

Минобрнауки России

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт искусственного интеллекта   
Базовая кафедра №252 – информационной безопасности

**курсовая работа по предмету**

**«МаТЕМАТИЧЕСКАЯ ЛОГИКА**

**и теория алгоритмов»**

**Тема работы:** Arduino. Реализация мультизадачности (эмуляция ее) при обработке событий.

|  |  |  |
| --- | --- | --- |
| **Студент группы ККСО-01-20** |  | *Семин В.В.* |
| **Руководитель** |  | *Старший преподаватель*  *Кузнецов Владимир*  *Сергеевич* |

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc153151566)

[1 ПОДХОДЫ К ЭМУЛЯЦИИ МНОГОЗАДАЧНОСТИ В ARDUINO 6](#_Toc153151567)

[1.1 ФУНКЦИЯ YIELD() 6](#_Toc153151568)

[1.2 МНОГОЗАДАЧНОСТЬ С MILLIS() 6](#_Toc153151569)

[1.3 ОПЕРАЦИОННАЯ СИСТЕМА 7](#_Toc153151570)

[1.4 ПРЕРЫВАНИЯ ПО АППАРАТНОМУ ТАЙМЕРУ 8](#_Toc153151571)

[1.5 МНОГОЗАДАЧНОСТЬ НА КОНЕЧНЫХ АВТОМАТАХ 10](#_Toc153151572)

[2 ТЕОРЕТИЧЕСКОЕ ОПИСАНИЕ МНОГОЗАДАЧНОГО ВСТРОЕННОГО ПРИЛОЖЕНИЯ НА ДКА 12](#_Toc153151573)

[2.1 ОБЩАЯ СТРУКТУРА 12](#_Toc153151574)

[2.2 ВИДЫ КОНЕЧНЫХ АВТОМАТОВ 12](#_Toc153151575)

[2.3 АВТОМАТ КАК ЗАДАЧА В ВСТРОЕННОМ ПРИЛОЖЕНИИ 13](#_Toc153151576)

[2.4 МЕНЕДЖЕР И ГЕНЕРАТОР СООБЩЕНИЙ 15](#_Toc153151577)

[2.5 ГЕНЕРАТОР СООБЩЕНИЙ КАК АВТОМАТ МИЛИ 16](#_Toc153151578)

[2.6 МЕНЕДЖЕР СООБЩЕНИЙ КАК АВТОМАТ МУРА 17](#_Toc153151579)

[2.7 АВТОМАТ ОБРАБОТКИ ВНЕШНИХ СОБЫТИЙ 18](#_Toc153151580)

[2.8 АВТОМАТ ВЗАИМОДЕЙСТВИЯ С ВНЕШНИМИ УСТРОЙСТВАМИ 19](#_Toc153151581)

[3 ПРИМЕР РЕАЛИЗАЦИИ ПРИЛОЖЕНИЯ «УМНЫЙ ДОМ» 21](#_Toc153151582)

[3.1 АВТОМАТ КЛИМАТА 21](#_Toc153151583)

[3.2 АВТОМАТ УПРАВЛЕНИЯ УМНЫМ ПЫЛЕСОСОМ 22](#_Toc153151584)

[3.3 АВТОМАТ БЕЗОПАСНОСТИ 23](#_Toc153151585)

[3.4 ВНЕШНИЕ СОБЫТИЯ 25](#_Toc153151586)

[3.5 ВЗАИМОДЕЙСТВИЕ С ВНЕШНИМИ УСТРОЙСТВАМИ 25](#_Toc153151587)

[3.6 ГЕНЕРАТОР СООБЩЕНИЙ 25](#_Toc153151588)

[ЗАКЛЮЧЕНИЕ 28](#_Toc153151589)

[ЛИТЕРАТУРА 29](#_Toc153151590)

# ВВЕДЕНИЕ

Разработка многозадачных приложений для Arduino становится всё более актуальной задачей в силу растущей сложности и функциональности устройств, использующих эту микроконтроллерную платформу. Arduino, с его простотой и доступностью, стал популярным выбором для широкого круга разработчиков, создающих умные устройства и системы. Однако, с увеличением функциональных требований, необходимость эффективного управления множеством задач и периферийных устройств становится неотъемлемой частью процесса разработки.

Многозадачность позволяет Arduino одновременно обрабатывать несколько задач, таких как сбор и обработка данных от датчиков, управление различными устройствами вывода, взаимодействие с внешними событиями и выполнение других задач. Это особенно важно в контексте "Интернета вещей" (IoT), где устройства должны эффективно взаимодействовать с окружающей средой и обеспечивать отзывчивость на различные события. Разработка многозадачных приложений для Arduino предоставляет разработчикам возможность создавать более сложные и гибкие системы, способные эффективно реагировать на динамически меняющиеся условия и требования.

Актуальность исследования. Исследование в области многозадачных систем крайне актуально в современном информационном обществе, где устройства становятся все более взаимосвязанными и требовательными к эффективному управлению ресурсами. Эта область имеет прямое воздействие на различные сферы, включая Интернет вещей, автоматизацию и робототехнику, предоставляя возможность оптимизировать использование ресурсов, обеспечивать отзывчивость систем и повышать их гибкость. Результаты исследований в этой области влияют на технологический прогресс и повседневную жизнь, а также способствуют созданию более эффективных и интеллектуальных систем для различных сфер деятельности.

Объест исследования = многозадачные приложения на Arduino.

Предмет исследования – построение многозадачных приложений для Arduino на основе конечных автоматов.

Цель исследования: описать математическую модель многозадачного приложения на Arduino на основе теории конечных автоматов, привести пример построения приложения «Умный дом» в рамках описанной модели.

Задачи исследования:

1. Рассмотреть различные варианты эмуляции многозадачности в приложениях для Arduino.
2. Построить математическую модель приложения для Arduino на основе теории автоматов
3. Привести пример построения приложения «Умный дом» в рамках описанной модели.

Гипотеза исследования. Автоматный метод построения многозадачных приложений для Arduino является наилучшим в большинстве задач.

Методы исследования: изучение отечественной и зарубежной литературы по теме, анализ, абстрагирование, сравнение, моделирование.

# 1 ПОДХОДЫ К ЭМУЛЯЦИИ МНОГОЗАДАЧНОСТИ В ARDUINO

## 1.1 ФУНКЦИЯ YIELD()

Если объявить функцию yield(), то расположенный внутри неё код будет выполняться во время работы любой задержки delay() в программе. Это позволяет быстро реализовать пару параллельно выполняющихся задач. Рассмотрим простой пример: стандартный мигающий светодиод, но с опросом кнопки (листинг 1.1) [1].



Листинг 1.1 – Пример многозадачности с использованием функции yield()

Такой подход позволяет достаточно просто и быстро реализовать многозадачность, однако в данном случае она будет ограничена только двумя одновременно выполняющимися задачами.

## 1.2 МНОГОЗАДАЧНОСТЬ С MILLIS()

При помощи функций времени millis() или micros() можно организовать программный таймер, по которому и выполнять нужные действия. Схема такая:

1. Заводим переменную для таймера типа unsigned long (uint32\_t) – именно этот тип возвращает millis().
2. Ищем разницу между текущим временем работы программы и переменной таймера.
3. Если разница больше необходимого периода – выполняем нужный код и сбрасываем таймер.

Например, если мы хотим выполнять одно действие два раза в секунду, второе – три, и третье – 10. Нам понадобится 3 переменные таймера и 3 конструкции с условием (листинг 1.2) [2].



Листинг 1.2 – Пример многозадачности с использованием функции millis()

Такой подход также достаточно прост и быстр в реализации, однако плох тем, что мы задаём достаточно грубо задаём временные рамки выполнения задач, что в некоторых случаях может привести к простоям процессора.

## 1.3 ОПЕРАЦИОННАЯ СИСТЕМА

Операционные системы (ОС), которые работают внутри встраиваемой электроники (embedded devices), в большинстве случаев являются операционными системами реального времени (RTOS – Real-Time Operating System). В этих системах критически важно, чтобы задачи исполнялись в строго отведенное для них время. Задачи реального времени в этих системах подразумевают под собой то, что время ответа на определенное событие всегда занимает строго фиксированное время. Операционные системы реального времени (RTOS) спроектированы таким образом, чтобы обеспечивать запуск приложений с максимально точным таймингом и гарантировать им высокую надежность работы. Также ОС реального времени помогают реализовать многозадачность в микроконтроллерах (микропроцессорах) с одним ядром [3].

Данный подход позволяет сделать многозадачность максимально эффективной, и в сложных случаях без него не обойтись. Однако использование операционной системы требует большого количества ресурсов, которые в случае применения микроконтроллеров достаточно ограничены. Кроме этого, создание подобной операционной потребует от программиста достаточной осведомлённости о архитектуре выбранного микроконтроллера и высоких навыков в низкоуровневом программировании, а использование готовых решений может потребовать долгого разбирательства в чужом коде.

## 1.4 ПРЕРЫВАНИЯ ПО АППАРАТНОМУ ТАЙМЕРУ

Как в реальной жизни, в микроконтроллерах таймер — это нечто, что вы устанавливаете для запуска предупреждения в определенный момент в будущем. Когда этот момент наступает, предупреждение прерывает микропроцессор, напоминая ему что-то сделать, например, выполнить определенный кусок кода.

Таймеры работают, увеличивая переменную-счетчик, известную как регистр счетчика. Регистр счетчика может считать до определенного значения в зависимости от его размера (обычно 8 или 16 бит). Таймер увеличивает этот счетчик шаг за шагом до достижения его максимального значения, после чего счетчик переполняется и сбрасывается обратно на ноль. Таймер обычно устанавливает флаг, чтобы предупредить вас о том, что произошло переполнение. Этот флаг можно проверить вручную, или вы можете настроить таймер на вызов прерывания сразу после установки флага. Как и в случае с любым другим прерыванием, вы можете указать обработчик прерывания (ISR), чтобы выполнить свой собственный код, когда происходит переполнение таймера. ISR автоматически сбросит флаг переполнения, поэтому использование прерываний обычно является лучшим вариантом для простоты и скорости.

Для увеличения значения счетчика через регулярные интервалы таймеру необходим источник тактового сигнала. Этот источник предоставляет постоянный сигнал. Каждый раз, когда таймер обнаруживает этот сигнал, он увеличивает свой счетчик на единицу.

Вы можете подавать внешний тактовый сигнал для использования с таймерами, но обычно используется внутренний тактовый сигнал микросхемы [4].

Кодом, который будет выступать в качестве таймерного прерывания, может стать простая функция переключения задач или некоторый более сложный планировщик задач.

Положительной стороной такого подхода является то, что основной код программы не будет нагружен лишними конструкциями, так как код таймерного прерывания существует и исполняется отдельно. По этой же причине работа данного метода будет более чёткой и стабильной, чем в случае использования программного таймера, описанного в пункте 1.2.

Отрицательной стороной является то, что нужно хранить большую таблицу временных интервалов вызова прерываний. Можно, конечно, строить таблицу динамически: каждый раз, когда приходит срок очередного события, программа обработки прерывания спрашивает у обслуживаемого устройства, каким будет следующее событие и когда оно должно произойти. В этом случае достаточно будет таблицы с одной строчкой на каждый канал.

Однако при этом подходе процедура обслуживания события явно устанавливает величину интервала ожидания следующего события по тому же каналу. Это значит, что мы не можем запустить отсчет следующего интервала до завершения текущей процедуры обслуживания события, поскольку еще не знаем, какой из интервалов минимален. Из этого следует, что очередной интервал удлиняется на время выполнения процедуры, а это снижает точность отсчета времени, особенно в случаях, когда обслуживание события длится долго.

Также может возникнуть ситуация, когда до завершения обработки прерывания возникает еще одно прерывание, если оно не запрещено аппаратно. Поскольку у процедур обработки прерывания обычно проблемы с реентерабельностью, результат такой вложенности непредсказуем. Можно запретить повторные прерывания на время их обработки, но в этом случае есть риск их потерять, если они следуют достаточно часто [5].

## 1.5 МНОГОЗАДАЧНОСТЬ НА КОНЕЧНЫХ АВТОМАТАХ

Детерминированный конечный автомат состоит из следующих компонентов.

1. Конечное множество состояний, обозначаемое обычно как .
2. Конечное множество входных символов, обозначаемое обычно как .
3. Функция переходов, аргументами которой являются текущее состояние и входной символ, а значением — новое состояние. Функция переходов обычно обозначается как .
4. Начальное состояние, одно из состояний в .
5. Множество заключительных, или допускающих, состояний . Множество является подмножеством .

То есть ДКА – это пятёрка где — имя ДКА, — множество состояний, — множество входных символов, — функция переходов, — начальное состояние и — множество допускающих состояний [6 с. 62].

Любую задачу, которую будет решать встроенное приложение можно описать языком теории конечных автоматов. После этого можно будет создать некоторый цикл, который будет периодически заставлять полученные автоматы переходить в новое состояние. Также нужно будет создать систему сообщений, которая будет передавать последовательность выходных символов одного автомата на вход другому, позволяя задачам работать синхронно.

Главным достоинством автоматного подхода к созданию встроенного приложения является то, что подобную программу легко модифицировать и документировать. Ещё одним преимуществом является повторное использование кода. Фактически, программа в данном случае состоит из компонентов, являющихся в высокой степени автономными «сущностями». Компонент имеет ограниченное количество связей с остальной программой, его можно разрабатывать и тестировать отдельно, а применять многократно, и именно это свойство подобных систем делает разработку быстрой и удобной [7].

Негативной стороной является то, что если один из автоматов медленно функционирует, или вовсе впадает в бесконечный цикл или бесконечное ожидание, то это отрицательно сказывается на производительности всего приложения. Однако, эта проблема имеет решение, которое будет рассмотрено далее.

Таким образом, автоматный подход к созданию встроенных приложений является наиболее оптимальным для большинства задач, так как имеет неоспоримые плюсы для программиста, и его минусы достаточно легко исправимы. Поэтому теоретическому описанию данного подхода и будет посвящена следующая глава.

# 2 ТЕОРЕТИЧЕСКОЕ ОПИСАНИЕ МНОГОЗАДАЧНОГО ВСТРОЕННОГО ПРИЛОЖЕНИЯ НА ДКА

## 2.1 ОБЩАЯ СТРУКТУРА

Каждый конечный автомат описан своим программным модулем и имеет как минимум две внешние функции – функция инициализации и функция, отвечающая за работу автомата.

Главная особенность подхода в том, что функция работы не должна выполнять продолжительных во времени действий, связанных с ожиданием какого-либо флага или с истечением временного интервала.

Задачей главного цикла программы является поочередный вызов функций работы всех автоматов, составляющих программу.

Многозадачность системы обеспечивается тем, что в каждой итерации главного цикла поочередно вызываются функции работы каждого автомата – каждому автомату выделяется время для выполнения какого-либо элементарного действия (или, что тоже может быть, просто для передачи управления далее по списку). То есть если какой-либо из автоматов «зависает», то «зависает» вся система. Именно для предотвращения подобной ситуации вводится явный запрет на действия в автоматах, которые занимают продолжительное или неопределенное время [7].

## 2.2 ВИДЫ КОНЕЧНЫХ АВТОМАТОВ

Автомат Мура был впервые описан Эдвардом Ф. Муром в [8].

Автомат Мура может быть определён как кортеж из 6 элементов, включающий:

* множество внутренних состояний (внутренний алфавит);
* начальное состояние ;
* множество входных сигналов (входной алфавит);
* множество выходных сигналов (выходной алфавит);
* функция переходов ;
* функция вывода .

Автомат Мили был впервые описан Джорджем Мили в [9].

Автомат Мили – совокупность , где

* — конечное непустое множество состояний автомата;
* — конечное непустое множество входных символов;
* — конечное непустое множество выходных символов;
* — функция переходов, отображающая пары состояние/входной символ на соответствующее следующее состояние;
* — функция выходов, отображающая пары состояние/входной символ на соответствующий выходной символ;
* — начальное состояние.

Основное отличие автомата Мили состоит в том, что в нём выходная последовательность зависит от состояния автомата и входных сигналов, тогда как в автомате Мура она зависит лишь от текущего состояния автомата.

Другими словами, в автомате Мура выходная последовательность определяется самими состояниями, а в автомате Мили — переходами между ними.

Также существуют смешанные автоматы, в которых выходные состояния могут определяться как самими состояниями, так и переходами между ними.

## 2.3 АВТОМАТ КАК ЗАДАЧА В ВСТРОЕННОМ ПРИЛОЖЕНИИ

В данном случае под входом понимается сообщение, срабатывание таймера, результат выражения, которое имеет логическое значение (в том числе возвращаемое функцией логическое значение). Доступ к аппаратным ресурсам микроконтроллера (регистрам периферийных устройств, портам ввода–вывода и т. п.) выполняется также путем вызова функций.

Под выходом автомата понимается отправка сообщения; запуск, останов или сброс таймера; вызов функции.

При этом выделяется особая переменная состояния – переменная, которая определяет текущее состояние автомата. Эта переменная должна быть доступна только своему автомату, ее изменение из других автоматов недопустимо.

Автомат может осуществлять действия (action) и деятельности (activity) автомата. И деятельность, и действие автомата относятся к его выходной активности.

При этом действием называется выходное воздействие, выполняемое однократно при входе в состояние или на переходе, а деятельностью – выходное воздействие, выполняющееся непрерывно в течение всего времени нахождения автомата в определенном состоянии. Также возможна реализация действий, выполняющихся на выходе из состояния.

Автомат Мура и автомат смешанного типа могут выполнять как действия, так и деятельности, а автомат Мили может выполнять только действия.

Автомат в минимальном виде представлен на Рисунке 2.1.

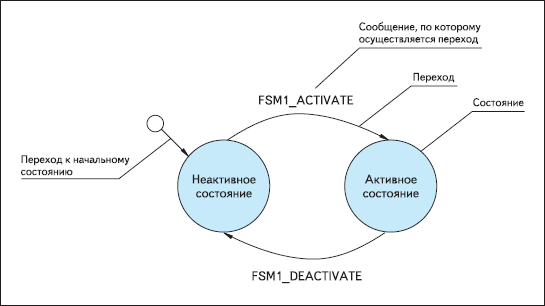


Рисунок 2.1 — Автомат в минимальном виде.

Данный автомат выполняет действия на переходах из одного состояния в другое (именно поэтому это автомат Мили), однако его достаточно легко преобразовать в автомат Мура или в смешанный автомат. Представленный на рисунке автомат не делает ничего полезного, это просто шаблон, на основе которого можно строить автоматы с более сложным поведением [7].

## 2.4 МЕНЕДЖЕР И ГЕНЕРАТОР СООБЩЕНИЙ

Для отправки сообщения другому автомату действующий автомат должен решить следующие вопросы: содержание, канал передачи, получатель, протокол передачи.

Во встроенном приложении мы не можем позволить доверить каждому автомату решать данные вопросы самостоятельно, так как в общем случае время их решения не является константным, а значит операции, связанные с этим вопросом, могут сильно замедлить работу приложения.

Следовательно, формирование и пересылка сообщений должна быть централизована. Для этого предлагается ввести в рассматриваемую модель встроенного приложения новые сущности: генератор и менеджер сообщений.

Генератор сообщений будет работать по следующему алгоритму. Он принимает на вход символ из множества выходных последовательностей одного из автоматов. С помощью этого символа по некоторому заданному условиями прикладной задачи набору правил он определяет содержание и получателя сообщения. Далее он формирует сообщение по заданному программистом протоколу и помещает его в очередь сообщений.

Менеджер сообщений занимается распределением сообщений в очереди по получателям. Алгоритм работы менеджера сообщений следующий. Во время инициализации работы приложения автоматы, которые будут принимать сообщения, должны быть зарегистрированы в менеджере сообщений. Сообщения, созданные генератором, помещаются в очередь сообщений. Менеджер выбирает по некому алгоритму, какое сообщение будет послано следующим. Далее он определяет получателя и отправляет ему сообщение.

Стоит учесть, что в главном цикле программы должно быть также выделено процессорное время для менеджера и генератора сообщений.

Для единообразия рассматриваемой модели встроенного приложения генератор и менеджер сообщений можно описать как ДКА.

## 2.5 ГЕНЕРАТОР СООБЩЕНИЙ КАК АВТОМАТ МИЛИ

1. Множеством входных символов данного автомата является объединения выходных множеств всех автоматов и сигнал о окончании обработки сообщения.
2. Выходом являются сообщения.
3. Состояний у данного автомата два: «Ожидание выходного символа автомата-процесса» и «Формирование сообщения».
4. Начальное состояние «Ожидание выходного символа автомата-процесса».
5. Функция переходов:
   1. В состоянии «Ожидание выходного символа автомата-процесса»:

* Если выход автомата-процесса требует формирования сообщения, автомат переходит в состояние «Формирование сообщения».
* Если не требует, автомат остаётся в состоянии «Ожидание выходного символа автомата-процесса».
  1. После того, как сообщение сформировано автомат из состояния «Формирование сообщения» возвращается в состояние «Ожидание выходного символа автомата-процесса».

1. Функция выходов: при состоянии «Формирование сообщения» и входном символе, требующем обработки сообщения, генерируется сообщение на основе этого символа.

Граф ДКА, реализующего генератор сообщений изображён на Рисунке 2.2.

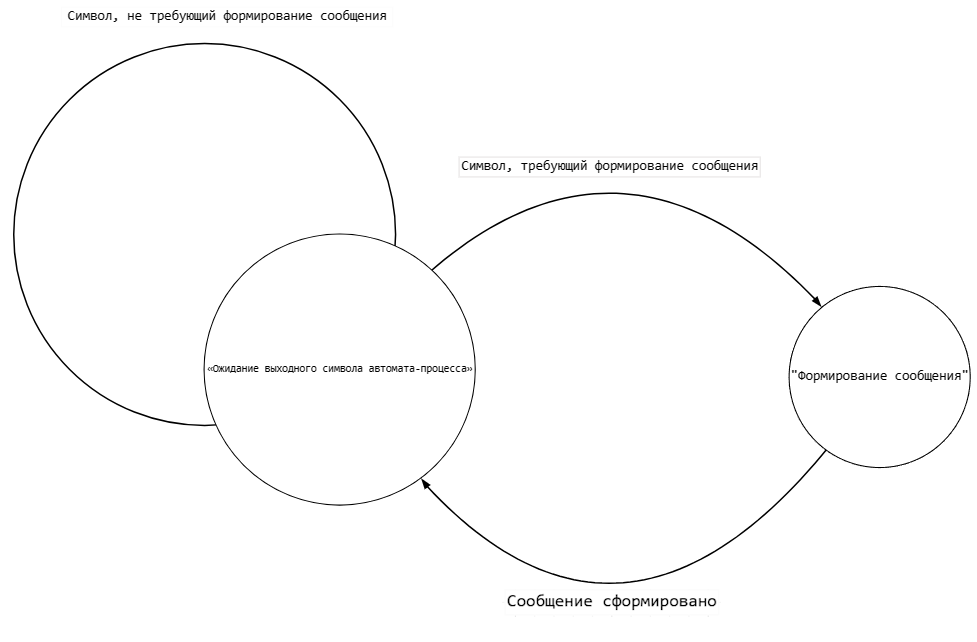


Рисунок 2.2 – ДКА, реализующий генератор сообщений.

## 2.6 МЕНЕДЖЕР СООБЩЕНИЙ КАК АВТОМАТ МУРА

1. Входом является информация о наличии или отсутствии сообщений в очереди.
2. Состояний снова два: «Простой» и «Обработка сообщений».
3. Начальное состояние – «Простой».
4. Выходное множество – деятельность по пересылке сообщения.
5. Функция перехода:
   1. В состоянии «Простой»

* Если на вход поступает информация о наличии сообщений в очереди, перейти в состояние «Обработка сообщений».
* Если на вход поступает информация о отсутствии сообщений, остаться в состоянии «Простой»
  1. Когда на вход поступает информация о отсутствии сообщений в очереди, автомат возвращается из состояния «Обработка сообщений» в состояние «Простой».

1. Функция выходов: В состоянии «Обработка сообщений» автомат выполняет деятельность отправки сообщений.

Граф ДКА, реализующего менеджер сообщений изображён на Рисунке 2.3.

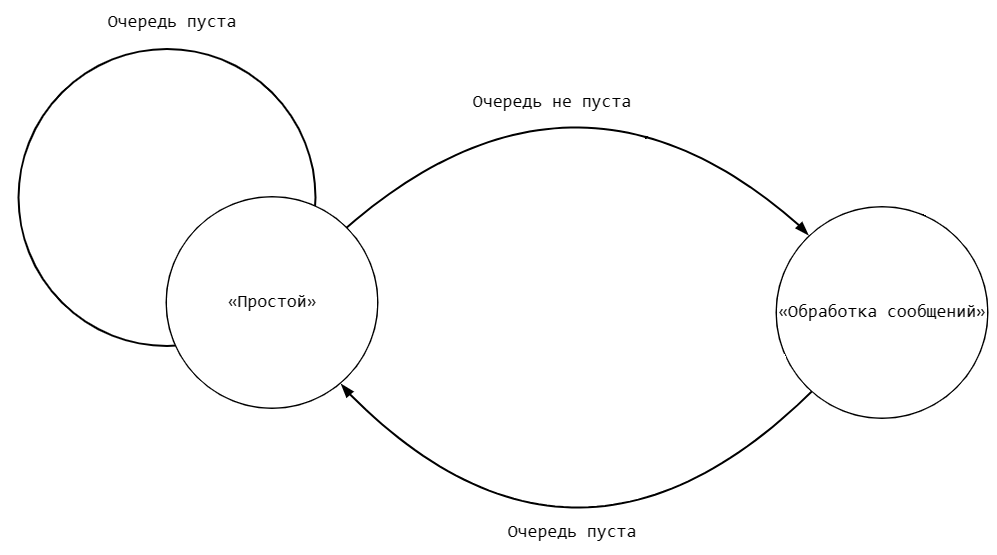


Рисунок 2.3 – ДКА, реализующий менеджер сообщений.

## 2.7 АВТОМАТ ОБРАБОТКИ ВНЕШНИХ СОБЫТИЙ

Добавление в рассматриваемую модель генератора и менеджера сообщений позволяет эффективно обрабатывать внешние события, такие как сигналы датчиков, прерывания таймера, ввод пользователя и тому подобное.

Для этого добавим в модель автомат обработки внешних событий, вызываемый прерыванием от внешнего устройства и генерирующим сигналы внешних событий, которые будут обрабатываться генератором сообщений по тому же алгоритму, что и выход автоматов-процессов.

Автомат обработки внешних событий как автомат Мили имеет следующее описание:

1. Входным множеством является прерывания от внешних устройств и сигнал об окончании обработки прерывания.
2. Состояний два – «Ожидание прерывания» и «Обработка прерывания».
3. Начальное состояние – «Ожидание прерывания».
4. Выходное множество – сигналы о внешних событиях, которые в дальнейшем будут обрабатываться генератором сообщений.
5. Функция переходов: при прерывании от внешнего устройства автомат переходит из состояния «Ожидание прерывания» в состояние «Обработка прерывания». По окончании обработки автомат возвращается в состояние «Ожидание прерывания».
6. Функция выходов: при состоянии «Обработка прерывания» и прерывании от внешнего устройства автомат выдаёт сигнал о внешнем событии .

Граф ДКА, реализующего обработку внешних событий изображён на Рисунке 2.4.

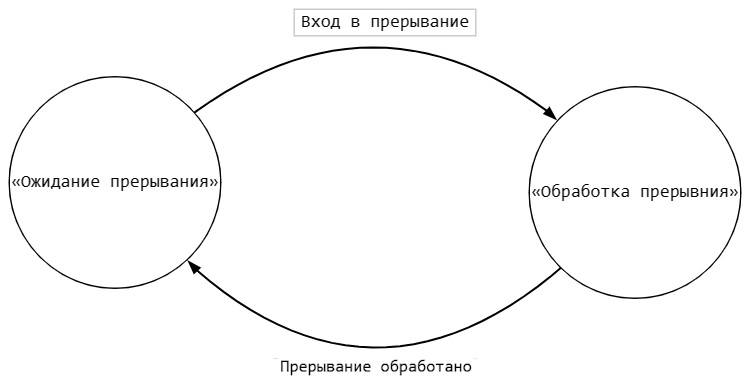


Рисунок 2.4 – ДКА, реализующий обработку внешних событий.

## 2.8 АВТОМАТ ВЗАИМОДЕЙСТВИЯ С ВНЕШНИМИ УСТРОЙСТВАМИ

Кроме задачи обработки внешних событий нужно решить и обратную задачу, когда автомату-процессу нужно подать сигнал внешнему устройству. Для этого введём в модель автомат взаимодействия с внешними устройствами.

Автомат взаимодействия с внешними устройствами как автомат Мили:

1. Вход: сообщение о необходимости обратиться к устройству и сигнал об окончании обработки запроса.
2. Состояния: «Холостое» и «Обработка запроса».
3. Начальное состояние: «Холостое».
4. Выход: обращение к устройству .
5. Функция перехода: при поступлении сообщения о необходимости обратиться к устройству автомат переходит из состояния «Холостое» в состояние «Обработка запроса». После окончания обработки он возвращается в «Холостое состояние»
6. Функция выходов: в состоянии «Обработка запроса» при наличии сообщения о необходимости обратиться к устройству автомат выполняет действие обращения к устройству .

Граф ДКА, реализующего взаимодействие с внешними устройствами изображён на Рисунке 2.5.



Рисунок 2.4 – ДКА, реализующий взаимодействие с внешними устройствами.

# 3 ПРИМЕР РЕАЛИЗАЦИИ ПРИЛОЖЕНИЯ «УМНЫЙ ДОМ»

Данная глава будет посвящена прикладному применению вышеописанной модели на примере приложения «Умный дом». Приложение будет иметь три процесса: контроль климата, контроль умного пылесоса и безопасность.

## 3.1 АВТОМАТ КЛИМАТА

1. Вход: сообщение о низкой температуре, сообщение о высокой температуре, сигнал «Температура понижена» и сигнал «Температура повышена», сообщение «Изменить стандартную температуру», сигнал «Стандартная температура изменена»
2. Состояния: «Холостое», «Температура понижена», «Температура повышена», «Изменить стандартную температуру».
3. Начальное состояние: «Холостое».
4. Выход: сигнал «Понизить температуру», сигнал «Повысить температуру», сигнал о изменении стандартной температуры
5. Функция переходов:
   1. Из состояния «Холостое»:

* При поступлении сообщения о низкой температуре совершается переход в состояние «Температура понижена».
* При поступлении сообщения о высокой температуре совершается переход в состояние «Температура повышена».
* При поступлении от пользователя запроса на изменение стандартной температуры автомат переходит в состояние «Изменить стандартную температуру»

1. Из состояния «Температура понижена»: после повышения температуры автомат возвращается в состояние «Холостое».
2. Из состояния «Температура повышена»: после понижения температуры автомат возвращается в состояние «Холостое».
3. Из состояния «Изменить стандартную температуру» после её изменения автомат возвращается в состояние «Холостое».
4. Функция выходов: в состоянии «Температура понижена» автомат выводит сигнал «Повысить температуру», в состоянии «Температура повышена» автомат выводит сигнал «Понизить температуру», в состоянии «Изменить стандартную температуру» автомат посылает электронному термометру сигнал о изменении стандартной температуры.

Граф автомата климата изображён на Рисунке 3.1.

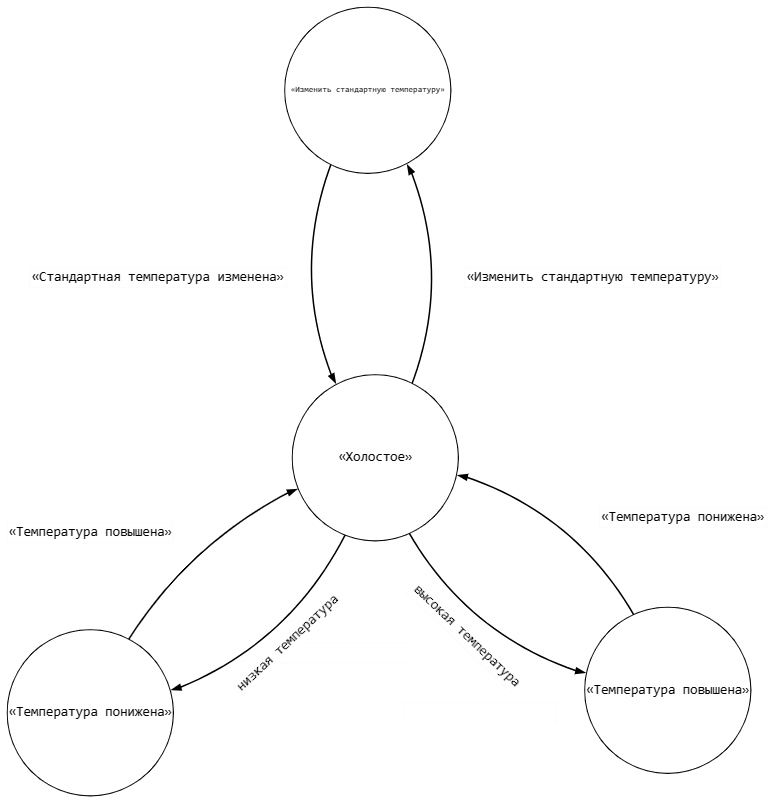


Рисунок 3.1 – Граф автомата климата.

Данный автомат является автоматом Мура.

## 3.2 АВТОМАТ УПРАВЛЕНИЯ УМНЫМ ПЫЛЕСОСОМ

1. Состояния: «Простой», «Зарядка», «Уборка»
2. Вход: сообщение от системного таймера о том, что пора начинать уборку, сообщение от умного пылесоса о том, что он вернулся на базу, сообщение от зарядного устройства о конце зарядки.
3. Начальное состояние: «Простой»
4. Выход: сигнал «Начать уборку», сигнал «Включить зарядку», сигнал «Выключить зарядку».
5. Функция переходов:
6. В состоянии «Простой» при поступлении сообщении о том, что пора начинать уборку автомат переходит в состояние «Уборка».
7. В состоянии «Уборка» при поступлении сообщения о том, что пылесос вернулся на базу автомат переходит в состояние «Зарядка»
8. В состоянии «Зарядка» при поступлении сообщения о конце зарядки автомат переходит в состояние «Простой»
9. Функция выходов: на входе в состояние «Уборка» автомат посылает сигнал пылесосу «Начать уборку», на входе в состояние «Зарядка» автомат посылает сигнал «Включить зарядку», на входе в состояние «Простой» автомат посылает сигнал «Выключить зарядку».

Граф автомата управления умным пылесосом представлен на Рисунке 3.2.

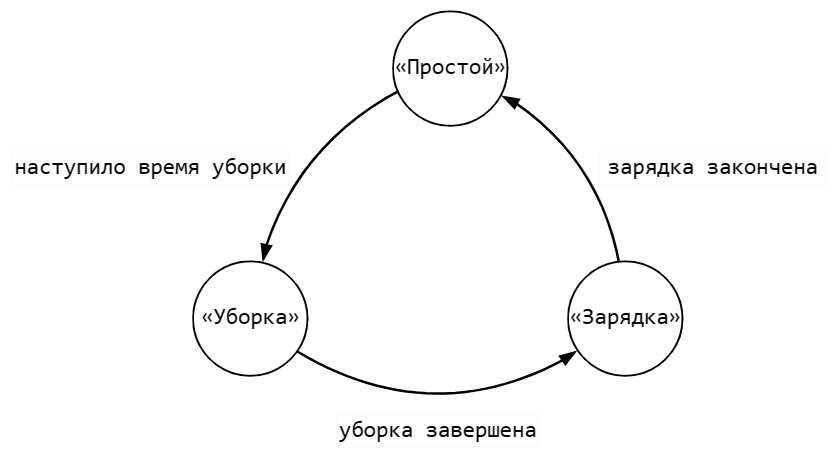


Рисунок 3.2 – Граф автомата управления умным пылесосом.

Данный автомат является автоматом Мура.

## 3.3 АВТОМАТ БЕЗОПАСНОСТИ

1. Вход: сообщение о включении системы безопасности, сообщение от датчика движения, сообщение о конце тревоги, сообщение о выключении системы безопасности.
2. Состояния: «Выключено», «Наблюдение», «Тревога»
3. Начальное состояние: «Выключено».
4. Выходы: сигнал тревоги.
5. Функция переходов:
6. Из состояния «Выключено» при поступлении сообщения о включении автомат переходит в состояние «Наблюдение».
7. Из состояния «Наблюдение» при поступлении сообщения от датчика движения автомат переходит в состояние «Тревога».
8. Из состояния «Наблюдение» при поступлении сообщения о выключении автомат переходит в состояние «Выключено».
9. Из состояния «Тревога» при поступлении сообщения о конце тревоги автомат переходит в состояние «Наблюдение».
10. Из состояния «Тревога» при поступлении сообщения при поступлении сообщения о выключении автомат переходит в состояние «Выключено».
11. Функция выходов: в состоянии «Тревога» автомат посылает сигнал тревоги.

Граф автомата безопасности представлен на Рисунке 3.3.

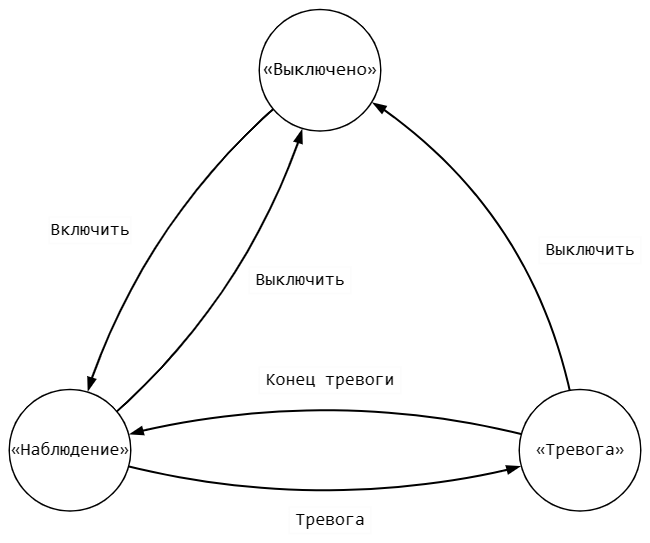


Рисунок 3.3 = Граф автомата безопасности.

## 3.4 ВНЕШНИЕ СОБЫТИЯ

Обработчик внешний событий будет обрабатывать следующие прерывания: сигнал от термометра о пониженной температуре, сигнал от термометра о повышенной температуре, команда от пользователя о изменении стандартной температуры, сигнал от системного таймера о начале уборки, сигнал от умного пылесоса о прибытии на базу, сигнал от зарядного устройства о окончании зарядки пылесоса, команда от пользователя о включении системы безопасности, команда от пользователя о выключении системы безопасности, сигнал от датчика движения.

После обработки данных событий информация о них попадёт в генератор сообщений.

## 3.5 ВЗАИМОДЕЙСТВИЕ С ВНЕШНИМИ УСТРОЙСТВАМИ

Автомат взаимодействия с внешними устройствами будет взаимодействовать со следующими устройствами:

1. Электронный термометр: изменение стандартной температуры.
2. Умный пылесос: начать уборку, включить зарядку, выключить зарядку.
3. Телефон пользователя: послать сигнал тревоги.

Для того чтобы выполнить данные обращения автомат взаимодействия с внешними устройствами должен получить соответствующие сообщения от менеджера сообщений.

## 3.6 ГЕНЕРАТОР СООБЩЕНИЙ

Генератор сообщений будет работать по следующим правилам:

1. При поступлении сигнала от термометра о пониженной температуре будет формировать сообщение для автомата климата о пониженной температуре.
2. При поступлении сигнала от термометра о повышенной температуре будет формировать сообщение для автомата климата о повышенной температуре.
3. При поступлении команды от пользователя о изменении стандартной температуры будет формировать сообщение для автомата климата о изменении стандартной температуры.
4. При поступлении сигнала от системного таймера о начале уборки будет формировать сообщение для автомата управления умным пылесосом о начале уборки.
5. При поступлении сигнала от умного пылесоса о прибытии на базу будет формировать сообщение для автомата управления умным пылесосом о окончании уборки.
6. При поступлении сигнала от зарядного устройства о окончании зарядки пылесоса будет формировать сообщение для автомата управления умным пылесосом о окончании зарядки.
7. При поступлении команды от пользователя о включении системы безопасности будет формировать сообщение для автомата безопасности о включении системы безопасности.
8. При поступлении команда от пользователя о выключении системы безопасности будет формировать сообщение для автомата безопасности о выключении системы безопасности.
9. При поступлении сигнала от датчика движения будет формировать сообщение для автомата безопасности о начале тревоги.
10. При получении сигнала от автомата климата о изменении стандартной температуры будет формировать сообщение для автомата взаимодействия с внешними устройствами о взаимодействии с термометром для изменения стандартной температуры.
11. При получении сигнала от автомата управления умным пылесосом о начале уборки будет формировать сообщение для автомата взаимодействия с внешними устройствами о взаимодействии с умным пылесосом для начала уборки.
12. При получении сигнала от автомата управления умным пылесосом о начале зарядки будет формировать сообщение для автомата взаимодействия с внешними устройствами о взаимодействии с умным пылесосом для начала зарядки.
13. При получении сигнала от автомата управления умным пылесосом о конце зарядки будет формировать сообщение для автомата взаимодействия с внешними устройствами о взаимодействии с умным пылесосом для окончания зарядки.
14. При получении сигнала от автомата безопасности о тревоге будет формировать сообщение для автомата взаимодействия с внешними устройствами о посылании на телефон пользователя информации о тревоге.

После того, как данные сообщения будут сформированы, они попадут в очередь менеджера сообщений, который разошлёт их по соответствующим автоматам.

# ЗАКЛЮЧЕНИЕ

Разработка многозадачных приложений для Arduino становится неотъемлемой задачей в свете растущей сложности устройств, использующих эту микроконтроллерную платформу. В контексте "Интернета вещей" и повышенных функциональных требований, эффективное управление множеством задач и периферийных устройств становится критическим компонентом разработки. Исследование в области многозадачных систем оказывает важное влияние на различные сферы, включая Интернет вещей, автоматизацию и робототехнику, обеспечивая оптимизацию ресурсов, отзывчивость и гибкость систем.

Целью исследования было построение математической модели многозадачного приложения для Arduino на основе теории конечных автоматов. Пройденные этапы исследования включали рассмотрение вариантов эмуляции многозадачности, построение математической модели приложения и приведение примера приложения "Умный дом" в соответствии с описанной моделью. Гипотеза, что автоматный метод является наилучшим в большинстве задач, была подтверждена.

В ходе работы использовались методы анализа, абстрагирования, сравнения и моделирования, а также изучение литературы по теме. Результаты исследования предоставляют разработчикам возможность строить более сложные и гибкие системы для Arduino, способные эффективно реагировать на изменяющиеся условия и требования в различных областях деятельности.

# ЛИТЕРАТУРА

1. Функции времени [Электронный ресурс]. Режим доступа: <https://alexgyver.ru/lessons/time/> (Дата обращения 05.11.23).
2. Многозадачность в Arduino [Электронный ресурс]. Режим доступа: <https://alexgyver.ru/lessons/how-to-sketch/> (Дата обращения 05.11.23).
3. Как использовать FreeRTOS в Arduino – руководство для начинающих [Электронный ресурс]. Режим доступа: <https://microkontroller.ru/arduino-projects/kak-ispolzovat-freertos-v-arduino-rukovodstvo-dlya-nachinayushhih/> (Дата обращения 07.11.23).
4. Timer interrupts [Электронный ресурс]. Режим доступа: <https://arduinodiy.wordpress.com/2012/02/28/timer-interrupts/> (Дата обращения 07.11.23).
5. Многозадачность во встроенном приложении. Часть 1. [Электронный ресурс]. Режим доступа: <https://club.shelek.ru/viewart.php?id=362> Дата обращения 07.11.23).
6. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений, 2-е изд. : Пер. с англ. — М.:Издательский дом «Вильямс», 2002. — ISBN 5-8459-0261-4
7. Конечные автоматы в микроконтроллерах AVR [Электронный ресурс]. Режим доступа: <https://microkontroller.ru/programmirovanie-mikrokontrollerov-avr/konechnye-avtomaty-v-mikrokontrollerah-avr/> Дата обращения 21.11.23).
8. Moore, Edward F. Gedanken-experiments on Sequential Machines // Automata Studies, Annals of Mathematical Studies. — Princeton, N.J.: Princeton University Press, 1956. — No. 34. — P. 129—153.
9. Mealy, George H. A Method to Synthesizing Sequential Circuits — Bell Systems Technical Journal, 1955. — P. 1045—1079.