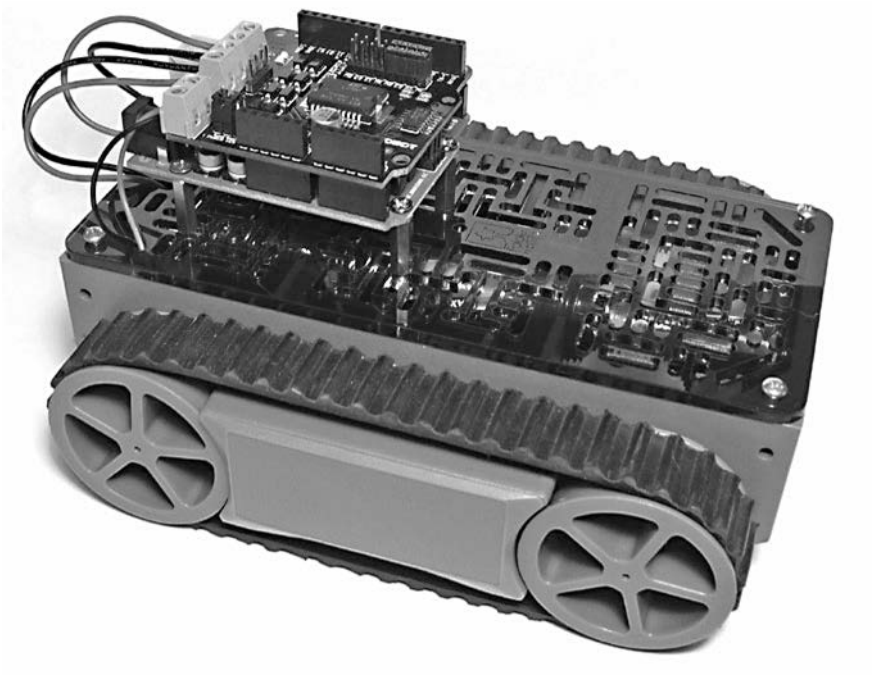

Рис. 12.20. Танк готов к испытаниям!

Скетч

Теперь заставим танк двигаться. Сначала создадим несколько функций, которые потом упростят управление движением. Так как танк приводится в движение двумя электродвигателями, нам потребуется реализовать четыре вида движений:

- движение вперед;
- движение назад;
- поворот по часовой стрелке;
- поворот против часовой стрелки.

Наша плата расширения управляет каждым двигателем с помощью двух цифровых выходов: один регулирует скорость вращения с применением ШИМ (как демонстрировалось в проекте 39), а другой определяет направление вращения.

Четырем видам движения в скетче соответствуют четыре функции: `goForward()`, `goBackward()`, `rotateLeft()` и `rotateRight()`. Каждая принимает значение в миллисекундах, определяющее время, в течение которого будут включены электродвигатели, и скорость в виде значения ШИМ от 0 до 255. Например, чтобы заставить танк двигаться вперед на полной скорости в течение 2 секунд, нужно выполнить вызов `goForward(2000, 255)`.

Введите и сохраните следующий скетч (но пока не загружайте его):

```
// Проект 40 - Роботизированный танк и управление им
int m1speed=6; // цифровые выходы, управляющие скоростью
int m2speed=5;
int m1direction=7; // цифровые выходы, управляющие направлением
int m2direction=4;

void setup()
{
  pinMode(m1direction, OUTPUT);
  pinMode(m2direction, OUTPUT);
  delay(5000);
}

void goForward(int duration, int pwm)
{
❶  digitalWrite(m1direction,HIGH); // вперед
  digitalWrite(m2direction,HIGH); // вперед
  analogWrite(m1speed, pwm);      // скорость
  analogWrite(m2speed, pwm);
  delay(duration);
  analogWrite(m1speed, 0);        // скорость
  analogWrite(m2speed, 0);
}

void goBackward(int duration, int pwm)
{
  digitalWrite(m1direction,LOW); // назад
❷  digitalWrite(m2direction,LOW); // назад
  analogWrite(m1speed, pwm);     // скорость
  analogWrite(m2speed, pwm);
  delay(duration);
  analogWrite(m1speed, 0);       // скорость
  analogWrite(m2speed, 0);
}

void rotateRight(int duration, int pwm)
{
❸  digitalWrite(m1direction,HIGH); // вперед
  digitalWrite(m2direction,LOW);  // назад
  analogWrite(m1speed, pwm);      // скорость
  analogWrite(m2speed, pwm);
  delay(duration);
  analogWrite(m1speed, 0);        // скорость
  analogWrite(m2speed, 0);
}

void rotateLeft(int duration, int pwm)
{
❹  digitalWrite(m1direction,LOW);  // назад
  digitalWrite(m2direction,HIGH); // вперед
  analogWrite(m1speed, pwm);      // скорость
  analogWrite(m2speed, pwm);
```

```
    delay(duration);
    analogWrite(m1speed, 0);        // скорость
    analogWrite(m2speed, 0);
  }

  void loop()
  {
    goForward(1000, 255);
    rotateLeft(1000, 255);
    goForward(1000, 255);
    rotateRight(1000, 255);
    goForward(1000, 255);
    goBackward(2000, 255);
    delay(2000);
}
```

Направление вращения каждого электродвигателя устанавливается вызовом:

```
digitalWrite(m1direction,direction);
```

Значение `HIGH` в параметре `direction` соответствует вращению в прямом направлении, а `LOW` — в обратном. То есть, чтобы танк двигался вперед, необходимо установить одно и то же направление для обоих электродвигателей, что и делается в строках ❶ и ❷. Скорость вращения устанавливается вызовом:

```
analogWrite(m1speed, pwm);
```

Параметр `pwm` определяет скорость и может принимать значения от 0 до 255. Чтобы танк повернул влево или вправо, электродвигатели должны вращаться в противоположных направлениях, как это делается в строках ❸ и ❹.

ВНИМАНИЕ

Когда вы будете готовы загрузить скетч, поднимите танк над поверхностью стола, чтобы его гусеницы свободно вращались в воздухе; если этого не сделать, то после загрузки скетча танк рванется вперед и может упасть со стола!

Загрузите скетч, отсоедините кабель USB и включите кабель от 9-вольтовой батареи в гнездо питания на плате Arduino. Затем поместите танк на ковер или другую поверхность и отпустите его. Поэкспериментируйте с функциями движения в проекте 40; это поможет вам лучше понять суть задержек и их влияние на величину пройденного расстояния.

Определение столкновений

Теперь, научив наш танк двигаться, мы добавим дополнительные средства, такие как датчики определения, которые сообщат танку о произошедшем столкновении или измерят расстояние между танком и объектом на его пути, чтобы танк мог свернуть

и избежать столкновения. Мы будем использовать три способа предотвращения столкновений: микровыключатели, инфракрасные и ультразвуковые датчики.

Проект № 41: Определение столкновений с помощью микровыключателя

Микровыключатель действует как обычная кнопка без фиксации, которую мы использовали в главе 4; отличие в том, что микровыключатель имеет большие размеры и длинную металлическую пластину, играющую роль рычага (рис. 12.21).

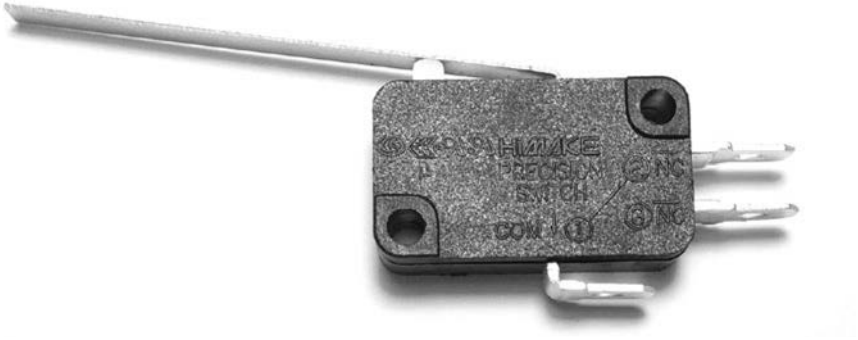

Рис. 12.21. Микровыключатель

Обычно при использовании микровыключателя один провод подключается к нижнему выводу, а другой — к выводу с меткой NO (normally open — нормально разомкнутый), чтобы ток тек только при нажатом рычаге. Мы смонтируем микровыключатель на передней панели танка, и когда танк достигнет какого-то препятствия, рычаг замкнет микровыключатель, ток потечет через его выводы и заставит танк изменить направление движения.

Схема

Подсоедините микровыключатель как обычную кнопку, как показано на рис. 12.22.

Скетч

Мы подключили микровыключатель к цифровому контакту 2, поддерживающему прерывания. Может показаться, что в функции, вызываемой по прерываниям, мы должны заставить танк двигаться в обратном направлении в течение какого-то времени, однако это невозможно, потому что функция `delay()` не работает внутри обработчиков прерываний. В данном случае необходимо найти какое-то другое решение.

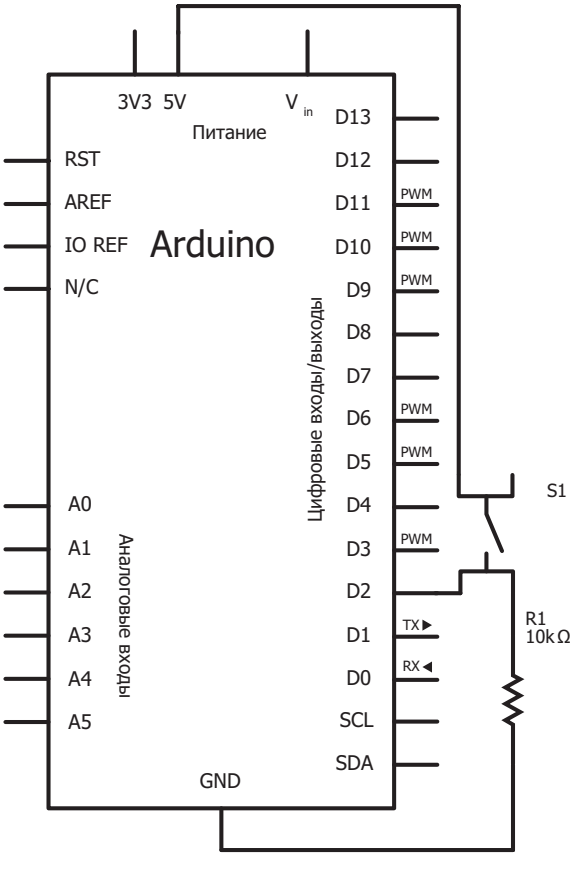

Рис. 12.22. Схема подключения микровыключателя для определения столкновений

Это решение заключается в следующем: функция `goForward()` будет включать электродвигатели только при соблюдении двух условий, задаваемых логическими переменными `crash` и `move`. Если `crash` имеет значение `true`, электродвигатели будут на 2 секунды включаться в обратном направлении с низкой скоростью, чтобы «выйти» из соприкосновения с препятствием.

Мы не можем вызывать функцию `delay()` из-за использования прерывания, поэтому время работы электродвигателей будет измеряться следующим образом: перед включением будет вызываться `millis()`, чтобы засечь момент включения двигателей, и затем в цикле каждое новое ее значение будет сравниваться с начальным. Когда разность сравняется или превысит требуемую продолжительность, функция присвоит переменной `move` значение `false` и остановит электродвигатели.

Введите и загрузите следующий скетч: