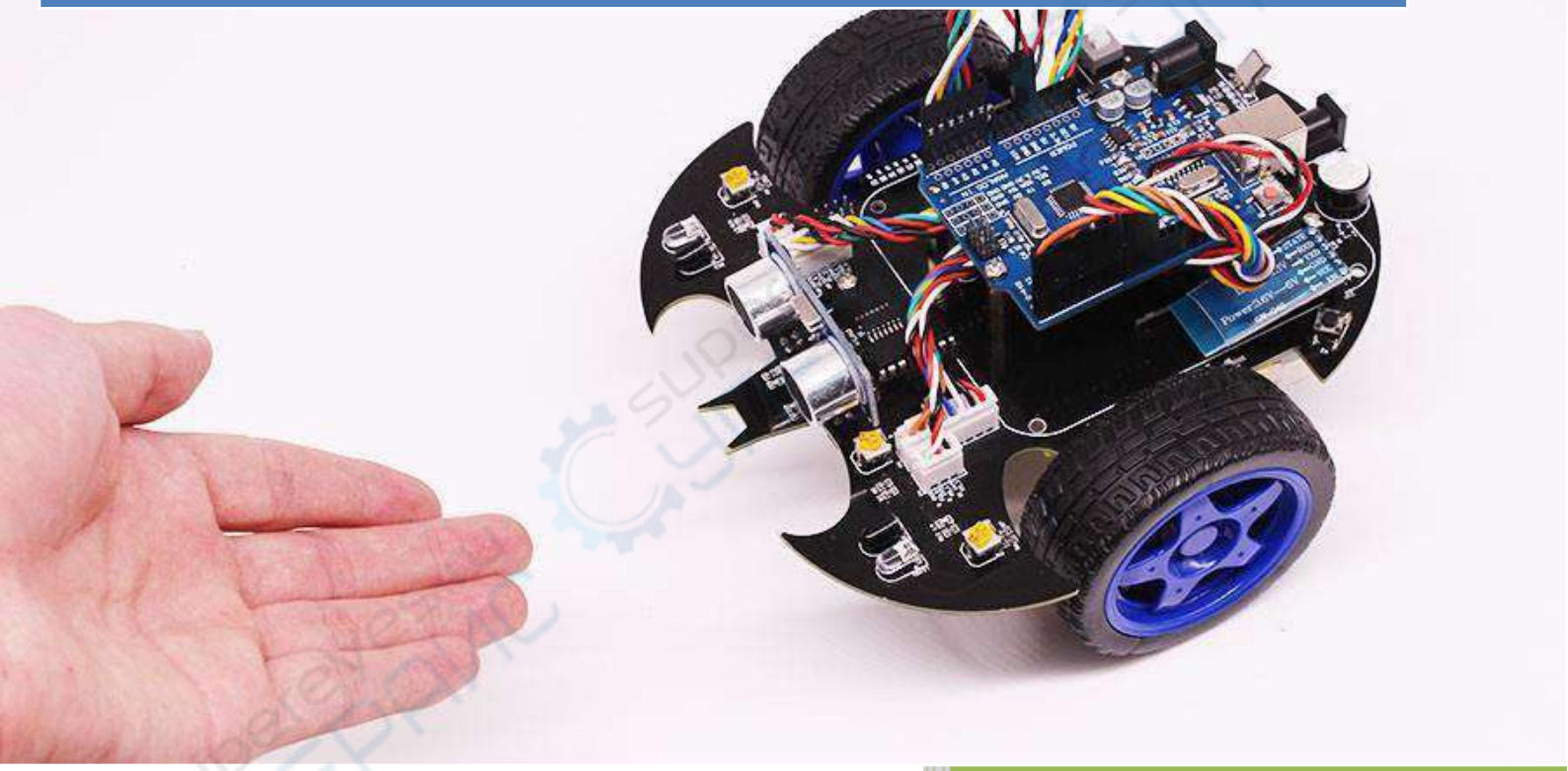
2021

BatmobileYahboom на базе Arduino

Руководство по программированию



Содержание

Урок 1. Движение вперед

Цель эксперимента

После загрузки программы и включения питания Batmobile начинает движение вперед после двух секунд простоя.

Список необходимых компонентов:

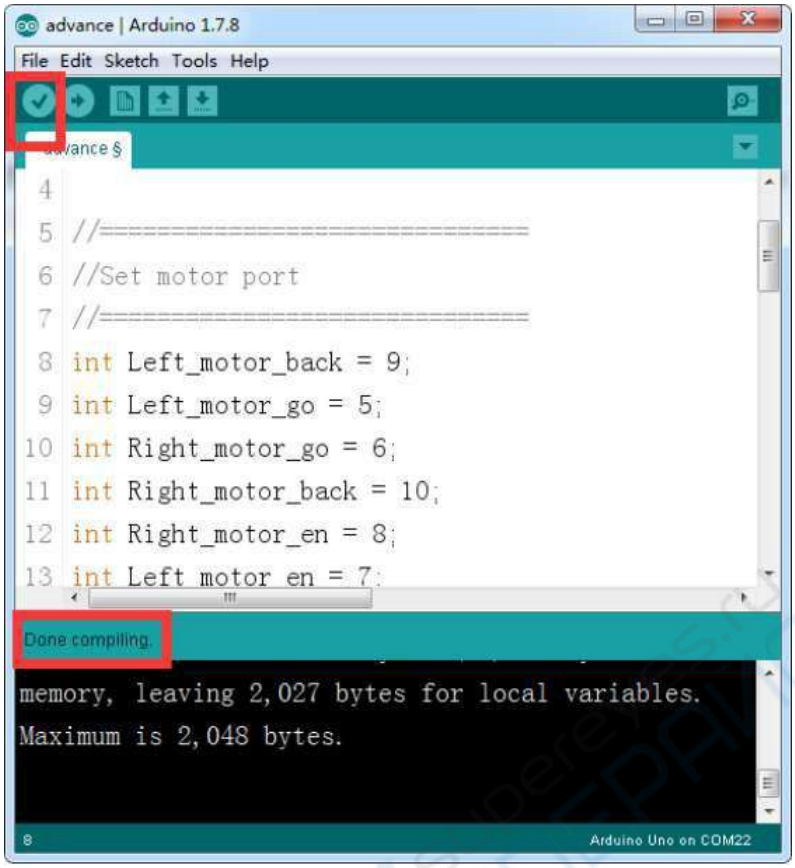
- Batmobile — 1 шт.
- Кабель USB — 1 шт.

Программный код:

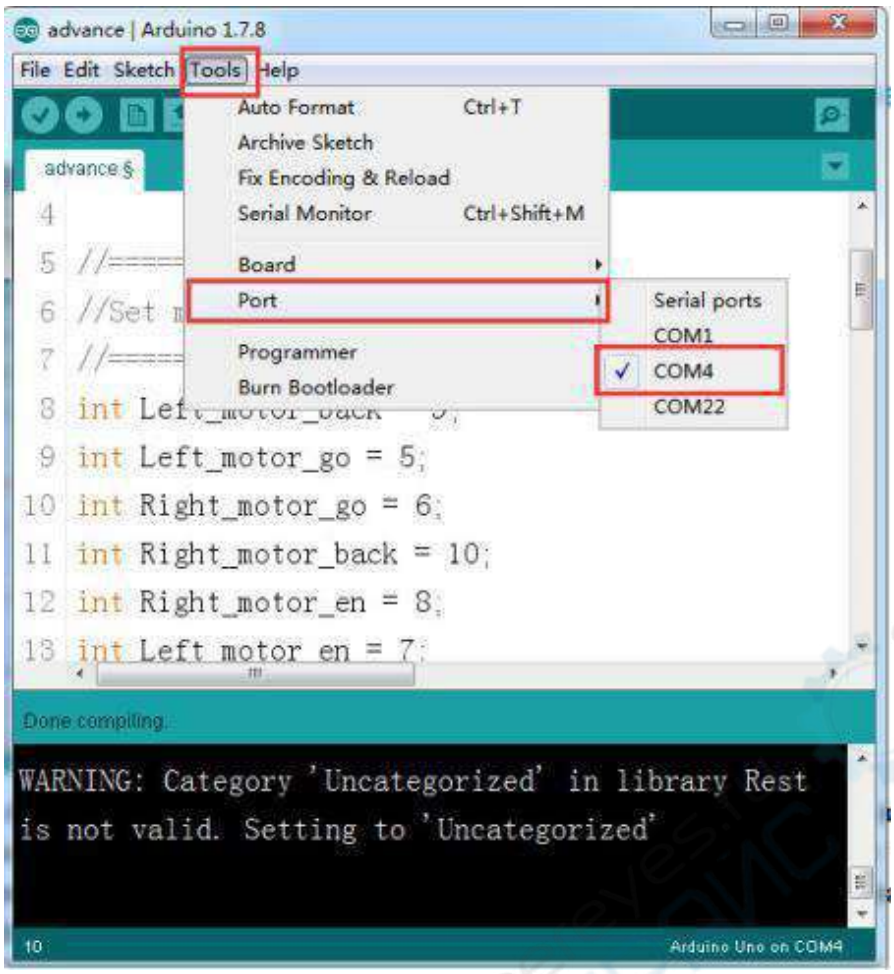
```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;

void setup()
{
  // Инициализация электродвигателей
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
}
void run(int time) // Движение вперед
{
  digitalWrite(Left_motor_en,HIGH); // Активировать левый электродвигатель
  digitalWrite(Right_motor_en,HIGH); // Активировать правый электродвигатель
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  delay(time * 100);      // Время работы
}
  void loop()
  {
  delay(2000); // Задержка 2 сек
  run(10);  // Запуск
  }
```

Этапы эксперимента

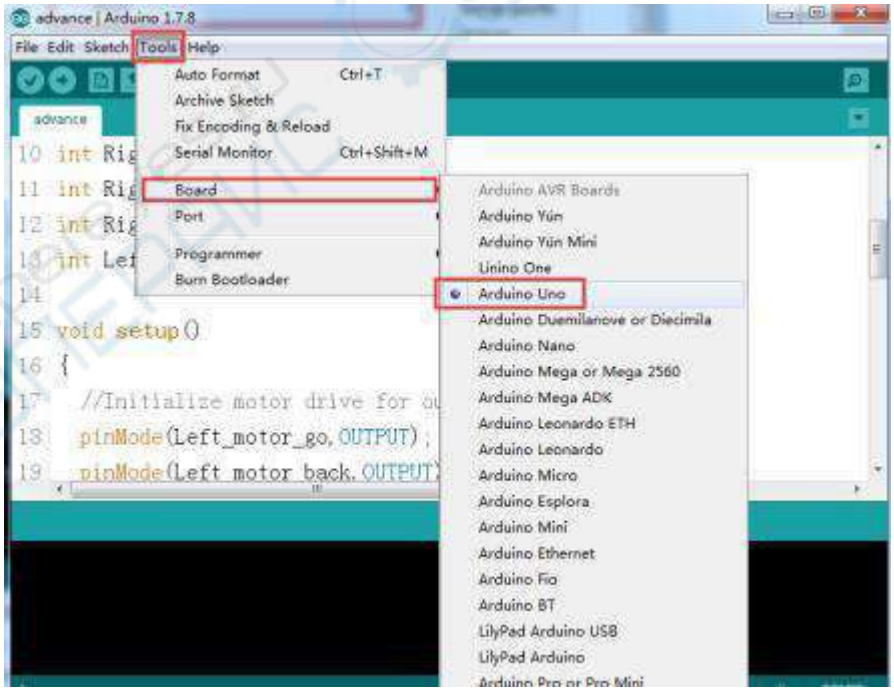
1. Откройте код **advance.ino**, нажмите на кнопку «√» под главным меню для компиляции кода и дождитесь сообщения «**Done compiling**» в нижнем правом углу, как показано на рисунке.



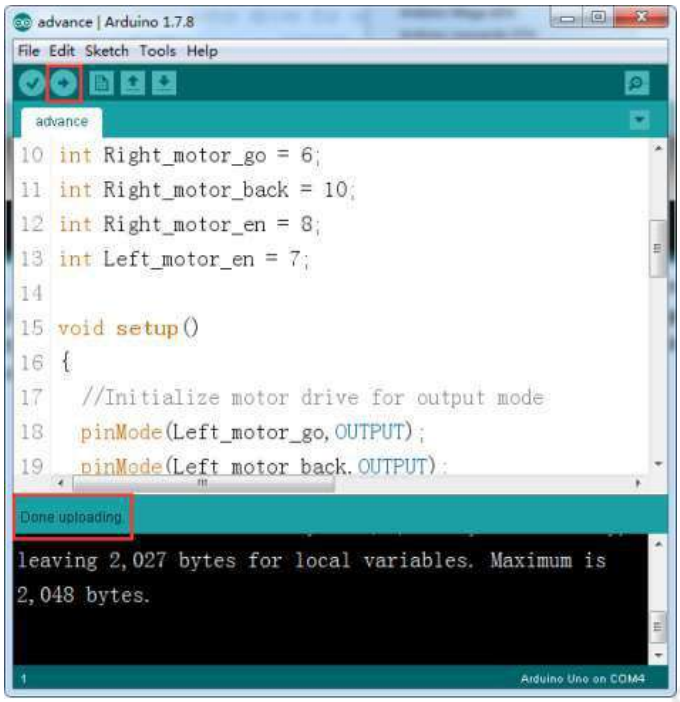
2. В меню Arduino IDE выберите пункт «Tools» —> «Port» и установите номер порта устройства, который отображается в Диспетчере устройств.



3. В меню «Tools» —> «Board» выберите пункт «Arduino Uno».



4. После завершения вышеперечисленных настроек нажмите на кнопку «→» под главным меню для загрузки кода в плату Arduino UNO. По окончании процесса загрузки в левом нижнем углу окна отображается сообщение «**Done uploading**».



5. После успешной загрузки программы, нажмите кнопку включения питания (рисунок 1 – Power switch). После 2-секундной паузы Batmobile начинает двигаться вперед.

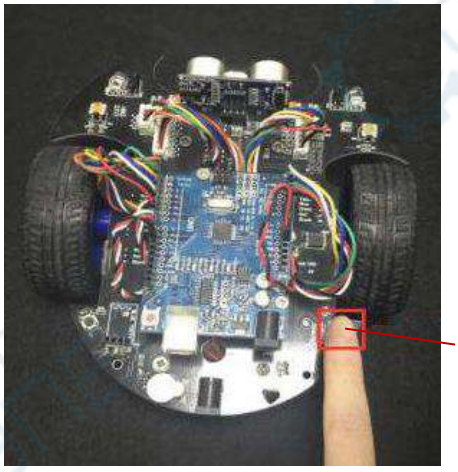


Рисунок 1

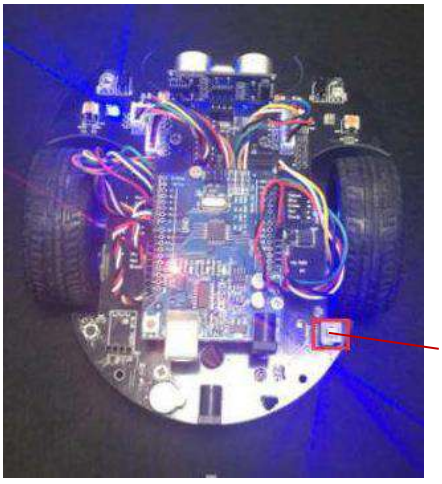


Рисунок 2

Урок 2. Запрограммированные движения после нажатия кнопки

Цель эксперимента

После включения питания и нажатия на кнопку K1 Batmobile издает короткий звуковой сигнал и движется в следующем порядке: вперед, назад, поворот налево, поворот направо, разворот налево, разворот направо.

Список необходимых компонентов:

- Batmobile — 1 шт.
- Кабель USB — 1 шт.

Программный код

```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/*Установить порт кнопки*/ int key=4;
/*Установить порт генератора звука*/ int beep=3;

void setup()
{
  // Инициализация двигатеей pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(key,INPUT); // Установить кнопку в качестве устройства ввода
  pinMode(beep,OUTPUT); // Установить генератор звука в качестве устройства вывода
 digitalWrite(key,HIGH); // Инициализация кнопки
digitalWrite(beep,HIGH); // Отключить звук
}
void run(int time)        // Запуск
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200); // ШИМ — широтно-импульсная модуляция (0-255).
  // Используется для управления скоростью.
```

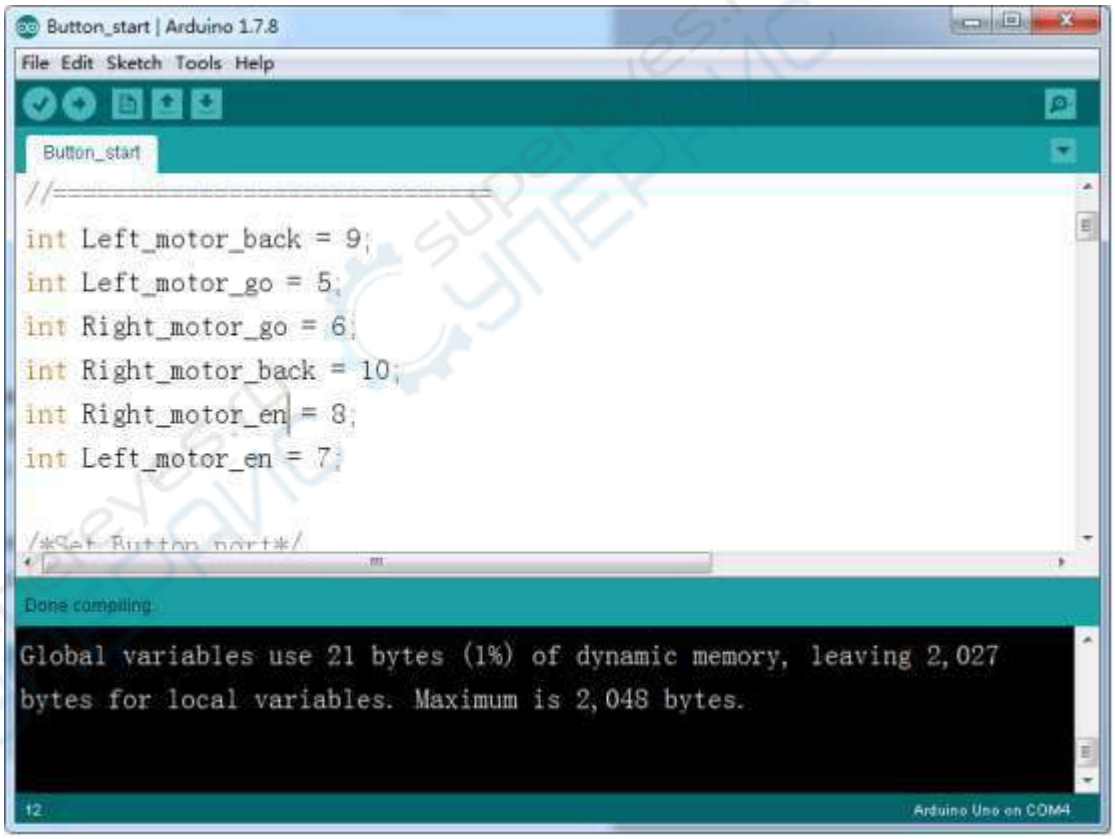
```
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Left_motor_back,0);
  delay(time * 100);      // Время работы, можно изменять
}
void brake(int time)              // Стоп
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
  delay(time * 100);
}
void left(int time) // Поворот налево
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение
  вперед digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW); // Левый электродвигатель: стоп
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0);
  delay(time * 100);
}
void spin_left(int time)      // Разворот налево
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,200); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW); // Левый электродвигатель: движение назад
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,200); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  delay(time * 100);
}
void right(int time)         // Поворот направо
{
  digitalWrite(Right_motor_go,LOW); // Правый электродвигатель: стоп
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель вперед
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
```

```
  analogWrite(Left_motor_back,0);
  delay(time * 100);
 }
 void spin_right(int time)       // Разворот направо
 {
  digitalWrite(Right_motor_go,LOW); // Правый электродвигатель: движение назад
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,200); // ШИМ — широтно-импульсная модуляция (0-255).
 // Используется для управления скоростью.
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Left_motor_back,0;
  delay(time * 100);
 }
 void back(int time)      // Движение назад
 {
  digitalWrite(Right_motor_go,LOW); // Правый электродвигатель: движение назад
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,150); // ШИМ — широтно-импульсная модуляция (0-255).
 // Используется для управления скоростью.
  digitalWrite(Left_motor_go,LOW); // Левый электродвигатель: движение назад
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,150); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  delay(time * 100);
 }
 void keysacn()
 {
  int val;
  val=digitalRead(key); // Значение состояния кнопки присваивается val
  while(digitalRead(key)) // Кнопка не нажата
  {
   val=digitalRead(key);
  }
  while(!digitalRead(key)) // Кнопка нажата
  {
   delay(10); // Пауза 10 мсек
    val=digitalRead(key); // Значение состояния кнопки присваивается val
    if(val==LOW)  // Двойная проверка нажатия кнопки
    {
      digitalWrite(beep,LOW); // Звуковой сигнал
      delay(100); // Задержка 100 мсек
       while(!digitalRead(key)) // Определение состояния кнопки
        digitalWrite(beep,HIGH); // Выключить звук
    }
    else
      digitalWrite(beep,HIGH); // Выключить звук
  }
```

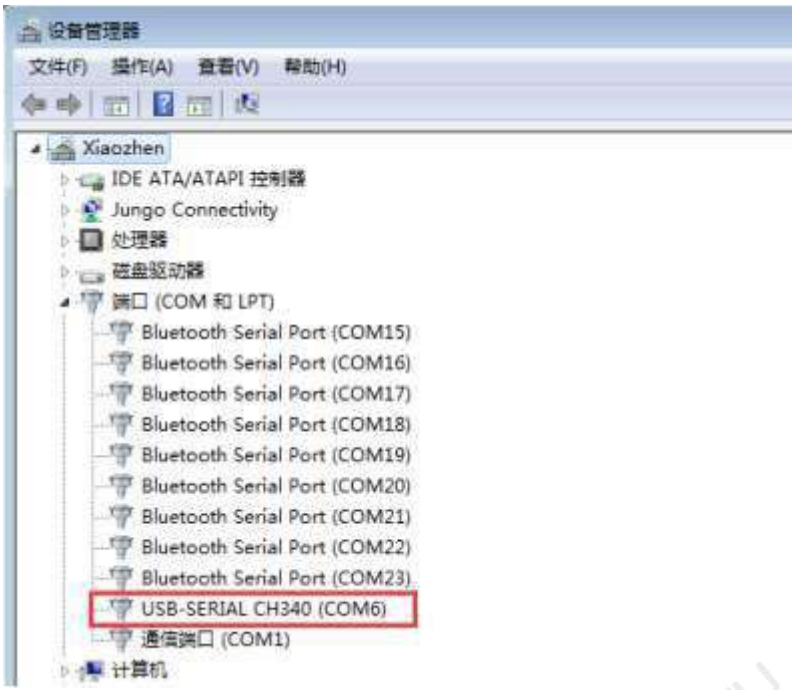
```
}
void loop()
{
  delay(2000); // Пауза 2 сек
  keysacn();
  back(10); // Движение назад 1 сек
  brake(5); // Останов 0,5 сек
  run(10); // Движение вперед 1 сек
  brake(5); // Останов 0,5 сек
  left(10); // Поворот налево 1 сек
  right(10); // Поворот направо 1 сек
  spin_left(20); // Разворот налево 2 сек
  spin_right(20); // Разворот направо 2 сек
  brake(5);  // Останов 0,5 сек
}
```

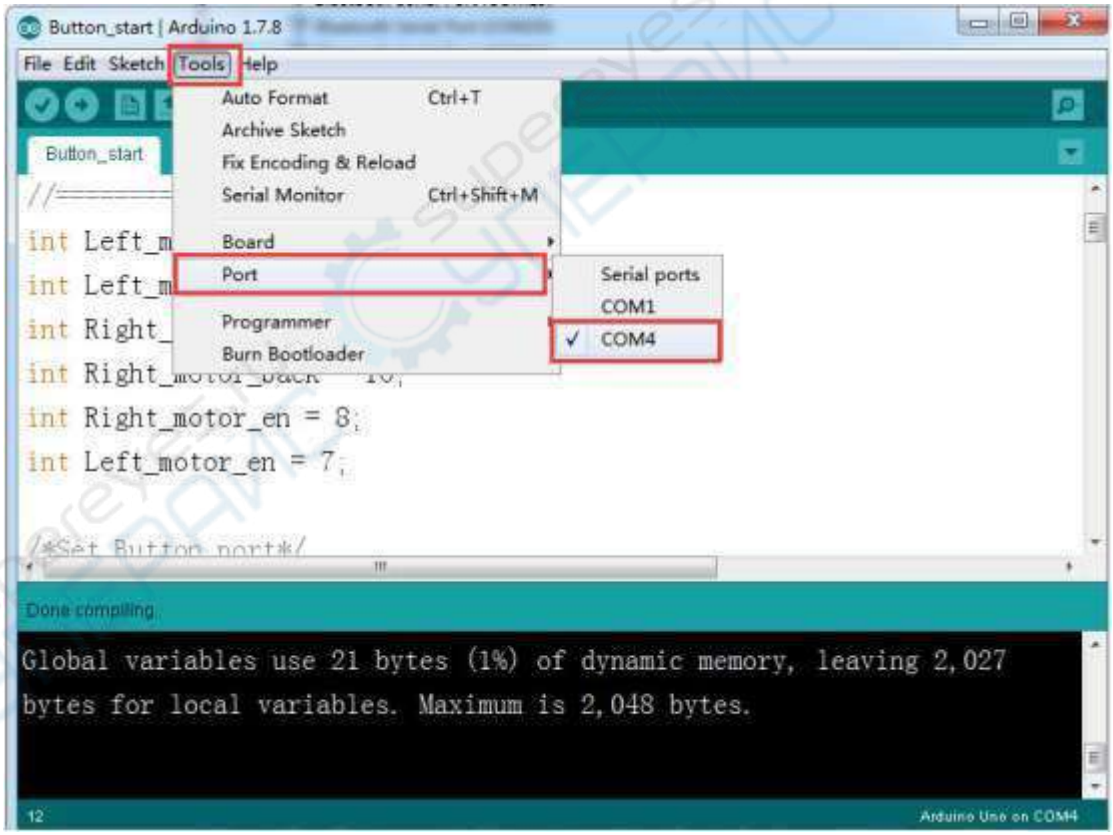
Этапы эксперимента

1. Откройте код **Button_start.ino**, нажмите на кнопку «√» под главным меню для компиляции кода и дождитесь сообщения «**Done compiling**» в нижнем правом углу, как показано на рисунке.

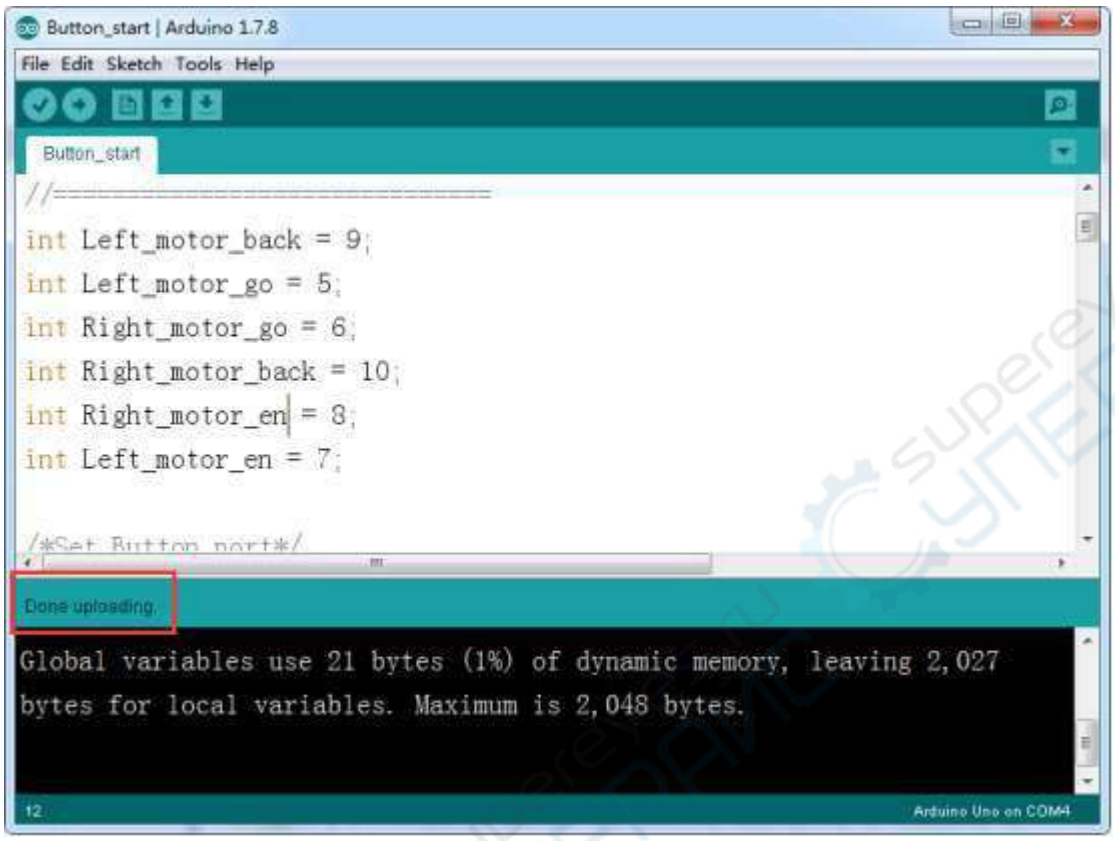


2. В меню Arduino IDE выберите пункт «Tools» —> «Port» и установите номер порта устройства, который отображается в Диспетчере устройств.

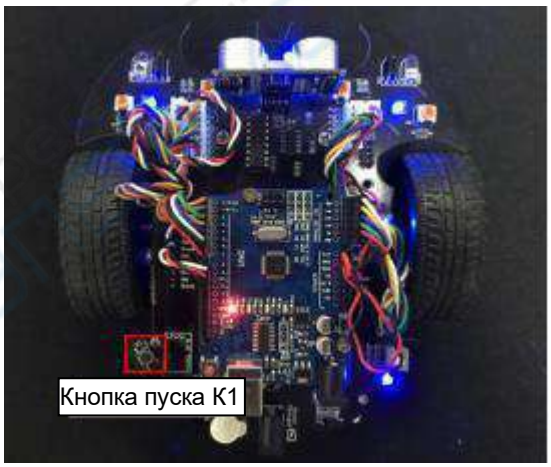


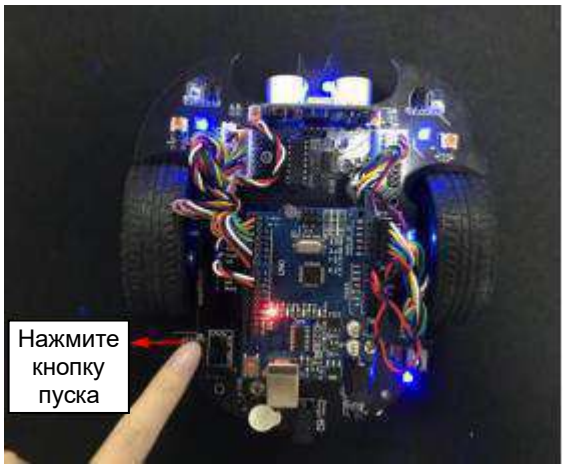


3. После завершения вышеперечисленных настроек нажмите на кнопку «→» под главным меню для загрузки кода в плату Arduino UNO. По окончании процесса загрузки в левом нижнем углу окна отображается сообщение «**Done uploading**».



4. Отключите кабель USB, установите Batmobile на открытую площадку и нажмите кнопку питания. Batmobile не начнет движение, пока не будет нажата кнопка пуска «K1». После нажатия на эту кнопку Batmobile движется в следующем порядке: вперед, назад, поворот налево, поворот направо и т.д.





Урок 3. Движение по линии

Цель эксперимента

После нажатия на кнопку Batmobile движется вдоль черной линии. Необходимо предварительно не только загрузить программу, но и отрегулировать чувствительность фотодатчика потенциометрами SW3 и SW4.

Список необходимых компонентов:

- Batmobile — 1 шт.
- Кабель USB — 1 шт.
- Трек с черной линией — 1 шт.

Программный код

```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/*Установка порта кнопки*/ int key=4;
/*Установка порта генератора звка*/ int beep=3;
/*Движение по линии*/
const int SensorRight = A3;  // Установить порт правого ИК-датчика
const int SensorLeft = A2;    // Установить порт левого ИК-датчика
int SL;     // Состояние левого ИК-датчика
int SR;     // Состояние правого ИК-датчика
void setup()
{
  // Инициализация электродвигателей
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(key,INPUT); // Установить режим ввода для
  кнопки
  pinMode(beep,OUTPUT); // Установить режим вывода для генератора звука
  pinMode(SensorRight, INPUT); // Установить режим ввода для правого ИК-датчика
  pinMode(SensorLeft, INPUT); // Установить режим ввода для левого ИК-датчика

  digitalWrite(key,HIGH); // Инициализация кнопки
  digitalWrite(beep,HIGH); // Отключить звук
```

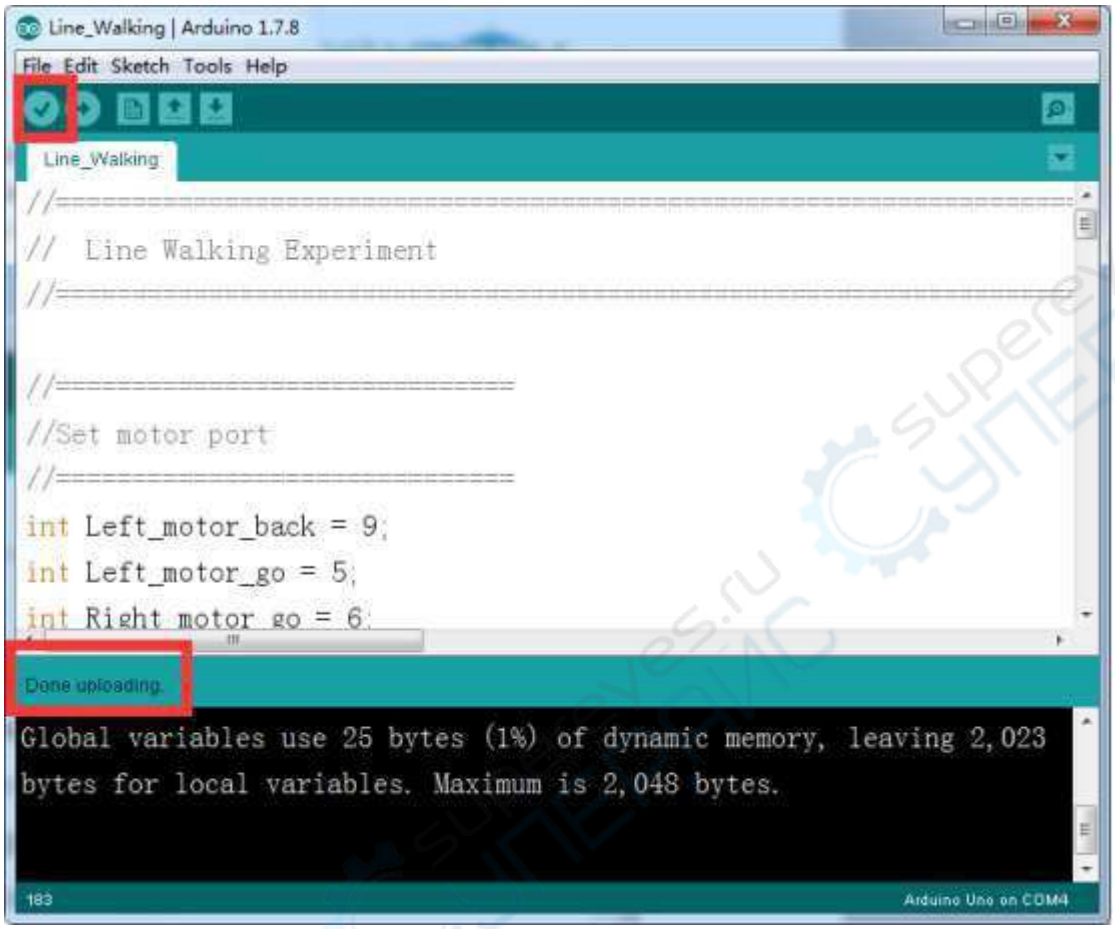
```
}
 //=======================Электродвигатели=========================
void run()
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
   digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение
  вперед
   digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Left_motor_back,0);
}
void brake() // Стоп
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
}
void left() // Поворот налево
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);
  analogWrite(Right_motor_back,0); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  digitalWrite(Left_motor_go,LOW);        // Левый электродвигатель: стоп
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
}
void spin_left(int time)            // Разворот налево
{
 digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
 digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);        // Левый электродвигатель: движение назад
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,100); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  delay(time * 100);
}
void right() // Поворот направо

{
  digitalWrite(Right_motor_go,LOW);        // Правый электродвигатель: стоп
  digitalWrite(Right_motor_back,LOW); analogWrite(Right_motor_go,0);
```

```
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100);
 analogWrite(Left_motor_back,0); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
 }
 void spin_right(int time)          // Разворот направо
 {
  digitalWrite(Right_motor_go,LOW);       // Правый электродвигатель: движение назад
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,200); // ШИМ — широтно-импульсная модуляция (0-255).
 // Используется для управления скоростью.
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,200);
  analogWrite(Left_motor_back,0); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  delay(time * 100);
 }
 void back(int time)  // Движение назад
 {
  digitalWrite(Right_motor_go,LOW); // Правый электродвигатель: движение назад
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,150); // ШИМ — широтно-импульсная модуляция (0-255).
 // Используется для управления скоростью.
  digitalWrite(Left_motor_go,LOW); // Левый электродвигатель: движение назад
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,150); // ШИМ — широтно-импульсная модуляция (0-255).
 // Используется для управления скоростью.
  delay(time * 100);
 }
 //==========================================================
 void keysacn()
 {
  int val;
  val=digitalRead(key); // Значение состояния кнопки присваивается val
  while(digitalRead(key)) // Кнопка не нажата
  {
    val=digitalRead(key);
  }
  while(!digitalRead(key)) // Кнопка нажата
  {
 delay(10); // пауза 10 мсек
    val=digitalRead(key); // Значение состояния кнопки присваивается val
    if(val==LOW)  // Двойная проверка нажатия кнопки
    {

      digitalWrite(beep,LOW); // Звуковой сигнал
      delay(50); // Пауза 50 мсек
```

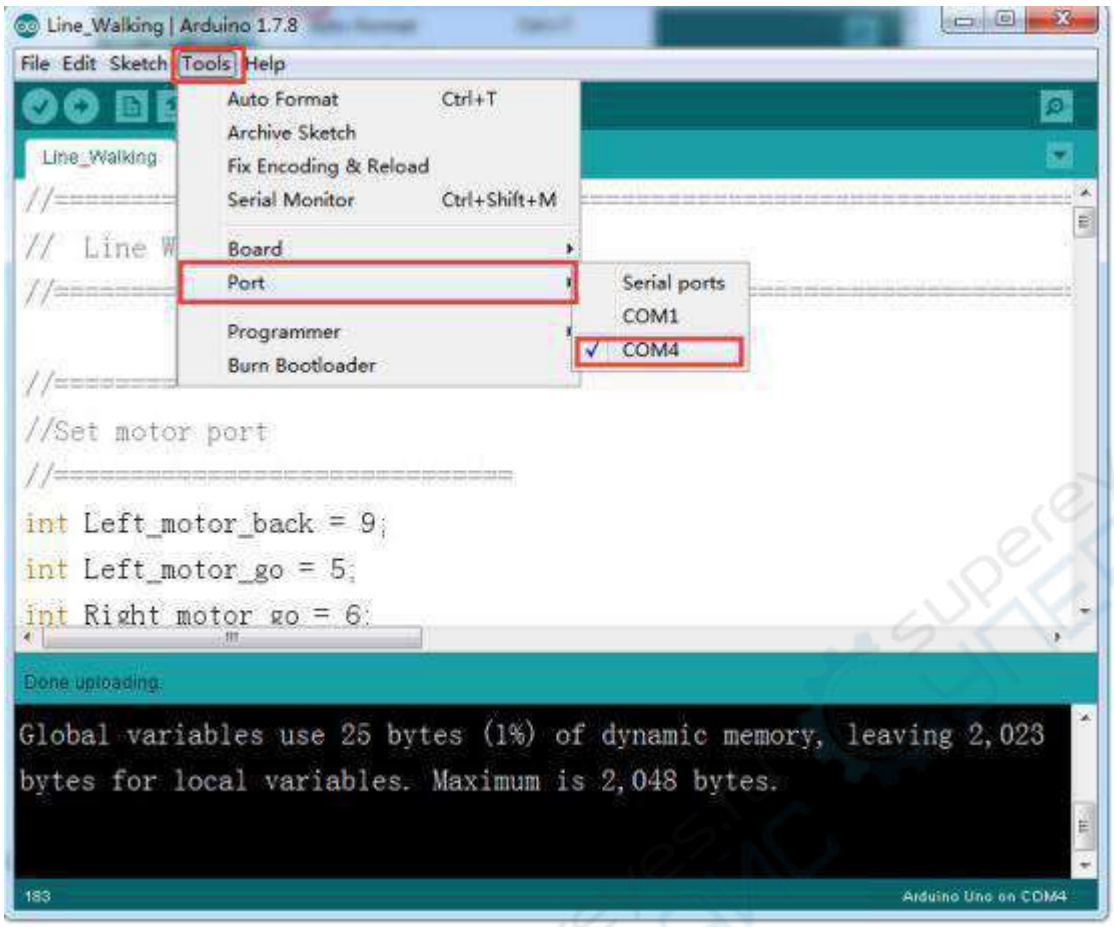
```
  while(!digitalRead(key)) // Определение состояния кнопки
        digitalWrite(beep,HIGH); // Выключить звук
    }
    else
      digitalWrite(beep,HIGH); // Выключить звук
  }
}
/*Главный цикл*/
 void
loop()
{
  keysacn(); // Нажмите кнопку для запуска
  while(1)
  {
  /*********************************************************************************
  При обнаружении белой поверхности — низкий уровень, светодиод включается
  При обнаружении черной поверхности — высокий уровень, светодиод гаснет.
  *********************************************************************************/
  SR = digitalRead(SensorRight); // Правый ИК-датчик находится над белой
поверхностью. LED[L2] включается над белой поверхностью и гаснет над черной.
  SL = digitalRead(SensorLeft); // Левый ИК-датчик находится над белой поверхностью.
LED[L3] включается над белой поверхностью и гаснет над черной.
  if (SL ==LOW&&SR== LOW) // Черные линии не обнаружены
  run();      // Движение вперед
  else if (SL == LOW & SR == HIGH) // Левый датчик над белой поверхностью, а правый
//датчик над черной поверхностью; машина сошла с линии, необходимо повернуть
//направо.
    right();
  else if (SR == LOW & SL == HIGH) // Правый датчик находится над белой
//поверхностью, а левый датчик — над черной поверхностью; машина сошла с линии,
//необходимо повернуть налево.
    left();
  else // Черные линии обнаружены обоими датчиками, останов машины
  brake();
  }
}
```

Этапы эксперимента

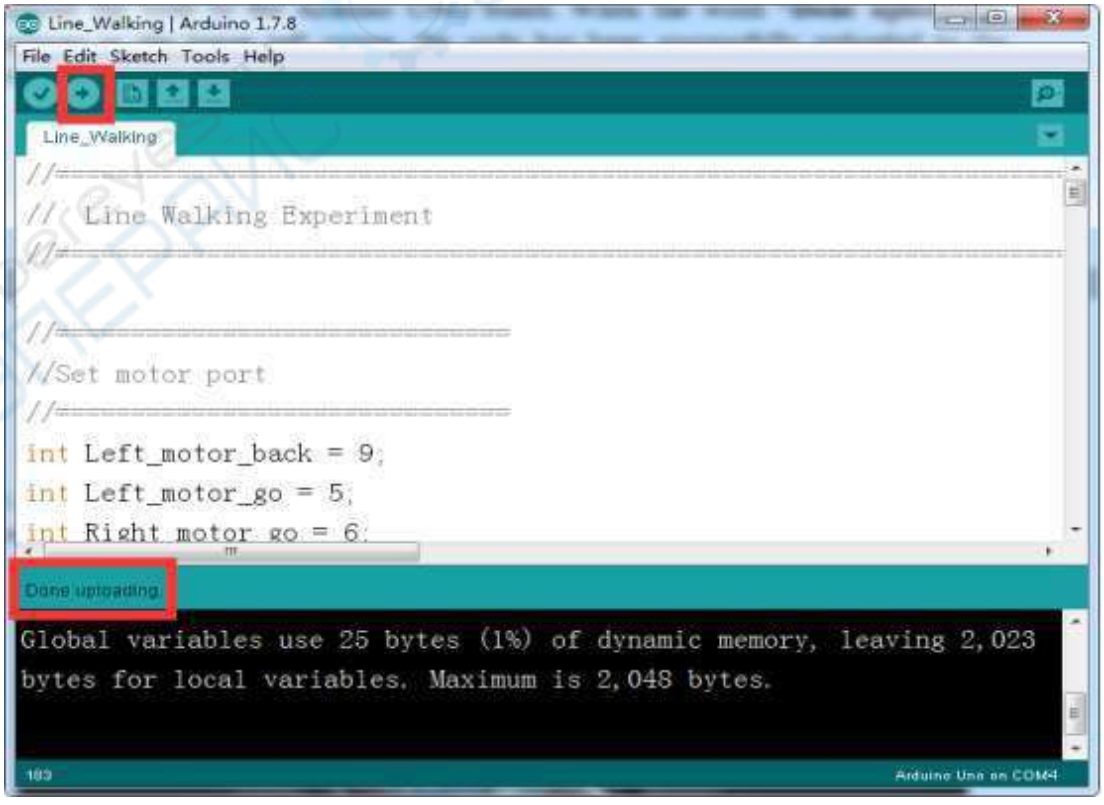
1. Откройте код **Line_Walking.ino**, нажмите на кнопку «√» под главным меню для компиляции кода и дождитесь сообщения «**Done compiling**» в нижнем правом углу, как показано на рисунке.



2. В меню Arduino IDE выберите пункт «Tools» —> «Port» и установите номер порта устройства, который отображается в Диспетчере устройств.



3. После завершения вышеперечисленных настроек нажмите на кнопку «→» под главным меню для загрузки кода в плату Arduino UNO. По окончании процесса загрузки в левом нижнем углу окна отображается сообщение «**Done uploading**».

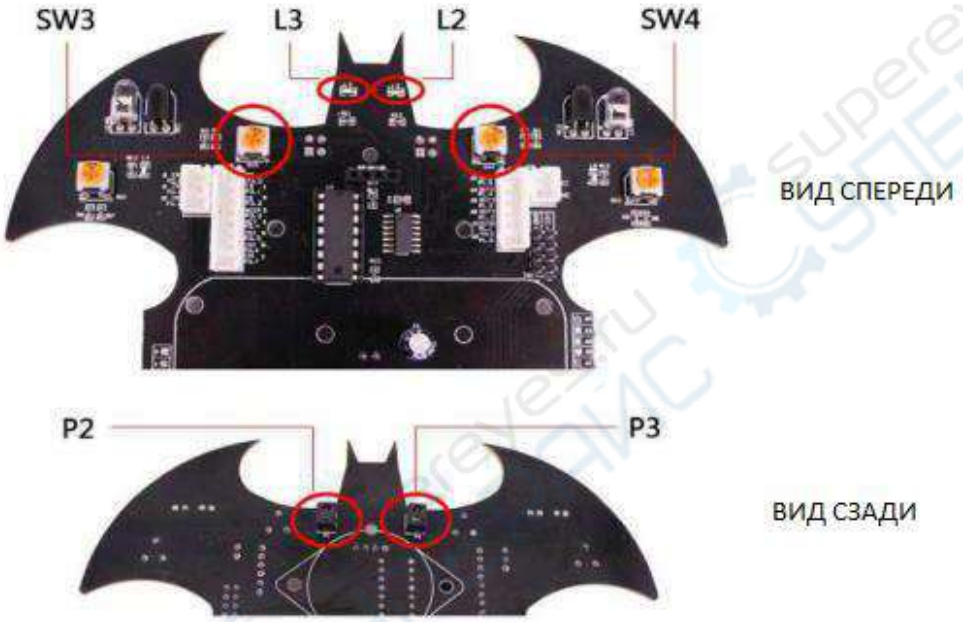


4. После загрузки программы отрегулируйте настройку фотодатчика потенциометрами SW3 и SW4.

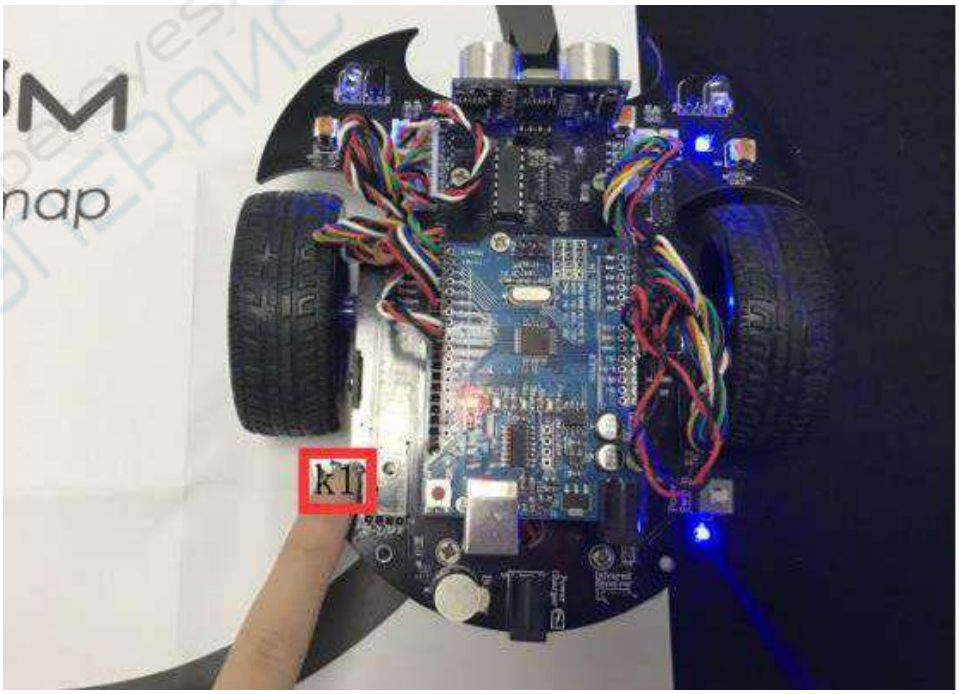
Отладка

- Отрегулируйте потенциометром SW3 работу фотодатчика P3 таким образом, чтобы светодиод L3 включался при нахождении датчика P3 над белой поверхностью и гас при нахождении датчика P3 над черной поверхностью.
- Отрегулируйте потенциометром SW4 работу фотодатчика P2 таким образом, чтобы светодиод L2 включался при нахождении датчика P2 над белой поверхностью и гас при нахождении датчика P2 над черной поверхностью.

Внимание: поворачивать потенциометр нужно в пределах 30°.



5. Установите Batmobile на трек и нажмите кнопку K1. Batmobile начинает движение вдоль черной линии.



Урок 4. Объезд препятствий

Цель эксперимента

Отрегулируйте потенциометры SW1 и SW2 в соответствии с описанием в конце урока. Включите питание, после чего Batmobile издает короткий звуковой сигнал и начинает движение, объезжая препятствия.

Список необходимых компонентов:

- Batmobile — 1 шт.
- Кабель USB — 1 шт.

Программный код

```
int Left_motor_back = 9;
int Left_motor_go = 5;
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
/* Установить порт кнопки */ int key=4;
/* Установить порт генератора звука */ int beep=3;
/*Движение по линии*/
const int SensorRight = A3;  // Установить порт правого ИК-датчика
const int SensorLeft = A2;    // Установить порт левого ИК-датчика
int SL;     // Состояние левого ИК-датчика
int SR;     // Состояние правого ИК-датчика
/*объезд препятствий*/
const int SensorRight_2 = A4;         // Правый ИК-датчик
const int SensorLeft_2 = A5;         // Левый ИК-датчик
int SL_2;      // Состояние левого ИК-датчика
int SR_2;      // Состояние правого ИК-датчика
void setup()
{
  // Инициализация электродвигателей
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(key,INPUT); // Установить режим ввода для кнопки
  pinMode(beep,OUTPUT); // Установить режим вывода для генератора звука
  pinMode(SensorRight, INPUT); // Установить режим ввода для правого ИК-датчика
  //следования за линией
```

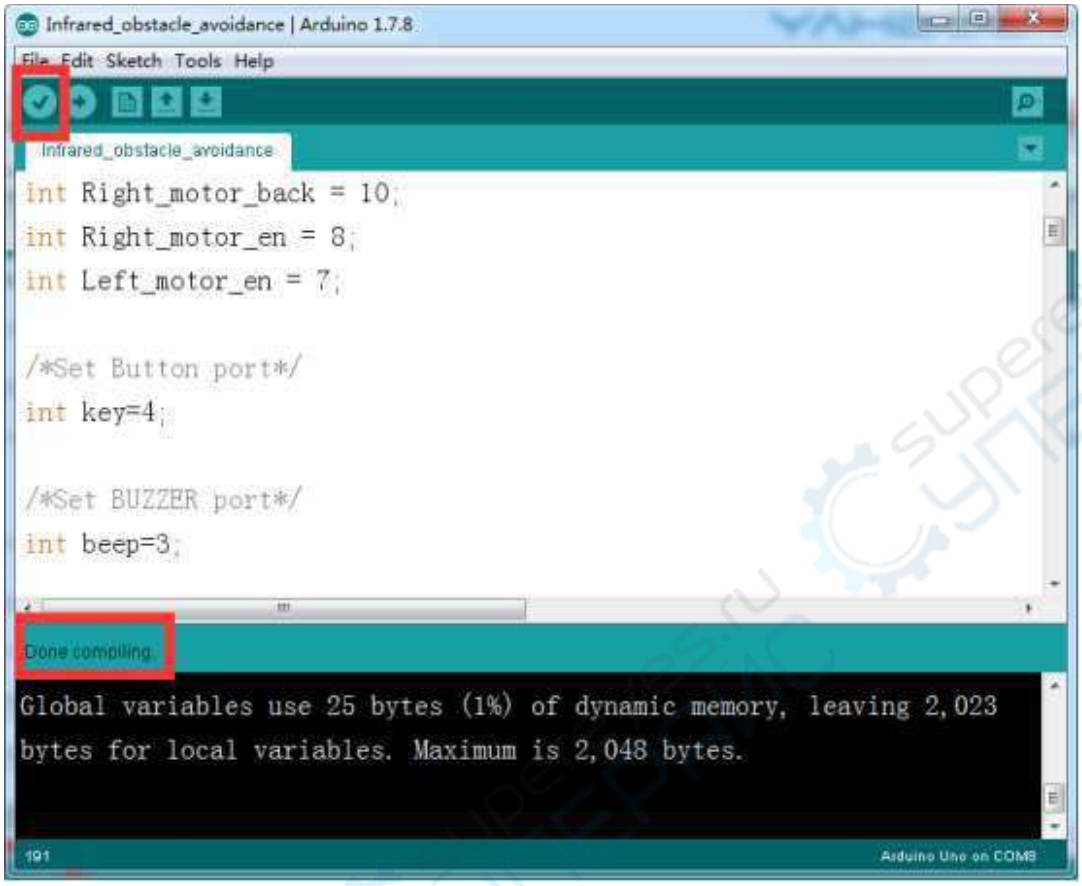
```
  pinMode(SensorLeft, INPUT); // Установить режим ввода для левого ИК-датчика
  // следования за линией
  pinMode(SensorRight_2, INPUT); // Установить режим ввода для правого ИК-датчика
  pinMode(SensorLeft_2, INPUT); // Установить режим ввода для левого ИК-датчика
  digitalWrite(key,HIGH); // Инициализация кнопки
  digitalWrite(beep,HIGH); // Отключить звук
}
 //======================Электродвигатели=========================
void run()
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
   digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
   digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,100); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Left_motor_back,0);
}
void brake() // Стоп
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
}
void left() // Поворот налево
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100);
  analogWrite(Right_motor_back,0); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  digitalWrite(Left_motor_go,LOW);        // Левый электродвигатель: стоп
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,0); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
}
void spin_left(int time)          // Разворот налево
{
 digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
 digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,100); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
  analogWrite(Right_motor_back,0);
  digitalWrite(Left_motor_go,LOW);        // Левый электродвигатель: движение назад
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,100); // ШИМ — широтно-импульсная модуляция (0-255).
```

```
 // Используется для управления скоростью.
   delay(time * 100);
 }
 void right() // Поворот направо
 {
   digitalWrite(Right_motor_go,LOW);          // Правый электродвигатель: стоп
   digitalWrite(Right_motor_back,LOW); analogWrite(Right_motor_go,0);
   analogWrite(Right_motor_back,0);
   digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
   digitalWrite(Left_motor_back,LOW);
   analogWrite(Left_motor_go,100);
  analogWrite(Left_motor_back,0); // ШИМ — широтно-импульсная модуляция (0-255).
// Используется для управления скоростью.
 }
 void spin_right(int time)           // Разворот направо
 {
   digitalWrite(Right_motor_go,LOW);        // Правый электродвигатель: движение назад
   digitalWrite(Right_motor_back,HIGH);
   analogWrite(Right_motor_go,0);
   analogWrite(Right_motor_back,200); // ШИМ — широтно-импульсная модуляция (0-255).
  // Используется для управления скоростью.
   digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
   digitalWrite(Left_motor_back,LOW);
   analogWrite(Left_motor_go,200);
   analogWrite(Left_motor_back,0); // ШИМ — широтно-импульсная модуляция (0-255).
 // Используется для управления скоростью.
   delay(time * 100);
 }
 void back(int time)  // Движение назад
 {
   digitalWrite(Right_motor_go,LOW); // Правый электродвигатель: движение назад
   digitalWrite(Right_motor_back,HIGH);
   analogWrite(Right_motor_go,0);
   analogWrite(Right_motor_back,150); // ШИМ — широтно-импульсная модуляция (0-255).
  // Используется для управления скоростью.
   digitalWrite(Left_motor_go,LOW); // Левый электродвигатель: движение назад
   digitalWrite(Left_motor_back,HIGH);
   analogWrite(Left_motor_go,0);
   analogWrite(Left_motor_back,150); // ШИМ — широтно-импульсная модуляция (0-255).
  // Используется для управления скоростью.
   delay(time * 100);
  }
  //=========================================================
 void keysacn()
 {
   int val;
   val=digitalRead(key); // Значение состояния кнопки присваивается val
   while(digitalRead(key)) // Кнопка не нажата
   {
     val=digitalRead(key);
   }
   while(!digitalRead(key)) // Кнопка нажата
   {
```

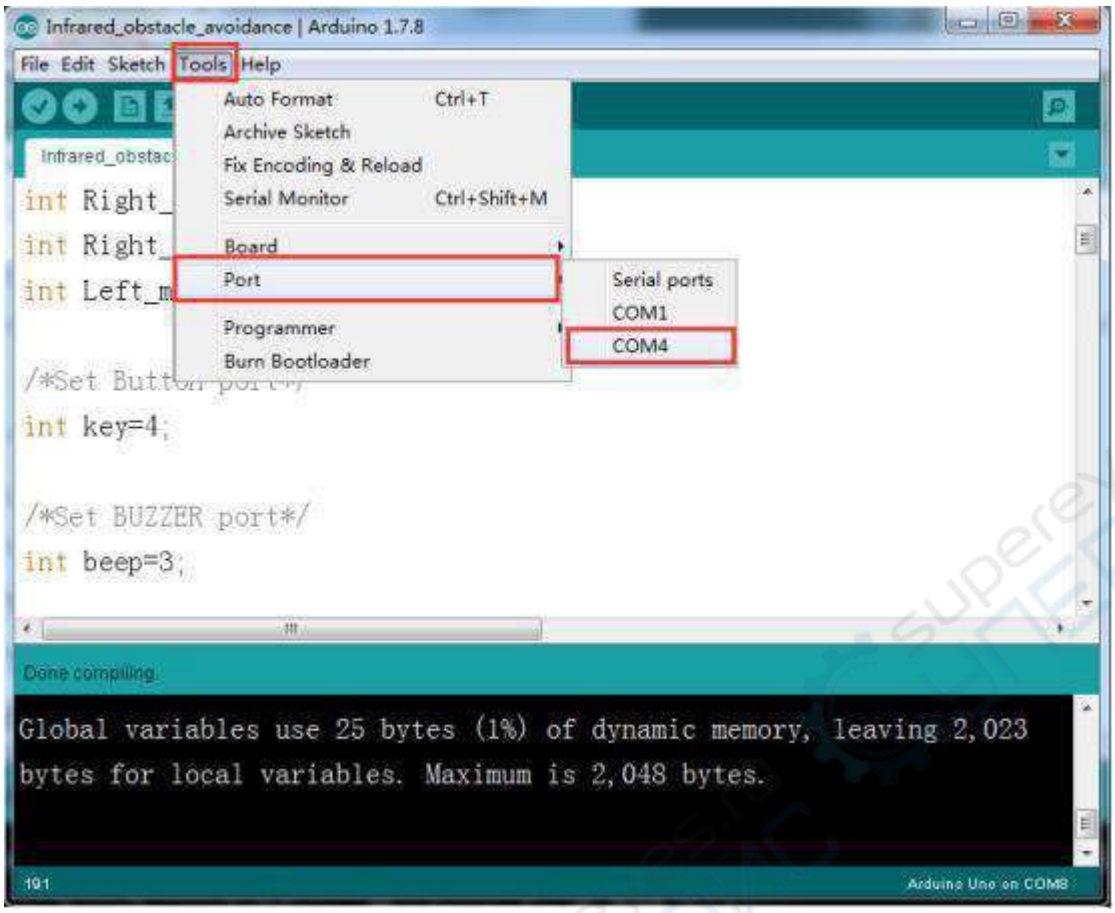
```
 delay(10); // пауза 10 мсек
   val=digitalRead(key); // Значение состояния кнопки присваивается val
   if(val==LOW)  // Двойная проверка нажатия кнопки
   {

     digitalWrite(beep,LOW); // Звуковой сигнал
     delay(50); // Пауза 50 мсек
 while(!digitalRead(key)) // Определение состояния кнопки
       digitalWrite(beep,HIGH); // Выключить звук
   }
   else
     digitalWrite(beep,HIGH); // Выключить звук
 }
}
/*Главный цикл*/
 void loop()
{
  keysacn(); // Нажмите кнопку для запуска
  while(1)
  {
  /*********************************************************************************
При обнаружении препятствия (отражении ИК-сигнала от препятствия)
формируется низкий уровень и загорается светодиод. Если ИК-сигнал не
возвращается (не отражается), формируется высокий уровень и светодиод
гаснет.
  *********************************************************************************/
    SR_2 = digitalRead(SensorRight_2); // Если правый ИК-датчик  обнаруживает
//препятствие, светодиод [L5] зажигается, в противном случае светодиод гаснет.
    SL_2 = digitalRead(SensorLeft_2); // Если правый ИК-датчик  обнаруживает
//препятствие, светодиод [L4] зажигается, в противном случае светодиод гаснет.
    if (SL_2 == HIGH&&SR_2==HIGH) // Препятствий нет, продолжать движение
    run();
    else if (SL_2 == HIGH & SR_2 == LOW) // Препятствие справа, светодиод [L4]
     // Включается, поворот налево.
      left();
    else if (SR_2 == HIGH & SL_2 == LOW) ) // Препятствие слева, светодиод [L4]
     // Включается, поворот нправо.

       right();
    else // // Есть препятствие, нужно сдать назад и скорректировать направление.
    {
      back(6); // Движение назад 600 мсек
      spin_right(3); // Разворот направо в течение 300 мсек для коррекции направления
    }
  }
}
```

Этапы эксперимента

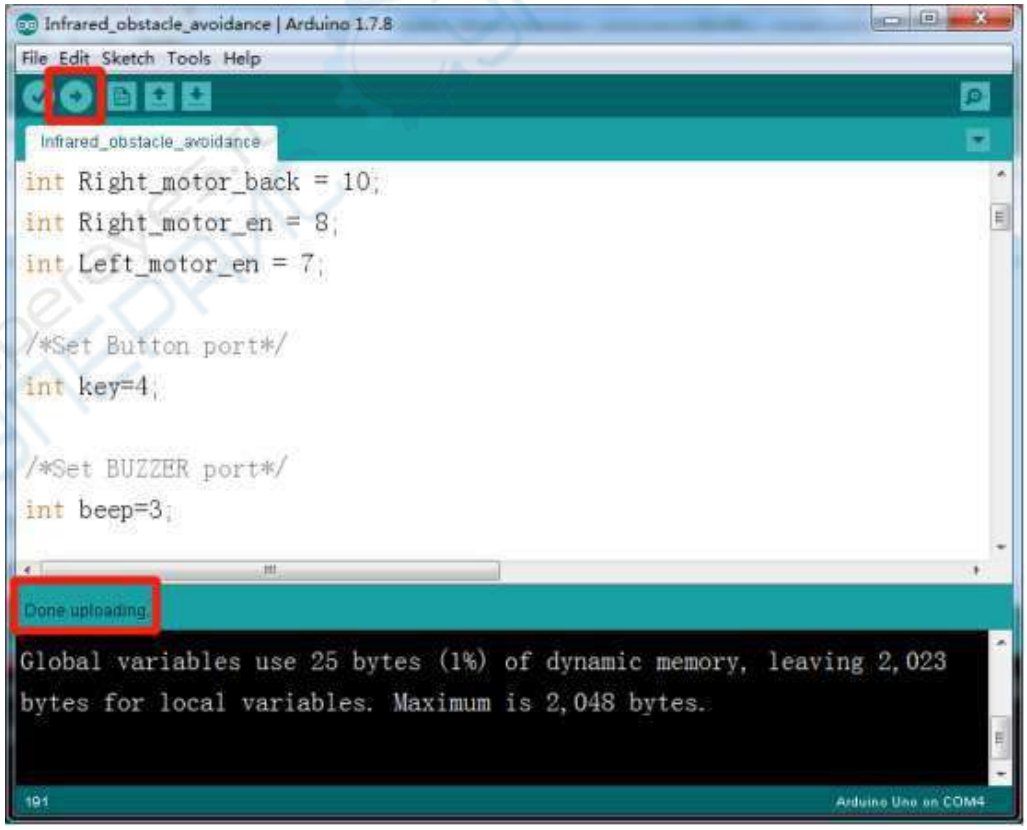
1. Откройте код **Infrared_obstacle_avoidance.ino**, нажмите на кнопку «√» под главным меню для компиляции кода и дождитесь сообщения «**Done compiling»** в нижнем правом углу, как показано на рисунке.



2. В меню Arduino IDE выберите пункт «Tools» —> «Port» и установите номер порта устройства, который отображается в Диспетчере устройств.



3. После завершения вышеперечисленных настроек нажмите на кнопку «→» под главным меню для загрузки кода в плату Arduino UNO. По окончании процесса загрузки в левом нижнем углу окна отображается сообщение «**Done uploading**».

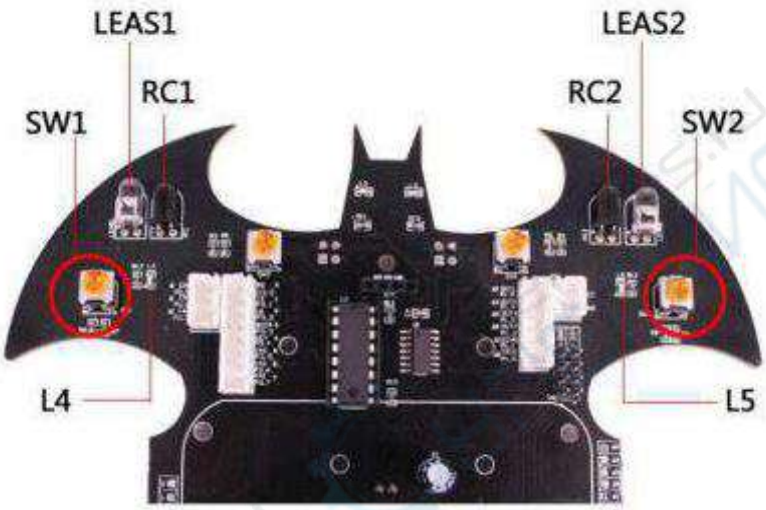


4. После загрузки программы проведите следующие настройки.

Отладка:

- Отрегулируйте потенциометром SW1 работу датчиков таким образом, чтобы при нахождении излучающего диода [LEAS1] и приемного диода [RC1] на расстоянии 10 см от препятствия и менее включался светодиод [L4]. Светодиод [L4] должен гаснуть при увеличении расстояния от препятствия до значений более10 см.
- Отрегулируйте потенциометром SW2 работу датчиков таким образом, чтобы при нахождении излучающего диода [LEAS2] и приемного диода [RC2] на расстоянии 10 см от препятствия и менее включался светодиод [L5]. Светодиод [L5] должен гаснуть при увеличении расстояния от препятствия до значений более10 см.

  Внимание: поворачивать потенциометр нужно в пределах 30°.



5. Установите Batmobile на открытую площадку и расставьте несколько картонных коробок в качестве препятствий. Включите питание Batmobile и нажмите кнопку пуска K1. Batmobile издаст короткий звуковой сигнал и начнет движение, объезжая препятствия.

Урок 5. Управление с помощью пульта

Цель эксперимента

Управление Batmobile с помощью пульта управления в затемненном помещении. Пульт необходимо направлять на ИК-приемник в задней части Batmobile. Управление осуществляется цифровыми кнопками.

Список необходимых компонентов:

- Batmobile — 1 шт.
- Кабель USB — 1 шт.
- ИК-пульт – 1 шт.



Программный код

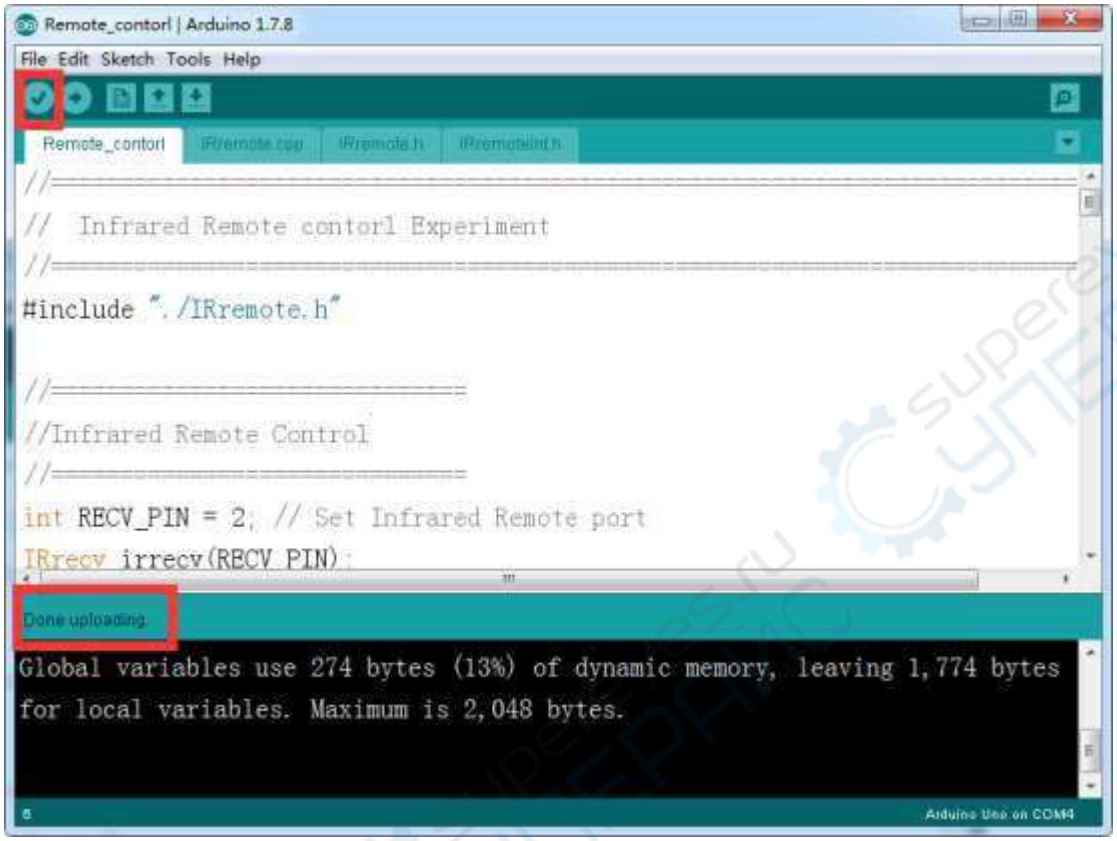
```
//==========================================================
===========
//  Управление с помощью пульта
//==========================================================
============
#include "./IRremote.h"
//==============================
//Управление с помощью ИК-пульта управления
//==============================
int RECV_PIN = 2; // Установить порт ИК-управления
IRrecv irrecv(RECV_PIN);
decode_results results; // Сохранить декодирование
data unsigned long last = millis();
int on = 0; // Флаг ИК-приемника
long run_car = 0x00FF18E7; // Кнопка 2
long back_car = 0x00FF4AB5; // Кнопка 8
long left_car = 0x00FF10EF; // Кнопка 4
long right_car = 0x00FF5AA5; // Кнопка 6
long stop_car = 0x00FF38C7; // Кнопка 5
long left_turn = 0x00ff30CF; // Кнопка 1
long right_turn = 0x00FF7A85; // Кнопка 3
//==============================
//Установить порт электродвигателя
//==============================
int Left_motor_back = 9;
int Left_motor_go = 5;
```

```
int Right_motor_go = 6;
int Right_motor_back = 10;
int Right_motor_en = 8;
int Left_motor_en = 7;
void setup()
{
  // Инициализация электродвигателей
  pinMode(Left_motor_go,OUTPUT);
  pinMode(Left_motor_back,OUTPUT);
  pinMode(Right_motor_go,OUTPUT);
  pinMode(Right_motor_back,OUTPUT);
  pinMode(13, OUTPUT); // Показать сигнал ИК-пульта
  irrecv.enableIRIn(); // Запустить приемник
}
void run()
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение
  вперед
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
  digitalWrite(Left_motor_back,LOW);
}
void brake() // Стоп
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);

void left() // Поворот налево
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
      digitalWrite(Right_motor_back,LOW);
digitalWrite(Left_motor_go,LOW);              // Левый электродвигатель: стоп
      digitalWrite(Left_motor_back,LOW);
}
void spin_left()          // Разворот налево
{
  digitalWrite(Right_motor_go,HIGH); // Правый электродвигатель: движение вперед
digitalWrite(Right_motor_back,LOW);
digitalWrite(Left_motor_go,LOW);             // Левый электродвигатель: движение назад
digitalWrite(Left_motor_back,HIGH);
}
void right() // Поворот направо
{
  digitalWrite(Right_motor_go,LOW);          // Правый электродвигатель: стоп
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение
  вперед
  digitalWrite(Left_motor_back,LOW);
}
void spin_right()          // Разворот направо
{
  digitalWrite(Right_motor_go,LOW);        // Правый электродвигатель: движение назад
```

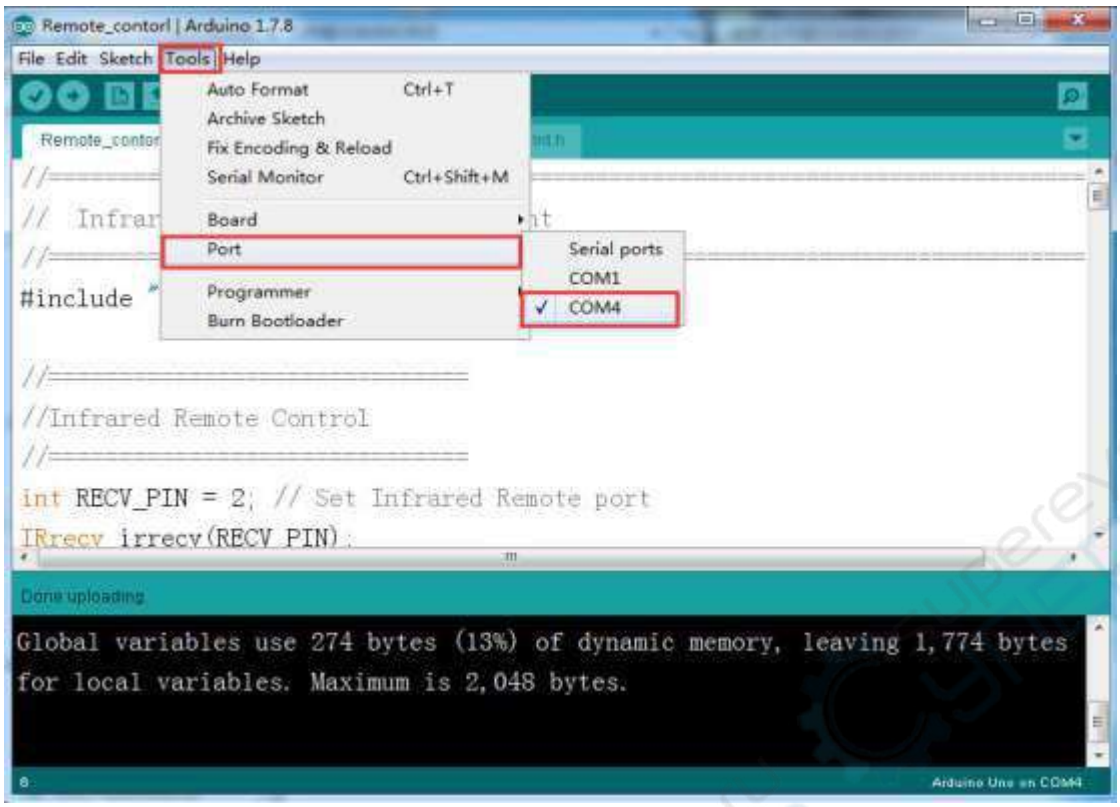
```
  digitalWrite(Right_motor_back,HIGH);
  digitalWrite(Left_motor_go,HIGH); // Левый электродвигатель: движение вперед
  digitalWrite(Left_motor_back,LOW);
}
void back()  // Движение назад
{
  digitalWrite(Right_motor_go,LOW); // Правый электродвигатель: движение назад
  digitalWrite(Right_motor_back,HIGH);
  digitalWrite(Left_motor_go,LOW); // Левый электродвигатель: движение назад
  digitalWrite(Left_motor_back,HIGH);
}
//=========================================================
void dump(decode_results *results) //декодирование принятых сигналов ИК-пульта
{
  int count = results->rawlen;
  if (results->decode_type == UNKNOWN)
  {   brake();
  }
}
void loop()
{
  if (irrecv.decode(&results)) // Получен ИК-сигнал
  {
    if (millis() - last > 250) // Проверка получения
    {
      on = !on; //инверсия
      digitalWrite(13, on ? HIGH : LOW); //LED blink
      dump(&results); // Декодирование
    }
    if (results.value == run_car ) // Кнопка "2"
    run(); // Движение вперед
    if (results.value == back_car ) // Кнопка "8"
    back(); // Движение назад
    if (results.value == left_car ) // Кнопка "4"
    left(); // Поворот налево
    if (results.value == right_car ) // Кнопка "6"
    right(); // Поворот направо
    if (results.value == stop_car ) // Кнопка "5"
    brake(); // Стоп
    if (results.value == left_turn ) // Кнопка "1"
    spin_left(); // Разворот налево
    if (results.value == right_turn ) // Кнопка "2"
    spin_right(); // Поворот направо
    last = millis();
    irrecv.resume(); // Получить следующее значение
  }
}
```

Этапы эксперимента

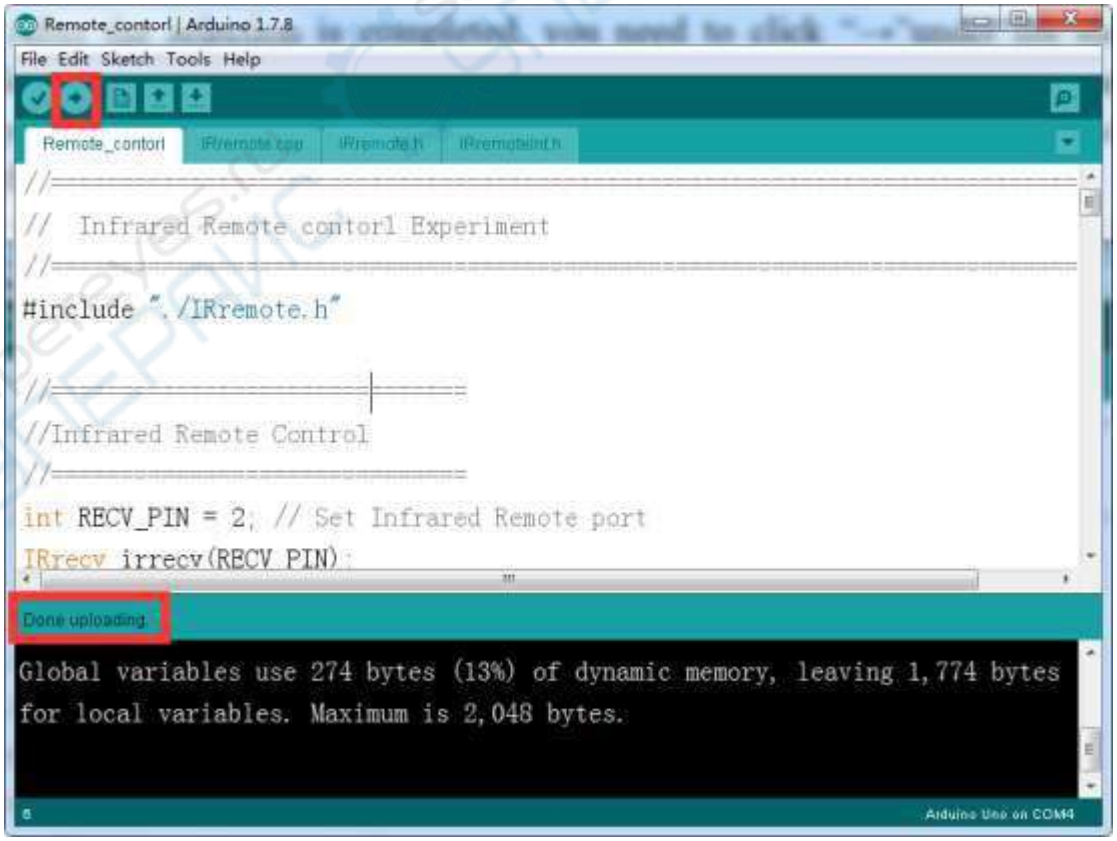
1. Откройте код **Remote_contorl.ino**, нажмите на кнопку «√» под главным меню для компиляции кода и дождитесь сообщения «**Done compiling**» в нижнем правом углу, как показано на рисунке.



2. В меню Arduino IDE выберите пункт «Tools» —> «Port» и установите номер порта устройства, который отображается в Диспетчере устройств.



2. После завершения вышеперечисленных настроек нажмите на кнопку «→» под главным меню для загрузки кода в плату Arduino UNO. По окончании процесса загрузки в левом нижнем углу окна отображается сообщение «**Done uploading**».



3. После загрузки программы **«Remote control»** установите Batmobile на площадку в затемненном помещении (закройте шторы для блокирования внешнего освещения). Направьте пульт на ИК-приемник в задней части бэтмобиля и нажмите необходимую кнопку. Назначение кнопок ИК-пульта приведено в таблице ниже.

Кнопка ИК-пульта	Пользовательский код	Действие
0	0x00FF6897	Нет
—	0x00FF9867	Нет
C	0x00FFB04F	Нет
1	0x00ff30CF	Разворот налево
2	0x00FF18E7	Движение вперед
3	0x00FF7A85	Разворот направо
4	0x00FF10EF	Поворот налево
5	0x00FF38C7	Останов
6	0x00FF5AA5	Поворот направо
7	0x00FF42BD	Нет
8	0x00FF4AB5	Движение назад
9	0x00FF52AD	Нет

Пользовательский код: 00FF