

Аппаратное обеспечение IoT/CPS

Лекция 1

А. А. Подшивалов
apodshivalov@miem.hse.ru

О себе



- ▶ Александр Анатольевич Подшивалов
apodshivalov@miem.hse.ru
- ▶ Роскосмос, несколько сколковских стартапов, Huawei RRI
- ▶ Интересы — сети связи и их математическое моделирование, программирование встраиваемых систем, вычислительная математика и цифровая обработка сигналов

Несколько вводных слов

- ▶ Формула оценивания

$$0,4 \times \text{Практика} + 0,4 \times \text{Экзамен} + 0,2 \times \text{Доклад}$$

- ▶ На практических занятиях предлагаются задания различной сложности (1 балл — повторить пример из документации, 10 баллов — серьезный вклад в opensource-проект), суммарно можно набрать до 10 баллов (арифметика «с насыщением»)
- ▶ «Удаленки» **не будет**, контроля посещаемости **не будет**
- ▶ На экзамене можно пользоваться **любой** литературой, конспектами, интернетом, ...

Примерное содержание курса

- ▶ Программирование микроконтроллеров
 - ▶ Cortex-M (семейства STM32, nRF52, CC26xx), RISC-V
 - ▶ Язык программирования C
 - ▶ Операционная система Riot
- ▶ Сетевые технологии (LoRaWAN, 6LoWPAN)
- ▶ «Экзотика» — энергосбережение, безопасность и т. п.

Лекции

1. Вводная
2. GPIO, таймеры, прерывания
3. RTOS на примере Riot
4. Интерфейсы UART, I2C, SPI
5. АЦП и ЦАП
6. Беспроводные технологии интернета вещей
7. Энергосберегающие режимы работы
8. Память, DMA
9. Цифровая обработка сигналов
10. Функциональная безопасность встраиваемых систем

Примерные темы докладов

1. Архитектура ARM
2. Архитектура RISC-V и ОС Riot
3. Внешние шины передачи данных (USB, CAN и т. п.)
4. MEMS-датчики (акселерометры, гироскопы, магнетометры, ...)
5. Датчики концентрации газов
6. Алгоритм CSMA/CA в стандартах IEEE 802.11 и IEEE 802.15.4
7. Mesh-сети, IEEE 802.15.4 и 6LoWPAN
8. Протокол маршрутизации RPL
9. Bluetooth Low Energy
10. 6LoWPAN поверх BLE
11. Link 16 (TADIL J)
12. Локальные и глобальные системы позиционирования, ГНСС
13. Фильтр Калмана и его применение для позиционирования
14. Многочастичный фильтр
15. Нейросети на микроконтроллерах

Интернет вещей и киберфизические системы

Интернет вещей

Интернет вещей

Русская Википедия: «концепция вычислительной сети физических предметов („вещей“), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой, рассматривающая организацию таких сетей как явление, способное перестроить экономические и общественные процессы, исключающее из части действий и операций необходимость участия человека»

Интернет вещей

Русская Википедия: «концепция вычислительной сети физических предметов („вещей“), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой, рассматривающая организацию таких сетей как явление, способное перестроить экономические и общественные процессы, исключающее из части действий и операций необходимость участия человека»

Англоязычная Википедия: «physical objects (or groups of such objects) with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks»

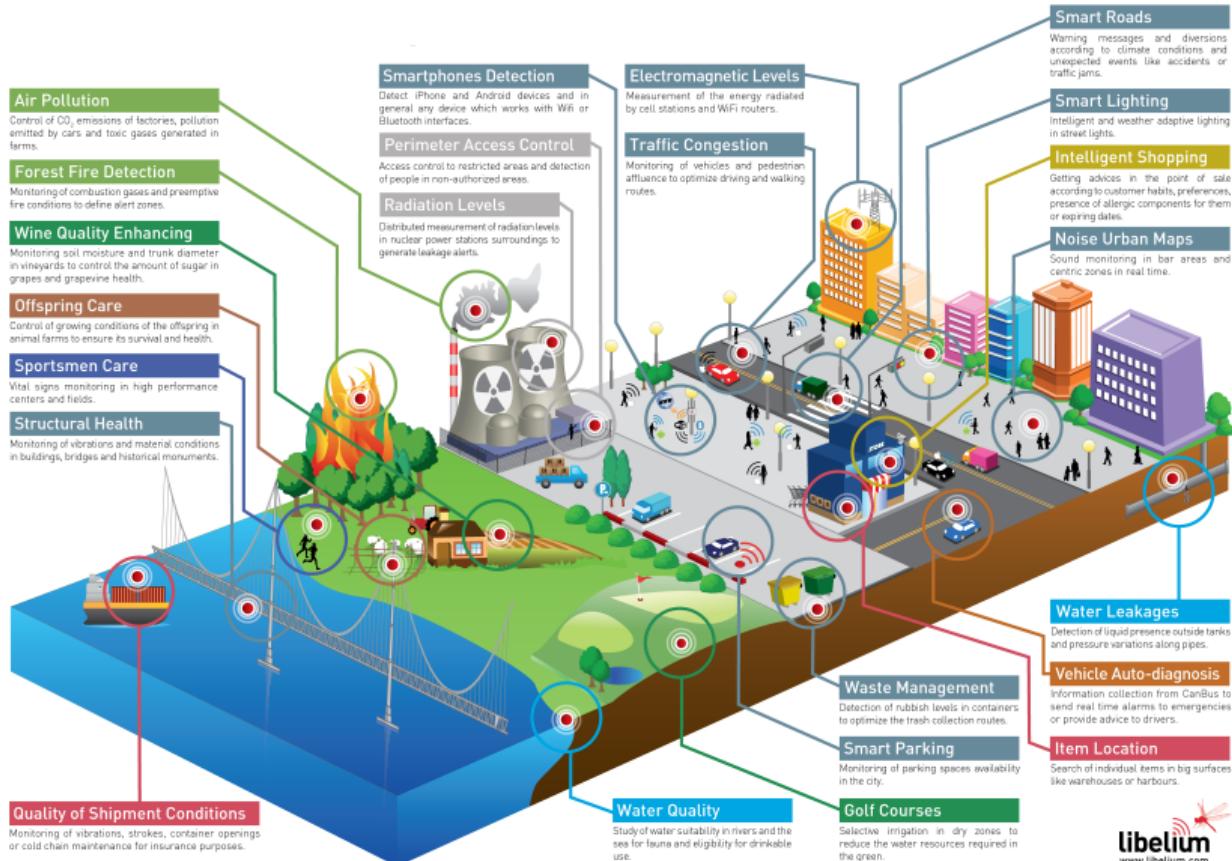
Интернет вещей

Русская Википедия: «концепция вычислительной сети физических предметов („вещей“), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой, рассматривающая организацию таких сетей как явление, способное перестроить экономические и общественные процессы, исключающее из части действий и операций необходимость участия человека»

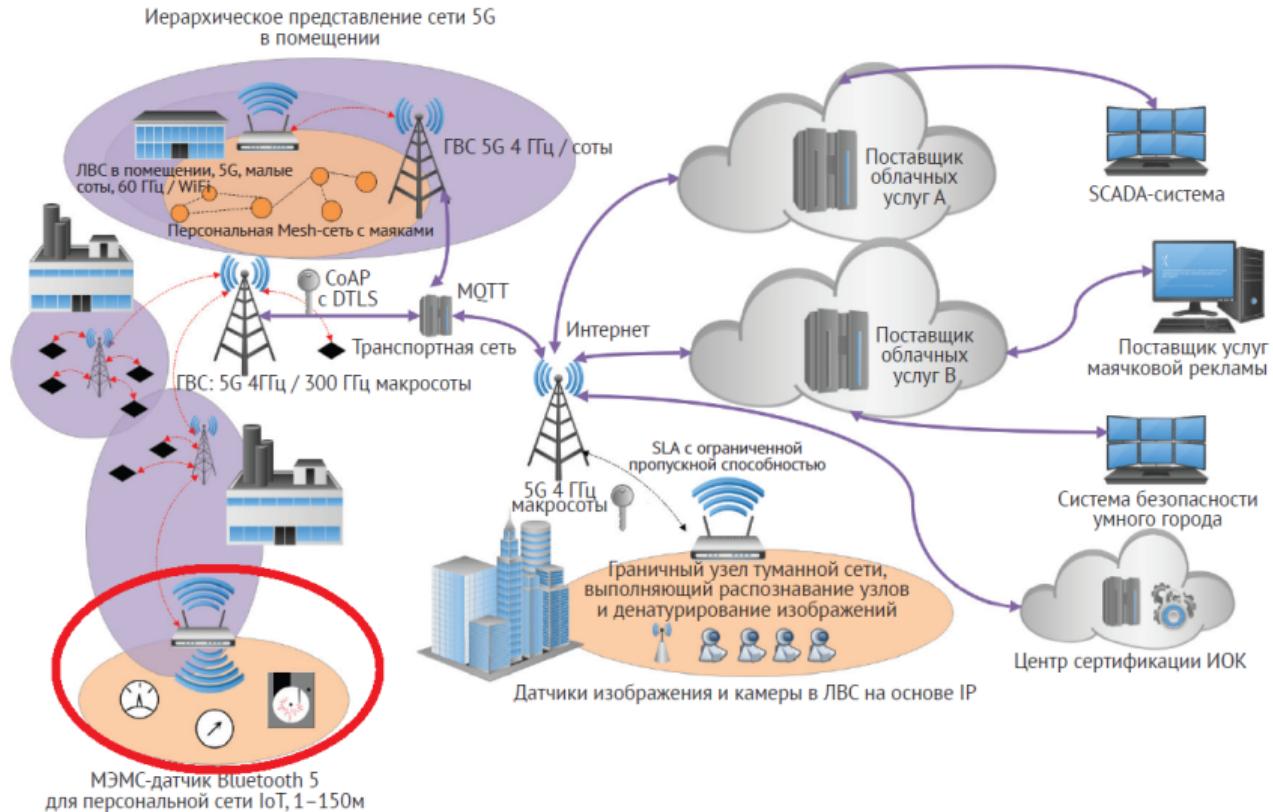
Англоязычная Википедия: «physical objects (or groups of such objects) with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks»

Мне вот такое определение нравится: **технологии и подходы к проектированию систем, обеспечивающие сбор и передачу данных там, где раньше это было технически невозможно или экономически неэффективно**

Некоторые примеры систем IoT



Место систем сбора данных в архитектуре IoT



Требования к «вещам» для IoT противоречивы

- ▶ Дешевизна — 10 \$ за устройство
- ▶ Компактность — размеры не более спичечного коробка
- ▶ Экономичность — месяцы/годы на одной батарейке
- ▶ Дальность связи
- ▶ Безлицензионность



Киберфизические системы

Киберфизические системы

Русская Википедия: «информационно-технологическая концепция, подразумевающая интеграцию вычислительных ресурсов в физические сущности любого вида, включая биологические и рукотворные объекты»

Киберфизические системы

Русская Википедия: «информационно-технологическая концепция, подразумевающая интеграцию вычислительных ресурсов в физические сущности любого вида, включая биологические и рукотворные объекты»
Англоязычная Википедия: «a computer system in which a mechanism is controlled or monitored by computer-based algorithms»

Киберфизические системы

Русская Википедия: «информационно-технологическая концепция, подразумевающая интеграцию вычислительных ресурсов в физические сущности любого вида, включая биологические и рукотворные объекты»

Англоязычная Википедия: «a computer system in which a mechanism is controlled or monitored by computer-based algorithms»

Очень широкие определения, относящиеся практически к любым **встраиваемым системам (embedded systems)**

Современные микроконтроллеры: основные архитектуры

Микроконтроллер vs микропроцессор

- Микроконтроллер — «однокристальная микро-ЭВМ», объединение CPU, памяти (ROM и RAM) и некоторых периферийных устройств на одном чипе
- «Промежуточные» варианты — SoC, SiP

	MCU	CPU
ОЗУ (RAM)	встроенное	внешнее
Объем ОЗУ	< 1 Мб	$\gg 1$ Мб
Постоянная память (ROM)	встроенная	внешняя
Объем ROM	< 1 Мб	$\gg 1$ Мб
Периферия	в основном встроенная	в основном внешняя

Типичные характеристики микроконтроллеров

- ▶ STM32L151CC
 - ▶ Ядро Cortex-M3, тактовая частота до 32 МГц
 - ▶ 256 кБ Flash-памяти (ROM), 8 кБ EEPROM
 - ▶ 32 кБ оперативной памяти
- ▶ nRF52832-QFAA
 - ▶ Ядро Cortex-M4F, тактовая частота 64 МГц
 - ▶ 512 кБ Flash-памяти (ROM)
 - ▶ 64 кБ оперативной памяти
- ▶ На 3-6 порядков хуже «настольного» компьютера по производительности и объему памяти

Типичные характеристики микроконтроллеров

- ▶ STM32F100C4T6B
 - ▶ Ядро Cortex-M3, тактовая частота до 24 МГц
 - ▶ 16 кБ Flash-памяти (ROM)
 - ▶ 4 кБ оперативной памяти
- ▶ ATmega8A
 - ▶ Ядро AVR, тактовая частота до 16 МГц
 - ▶ 8 кБ Flash-памяти (ROM), 512 байт EEPROM
 - ▶ 1 кБ оперативной памяти
- ▶ PIC16F84A
 - ▶ Ядро PIC16, тактовая частота до 20 МГц
 - ▶ 1024 слова (14 бит) памяти программ (ROM), 64 байта EEPROM
 - ▶ 68 байт оперативной памяти

Классификация устройств Интернета вещей по RFC 7228

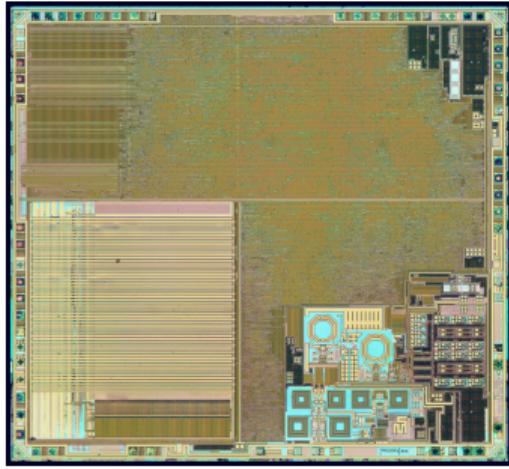
	Память данных (RAM)	Память программ (ROM)
Класс 0	$\ll 10$ Кб	$\ll 100$ Кб
Класс 1	~ 10 Кб	~ 100 Кб
Класс 2	~ 50 Кб	~ 250 Кб

Микроконтроллер: фото вживую

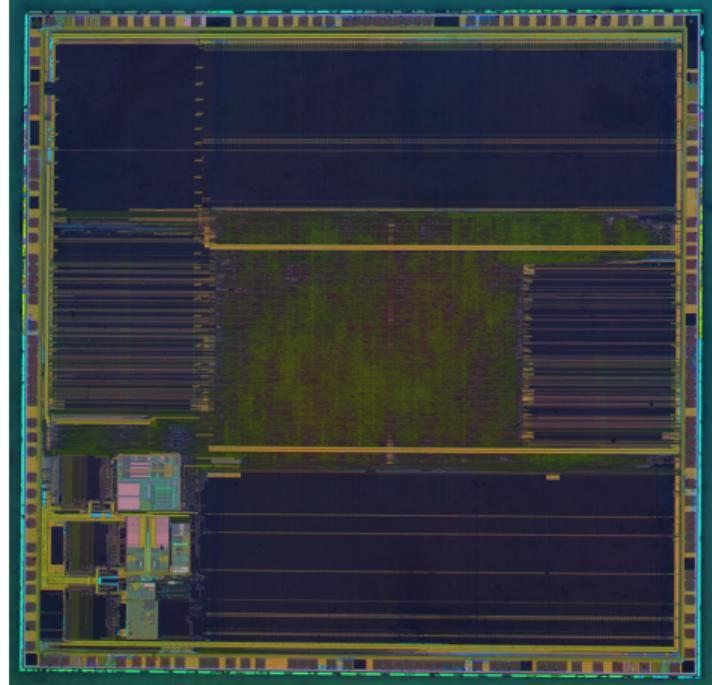


Верху — nRF51822, Cortex-M0, 256
Кб Flash, 32 Кб RAM, встроенный
радиомодем, корпус QFN48, 6 × 6
мм; справа — STM32F103VGT6,
Cortex-M3, 1 Мб Flash, 96 кБ RAM,
LQFP100, 14 × 14 мм

Микроконтроллер: фото кристалла

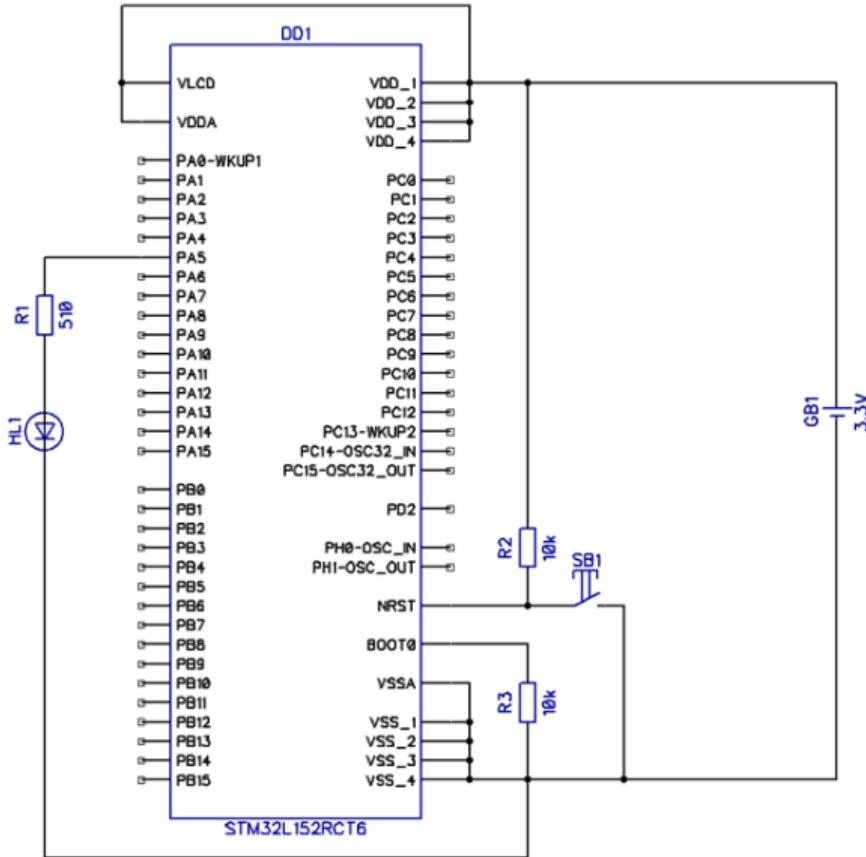


Верху — nRF51822, Cortex-M0, 256
Кб Flash, 32 Кб RAM, встроенный
радиомодем, 3833×3503 мкм;
справа — STM32F103VGT6, 1 Мб
Flash, 96 кБ RAM, 5339×5188 мкм,
техпроцесс 180 нм



Микроконтроллер: минимальная схема

- ▶ Нужно только питание (3,3 В)
- ▶ Встроенный тактовый генератор не требует внешних компонентов
- ▶ Не забываем про схему сброса и выбор источника загрузки (если есть)
- ▶ Вопросов «прошивки» и подключения отладчика пока не касаемся



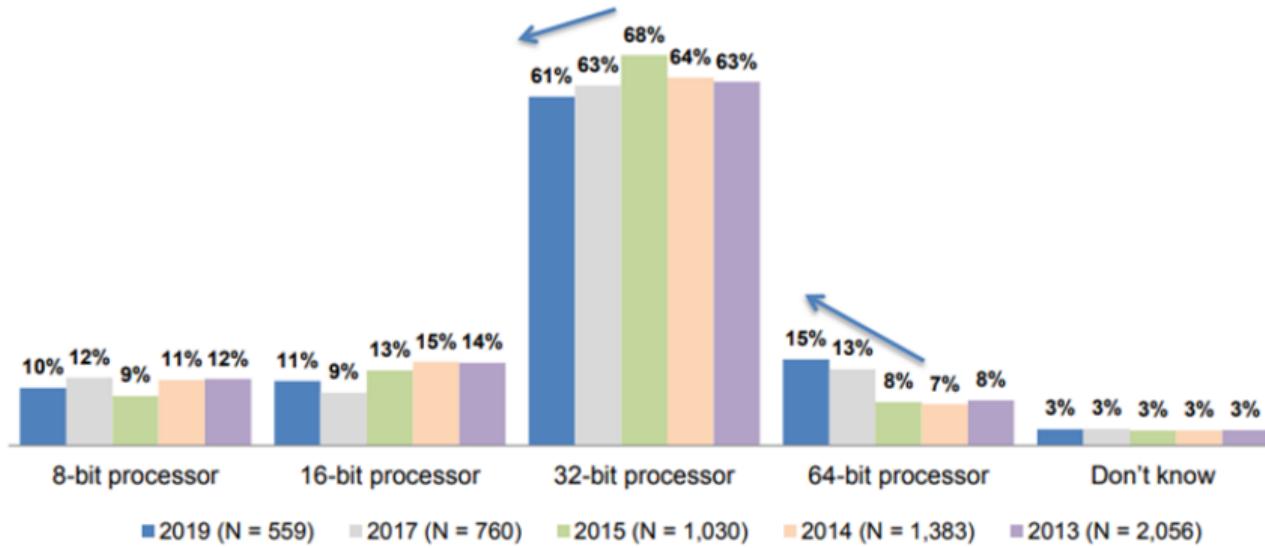
Распространенные архитектуры микроконтроллеров

8 бит	Microchip PIC	Microchip AVR	(Intel) 8051	STMicro STM8	Zilog Z80
16 бит	TI MSP430	Microchip PIC24	Infineon C166		
32 бита	ARM Cortex-M	RISC-V	Microchip AVR32	Microchip PIC32	TenSilica Xtensa

Популярные архитектуры микроконтроллеров



My current embedded project's main processor is a:



71% of EMEA users use 32-bit chips as their main processor.

Популярные архитектуры микроконтроллеров

8 бит	Microchip PIC	Microchip AVR	(Intel) 8051	STMicro STM8	Zilog Z80
16 бит	TI MSP430	Microchip PIC24	Infineon C166		
32 бита	ARM Cortex-M	RISC-V	Microchip AVR32	Microchip PIC32	TenSilica Xtensa

В 2023 году 8- и 16-битные микроконтроллеры имеет смысл использовать лишь в особых случаях

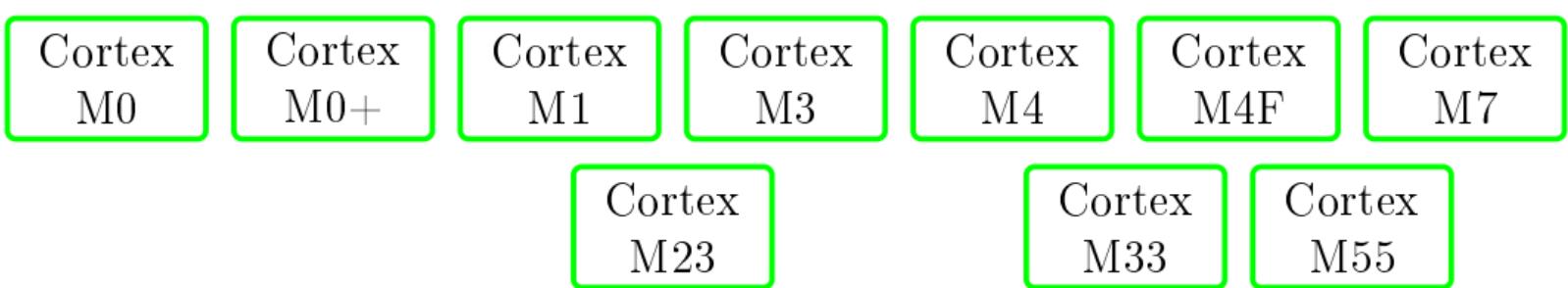
Семейство ARM Cortex

Cortex-M
Микроконтроллеры

Cortex-R
Системы реального
времени

Cortex-A
Процессоры общего
назначения

Семейство ARM Cortex-M



Некоторые производители ARM Cortex-M



Почему Cortex-M?

- ▶ Отличная производительность (1,25 DMIPS/МГц)
- ▶ Богатый выбор инструментов для программиста
- ▶ Огромный выбор различных моделей
- ▶ Очень богатый набор встроенной периферии
- ▶ Программная конфигурация процессора «на лету»
- ▶ Низкое энергопотребление и продвинутое управление питанием
- ▶ Низкая стоимость



STM32 MCUs

32-bit Arm® Cortex®-M



High
Performance

STM32F2
398 CoreMark
120 MHz Cortex-M3

STM32F4
608 CoreMark
180 MHz Cortex-M4

STM32F7
1082 CoreMark
216 MHz Cortex-M7

STM32H7
Up to 3224 CoreMark
Up to 550 MHz
Cortex-M7
240 MHz Cortex-M4



Mainstream

STM32G0
142 CoreMark
64 MHz Cortex-M0+

STM32G4 ●
569 CoreMark
170 MHz Cortex-M4

STM32F0
106 CoreMark
48 MHz Cortex-M0

STM32F1
177 CoreMark
72 MHz Cortex-M3

STM32F3 ●
245 CoreMark
72 MHz Cortex-M4

Optimized for
mixed-signal applications



Ultra-low-
power

STM32L0
75 CoreMark
32 MHz Cortex-M0+

STM32L1
93 CoreMark
32 MHz Cortex-M3

STM32L4+
409 CoreMark
120 MHz Cortex-M4

STM32U5
651 CoreMark
160 MHz Cortex-M33

STM32L4
273 CoreMark
80 MHz Cortex-M4

STM32L5
443 CoreMark
110 MHz Cortex-M33



Wireless

STM32WL
162 CoreMark
48 MHz Cortex-M4
48 MHz Cortex-M0+

STM32WB ●
216 CoreMark
64 MHz Cortex-M4
32 MHz Cortex-M0+

Cortex-M0+
Radio co-processor

Почему не Cortex-M?

- ▶ Предыдущие слайды были сделаны в 2018 году

Почему не Cortex-M?

- ▶ Предыдущие слайды были сделаны в 2018 году
- ▶ «Кризис полупроводников», санкции
- ▶ Альтернативы?

Почему не Cortex-M?

- ▶ Предыдущие слайды были сделаны в 2018 году
- ▶ «Кризис полупроводников», санкции
- ▶ Альтернативы?
- ▶ RISC-V
 - ▶ Открытая архитектура
 - ▶ Российские производители («Миландр», «Микрон»)

С чем будем работать

- ▶ STM32
 - ▶ STM32L151CC — в «конструкторе» Unwired Devices
 - ▶ STM32WL55JC — микроконтроллер со встроенным радиомодемом LoRa
- ▶ nRF52832 — микроконтроллер с Bluetooth Low Energy
- ▶ TI CC2650 — микроконтроллер с поддержкой IEEE 802.15.4
- ▶ SiFive FE310 — RISC-V начального уровня

Пробуем программировать микроконтроллер

Документация на микроконтроллер

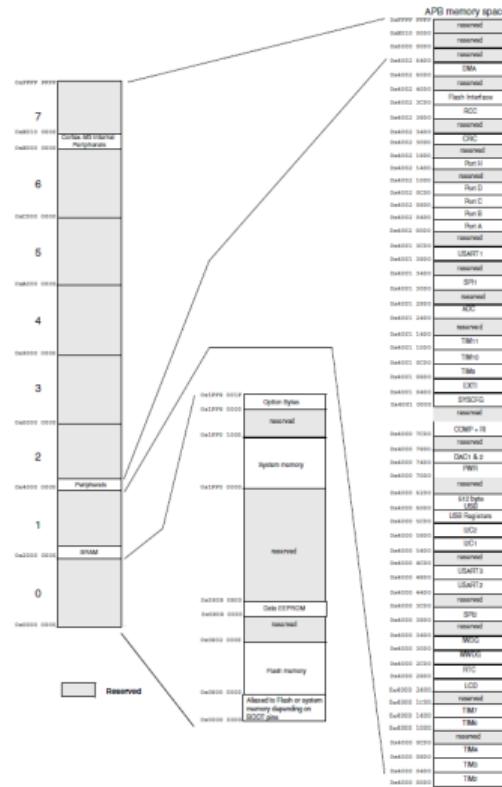
- ▶ **Datasheet** — описание в основном «электрических» параметров, 100–200 страниц
- ▶ **Reference Manual** — руководство программиста, порядка 1000 страниц
- ▶ **Errata** — список ошибок в предыдущих двух документах

Регистры и адреса

- ▶ Вся периферия микроконтроллера управляется записью в память по определенным фиксированным адресам (**регистры** периферийных устройств микроконтроллера)
- ▶ О состоянии периферии можно узнать, читая соответствующие регистры из памяти

Memory map

- Общий размер адресного пространства у 32-битного ядра — 4 Гб
- В единое адресное пространство проецируются:
 - ROM
 - RAM
 - Регистры периферии
- Обращение к несуществующему адресу приводит к остановке работы микроконтроллера



Попробуем включить и выключить светодиод

- ▶ В Reference Manual находим описание регистров GPIO
- ▶ Чтобы включить вывод («пин») микроконтроллера «на выход», надо записать «01» в определенную пару битов регистра MODER
- ▶ Чтобы установить на выходе определенный логический уровень, надо записать «1» в соответствующий бит регистра BSRR
- ▶ Вопросы настройки тактирования и тому подобные пока опустим

Looks like it's back to pixels for you!



Getting an Arduino LED to Blink

And Then Losing Interest

O RLY?

@ThePracticalDev

Регистр MODER

7.4.1 GPIO port mode register (**GPIOx_MODER**) (x = A..H)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw										

Bits 2y:2y+1 **MODERy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O direction mode.

00: Input (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

Регистр BSRR

7.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..H)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy**: Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy**: Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

Первая попытка

```
*((uint32_t*) 0x40020000) = (1UL << (5*2));
while(1) {
    delay_ms(500);
    *((uint32_t*) 0x40020018) = (1UL << 5);
    delay_ms(500);
    *((uint32_t*) 0x40020018) = (1UL << (5 + 16));
}
```

- ▶ GPIOA находится по адресу 0x40020000
- ▶ Регистр MODER — первый регистр GPIO, смещение нулевое
- ▶ Регистр BSRR расположен со смещением 0x18, то есть по адресу 0x40020018

CMSIS

- ▶ Работать напрямую с регистрами неудобно и совершенно непереносимо между микроконтроллерами даже в пределах одного семейства
- ▶ Макросы CMSIS (**Cortex Microcontroller Software Interface Standard**) — набор определений, описывающих регистры микроконтроллера (их имена и адреса)
- ▶ Заголовочные файлы с определениями CMSIS обычно предоставляются производителем микроконтроллера
- ▶ Обычно код с использованием CMSIS переносим (с ограничениями) между микроконтроллерами одного семейства

То же самое с CMSIS

```
uint32_t tmpreg;
tmpreg = GPIOA->MODER;
tmpreg &= ~GPIO_MODER_MODER5;
tmpreg |= GPIO_MODER_MODER5_0;
GPIOA->MODER = tmpreg;
while(1) {
    delay_ms(500);
    GPIOA->BSRR = GPIO_BSRR_BS_5;
    delay_ms(500);
    GPIOA->BSRR = GPIO_BSRR_BR_5;
}
```

Hardware Abstraction Layer

- ▶ Для переносимости кода между разными микроконтроллерами необходимы дальнейшие уровни абстракции
 - ▶ STM32 SPL, HAL, LL
 - ▶ TI driverlib, Simplelink
 - ▶ libopencm3
 - ▶ И даже Arduino
- ▶ С помощью этих средств можно обеспечить переносимость исходного кода даже между разными семействами микроконтроллеров разных производителей

То же самое с libopencm3

```
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/stm32/gpio.h>

int main(void){
    rcc_clock_setup_pll(
        &rcc_clock_config[RCC_CLOCK_VRANGE1_HSI_PLL_32MHZ]);
    rcc_periph_clock_enable(RCC_GPIOA);
    gpio_mode_setup(GPIOA, GPIO_MODE_OUTPUT, GPIO_PUPD_NONE, GPIO5);
    while(1) {
        delay_ms(500);
        gpio_set(GPIOA, GPIO5);
        delay_ms(500);
        gpio_clear(GPIOA, GPIO5);
    }
}
```

Операционные системы для микроконтроллеров

Операционные системы реального времени

Операционные системы реального времени

- ▶ **Расширенная машина:** унифицированный HAL, переносимость между различными семействами и архитектурами МК
- ▶ **Менеджер ресурсов:** управление процессорным временем, доступом к памяти и периферии

Операционные системы реального времени

- ▶ **Расширенная машина:** унифицированный HAL, переносимость между различными семействами и архитектурами МК
- ▶ **Менеджер ресурсов:** управление процессорным временем, доступом к памяти и периферии
- ▶ **Реальное время:** гарантированное время реакции на события



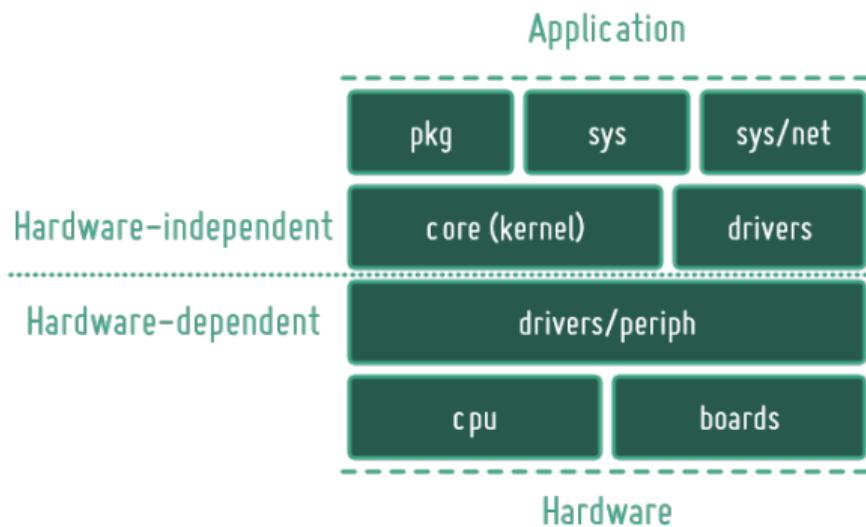
ARM mbed

IoT Device Platform

ARM KEIL®

Microcontroller Tools

Операционная система Riot



- ▶ Открытый исходный код, LGPL
- ▶ Разработана INRIA, FU Berlin и HAW Hamburg
- ▶ Ядро ОС
- ▶ Поддержка различных архитектур (ARM, RISC-V, AVR, ...)
- ▶ Поддержка различных семейств микроконтроллеров
- ▶ Драйверы внешних устройств
- ▶ Сетевой стек GNRC
- ▶ Поддержка программных пакетов сторонних разработчиков

Пример простейшей программы для Riot

```
#include <stdio.h>

#include "board.h"
#include "periph/gpio.h"
#include "xtimer.h"

int main(void) {
    puts("Hello, world!");
    gpio_init(LED0_PIN, GPIO_OUT);
    while(1) {
        xtimer_msleep(500);
        gpio_set(LED0_PIN);
        xtimer_msleep(500);
        gpio_clear(LED0_PIN);
    }
    return 0;
}
```

Что дальше?



На практическом занятии

- ▶ Как скомпилировать эту программу?
- ▶ Как загрузить ее в микроконтроллер?
- ▶ «Нулевой» блок заданий:
 - ▶ Установка среды сборки для Riot
 - ▶ Настройка VSCode в качестве IDE
 - ▶ Опрос стоимостью 1 балл