Вбудована розробка для мікроконтролерів

або

EMBEDDED

Палій Святослав

Український Католицький Університет

APPLIED SCIENCES FACULTY

Мета курсу

- Поєднати знання трьох дисциплін

  1.Організація ЕОМ
  2.Програмування
  3.Базова електроніка

- Опанувати взаємодію цифрових систем з реальним світом

- Навчитися якісно проектувати вбудовані рішення

Структура курсу

15 лекцій

1. **Вступне слово. Архітектура вбудованих систем**
2. **Засоби та інструменти для проектування.**
3. **Пам'ять та периферія мікропроцесора та мікроконтролера**
4. **Переривання**
5. **Цифрова комунікація.**
6. **Таймери**
7. **Аналогова периферія**
8. **Особливості роботи з реальним світом.**
9. **Особливості роботи з реальним світом (прод.)**
10. **Силова периферія**
11. **Розширена та конфігурована периферія**
12. **Прямий доступ до пам’яті (DMA)**
13. **Ефективний програмний код**
14. **Галузеві стандарти**
15. **Сучасні підходи до embedded**

Структура курсу

8 практичних завдань

1. **Базовий ввід/вивід**
2. **Аналоговий та розширений ввід/вивід**
3. **Ввід/вивід на апаратному рівні**
4. **Процеси реального часу**
5. **Інтерфейси та комунікації в деталях**
6. **Комплексний проект з паралельними сценаріями обробки**
7. **Графічний інтерфейс користувача, використання готових бібліотек**
8. **Статичний аналіз коду**

Оцінювання

1. **Практичні роботи – 30 балів**
    - **4 бали за виконане практичне завдання.**
    - **Завдання здане після дедлайну враховується з коефіцієнтом 0.5**
2. **Проект вільного вибору – 30 балів**

    Проект може бути спільний з ПОК
3. **Мідтерм – 20 балів**
4. **Екзамен – 20 балів**

Основи вбудованих систем

Визначення та характеристики вбудованих систем

Визначення вбудованих систем

Вбудовані системи — це комп'ютерні системи, інтегровані в інші пристрої для виконання спеціалізованих завдань.

Обмежені ресурси

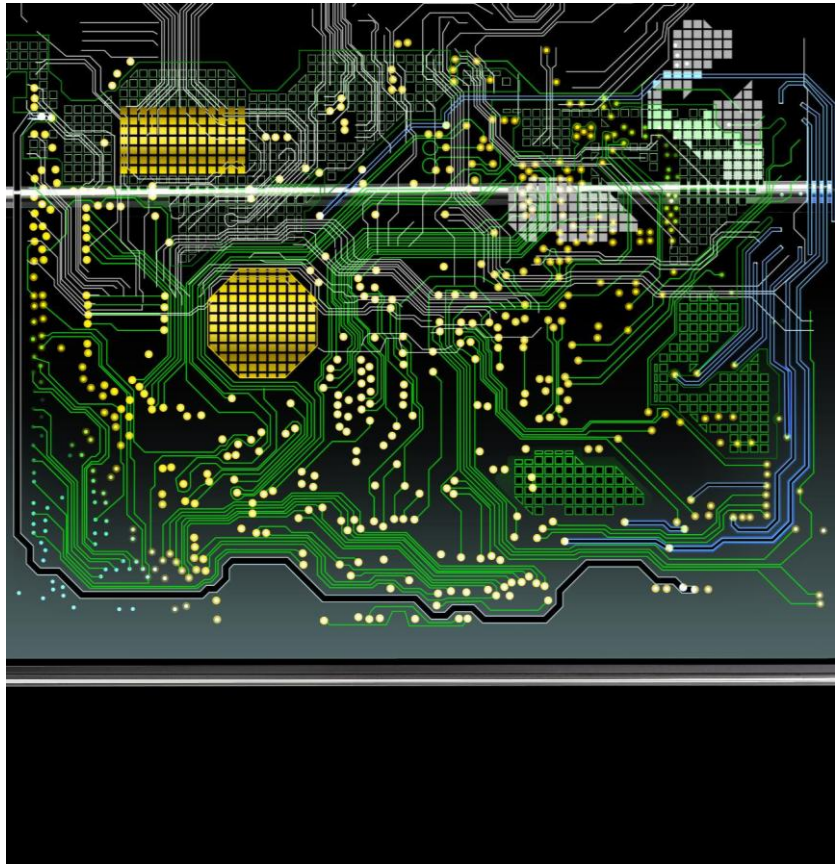
Вбудовані системи зазвичай мають обмежену продуктивність, енергоспоживання та пам'ять, що впливає на їх здатність виконувати складні завдання.

Енергоефективність

Важливою характеристикою вбудованих систем є енергоефективність для забезпечення тривалої роботи пристроїв, особливо тих що живляться від батарей.

Надійність та реальний час

Вбудовані системи повинні бути надійними і мати специфічні вимоги до реального часу для успішного виконання поставлених на них завдань.



Приклади вбудованих систем

Побутові електронні пристрої

Вбудовані системи використовуються в побутових електронних пристроях, таких як пральні машини, мікрохвильові печі та холодильники, і навіть новітні USB кабелі.

Автомобільні системи управління

Сучасні автомобілі оснащені вбудованими системами, які контролюють управління, безпеку та комфорт під час водіння.

Медичні пристрої

Пристрої призначені для діагностики та допомоги лікарю у лікуванні. Носимі та імплантовані портативні медичні пристрої.

Промислова автоматика

Космос та інше.

Безпека та надійність у вбудованих системах

Критичність безпеки

Безпека є критично важливою в вбудованих системах, особливо в автомобільній та медичній сферах, де помилки можуть мати серйозні наслідки.

Вразливості в проектуванні

На етапі проектування необхідно враховувати потенційні загрози та вразливості, щоб забезпечити надійність вбудованих систем.

Протидія загрозам та злому

Ефективні стратегії протидії загрозам необхідні для захисту вбудованих систем від зовнішніх та внутрішніх ризиків.

Компоненти вбудованих систем

Мікроконтролери

Мікроконтролери а деколи і мікропроцесори є основою вбудованих систем, які контролюють інші компоненти та виконують програми.

Пам'ять

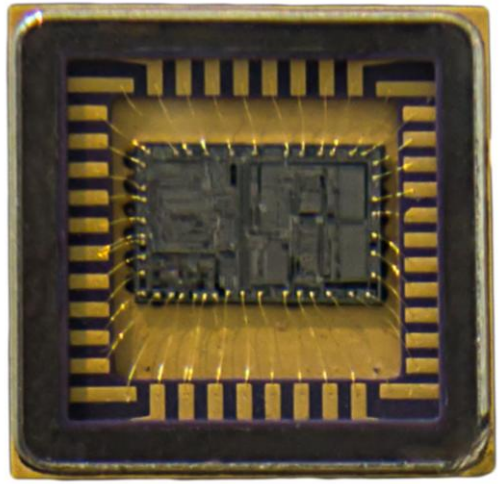
Пам'ять зберігає дані та програми, необхідні для роботи вбудованих систем, забезпечуючи швидкий доступ до інформації.

Давачі

Давачі вимірюють фізичні величини, такі як температура, тиск, світло, напруга відстань, звук та інше надаючи дані для подальшої обробки.

Виконавчі механізми

Виконавчі механізми виконують фізичні дії, такі як рух або активація, на основі команд від мікроконтролера.



Мікроконтролер чи мікропроцесор

Мікропроцесор (CPU) - це серце загально-цільових комп'ютерів, призначене для високопродуктивних завдань з великою кількістю обчислень.

Мікроконтролер (MCU) - це універсальний інструмент для вбудованих систем, що потребують меншої потужності та виконують конкретні завдання ефективно.

CPU

- **Ціль:** Призначений для загальних обчислень, виконує широкий спектр завдань.
- **Комплексність:** Зазвичай більш потужний, зазвичай має кілька ядер, високу тактову частоту, багато кеш-пам'яті.
- **Використання:** Використовується у персональних комп'ютерах, ноутбуках, серверах, смартфонах.
- **Периферійні пристрої:** Вимагає наявності зовнішніх компонентів для управління пам'яттю, введення/виведення та інших задач.
- **Енерговитрати:** Зазвичай вищі, оскільки обробляє складніші завдання.

MCU

- **Ціль:** Призначений для вбудованих систем, виконує конкретні завдання у керованих середовищах.
- **Комплексність:** Менш потужний, але інтегрує процесор, пам'ять та периферійні пристрої на одному чіпі.
- **Використання:** Використовується у різних пристроях, IoT, гаджети, побутова техніка, автомобілі, тощо.
- **Периферійні пристрої:** Має вбудовану оперативну пам'ять, флеш-пам'ять, контролери введення/виведення, аналогову периферію, що робить його більш автономним.
- **Енерговитрати:** Зазвичай нижчі, оскільки виконує спеціалізовані завдання з меншою складністю.

Мікроконтролер + мікропроцесор

В багатьох достатньо складних системах поєднується мікропроцесор котрий забезпечує обчислювальну потужність та один чи декілька мікроконтролерів, котрі забезпечують взаємодію з навколишнім світом.

Розрядність шини та регістрів

- Мікроконтролери бувають різної розрядності, і їх можна класифікувати за розрядністю шини даних та адресної шини. Ось основні типи:
- **8-розрядні мікроконтролери:** Ці мікроконтролери мають 8-бітну шину даних і можуть обробляти 8-бітні дані за один цикл. Приклади включають серії мікроконтролерів AVR (наприклад, ATmega328) та PIC.
- **16-розрядні мікроконтролери:** Вони мають 16-бітну шину даних і можуть обробляти 16-бітні дані. Ці мікроконтролери забезпечують більшу продуктивність порівняно з 8-бітними моделями. Прикладом є серія MSP430 від Texas Instruments.
- **32-розрядні мікроконтролери:** Мають 32-бітну шину даних і можуть обробляти 32-бітні дані. Ці мікроконтролери використовуються для складніших завдань і забезпечують високу продуктивність. Приклади включають серії ARM Cortex-M.
- **64-розрядні мікроконтролери:** Вони менш поширені, але використовуються в специфічних областях, де потрібна висока обчислювальна потужність і велика адресна пам'ять.

А що це значить для програміста

```
int main()
{

  // Вичитуємо значення АЦП
  uint8_t levelLo = ADC_LO_REG;
  uint8_t levelHi = ADC_HI_REG;


  uint16_t value = 256*levelHi + levelLo;
  printf("Значення: %d\n", value);
}
```

А що це значить для програміста

```
int main()
{

  // Вичитуємо значення АЦП
  uint8_t levelLo = ADC_LO_REG;
  uint8_t levelHi = ADC_HI_REG;


  uint16_t value = 256*levelHi + levelLo;
  printf("Значення: %d\n", value);
}
```

Якщо значення АЦП коливаються між 0x0100 та 0x00FF ми ризикуємо зчитати в момент коли HI частина буде від одного значення а LO від іншого, наприклад 0x01FF чи 0x0000, що буде геть неправильним значенням.

Потрібно вживати додаткових заходів що гарантують «атомарне» зчитування на апаратному чи програмному рівнях.

Архітектура мікроконтролерів

- **Harvard Architecture**
  - **Опис:** Використовує окремі шини для команд і даних.
  - **Переваги:** Вища швидкість виконання завдань, оскільки команди та дані можуть передаватися паралельно.
  - **Приклади:** AVR (наприклад, мікроконтролери Atmel), PIC мікроконтролери.
- **Von Neumann Architecture**
  - **Опис:** Використовує одну шину для команд і даних.
  - **Переваги:** Простота у проектуванні та розробці.
  - **Недоліки:** Можливе зниження продуктивності через "вузьке горло" при доступі до пам'яті.
  - **Приклади:** Частина мікроконтролерів ARM, наприклад ARM Cortex-M0.
- **Модифікована Harvard Architecture**
  - **Опис:** Пам’ять одна, а шини для команд і даних різні.
  - **Переваги:** Намагання поєднати позитивні сторони двох інших архітектур
  - **Приклади:** Частина мікроконтролерів ARM, наприклад ARM Cortex-M3.

А що це значить для програміста?

```
int main()
{
  // Приклад адреси пам'яті
  unsigned int address = 0x1000;

  // Вказівник на адресу
  int *ptr = (int*)address;

  // Читання значення з пам'яті за адресою
  int value = *ptr;
  printf("Значення за адресою 0x%X: %d\n", address, value);

  // Запис значення в пам'ять за адресою
  *ptr = 42;
}
```

А це адреса пам'яті програм чи даних? А так?

```
const int *ptr = (int*)address;
```

Навіть у Von Neumann так записати до Flash пам'яті неможливо

Endiannes

Endianness визначає порядок байтів у слові даних, і вона може бути двох видів: Big Endian та Little Endian.

Little Endian

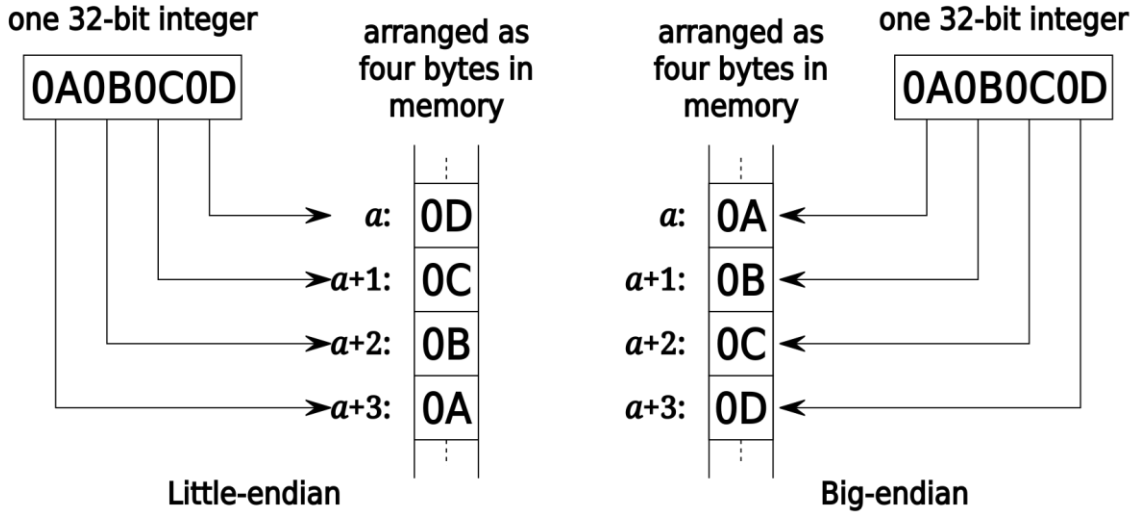
У Little Endian порядку молодший байт (менш значущий байт, LSB) записується першим, або за молодшою адресою, а старший байт (більш значущий байт, MSB) останнім. Це означає, що байти зберігаються у зворотному порядку відносно того, як ми зазвичай читаємо числа.

Наприклад при збереженні числа 0x12345678 у пам'ять у порядку Little Endian, байти будуть розташовані таким чином: 0x78 0x56 0x34 0x12.

Endiannes

- **Big Endian**

У Big Endian порядку старший байт записується першим, а молодший байт — останнім. Прикладом може бути 0x12345678, де байти будуть розташовані як 0x12 0x34 0x56 0x78.



А що це значить для програміста,

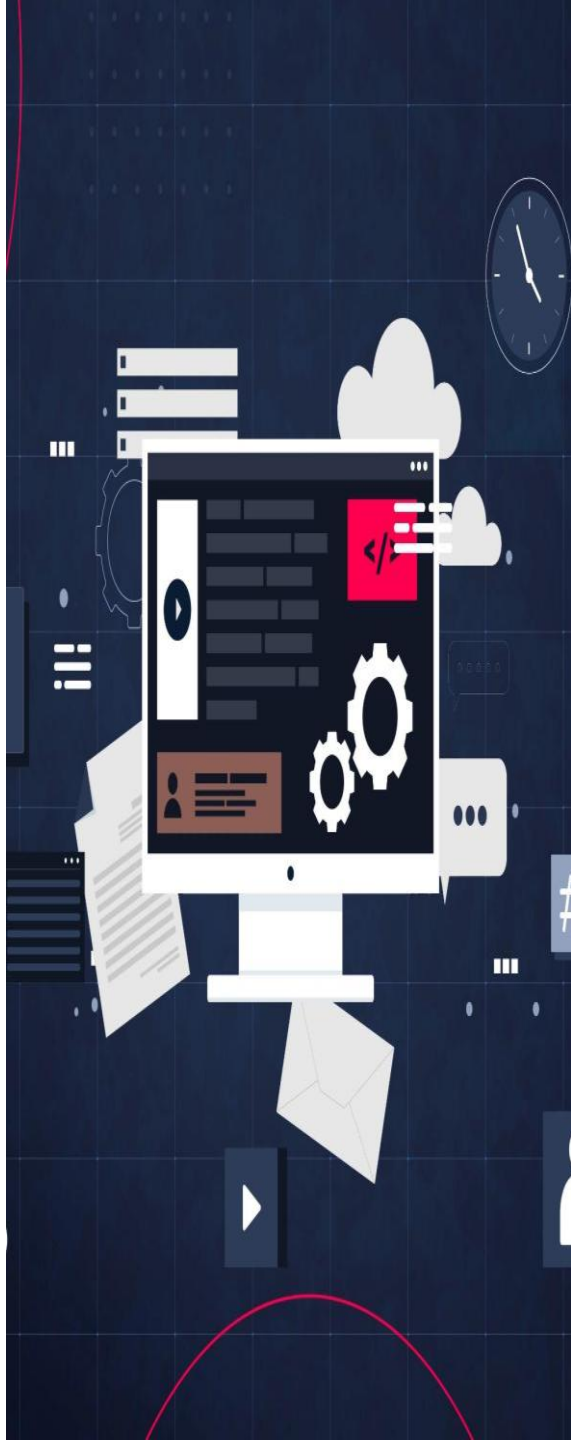
```
int main()
{
  unsigned int value = 0x12345678;
  unsigned char *ptr = (unsigned char*)&value;

  printf("Порядок байтів:\n");
  for (int i = 0; i < sizeof(value); ++i)
  {
    printf("%02x ", ptr[i]);
  }

  return 0
}
```

Мови програмування в embedded

- C
- C++
- Rust
- Інші менш популярні в промисловому сегменті, наприклад microPython, Lua



C/C++

- **Історія мови програмування C:**
- **1969-1973**: Розробка C була розпочата Денісом Рітчі (Dennis Ritchie) у Bell Labs. Мова спочатку була створена для операційної системи Unix і базувалася на мові B, яка, в свою чергу, походила від мови BCPL.
- **1978**: Опублікована книга "The C Programming Language" авторів Брайана Кернігана (Brian Kernighan) і Деніс Рітчі, яка стала відомою як "K&R C". Ця книга зробила мову C популярною серед програмістів.
- **1983**: ANSI (Американський національний інститут стандартів) сформував комітет для стандартизації мови C, щоб уникнути її різних реалізацій.
- **1989**: Був прийнятий перший стандарт мови C під назвою ANSI C або C89.
- **1990**: Міжнародна організація зі стандартизації (ISO) прийняла ANSI C зі своїми коригуваннями під назвою C90.
- **1999**: Був прийнятий новий стандарт під назвою C99, який додав багато нових функцій, таких як змінні довжини масивів, нові типи даних і покращені можливості вводу/виводу.
- **2011**: З'явився стандарт C11, який додав багатопоточність, нові функції та покращення продуктивності.
- **2018**: Прийнятий стандарт C18, який включає технічні коригування і покращення для попереднього стандарту C11.

- **Історія мови програмування C++:**
- **1979**: Б'ярне Страуструп (Bjarne Stroustrup) у Bell Labs почав розробку C++, щоб додати об'єктно-орієнтовані можливості до мови C. Спочатку мова називалася "C with Classes".
- **1983**: Мова отримала назву "C++", де "++" є оператором інкремента у мові C, що символізує вдосконалення мови.
- **1985**: Вийшла перша комерційна версія мови C++, а також перша книга Б'ярне Страуструпа "The C++ Programming Language".
- **1990**: З'явилися численні компілятори для C++, що сприяло його поширенню.
- **1998**: ISO прийняла перший стандарт мови C++, відомий як C++98.
- **2003**: З'явився стандарт C++03, який включав деякі технічні виправлення та уточнення до C++98.
- **2011**: Прийнятий стандарт C++11, який додав багато нових можливостей, таких як лямбда-вирази, нові типи даних, діапазонні цикли та інші покращення.
- **2014**: Прийнятий стандарт C++14, який включає подальші покращення.
- **2017**: Прийнятий стандарт C++17, який додає ще більше можливостей.
- **2020**: Прийнятий стандарт C++20, який включає значні вдосконалення, такі як концепти, модули та корутини.

Rust

Історія мови програмування Rust:

- **2006**: Розробка Rust почалася як хобі-проект Грейдона Хоара (Graydon Hoare) у компанії Mozilla.
- **2010**: Проект був представлений публіці, і Mozilla офіційно почала підтримувати розробку Rust.
- **2012**: Був випущений перший прототип компілятора Rust під назвою "rustc".
- **2015**: Випущена перша стабільна версія Rust 1.0, яка отримала позитивні відгуки за свою безпеку пам'яті та продуктивність.
- **З 2015 року**: Rust продовжує розвиватися, і кожні шість тижнів випускаються нові версії з покращеннями та новими можливостями.
- **2021**: Rust обраний основною мовою для деяких частин ядра операційної системи Linux, що підкреслює його надійність та прийнятність для системного програмування.

Модулі та бібліотеки

Спрощення розробки

Модулі та бібліотеки спрощують розробку програмного забезпечення, надаючи готові рішення для повторюваних завдань.

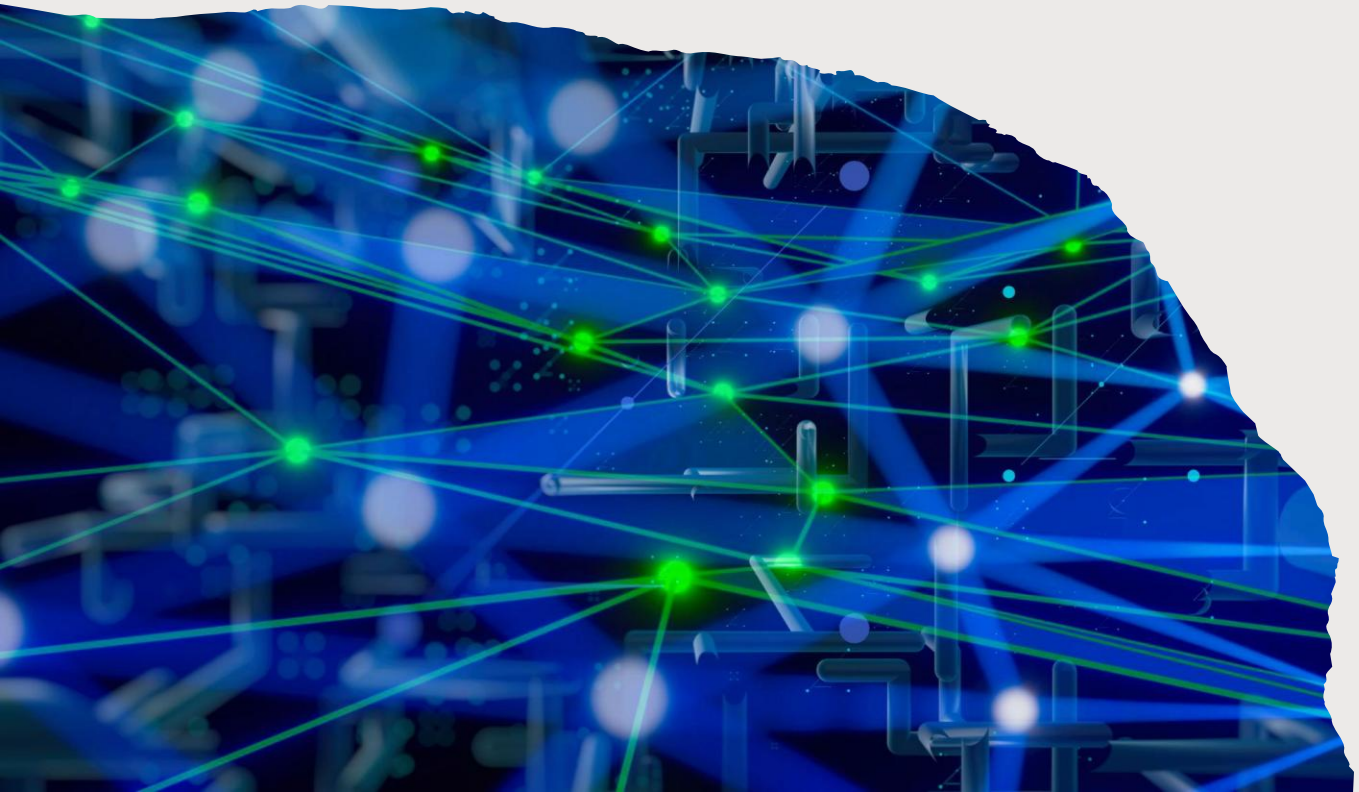
Зосередження на функціях

Використання модулів дозволяє розробникам зосередитися на специфічних функціях системи, підвищуючи ефективність.

Прискорення процесу

Завдяки бібліотекам розробка програмного забезпечення стає швидшою, що дозволяє зменшити час виходу на ринок.

Інтернет речей (IoT)



Взаємодія пристроїв

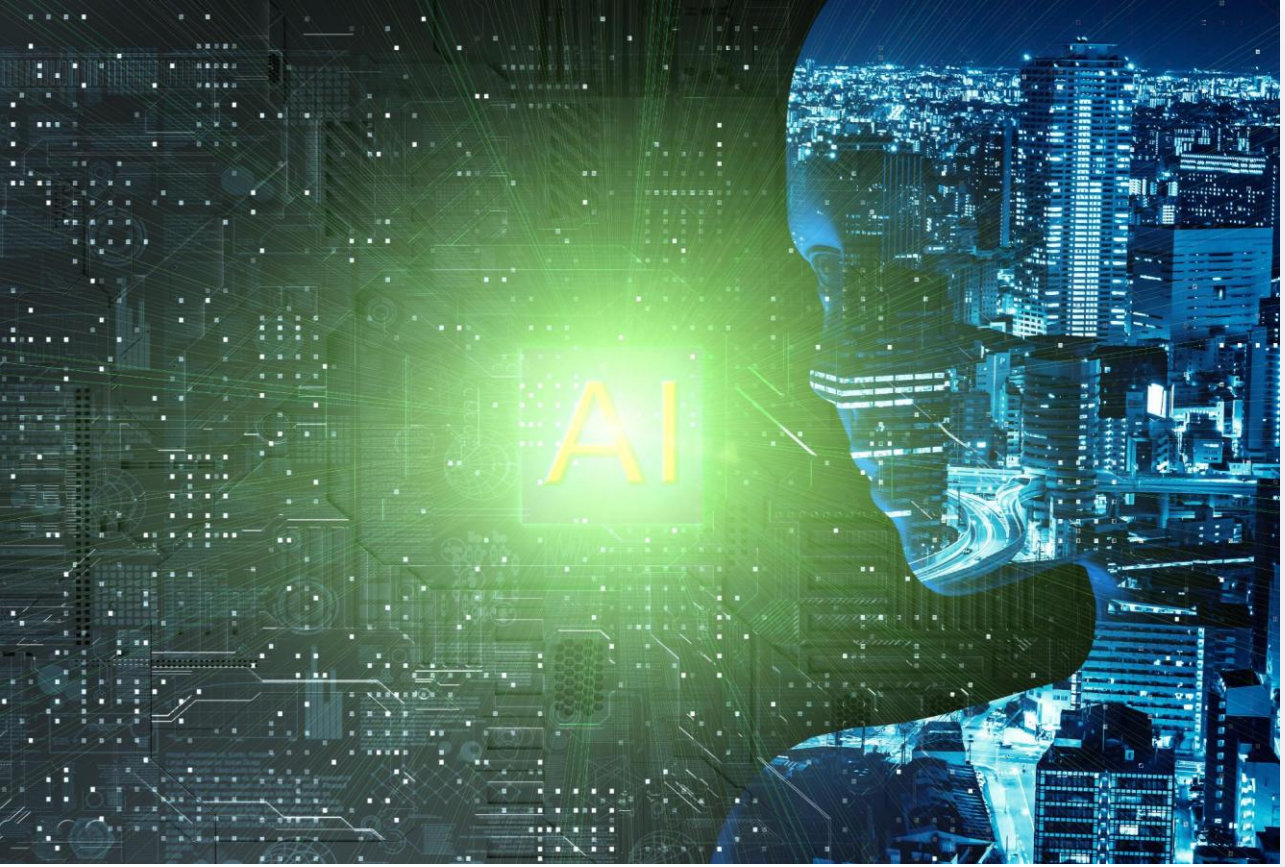
IoT дозволяє пристроям взаємодіяти через Інтернет, що веде до нових можливостей для автоматизації процесів.

Автоматизація процесів

IoT відкриває нові шляхи для автоматизації, що підвищує ефективність і знижує витрати в різних галузях.

Безпека систем

З IoT виникають нові вимоги до розробки безпечних систем для захисту даних та приватності користувачів.



Штучний інтелект та машинне навчання

Адаптація вбудованих систем

Штучний інтелект дозволяє вбудованим системам адаптуватися до змін у навколишньому середовищі для покращення їх роботи.

Підвищення ефективності

Впровадження машинного навчання в системи значно підвищує їх ефективність і продуктивність в різних застосуваннях.

Інноваційні можливості

Інтеграція штучного інтелекту відкриває нові горизонти для інновацій у багатьох галузях, змінюючи спосіб функціонування технологій.

MCU MARKET – 2023 MARKET REPARTITION

Medical
(Consumer medical, medical imaging, instruments, etc.)

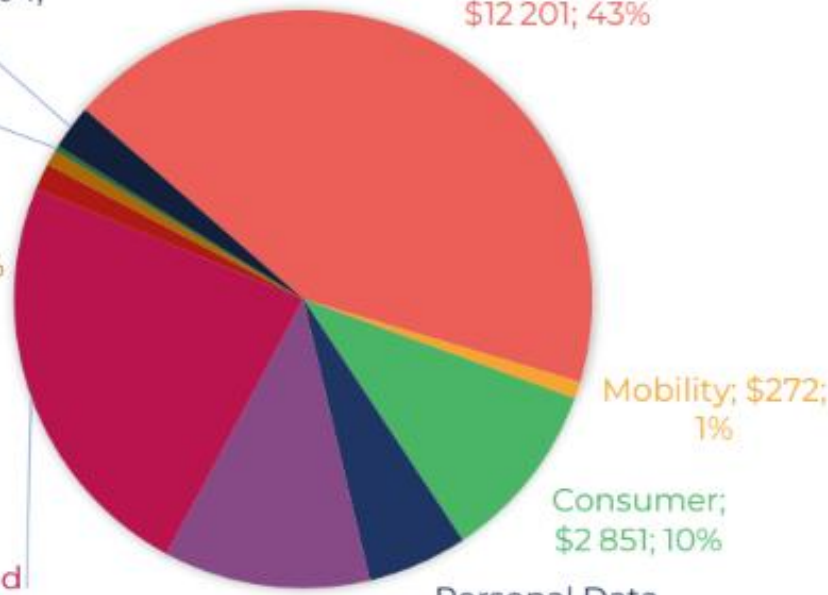
Defense & aerospace
(Installations, munitions, unmanned vehicles, etc.)

Telecommunications infrastructure
(Carrier infrastructure, enterprise and gov't networking, etc.)

Enterprise data processing
(Enterprise & gov't compute, enterprise & gov't office equipment)

Industrial and other
(Agriculture, logistics, factory automation, mining, etc.)

Smartcard & embedded security
(Smartcard, embedded security)



Automotive and mobility 44%

Automotive
(Personal car, light commercial vehicles)

Mobility
(Bus, trucks, maritime, train, etc.)

Mobile and consumer 16%

Consumer
(Smart home, set-top box, TV, fitness wearables, etc.)

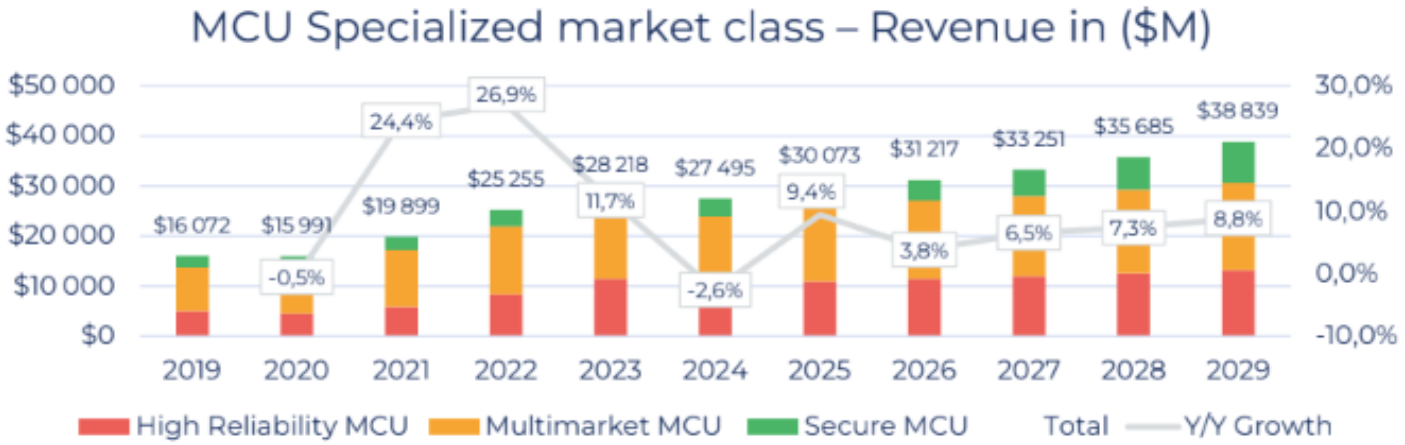
Personal data processing
(Desktop PC, media tablet, mobile phone, etc.)

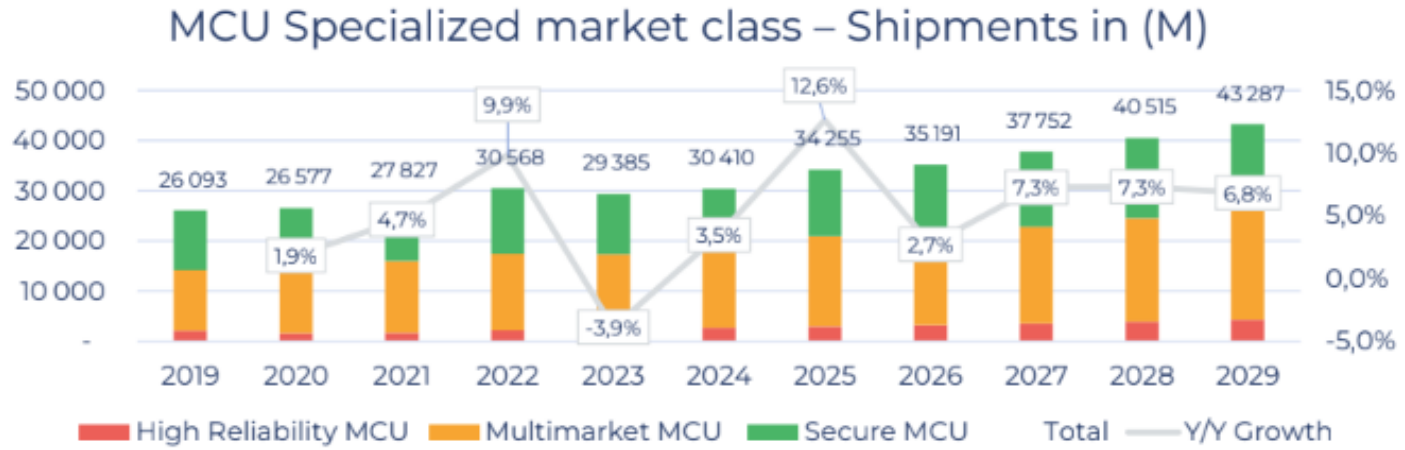
MCU REVENUE AND SHIPMENTS FORECAST

Specialized market-class trends

While bit class is mainly about balancing efficient performance vs. cost/energy/area, the specialized market classes have more to do with production techniques and meeting either specific packaging and longevity standards (such as high-reliability) or specialized security protocols for protecting user and system data and encrypting communications. Multimarket represents a base-level solution appropriate to the widest variety of general purpose or similar application-specific designs.

- Multimarket-class MCUs are by nature general purpose and gain economies of scale by servicing a wide variety of markets that have similar performance and interconnect demands. This market is expected to remain the dominant MCU specialization class far into the future.
- With the evolution in the automotive industry towards more central domain control for ADAS, telematics, and electrification, high-reliability MCUs are especially gaining in revenue due to gains in ASP. High reliability MCUs grew in early 2023 with strong demand for automotive safety solutions, but by early 2024, the automotive market has hit a slump, especially in EV sales in and around China. The expectation is that high Interest rates, a large barrier to big ticket sales, will start coming back down as early as 2025 and release pent up demand for large items that rely on specialized MCUs such as Automotive, Mobility and some Industrial applications.
- Secure MCUs are transitioning from a market dominated by low-cost smartcard ICs to more price- and performance-rich embedded secure MCUs. This transition is one of the strongest revenue drivers for MCUs overall and will continue for some time driving double digit growth for Secure MCU class revenue from 2023 to 2029.





MCU REVENUE AND SHIPMENTS FORECAST

Bit class trends

The 4/8-bit and 16-bit classes did not benefit from the surge in late 2022, early 2024 like the 32-bit class. The rapid growth of automotive and secure MCU applications, two major MCU markets had a great deal to do with the rise of 32-but. We predict the demand for 4/8-bit see modest growth opportunities due to demand for energy efficiency 16-bit will remain flat. New applications for MCUs will target 32-bit applications creating larger growth for the 32-bit market steadily through 2029.

- The product-mix growth that is bolstering ASP is also tied to the growing demand for 32-bit MCUs over that of 4/8-but and 16-bit. Much of the growth in automotive, for example, is in 32-bit and increased safety, security, and more demanding applications. Embedded Secure MCUs are the decline of smartcard MCU demand is largely product substitution driver of 32-bit MCUs.
- While the market for 16-bit shipments is forecasted relatively flat, growth continues to favor both cost sensitive 4/8-bit markets and 32-bit high performance and high reliability markets.. Many proprietary and general-purpose product lines are experiencing strong competition from low-cost 32-bit ARM and emerging 32-bit RISC-V solutions.
- The role of 16-bit MCUs is somewhat niche. There are several systems that need the mathematical proficiency of a 16-bit MCU to run timers, simple LCD displays and/or user interfaces, security and other similar applications, but still need low-cost and low-power that may put 32-bit solutions slightly out of range.
- Still, despite hype that 4/8/16-bit is being replaced, 4/8-bit markets are still growing strong. 16-bit remains flat. These markets are attractive for applications requiring simple programming, quick time-to-market, IP reuse, low-cost and low-power solutions. These are all key characteristics for why MCUs are so successful as a solution in deeply embedded applications.