Карпов В.Э.

Управление движением робота с использованием энкодеров

При управлении движением робота зачастую требуется, чтобы робот максимально точно отработал то или иное движение. Самый простой способ - использование временных задержек (скажем, включить оба двигателя и подождать какое-то количество секунд). Однако при этом точность движения бывает крайне неудовлетворительной: во-первых, двигатели обычно вращаются с разными скоростями, во-вторых, по мере просаживания напряжения питания также изменяется скорость вращения и т.п.

Одним из решений поставленной задачи является использование датчиков угла поворота, которые устанавливаются на выходные валы приводов. Разумеется, наличие таких датчиков - это еще не панацея, т.к. существует множество факторов, мешающих точной отработке движений (например, неизбежной проскальзывание колес). Тем не менее, подобного рода датчики значительно улучшают управляемость системы.

Энкодеры

Датчики угла поворота (или энкодеры) - это устройства, при помощи которых можно определять положение вращающихся валов. Различают абсолютные и инкрементальные энкодеры. На выходе у абсолютных энкодеров формируется цифровой код, указывающий положение вала. Обычно это дорогие и сложные устройства. Инкрементальные энкодеры значительно проще. При повороте на определённый угол на выходе генерируется выходной импульс. Вся остальная работа по подсчету этих импульсов ложится либо на управляющие схемы, либо на управляющую программу. Именно об инкрементальных энкодерах мы и будем говорить ниже.

Среди множества энкодеров наиболее точными и надежными являются оптические. Обычно они представляют собой оптопару - прибор, состоящий из излучателя света (обычно - светодиод) и фотоприемника (фототранзистор, фотодиод и т.п.).

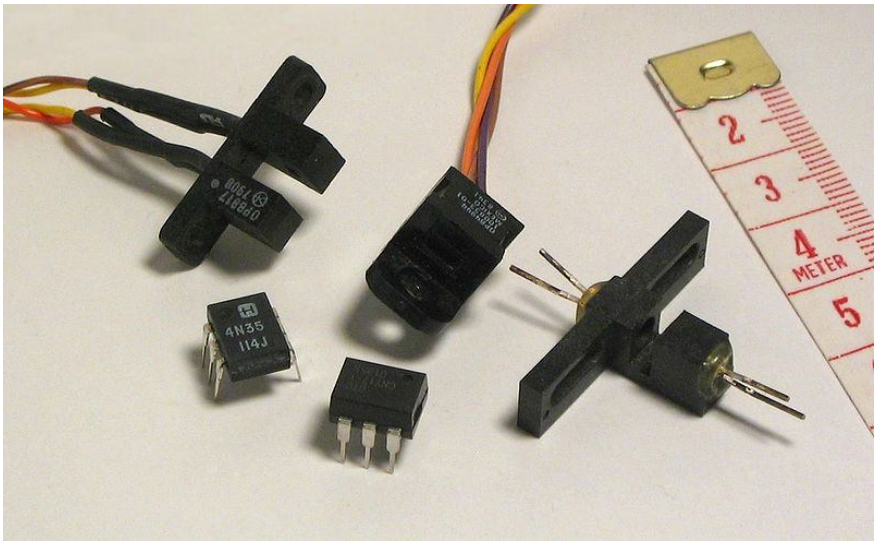

Рис.1. Оптроны. Фото с ru.wikipedia.org

В свою очередь, среди оптопар нас интересуют прежде всего т.н. щелевые датчики, т.е. устройства, у которых приемник и излучатель разделены некоторым промежутком - щелью. Иногда эти датчики называются датчиками просветного типа. Такие устройства позволяют обнаруживать наличие или отсутствие непрозрачного препятствия на пути

луча. Чаще всего в них используется ИК-излучение. Промежуток между излучателем и фотоприемником образует чувствительную область сенсора. При попадании какого-либо объекта в эту область ИК-луч прерывается, и ток через фотоприемник (фотодиод, фототранзистор) резко уменьшается.

Рис.2. Датчики просветного типа

По такому же принципу была устроена «опто-механическая» мышь. Обычно в них использовалась двойная оптопара (ИК-диод и два фототранзистора) и диск с отверстиями или лучевидными прорезями, перекрывающего световой поток по мере вращения. При перемещении мыши диск вращается и с фотодиодов снимается сигнал с частотой, соответствующей скорости перемещения мыши.

Рис.3. Оптическая система мыши. Фото с ru.wikipedia.org

Интересно, что для определения направления вращения диска использовались различные приемы: например, устанавливался второй фотодиод, смещенный на некоторый угол; использовались диски со смещённой системой отверстий/прорезей; ставились 2 оптопары. Все это нужно было для того, чтобы отловить задержку появления сигнала, которая зависела от направления движения диска.

Робот

Выбор подходящего датчика - это непростой вопрос. С одной стороны, щелевые (просветные) датчики - наиболее удобны с конструктивной точки зрений. Их легко крепить и главное, они генерируют достаточно мощный сигнал. Однако главной проблемой при их использовании является необходимость наличия подходящего диска с прорезями.

Другим вариантом является использование старой мыши, их которой извлекаются оптопары и диски. Именно этот вариант и будет рассмотрен далее. У этого варианта есть основной недостаток - малая точность. Дело в том, что мышиный диск имеет порядка 40-

50 прорезей (отверстий). При этом диск маленький. При больших скоростях вращения сигнал, снимаемый с фототранзистора, имеет очень малую амплитуду и зачастую отловить этот сигнал без использования дополнительных схем (например, компараторов) становится сложно.

Тем не менее, будем считать, что у нас имеется именно мышиный датчик. Выпаяв из платы мыши элементы оптопары, разместим их на макетной плате примерно на том же расстоянии, что и в исходной мыши (обычно излучатели - в прозрачном корпусе, а приемники - в непрозрачном). При этом будем полагать, что излучатель - это «обычный» ИК-диод, поэтому для его питания возьмем 5 В, не забывая при этом, что нам потребуется ограничивающий резистор номиналом порядка 1К.

Фотоприемник - это, фактически, фототранзистор. Со всеми вытекающими. Это, в частности, означает, что мы его можем напрямую подключить ко входам нашего «стандартного учебного» контроллера. Напомним, что сконфигурирован контроллер так, что его входы подтянуты к высокому уровню. Это означает, что при срабатывании фототранзистора (луч излучателя проходит сквозь прорезь диска) на соответствующем входе будет «0». Если же луч попадает на непрозрачный участок, то фототранзистор закрыт и на входе контроллера будет «1».

Самой ответственной операцией является установка диска на выходные валы и закрепление оптопар.

На рис.4. изображена такая системы. Обратите внимание на то, что нам «повезло» с редуктором, выходная ось которого вполне пригодна для установки диска энкодера.

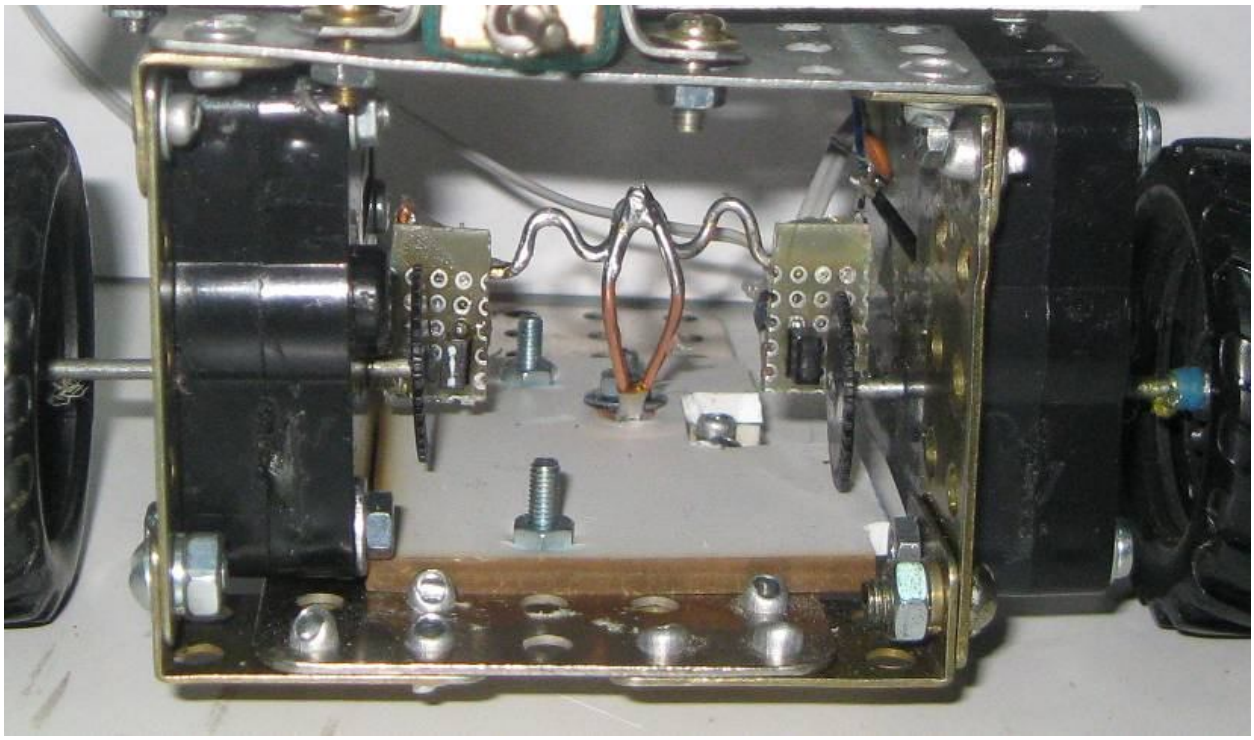

Рис.4. Энкодеры

Общий вид робота, оснащенного энкодерами и управляемого «учебным» контроллером показан на рис.5.

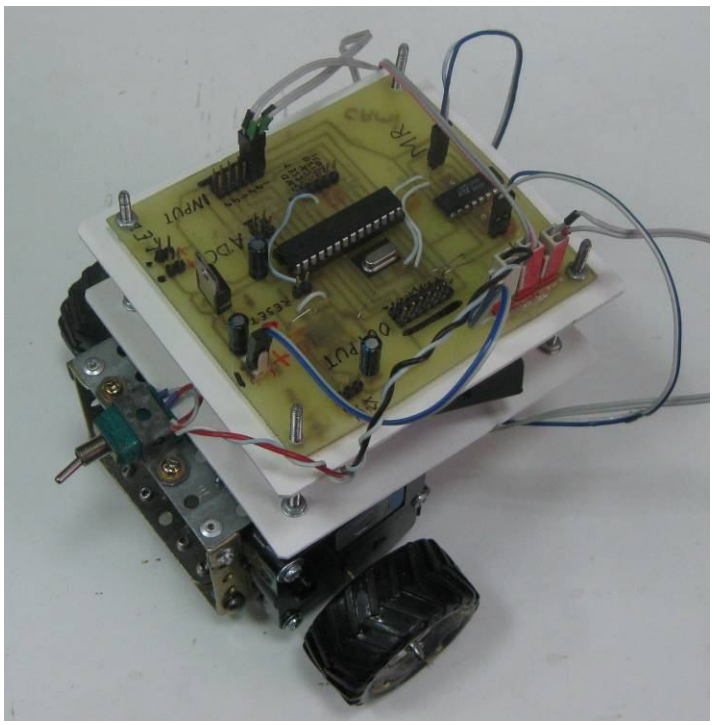

Рис.5. Общий вид

Программная часть

Мы не будем подробно описывать всю программу, демонстрирующую работу с энкодером. Будем считать, что у нас имеются все необходимые ходовые функции, определяющие направление вращения двигателей.

Далее мы определяем порты, на вход которых поступают сигналы от фотодатчиков.

```
// Подключение фотоприемников датчиков
#define enc_L sen_1
#define enc_R sen_2
```

Принцип управления прост. Сначала мы задаем направление движения робота (включаем нужные двигатели), а затем считаем необходимое количество шагов для каждого из них. Это делается функцией ***MakeNSteps****(int NumLeft, int NumRight)*

Параметры *NumLeft и NumRight* как раз определяют это количество шагов.

```
void MakeNSteps(int NumLeft, int NumRight)
// Сделать NumLeft и NumRight шагов и остановить двигатели
{
  int cnt_L, cnt_R; // Счетчики
  char wasL, wasR;  // Флаги датчиков (для ловли момента перехода)

  // Обнуляем
  cnt_L = cnt_R = 0;
  wasL = wasR = 0;

  // Продолжаем до тех пор, пока не досчитаем до конца
  while(cnt_L<NumLeft || cnt_R<NumRight)
  {
    if(enc_L==0)
    {
      if(wasL) // Переход из 1 в 0
      {
        wasL = 0;
        cnt_L++; // Увеличиваем счетчик шагов
```

```
      }
    }
    else wasL = 1;

    if(enc_R==0)
    {
      if(wasR) // Переход из 1 в 0
      {
        wasR = 0;
        cnt_R++; // Увеличиваем счетчик шагов
      }
    }
    else wasR = 1;

    // Если досчитали до конца, то останавливаем соответствующий двигатель
    if(cnt_L>NumLeft) MotorLeftStop();
    if(cnt_R>NumRight) MotorRightStop();
  }
}
```

Функция эта достаточно очевидна. Главное в ней - это определение момента поворота дисков, точнее - момента перехода их открытого в закрытое состояние. Для этого в функции используются флаги - переменные `wasL` и `wasR`. Все остальное - это банальный цикл.

Основная функция демонстрационной программы выглядит просто. Управление осуществляется по интерфейсу RS232 (при помощи какой-нибудь терминальной программы). При нажатии соответствующих клавиш устанавливается необходимое направление вращения колес и дальше вызывается функция ***MakeNSteps***. Она будет подсчитывать шаги, количество которых в программе фиксировано и определено макросами `NSTEP_LEFT NSTEP_RIGHT`.

```
// Основная программа

#define NSTEP_LEFT  100
#define NSTEP_RIGHT 100

void main(void)
{
  char c;

  InitCM8(); //Инициализация контроллера

  robotStop();

  while(1)
  {
    c = getchar(); //Считываем код клавиши
    switch(c)
    {
      case 'w':
        goFwd(); //Вперед
        MakeNSteps(NSTEP_LEFT, NSTEP_RIGHT);
        break;
      case 's':
        goBack(); //Назад
        MakeNSteps(NSTEP_LEFT, NSTEP_RIGHT);
        break;
      case 'a':
        goLeft(); //Разворот влево
        MakeNSteps(NSTEP_LEFT, NSTEP_RIGHT);
        break;
```

```
      case 'd':
        goRight(); //Разворот вправо
        MakeNSteps(NSTEP_LEFT, NSTEP_RIGHT);
        break;
      case ' ':
        robotStop(); //Стоп
        break;
    }
  }
}
```

Построенная на этих принципах тележка (робот) показывает достаточно устойчивые результаты управления. Малая скорость вращения (большой передаточный коэффициент редукторов) позволяет достаточно стабильно снимать сигналы с датчиков. 100 отсчитываемых шагов при 50 прорезях на диске заставляют колеса делать по 2 полных оборота, что можно достаточно просто контролировать (нанеся на колеса метки).

Характерно, что сами по себе двигатели имеют весьма большое рассогласование (мотор-редукторы взяты от какой-то дешевой и некачественной игрушки), а батарея питания просаживается очень быстро (двигатели еще и потребляют неприлично много). Из-за этого зачастую при движении вперед один двигатель уже останавливался (нужное количество оборотов уже сделано), а другой еще какое-то время дорабатывал. В этом смысле приведенная выше функция, конечно, далеко не оптимальна. «Настоящая» управляющая процедура должна уметь синхронизировать работу двигателей постоянно, однако это уже выходи за рамки нашего изложения.

Кроме того, для более качественного управления требуется, чтобы робот осуществлял старт и остановку **постепенно**. Дело в том, что робот останавливается не сразу, колеса еще некоторое время продолжают вращаться, что сказывается на точности управления.

Между прочим, добиться приемлемой точности и качества движения на *длинной траектории* можно достичь, если разбить всю траекторию на *множество участков*. Тогда робот будет перемещаться небольшими отрезками и разница в скоростях вращения колес будет не так заметна. Правда, при этом возникает новая проблема – накопление погрешностей управления.

И еще одно замечание. Существует такая задача, как **одометрия** – измерение пройденного пути. Энкодер позволяет решать ее достаточно качественно. Для измерения пройденного пути (или задания пути, который должен пройти робот) требуется знать два параметра – количество оборотов колеса и диаметр колеса.