Методы сборочного программирования в интернете вещей

Агамирзян Игорь Рубенович,

Профессор, Заведующий кафедрой,

вице-президент НИУ ВШЭ

Киберфизические системы и интернет вещей

- Киберфизические системы в целом и интернет вещей в частности, как правило, представляют собой распределённые вычислительные системы
  - с большим числом интеллектуальных узлов
  - узлы взаимодействуют с сенсорами и актуаторами
  - узлы взаимодействуют и между собой
  - передавая и получая как данные, так и управляющую информацию
- Программный код на конкретном контроллере может быть вполне тривиальным
  - но вся система в целом оказывается функционально весьма сложной

Сборочное программирование

- Сборочное программирование – «это метод создания сложных программ для ЭВМ на базе однажды разработанных, многократно используемых программных компонент»
  - Липаев В.В., Позин Б.А., Штрик А.А.
- Сборочное программирование это – «построения сложных программ из более простых программных элементов, которые записаны в современных языках программирования»
  - Лаврищева Е.М, Грищенко В.Н.
- В киберфизических системах и интернете вещей программный код переиспользуется по определению – в силу множественности однотипных «вещей»

Сборочное программирование в IoT

- Часто традиционно решаемые в классических вычислительных системах добавлением новых классов или процедур задачи в интернете вещей можно эффективнее решать добавлением новых контроллеров и устройств
  - это связано как с ресурсными ограничениями микроконтроллеров, так и со снижением затрат на распараллеливание вычислений
- Киберфизические системы принципиально архитектурно неоднородны
  - как правило, они реализуются с использованием множества различных архитектур процессоров
  - с различными ресурсами в зависимости от решаемых ими локальных задач
  - программируемых на множестве различных языков программирования
- В этом заключается принципиальное отличие от традиционных распределённых вычислительных систем
  - например, вычислительных кластеров

От вызовов к интерфейсам

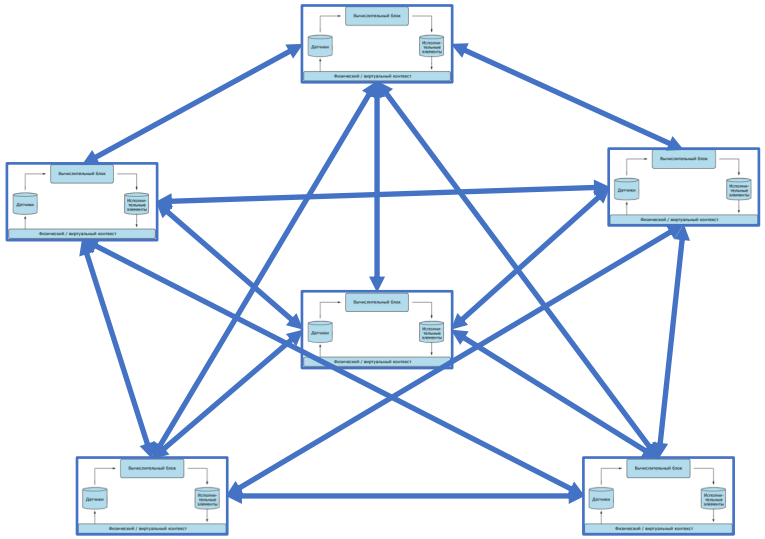
- Основным методом применения сборочного программирования в традиционных вычислительных системах является инструмент вызова (call) – процедуры, функции, метода
  - в некоторых случаях в параллельном режиме на том же или другом ядре или процессоре
- Интерфейсы в традиционном системном программном обеспечении существуют в зачаточном виде
  - Pipes в Unix и Linux
  - MPI (Message Passing Interface) в компьютерных кластерах
- Зато в сетевом мире интерфейсы занимают основное место

Вызовы и интерфейсы

- Киберфизические системы наследуют инструменты как классических вычислительных систем, так и сетевых систем
  - вызовы в первую очередь через библиотеки и сборочные пакеты в рамках одного узла
  - интерфейсы для взаимодействия
    - с сенсорами и актуаторами
    - между узлами (в сети)
- При этом ключевым способом определения взаимодействия становится протокол
  - протокол – это фактически аналог метода передачи параметров в вызовах традиционного программирования

Интерфейсы и протоколы

- Каждый узел киберфизической системы взаимодействует с сенсорами (датчиками) и актуаторами (исполнительными элементами) через интерфейсы по каким-то протоколам
- Каждый узел киберфизической системы взаимодействует с другими узлами через какие-то интерфейсы по каким-то протоколам
  - как правило, это другие интерфейсы и протоколы



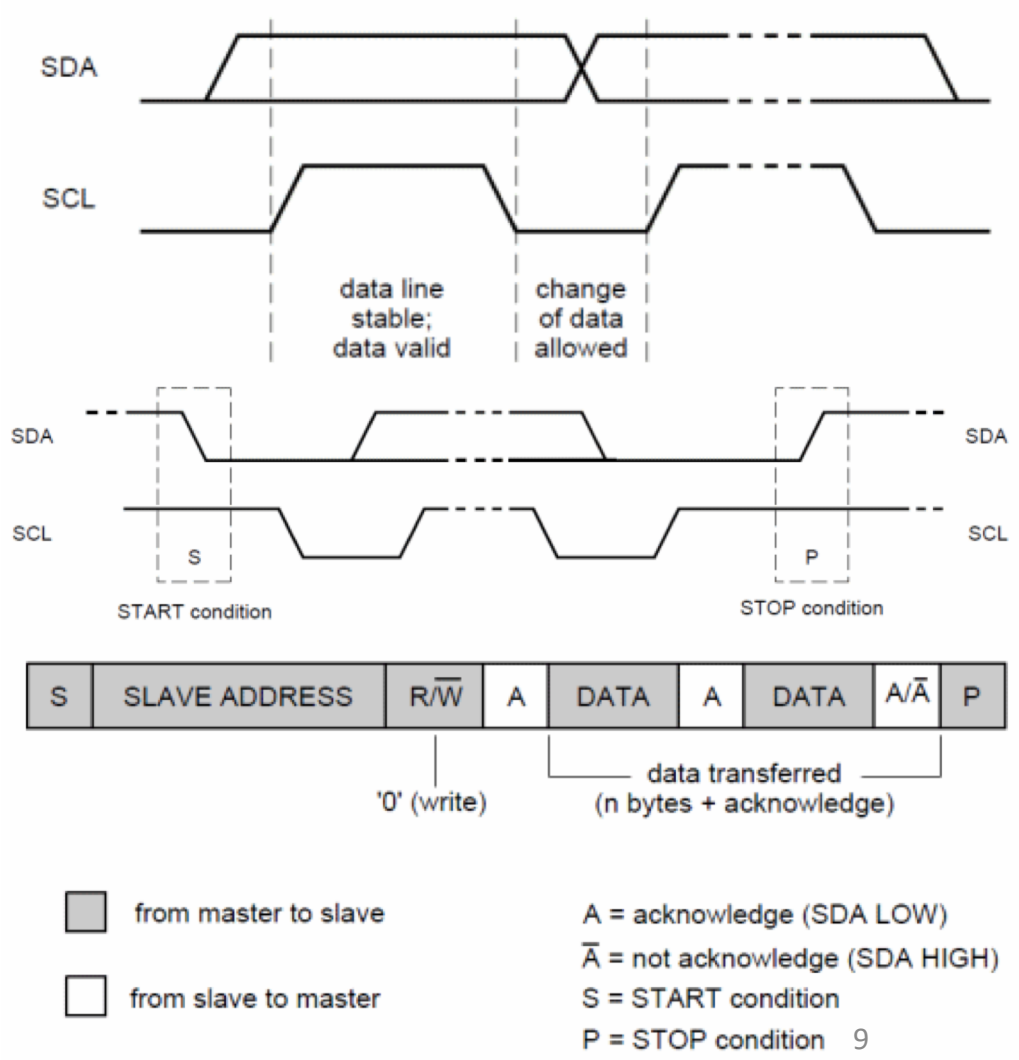
Интерфейсы

- Интерфейсом называется физический механизм передачи информации между
  - Контроллером и сенсором
  - Контроллером и актуатором
  - Двумя контроллерами

  с формализованным протоколом взаимодействия
- Например, провод, соединяющий датчик температуры с контроллером может считаться интерфейсом, если определено, каким образом по нему передаётся информация
  - Аналоговым
  - Цифровым в соответствии с каким-то протоколом
- Существует несколько категорий интерфейсов:
  - Ближнего действия (внутри платы, например)
  - Среднего действия (внутри устройства)
  - Дальнего действия (между устройствами)
- Ортогональная классификация интерфейсов:
  - Проводные интерфейсы
  - Беспроводные интерфейсы

Примеры интерфейсов в IoT - I2C

- Интерфейс I2C (https://i2c.info/i2c-bus-specification)
  - Ближний/средний проводной интерфейс
  - Шина I2C использует 2 провода для передачи информации (с общей землёй)
    - serial data (SDA)
    - serial clock (SCL)
  - Устройства на интерфейсе всегда являются Master или Slave
    - Master инициирует передачу информации и генерирует пульс (clock)
    - Slave отвечает на команды на шине
    - Каждый Slave на шине имеет уникальный адрес
    - На шине может быть несколько Master-устройств, и в этом случае необходим арбитраж
  - Передача команд ведётся в старт-стопном режиме



Примеры интерфейсов в IoT – 1Wire

- Интерфейс 1Wire (https://www.electronics.ru/files/article_pdf/0/article_668_639.pdf)
  - Средний/дальний проводной интерфейс
  - Шина 1Wire использует 1 провод для передачи информации (с общей землёй)
  - Устройства на интерфейсе всегда являются Master или Slave
  - На шине возможен только один Master
  - Устройства подключаются к шине по схеме с открытым стоком и подтягивающим резистором
  - У каждого устройства 1-Wire есть 64-разрядный идентификатор (ID)
- Интерфейс 1Wire требует более интеллектуальных Slave-устройств, чем интерфейс I2C

Примеры интерфейсов в IoT – SPI

- Serial Peripheral Interface (SPI) – синхронный последовательный дуплексный интерфейс ближнего действия
- Разработан Motorola в середине 1980-х годов и стал индустриальным стандартом
- Типичные применения – работа с SD-картами и LED-дисплеями
- Устройства на интерфейсе всегда являются Master или Slave
  - На шине возможен только один Master
- Особенностью интерфейса является адресация устройств по выделенной линии (SS)
  - Кроме того, для обеспечение дуплекса используются 2 линии (MISO и MOSI)
  - Третья общая линия (CS) используется для тактирования мастером
- Таким образом, в SPI используется всегда 3+N линий, где N – количество устройств на шине
- Обычно контроллер SPI встраивается в систему на кристалле микроконтроллера
- В некоторых случаях применяется для внутрисхемного программирования Flash-памяти микроконтроллеров

Примеры интерфейсов в IoT – Serial

- Интерфейс Serial
  - Проводной интерфейс среднего/дальнего действия
  - Исторически происходит от стандарта RS232 (см, например, http://www.softelectro.ru/rs232.html)
  - В микроконтроллерных технологиях может использовать всего два сигнальных провода (с общей землёй) – TXD и RXD
  - В оригинальном стандарте используется 9 сигнальных линий
- Реализуется как программно, так и аппаратно
  - Аппаратные решения – как правило, пины 0 и 1 микроконтроллера
  - Программное решение – реализуется библиотекой для любых пинов
- Часто используется как виртуальный интерфейс внутри USB-канала
  - для этого используются специализированные микросхемы
- Существует промышленный интерфейс сверхдальнего действия RS485, в котором обычно используется протокол Serial-интерфейса
- Serial-интерфейс требует весьма интеллектуальных устройств (буферизация и т.д.), и обычно используется между контроллерами

Примеры протоколов в IoT – UDP и TCP/IP

- Интернет-протоколы UDP и TCP/IP
  - Для работы интернет-протоколов необходим доступ контроллера к локальной сети через интерфейс
    - Ethernet
    - WiFi
  - Для различных контроллеров возможны либо платы расширения, либо встроенный интерфейс WiFi
    - ESP 8266 и ESP 32 имеют встроенный WiFi в диапазоне 2.4 GHz
    - Raspberry PI изначально имеет Ethernet-порт и встроенный WiFi начиная с версии 3
    - Существуют варианты Arduino с встроенным WiFi
  - При наличии доступа в сеть возможна работа со всеми стандартами интернета, включая HTTP, SSL/TLS и т.д.
  - Существуют специальные протоколы для интернета вещей, работающие поверх TCP/IP
    - MQTT – Message Queuing Telemetry Transport

Протокол MQTT (https://mqtt.org)

- Упрощённый сетевой протокол, работающий поверх TCP/IP, ориентированный на обмен сообщений между устройствами по принципу издатель-подписчик
- Первая версия – 1999 год
- Спецификация MQTT 3.1.1 была стандартизирована консорциумом OASIS в 2014 году
- Возможно использование как в локальной сети, так и в интернете
- Наиболее известный брокер – Mosquitto (https://mosquitto.org, может работать на Raspberry Pi)
- Множество доступных серверов в сети – например, https://www.emqx.io/mqtt/public-mqtt5-broker
- Множество инструментов для клиентских тестовых систем (например, MQTT fx – https://softblade.de/en/mqtt-fx/)
- Полностью Open Source – решение

Основные операции протокола MQTT

- Connect
  - MQTT::Connect con (CLIENT_ID);
  - con.set_auth (USER, PASSWORD);
  - con.set_will (WILLTOPIC, "Offline");
- Subscribe
  - client.set_callback (callback);
  - client.subscribe (CMNDTOPIC);
- Publish
  - client.publish (MQTT::Publish (WILLTOPIC, "Online").set_qos (1));
  - client.publish (MQTT::Publish (STATTOPIC, "ON").set_qos (1));
- Примеры из библиотеки <PubSubClient.h>

Функция Callback в MQTT

- При подписке на topic задаётся функция, которая будет вызвана, когда кто-то опубликует сообщение в этом топике, например:

```
void callback (const MQTT::Publish& pub) {
  state = (pub.payload_string () == "on");
  digitalWrite (relay, state);
}
```

- Топики устроены как дерево файлов, например, типичное имя топика ”stat/monitor/POWER/0”
- При публикации топика указывается его значение (payload) – это всегда просто строка

Клиенты MQTT-брокера

- Клиентами могут быть программы на
  - Микроконтроллерах
  - Компьютерах
  - Смартфонах
  - Любых других интеллектуальных устройствах, имеющих доступ к сети
- MQTT позволяет
  - Без особых затрат обмениваться информацией между устройствами, входящими в киберфизический проект
  - Осуществлять синхронизацию и оркестрацию работы комплекса устройств
  - Реализовывать сложные проекты интернета вещей, класса «Умный дом» или «Интеллектуальная транспортная система»

Примеры интерфейсов в IoT – с низким энергопотреблением

- Особенностью решений интернета вещей является в ряде случаев требование низкого энергопотребление
  - Особенно актуально для беспроводных устройств с батарейным питанием
  - Стандартные беспроводные протоколы (из группы стандартов WiFi) не являются энергоэффективными
- Примером такого интерфейса и группы протоколов является RF433
- Другим примером интерфейса и протокола является Bluetooth
- Однако основным стандартом для использования в решениях интернета вещей стал протокол Zigbee
- Для работы энергоэффективных протоколов в стандартных сетях требуются шлюзы
  - примером такого шлюза является Zigbee2MQTT

zigbee и Zigbee2MQTT

- Zigbee, так же как и Bluetooth является беспроводным протоколом для создания PAN (Personal area network)
  - радиус действия не меньше 10 и не больше 100 метров
- Базируется на стандарте IEEE 802.15.4
- В отличие от Bluetooth позволяет создавать WANET (Wireless ad hoc network) – децентрализованные беспроводные сети
  - В Zigbee сетях существуют три типа устройств – единственный координатор, роутеры и конечные устройства
  - Именно роутеры умеют договариваться между собой для организации Mesh-сети
- Разработан Zigbee Alliance (Connectivity Standards Alliance) – включает более 500 компаний, специализирующихся в решениях для интернета вещей
- Устройства на базе Zigbee отличаются миниатюрным размером и низким энергопотреблением, позволяющим обеспечить 1-3 года эксплуатации на стандартном элементе питания 3V типа 2032
- Zigbee2MQTT – Open Source шлюз, позволяющий Zigbee-устройствам работать в MQTT-сетях, требует наличия аппаратного координатора Zigbee и интерфейса Ethernet/WiFi, может работать на Raspberry Pi

Более высокоуровневые протоколы

- Иногда в киберфизических системах и решениях интернета вещей используются более высокоуровневые протоколы, характерные скорее для DevOps-решений
  - например, AMQP (Advanced Message Queuing Protocol) с использованием RabbitMQ

  но это скорее исключение, чем правило
- Зато в случае взаимодействия киберфизических систем с облачными сервисами широкое распространение получили высокоуровневые протоколы на базе HTTP
  - например, Яндекс в своих решениях для умного дома (Яндекс Диалоги), позволяющих интегрировать IoT с Алисой, применяет взаимодействие по HTTP

2021-11-14T21:17:03+03:00: Sending request to provider:
GET https://smarthome.address/api/yandex_smart_home/v1.0/user/devices
request id: 96d3edea-3eb6-448a-954a-89c7afc4d654

Проблемы с облачными решениями

- Киберфизические системы и решения интернета вещей могут пользоваться облачными сервисами
  - соответствующий инструментарий доступен
  - в ряде случаев облачные сервисы предоставляют возможности, которые трудно или невозможно обеспечить на уровне микроконтроллеров

  но должны обеспечивать эффективное базовое функционирование в случае недоступности облака
  - например, для умного дома при недоступности Алисы или другого голосового помощника
  - роботы и беспилотники не могут использовать облачные сервисы при принятии решений в силу требований реального времени
- Собственно, именно в силу этого IoT-решения должны рассматриваться как модель Edge-вычислений

Сборочное программирование в интернете вещей

- Сборочное программирование в интернете вещей - это **построение системы из программных и аппаратных компонентов с использованием программных и аппаратных интерфейсов**
- Такое определение является обобщением понятия сборочного программирования в традиционных вычислительных системах
- Технологии сопровождения и развития киберфизических систем в значительной мере определяют стоимость жизненного цикла таких систем
- На сегодняшний день нельзя говорить, что в этой области сформировались стандарты, в отличие от традиционной программной инженерии
- Существуют разнонаправленные тренды
  - Коммерческие поставщики IoT решений как правило, стремятся замкнуть потребителей в своей экосистеме
  - Одновременно существует сильный тренд на интеграцию решений от различных поставщиков на Open Source – платформе

Q&A