Программирование движения гексапода по пересеченной местности

Залатов Владислав Петрович

11 марта 2021 г.

1. Существующие подходы к программированию роботов

1.1. Написание программного кода под AVR микроконтроллеры

Множество робототехнических систем реализуются на AVR контроллерах компании Microchip. В таком случае разработчику необходимо писать прошивку для каждого контроллера, а использование операционных систем в привычном понимании невозможно. Тем не менее, основная линейка AVR-микроконтроллеров отличается низкой стоимостью и приемлемой производительностью, поэтому очень часто выбор падает именно на них. Самыми популярными моделями являются ATmega и ATtiny.

Для программирования под такие микроконтроллеры используются программаторы, например STK500. Сам код разрабатывается на C/C++ или AVR Assembler.

Основа при программирование таких устройств – работа с портами ввода/вывода. Для этого используются различные регистры, так, например, регистр DDRD соответствующего порта указывает, какой вывод микроконтроллера будет использован для ввода данных (т.е. на этот контакт будет поступать внешний сигнал), а какой для вывода (на этом контакте сигнал будет формировать сам микроконтроллер). Установка соответствующего бита в 1 говорит о использовании контакта на вывод, а 0 говорит, что на ввод. При выводе данных в порт передаётся обычный байт, в котором каждый бит отвечает за конкретный контакт, вся логика устройства строится подобным образом. Например, простая программа, которая переводит порт D (настроенный на вывод) в состояние порта B (настроен на вход) представлена на рисунке 1.

```
1 stop: clr R18
2 ser R19
3 out DDRB,R18
4 out DDRD,R19
5 in R20,PINB
6 out PORTD,R20
7 rjmp stop
```

Рис. 1. Программа, копирующая содержимое одного порта в другой (AVR Assembler)

1.2. Подходы при написании кода для робототехнической системы

Вне зависимости от платформы, идеи организации программного кода при разработке гексаподов или аналогичных систем отличаются. При разработке на контроллерах зачастую используются идеи процедурного программирования. Повторяющиеся наборы операторов объединяются в функции, а выполнение операторов происходит последовательно в цикле.

В рамках этой модели стоит помнить, что в процессе выполнения основной программы могут быть сгенерированы внешние прерывания, а результат их обработки может сказываться на выполнении основной части кода (например, возникла ошибка или поменялся какой-либо флаг). В таком случае программа становится чрезвычайно разветвлённой, на каждом этапе выполнения должно проверяться текущее состояние системы, что делает код очень громоздким и зачастую влияет на время выполнения. Тем не менее, такие реализации часто встречаются.

Эту проблему частично решает использование парадигмы событийно-ориентированного программирования. Ход выполнения такой программы управляется событиями, в нашем случае – событиями обратной связи, командами пользователя, а также внутренними программными условиями. Самой удачной реализацией такой парадигмы для роботостроения (а также для создания искусственного интеллекта) является машина конечных автоматов. Конечный автомат – абстрактный объект, который в один момент времени может находится лишь в одном из заранее определённых состояний (их число конечно), а переход из одного состояния в другое происходит из-за внешнего воздействия или по внутренним причинам. В программной реализации это выглядит следующим образом: заранее определён набор состояний, между которыми возможен переход, и набор событий, который и обуславливает переход от одного состояния к другому. Такой подход позволяет

отойти от использования простой последовательности действий и множества повторяющихся проверок к вполне определённой системе, управлять которой довольно просто и поведение которой в большей степени предсказуемо. Для соблюдения требований лабораторной была добавлена невероятно легендарная формула 1.

$$E = mc^2 \tag{1}$$

Содержание