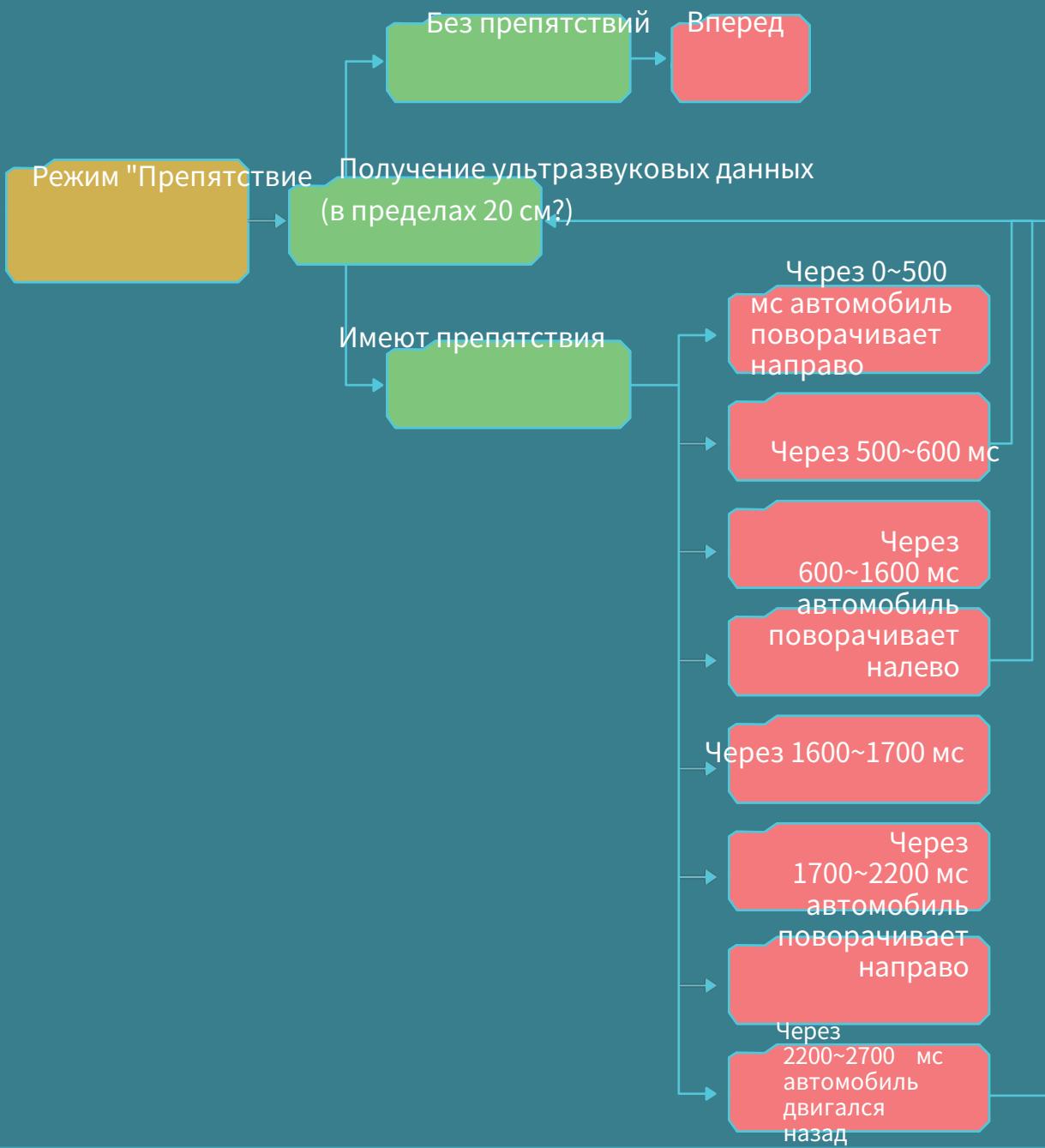
```

16:59:40.777 -> ULTRASONIC=6cm
16:59:40.810 -> ULTRASONIC=6cm
16:59:40.818 -> ULTRASONIC=6cm
16:59:40.852 -> ULTRASONIC=6cm
16:59:40.852 -> ULTRASONIC=6cm
16:59:40.886 -> ULTRASONIC=6cm
16:59:40.886 -> ULTRASONIC=6cm
16:59:40.926 -> ULTRASONIC=6cm
16:59:40.926 -> ULTRASONIC=6cm
16:59:40.960 -> ULTRASONIC=6cm
16:59:40.960 -> ULTRASONIC=6cm
16:59:40.995 -> ULTRASONIC=6cm
16:59:41.035 -> ULTRASONIC=6cm
16:59:41.035 -> ULTRASONIC=6cm
16:59:41.070 -> ULTRASONIC=6cm
16:59:41.070 -> ULTRASONIC=6cm
16:59:41.070 -> ULT

```

+ На этом работа с драйвером ультразвукового модуля завершена. Теперь рассмотрим общую схему принципа реализации режима уклонения от препятствий с помощью ультразвукового модуля, а затем проанализируем программу:

Принцип реализации Препятствие



+ Откройте Demo2 в текущей папке:

+ Рассмотрим определение относительных функций и переменных режима уклонения от препятствий:

```
C:/ // в ApplicationFunctionSet_xxx0.h #ifndef  
_ApplicationFunctionSet_xxx0_H_ #define  
_ApplicationFunctionSet_xxx0_H_  
  
#include <arduino.h>  
  
класс ApplicationFunctionSet  
{  
общественность:  
    void ApplicationFunctionSet_Init(void);  
    void ApplicationFunctionSet_Obstacle(void);  
  
частный:  
    volatile uint16_t UltrasoundData_mm;  
    volatile uint16_t UltrasoundData_cm;  
    boolean UltrasoundDetectionStatus = false;  
public:  
    boolean Car_LeaveTheGround = true;  
    const int ObstacleDetection = 20;  
};  
extern ApplicationFunctionSet Application_FunctionSet;  
#endif
```

Затем, для написания программы режима "Уход от препятствий", можно последовательно перечислить ситуации, описанные на блок-схеме при столкновении с препятствиями.

```
C:/ void ApplicationFunctionSet::ApplicationFunctionSet_Obstacle(void)  
{.....}
```

Загрузите программу. (При загрузке программы переключите кнопку "Upload-Cam" в положение "Upload"). После успешной загрузки программы поставьте машину на землю и включите выключатель. По умолчанию Smart Robot Car будет двигаться вперед, если перед ним нет препятствий. При этом будет реализован режим уклонения от препятствий.

интеллектуально зависит от различных ситуаций при столкновении с препятствием.