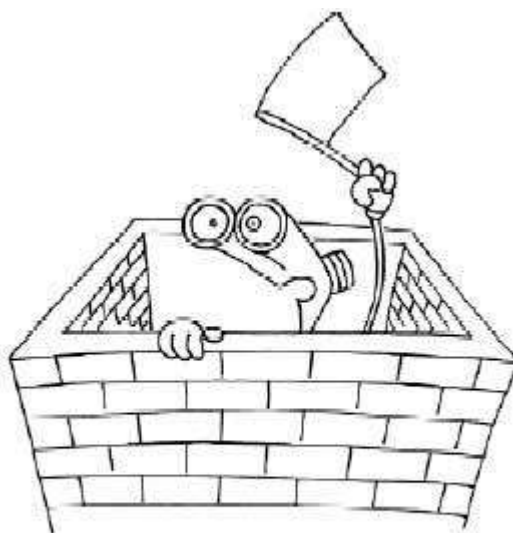




*...на пути движения будут возникать
серьезные повороты, вплоть до прямых углов.*

ПОВОРОТ ЗА УГОЛ

Следующим шагом необходимо организовать реакцию робота на «бесконечность». Как известно, когда в поле зрения датчика расстояния NXT равны 250 или 255 см. Если это число попадает на пропорциональный регулятор, робот начинает крутиться на месте. А в ситуации, когда роботу следует завернуть за угол, именно это и произойдет.

Для объезда предметов потребуется ввести контроль показаний датчика расстояния: при резком изменении робот должен делать вывод о возможном повороте, который надо будет производить с другими коэффициентами или просто с постоянным значением управляющего воздействия.



Рис. 10. Как и в первой конструкции, датчик располагается вертикально

Рассмотрим пример поворота направо «за угол». Если робот движется на расстоянии L от стены, то и поворот, очевидно, он будет выполнять по окружности с радиусом L (рис. 12).

Нетрудно рассчитать, каким должно быть отношение скоростей колес, чтобы радиус поворота оказался равен L . Для этого достаточно измерить расстояние между передними колесами. Пусть в нашем роботе оно будет равно $k = 16$ см, а его половина $d = 16/2 = 8$ см. Тогда левое и правое колеса движутся по окружностям радиусов, соответственно, $R_1 = L + d$ и $R_2 = L - d$. Пройденные ими пути за единицу времени должны быть пропорциональны радиусам, следовательно, скорости точек крепления колес v_1 и v_2 связаны следующим отношением

$$\frac{v_1}{v_2} = \frac{R_1}{R_2}$$

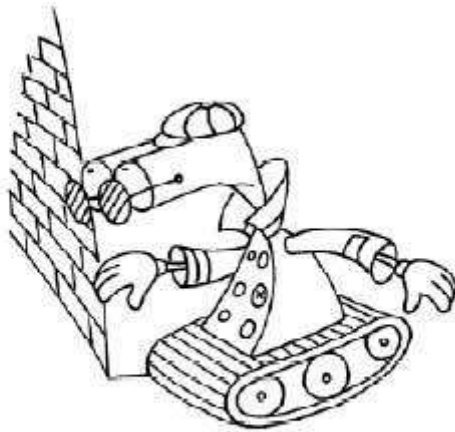
Выражая скорости перемещения ко лес через базовую скорость v и неизвестную x ,
а радиусы через L , получаем

$$\frac{v+x}{v-x} = \frac{L+d}{L-d},$$

$$vL + xL - vd - xd = vL + vd - xL - xd,$$

$$2xL = 2vd, \quad 2xL = 2vd, \quad 2xL = 2vd,$$

$$v_2 = v - \frac{vd}{L} = v\left(1 - \frac{d}{L}\right),$$



*Рассмотрим пример поворота направо
«за угол».*

Линейная скорость v пропорциональна угловой скорости колеса w , которая, в свою очередь, пропорциональна мощности, подаваемой на моторы (в режиме торможения). Мы привели закон управления к стандартному виду, что позволяет задать управляющее воздействие на время поворота за угол. Таким образом, получаем расчет для управления моторами нашего робота.

$$u = 50 \cdot 8 / L;$$

$$\text{motor}[\text{motorB}] = 50 + u;$$

$$\text{motor}[\text{motorC}] = 50 - u;$$

Когда расстояние до стены становится больше $2L$ (используем такой порог видимости), то есть открывается поворот за угол, управляющее воздействие начинает вычисляться по приведенным формулам (рис. 13).



Рис. 11. Увеличенное за счет корпуса робота расстояние до стены способствует расширению области обзора

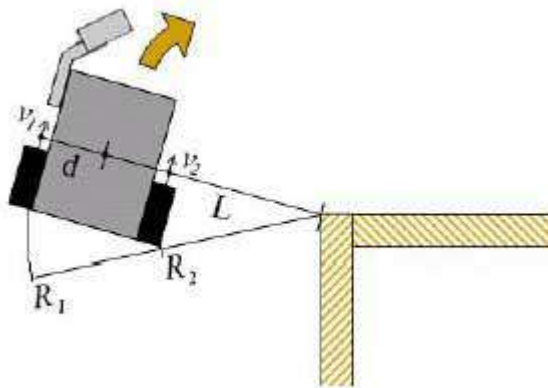


Рис. 12. Выполнение поворота при потере контакта со стенкой

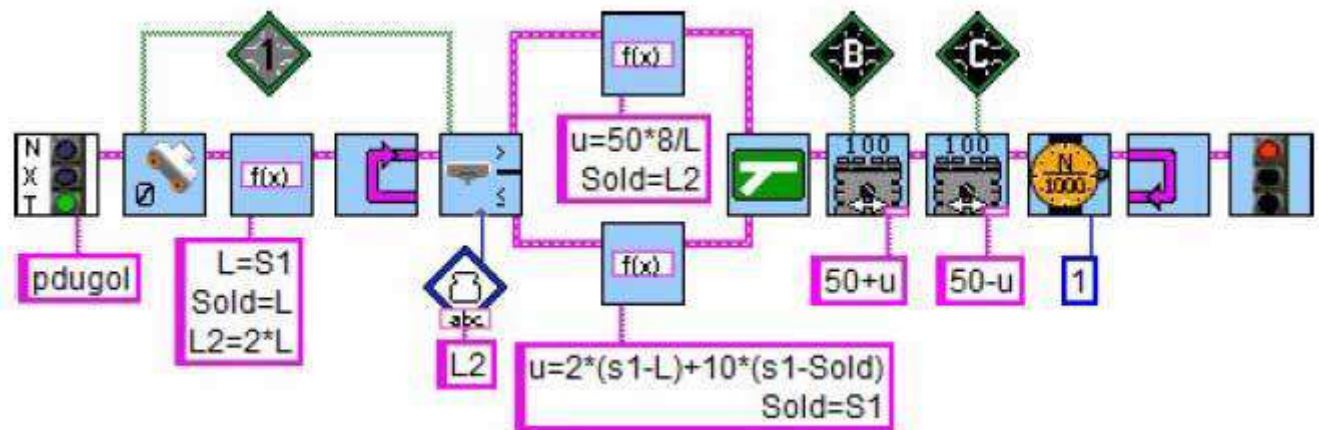


Рис. 13. Объезд предметов на заданном расстоянии по правилу правой руки
В программе на RobotC добавим переменных, чтобы сделать ее более строгой и наглядной. При этом не требуется вводить переменную L2, поскольку, в отличие от Robolab, здесь имеется возможность писать в условиях формулы.

```
task main() {
float u, k1=2, k2=10;
int v=50, d=8, Sold, L;
Sold=L=SensorValue[S1];
while(true) {
if (SensorValue[S1]>L*2) {
```

```

u=v*d/L; Sold=L*2;
}
else { u = k1*(SensorValue[S1]-L) + k2*(SensorValue[S1]-Sold);
Sold=SensorValue[S1];
} motor[motorB]=v+u;
motor[motorC]=v-u;
wait1Msec(1);
}
}

```

Надо заметить, что мы слегка обманываем робота, подставляя ему ограниченные сверху значения. Такой алгоритм будет стабильно работать на расстоянии L от 25 до 125 см.

ФИЛЬТРАЦИЯ ДАННЫХ

Наконец, для окончательной стабилизации робота следует ввести защиту от помех. В силу особенностей работы ультразвукового датчика, сигнал время от времени не попадает на глазок-приемник, и в результате не всегда поступают адекватные значения, что может внести серьезные возмущения в наш алгоритм. Например, небольшая щель в стене для робота выглядит целым проемом. Поскольку требования к скорости пока что невысоки, можно несколько замедлить реакцию датчика на изменения расстояния, установив программный фильтр его значений. Простейшая реализация такого фильтра состоит в усреднении очередного показания и предыдущего отфильтрованного значения:

$$S_{new} = (S_{new} + S1)/2.$$

Среднее арифметическое несколько сглаживает резкие скачки показаний датчика. Во всех формулах вместо S1 будет использоваться отфильтрованное значение S_{new}. Однако и этого может не хватить при слишком резких изменениях. Тогда следует ввести весовые коэффициенты c1 и c2 для усредненной и новой величины.

$$S_{new} = (c1*S_{new} + c2*S1)/(c1 + c2).$$

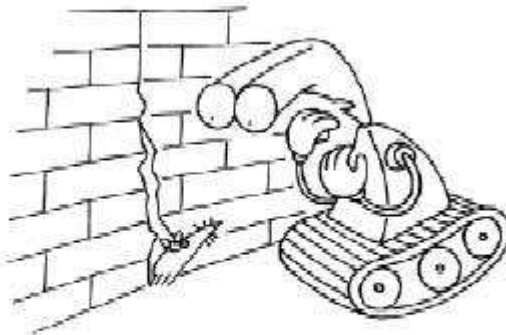
Раскрываем первые скобки

$$S_{new} = \frac{c1}{c1+c2} \cdot S_{new} + \frac{c2}{c1+c2} \cdot S1.$$

Легко заметить, что сумма полученных коэффициентов равна 1. Поэтому, заменив их, соответственно, на 1 - a и a, получим

$$S_{new} = (1 - a)*S_{new} + a*S1,$$

где $0 \leq a \leq 1$.



*...небольшая щель в стене для робота
выглядит целым проемом.*

Подбирая значение a , можно регулировать степень фильтрации. Для начала примем $a = 0.2$. Сравнение результатов фильтрации показаний датчика ультразвука в течение 1 с приведено на рис. 14.

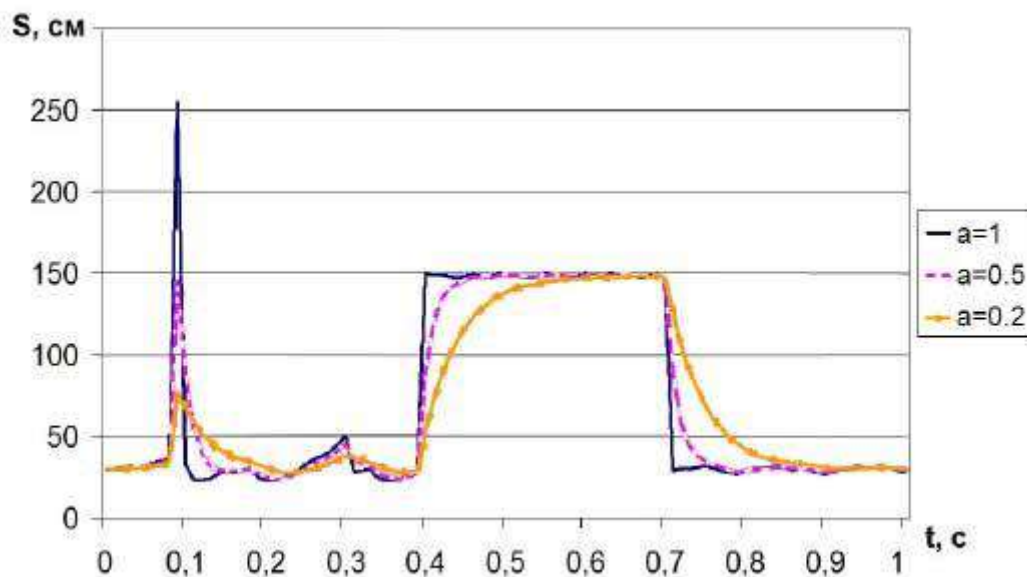


Рис. 14. Результаты фильтрации показаний датчика ультразвука с различными коэффициентами

Синий график показывает показания датчика, пурпурный . фильтр по среднему арифметическому, желтый . фильтр с параметром $a = 0.2$. По графикам видно, что при малых значениях параметра фильтрация происходит эффективнее, и случайные скачки показаний практически нивелируются. С другой стороны, более-менее постоянные высокие значения достигаются в доли секунды, причем переход становится достаточно точно плавным. Перепишем программы под новые отфильтрованные показания датчика. Поначалу результат может оказаться неожиданным, однако подбор коэффициентов все расставит по своим местам (рис. 15).

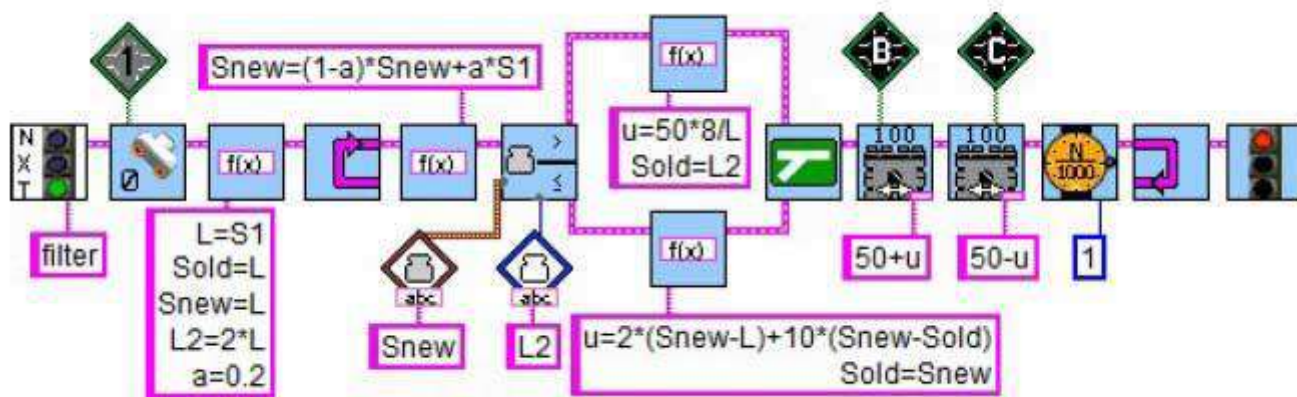


Рис. 15. Алгоритм движения вдоль стенки с фильтрацией данных

```
task main() {
float u, k1=2, k2=10, a=0.2, Snew; int v=50, d=8, Sold, L;
Snew=Sold=L=SensorValue[S1];
while(true) {
Snew=(1-a)*Snew+a*SensorValue[S1];
if (Snew>L*2) { u=v*d/L; Sold=L*2;

```

```

}
else { u = k1*(Snew-L) + k2*(Snew-Sold);
Sold=Snew;
} motor[motorB]=v+u;
motor[motorC]=v-u;
wait1Msec(1);
}
}

```

Фильтрация данных становится особенно актуальной, если на их основе требуется принимать решение о дальнейших действиях в долгосрочной перспективе. Например, увидев проем, остановиться или повернуть назад. Достаточно одной помехи, чтобы робот остановился не в том месте. Поэтому фильтры, хоть и затормаживают реакцию робота, но делают ее более стабильной и предсказуемой. Филиппов Сергей Александрович, учитель информатики физико-математического лицея № 239, методист.

Литература

1. С.А. Филиппов. Робототехника для детей и родителей. Под ред. А.Л. Фрадкова. СПб.: Наука, 2010.
2. М.С. Ананьевский, Г.И. Болтунов, Ю.Е. Зайцев, А.С. Матвеев, А.Л. Фрадков, В.В. Шиегин. Санкт Петербургские олимпиады по кибернетике. Под ред. А.Л. Фрадкова, М.С. Ананьевского. СПб.: Наука, 2006.
3. Сайт подразделения Lego Education: <http://www.lego.com/education/>.
4. Среда трехмерного моделирования Lego Digital Designer: <http://ldd.lego.com/>.
5. Среда программирования RobotC: <http://www.robotc.net/>.
6. Сайт поддержки пользователей Lego Mindstorms, Robolab 2.9.4 и пр.: <http://www.legoengineering.com/>.
7. Сайт о роботах, робототехнике и микроконтроллерах: <http://www.myrobot.ru/>.