В ходе выполнения работы была предложена архитектура и реализован макет учебного НКУ, создана программа управления имитатором СМКА и декодирования телеметрии. Разработанная система будет использоваться при тестировании и отработке нано и пикоспутников, а также при обучении студентов аэрокосмических специальностей Белорусского государственного университета.

Литература

1. Тяпичев Г. А. Компьютер на любительской радиостанции. СПб.: БХВ-Петербург, 2002.
2. Описание сетевого протокола AX.25. [Электронныйресурс]. Режимдоступа: http://goryham.qrz.ru/pr/ax25.html Дата доступа: 27.05.2016.

ОБЛАЧНАЯ СИСТЕМА УПРАВЛЕНИЯ АППАРАТНЫМИ КОНТРОЛЛЕРАМИ

С. И. Ганчук, С. В. Сергиенко

В статье описывается решение задачи программирования аппаратных контроллеров в облачной системе и способы их взаимодействия с удаленным сервером. Данная система предоставляет среду реализации и исполнения высокоуровневых алгоритмов для управления аппаратными устройствами на ресурсах облачного вычислительного кластера. Такой подход позволяет существенно облегчить разработку и отладку программного кода и ускорить процесс создания поведенческих алгоритмов для низкоуровневых аппаратных контроллеров.

Современные электронные устройства, как правильно, состоят из нескольких контроллеров, задача которых управлять системными модулями, агрегатами и датчиками, подключёнными к ним. При разработке электронного устройства одним из важнейших шагов является написание программного кода для управления его аппаратными контроллерами (микроконтроллерами). Программирование микроконтроллеров – достаточно трудоёмкий процесс, так как необходимо использовать низкоуровневые языки программирования, а также весьма затруднена отладка из-за сложности доступа к работающему устройству.

Реализованная облачная система позволяет перенести программирование аппаратных контроллеров в вычислительный облачный кластер. Таким образом, контроллеры устройств смогут передавать состояние входов в систему, а обратно принимать указание на установку состояния выходов, по результатам работы заданного алгоритма.

Для реализации такой системы потребовалось решить несколько задач. Первая задача состояла в разработке системы запуска алгоритма для обработки устройств. Алгоритм должен быть запущен изолированно, на

любом поддерживаемом языке программирования, и при этом должен потреблять минимум накладных вычислительных ресурсов.

ruby code:

```ruby
require_relative 'connection.rb'
require_relative 'helpers/bug.rb'
require_relative 'drivers/bugs/line_follower.rb'

class Worker
  def initialize connection
    @connection = connection

    @connection.mail_permission(["lh"])

    @driver = Drivers::Bugs::LineFollower::Main.new self
    puts 'forward'
    @driver.command 'forward'

    @ccc = ['left', 'forward']
  end

  def from_device msg
```

Рис.1. Пример алгоритма на языке Ruby

Вторая задача заключалась в реализации канала связи между микроконтроллером и облачной системой. Данный канал должен иметь возможность максимально быстро передавать небольшие порции данных. Также необходимо предоставить пользовательский интерфейс, который позволил бы писать и выполнять отладку исходного кода алгоритмов.

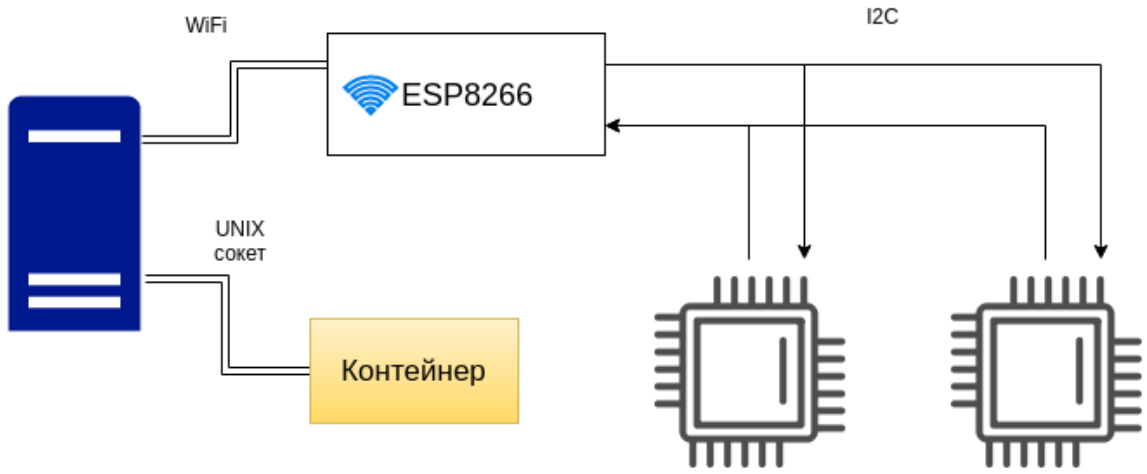

Рис. 2. Структура связи между контейнером и контроллером

Ввиду того, что алгоритмы могут писать пользователи, необходимо было организовать изоляцию алгоритма от основной системы и от других алгоритмов. Для запуска алгоритмов использовалась технология контейнеризации.

Контейнеризация – это метод виртуализации на уровне операционной системы, позволяющий запускать операционные системы внутри легковесных контейнеров [1]. Для каждого поддерживаемого языка программирования были созданы специальные “образы”, которые хранятся в

централизованной базе данных и имеют возможность исполнять пользовательские алгоритмы для подключенных устройств.

Далее требовалось организовать связь контейнера с контроллером. Для связи устройства и сервера используется модуль ESP8266 работающий по технологии Wi-Fi [2]. По средствам Wi-Fi, ESP8266 подключается к локальной сети или к сети интернет, и далее используя протокол WebSockets подключается к серверу по IP адресу или доменному имени. Протокол WebSockets предоставляет постоянное двунаправленное сетевое соединение [3].

Устройство может состоять из множества аппаратных контроллеров. Для того, чтобы минимизировать использование беспроводных модулей, все контроллеры были объединены в общую шину данных. В качестве ведущего микроконтроллера выступает модуль ESP8266. Он выполняет опрос, передачу и приём пакетов от контролеров, которые в дальнейшем может передавать по Wi-Fi.

Для организации связи сервера и контейнера были применены сокеты домена Unix. Сокеты домена Unix являются технологией, позволяющей наиболее эффективным способом передавать данные между процессами, запущенными на одной системе [4]. В конечном итоге устанавливается высокоскоростной канал связи между аппаратными контроллерами и облачным контейнером. Задержка в данном канале составляет несколько миллисекунд.

Пользовательский интерфейс представляет собой интерактивный редактор исходного кода алгоритма, работающий в веб-браузере. Он позволяет пользователю писать исходный код на выбранном языке программирования.

В результате проведенной работы была создана централизованная система управления низкоуровневыми аппаратными контроллерами, которая позволяет осуществить перенос программирования контроллеров в облачный вычислительный кластер и предоставляет ряд преимуществ:

- разработка поведенческих алгоритмов на высокоуровневых языках программирования, таких как Ruby, Python, Javascript;
- использование масштабируемых вычислительных ресурсов кластера для аппаратных контроллеров;
- осуществление взаимодействия между подключенными устройствами;
- удалённое управление и контроль загрузки ресурсов облачного кластера.

Благодаря данной системе предоставляется возможность существенно облегчить подход в программировании низкоуровневых аппаратных

контроллеров и сделать еще более доступным изучение основ объектно-ориентированного программирования и робототехники.

Литература

1. Статья “Контейнеризация на Linux в деталях” [Электронный ресурс] https://habrahabr.ru/company/FastVPS/blog/209072/ Дата доступа: 31.06.2016
2. Документация по “Arduino core for ESP8266 WiFi chip” [Электронный ресурс] https://github.com/esp8266/Arduino Дата доступа: 31.06.2016
3. Документация по протоколу WebSockets [Электронный ресурс] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API Дата доступа: 31.06.2016
4. Документация по сокетам домена UNIX [Электронный ресурс] http://man7.org/linux/man-pages/man7/unix.7.html Дата доступа: 31.06.2016

ЭЛЕКТРОФИЗИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПОЛИМЕРНЫХ МАТЕРИАЛОВ НА ОСНОВЕ ЭПОКСИДНОЙ СМОЛЫ, СОДЕРЖАЩЕЙ УГЛЕРОДНЫЕ НАНОТРУБКИ

М. В. Гринченко, И. Д. Парфимович

ВВЕДЕНИЕ

Открытые более четверти века назад углеродные нанотрубки (УНТ) привлекают к себе внимание многих специалистов и исследователей благодаря своим уникальным электрическим и механическим характеристикам. Использование УНТ в качестве добавок к другим материалам, в частности полимерам, также является интересным и перспективным, что уже успешно используется для создания качественно новых материалов в различных отраслях промышленности. [1,2] Весьма привлекательным является придание электропроводящих свойств полимерам, которые почти всегда являются хорошими электроизоляторами, путем добавления в них УНТ. Вследствие чего, открывается возможность создания прозрачных и проводящих покрытий, электростатической защиты, электростатических красок, покрытий для экранирования электропомех и поглощения СВЧ мощности. [3,4] Синтез одностеночных углеродных трубок, даже в настоящее время, достаточно дорог и при этом выход не высок. В этой связи для создания композитных материалов, содержащих наноразмерные углеродные образования, представляется интересным, использовать не дорогие материалы и доступные в больших количествах. К таким материалам можно отнести промышленно выпускаемые многостеночные УНТ.