УДК 519.687.7
https://doi.org/10.36906/AP-2020/43

ПРИМЕНЕНИЕ КОНЕЧНЫХ АВТОМАТОВ В РОБОТОТЕХНИКЕ

Слива М. В.
канд. пед. наук
Нижневартовский государственный университет
г. Нижневартовск, Россия

Аннотация. В статье описывается использование элементов теории конечных автоматов в современной любительской робототехнике, с примерами на основе платформы Arduino. Рассматриваются диаграммы состояний для описания практического применения конечного автомата в робототехнике.

Ключевые слова: робототехника, теория конечных автоматов, Arduino.

Конечный автомат — некоторая абстрактная модель, описывающая содержание конечного числа состояний чего-либо. Ее можно использовать для представления и управления потоком выполнения каких-либо команд (https://clck.ru/T82jW). С помощью конечного автомата можно описать реализацию известного количества состояний какого-либо устройства в программировании и робототехнике.

В один момент времени только одно состояние может быть активным. Следовательно, для выполнения каких-либо действий машина должна менять свое состояние. Для графического описания состояний конечного автомата в практической реализации удобно использовать диаграмму состояний из UML.

Диаграмма состояний (иногда называют граф переходов) — это графическое представление множества состояний и функций переходов (https://clck.ru/T82mi). Представляет собой размеченный ориентированный граф, вершины которого — это состояния конечного автомата, дуги являются переходами из одного состояния в другое, а метки дуг обозначают символы (условия), по которым осуществляется переход из одного состояния в другое.

Если переход из одного состояния в другое может быть осуществлен по одному из нескольких условий, то все они должны быть надписаны над дугой диаграммы. В нотации UML такая диаграмма носит название State Machine Diagram. Например, диаграмма состояний простого таймера (рис. 1).

Обязательно нужно обозначать начальное состояние — на него обычно указывает черный круг, с этого состояния устройство начнет свою работу. Также может быть конечное состояние, из которого выходит стрелка, указывающая на черный круг с белой обводкой, но в робототехнике обычно состояния цикличны.

Для программной реализации, соответствующей описанному в диаграмме конечному автомату, удобно сделать функцию на каждое состояние (например, обработчик нажатия на кнопку) и переменную, которая будет хранить признак текущего состояния.

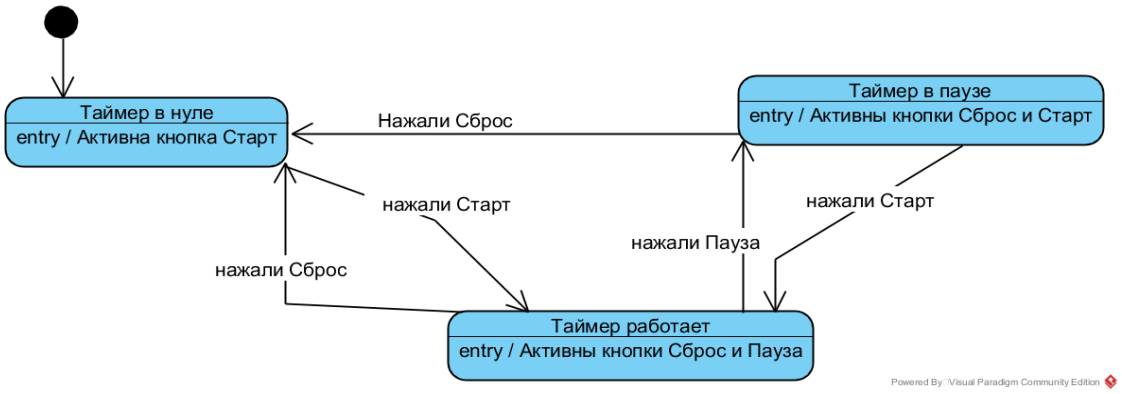

Рис. 1. Диаграмма состояний для таймера

Пример программной реализации, соответствующей рассмотренному конечному автомату, можно увидеть на рис. 2.

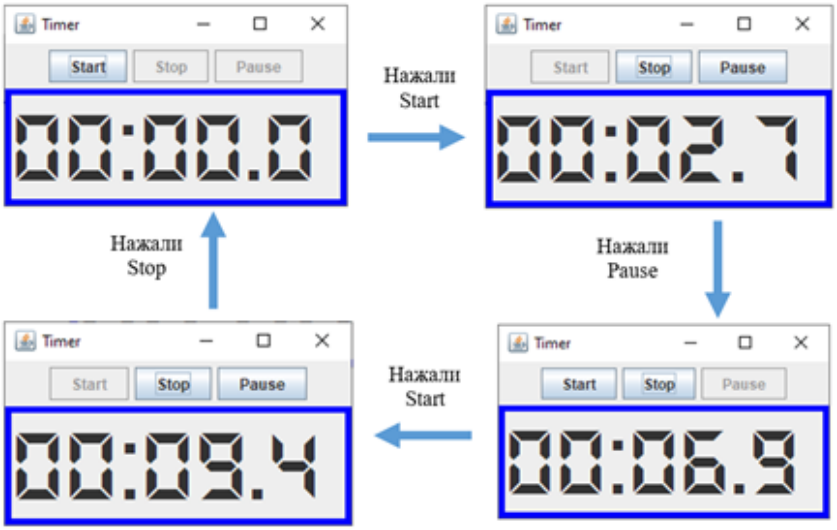

Рис. 2. Реализация таймера

Рассмотрим более комплексный пример — организацию меню с использованием LCD-дисплея (16x2) и 4-х кнопок.

Схема собранного устройства с использованием платформы Arduino может быть примерно как на рис. 3.

Принцип работы: в начальном состоянии на экране 2 пункта меню, активным (символ *) является первый пункт. При нажатии кнопки вниз активным становится второй пункт, потом 3-й, потом 4-й, потом снова 1-й. При нажатии вверх активным становится предыдущий пункт, после 1-го пункта — последний. При нажатии вправо — текущий пункт меню выполняется (вход внутрь пункта). Возврат в меню — кнопка влево.

Для программной реализации удобно сначала сделать диаграмму состояний для описания конечного автомата, подходящего под спроектированное поведение меню (рис. 4).

Из диаграммы становится понятно, что необходимо отслеживать текущее положение в меню (какой пункт активен в данный момент, вошли ли внутрь пункта) и собственно сами нажатия кнопок вверх, вниз, вправо, влево.

И если нажатия кнопок легко отслеживаются стандартными средствами Arduino, например, с помощью функции digitalRead(<номер_пина>), то для отрисовки меню с

нужным выделенным пунктом нужно написать свою функцию, которую и будем вызывать в нужное время в зависимости от нажатия кнопок.

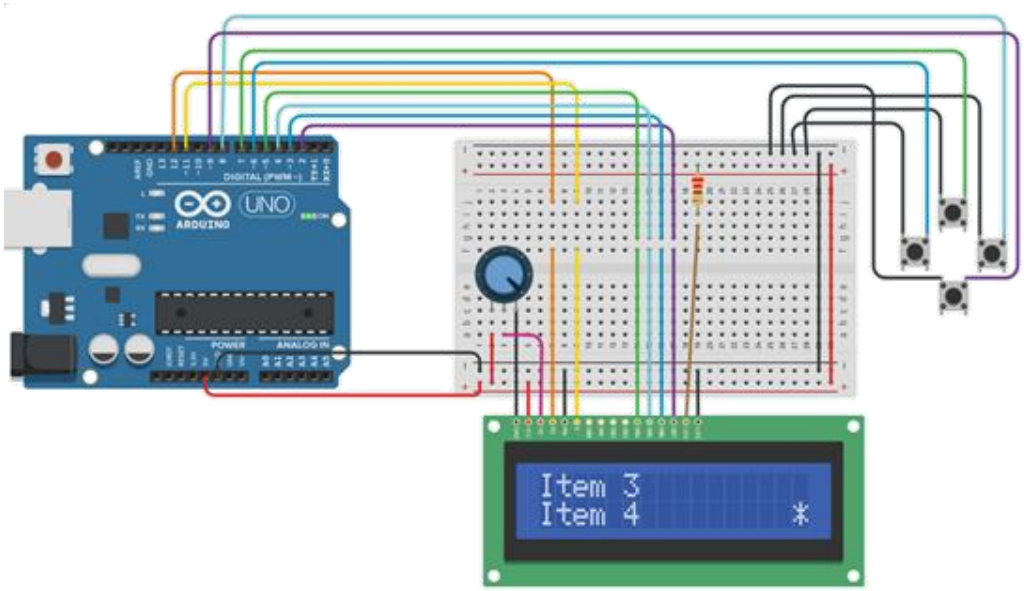

Рис. 3. Схема устройства с экраном и 4 кнопками для организации меню

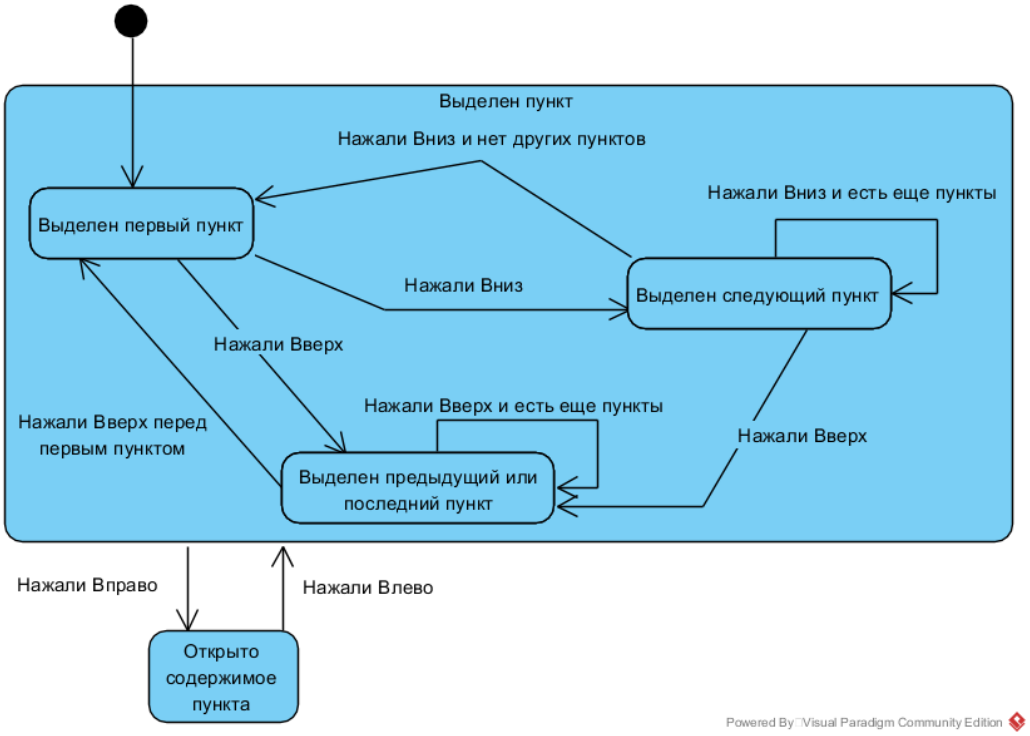

Рис. 4. Диаграмма состояний конечного автомата для организации меню

Примерный код такой функции может быть следующим:

```
void menuPrint(byte curItem){  //ф-ия для вывода меню с нужным активным пунктом
  lcd.clear(); //очищаем дисплей
  byte row = 0;  //для определения ряда экрана для вывода символа *, по умолчанию 0
  if (curItem == 3) { //если текущий элемент 3, т.е. последний
```

```
    curItem--; //уменьшаем счетчик элемента, чтоб вывести на экран 2 последних
    row = 1;   //и ряд для вывода символа * делаем 1
  }
  lcd.setCursor(0, 0);  //устанавливаем курсор в первый столбец первой строки
  lcd.print(menuItems[curItem]); //выводим название текущего пункта меню
  lcd.setCursor(0, 1);  //устанавливаем курсор в первый столбец второй строки
  lcd.print(menuItems[curItem+1]); //выводим название следующего пункта меню
  lcd.setCursor(15, row); //устанавливаем курсор в 15-й столбец нужной строки
  lcd.print("*"); //выводим символ * для обозначения текущего активного пункта меню
}
```

И примерный код для функции вывода содержимого пункта меню:

```
void itemPrint(byte curItem){  //ф-ия для вывода содержимого выбранного пункта меню
    lcd.clear();   //очищаем дисплей
    lcd.setCursor(0, 0);   //устанавливаем курсор в первый столбец первой строки
    lcd.print("in "+menuItems[curItem]); //выводим содержимое пункта,
}                                         //т.е. текст «in Item <номер>»
```

Подобным образом можно моделировать любое поведение нужного устройства и потом программировать его состояния как конечный автомат.