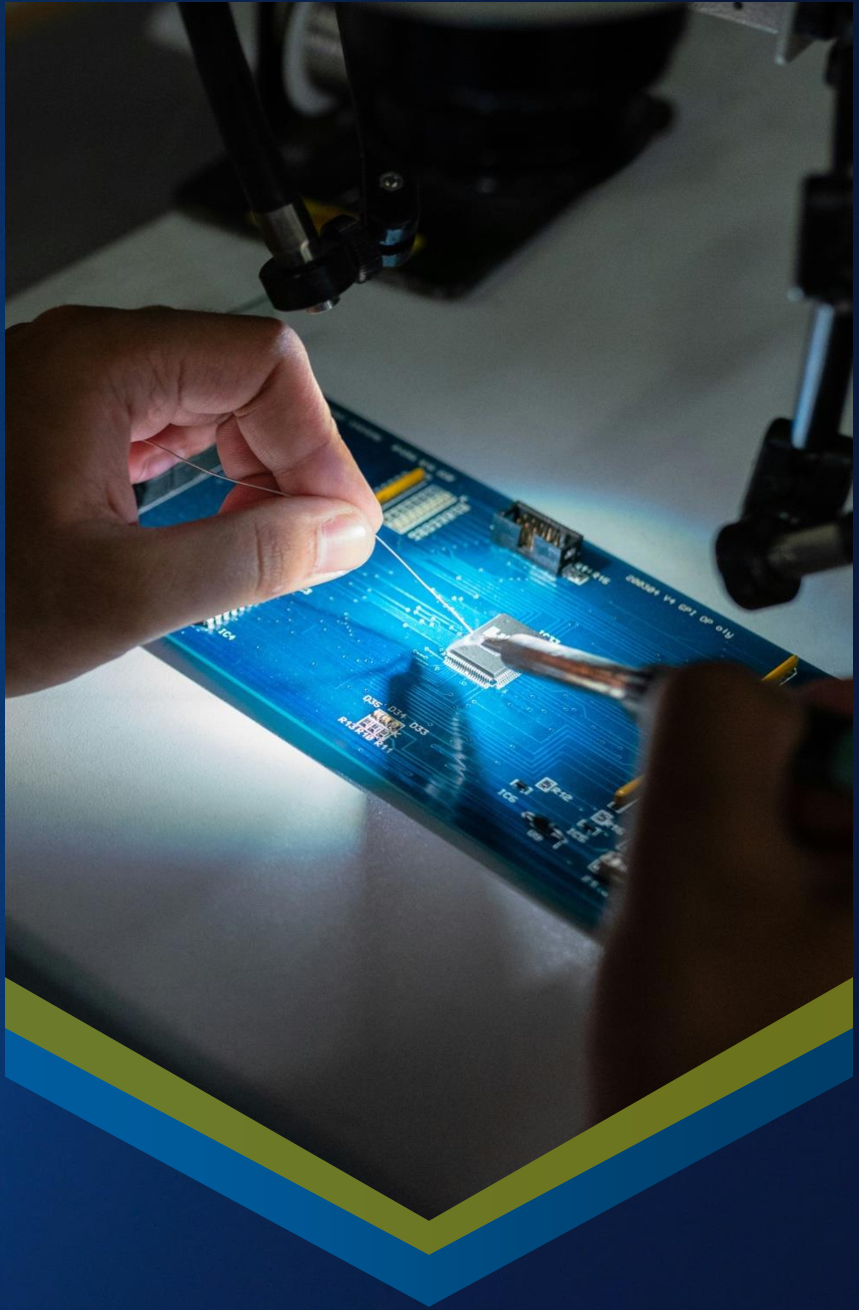
IOT & EMBEDDED SYSTEMS

COMPETENCE CENTER

Mocking the UART in C++: A Hands-On Approach



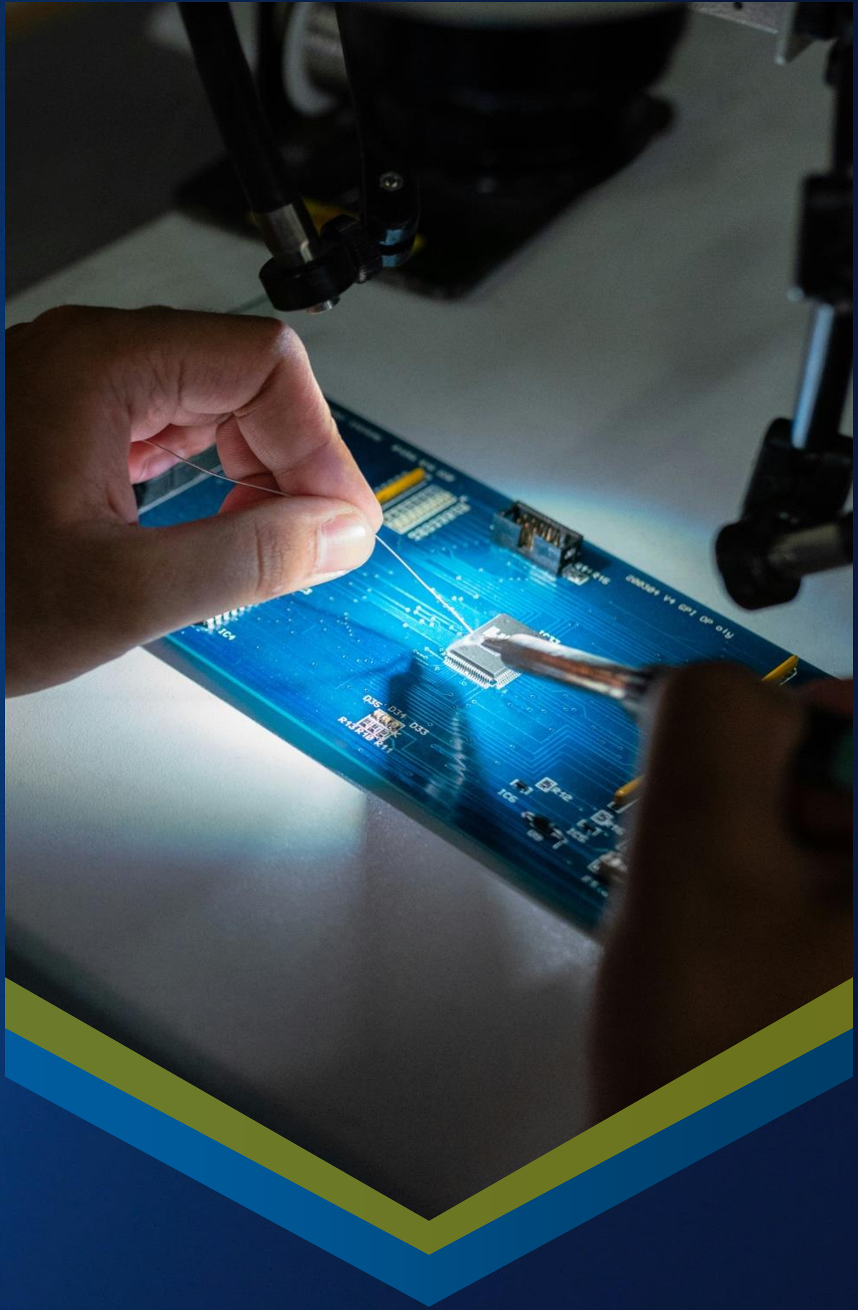
AGENDA

1 Get to know each other

2 KAS, Yocto, QEMU, Risc-V

3 C++ immediate functions

4 Interactive demo



AGENDA

1 Get to know each other

2 KAS, Yocto, QEMU, Risc-V

3 C++ immediate functions

4 Interactive demo

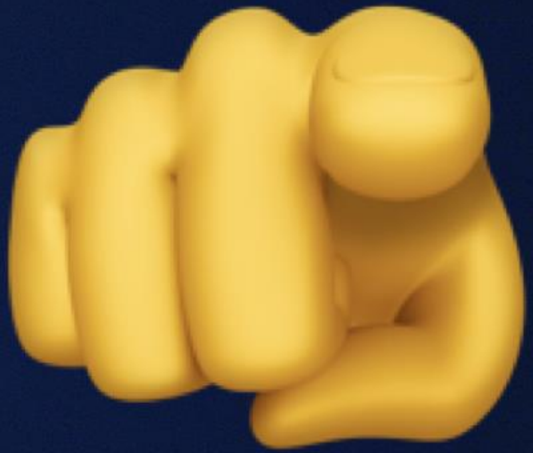
WHO AM I?

STEFANO FIORENTINO

- Head of Competence Center IoT & Embedded Systems
- Co-Founder at Italian Embedded
- Host at User Group C++ Lugano
- Lightning talks at EOC, ECC and it++
- C++ Standards Committee Member

I’M SORRY

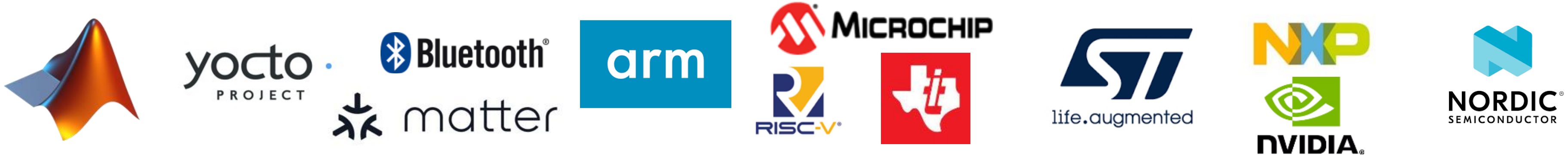
STEFANO FIORENTINO



ICE BREAKER – RAISE YOUR HAND IF YOU KNOW AT LEAST 3 LOGOS

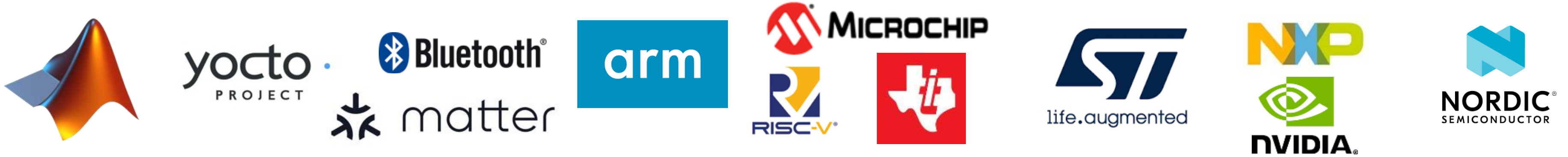
ICE BREAKER – RAISE YOUR HAND IF YOU KNOW AT LEAST 3 LOGOS

Design & prototiping

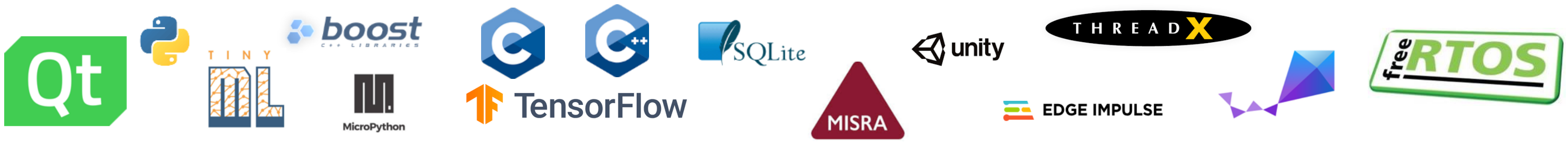


ICE BREAKER – RAISE YOUR HAND IF YOU KNOW AT LEAST 3 LOGOS

Design & prototyping



Development



ICE BREAKER – RAISE YOUR HAND IF YOU KNOW AT LEAST 3 LOGOS

Design & prototyping

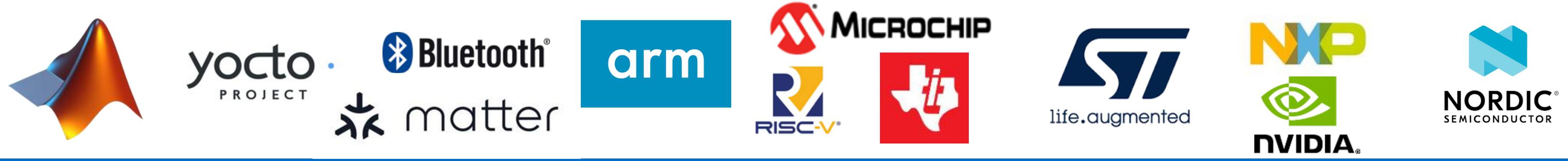
yocto PROJECT
Bluetooth®
matter
arm
MICROCHIP
RISC-V®
life.augmented
NXP
NVIDIA.
NORDIC® SEMICONDUCTOR

Development

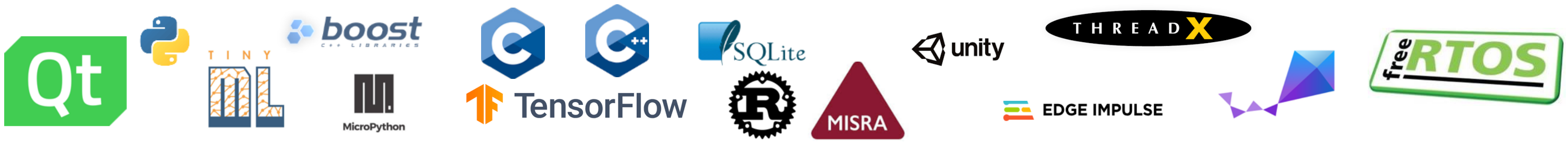
Qt
TINY ML
boost C++ LIBRARIES
MicroPython
TensorFlow
SQLite
MISRA
unity
THREADX
EDGE IMPULSE
free RTOS

ICE BREAKER – RAISE YOUR HAND IF YOU KNOW AT LEAST 3 LOGOS

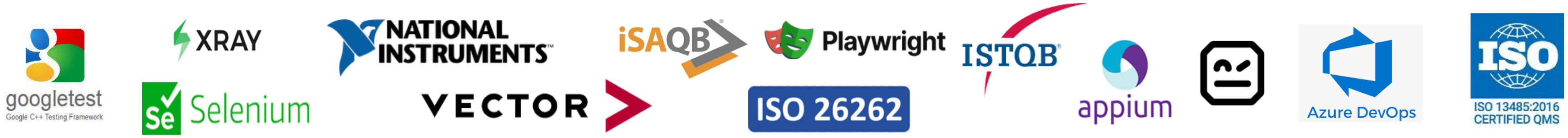
Design & prototiping



Development



Testing & Quality



ICE BREAKER – RAISE YOUR HAND IF YOU KNOW AT LEAST 3 LOGOS

Design & prototiping

yocto PROJECT
Bluetooth®
matter
arm
MICROCHIP
RISC-V®
life.augmented
NXP
NVIDIA.
NORDIC® SEMICONDUCTOR

Development

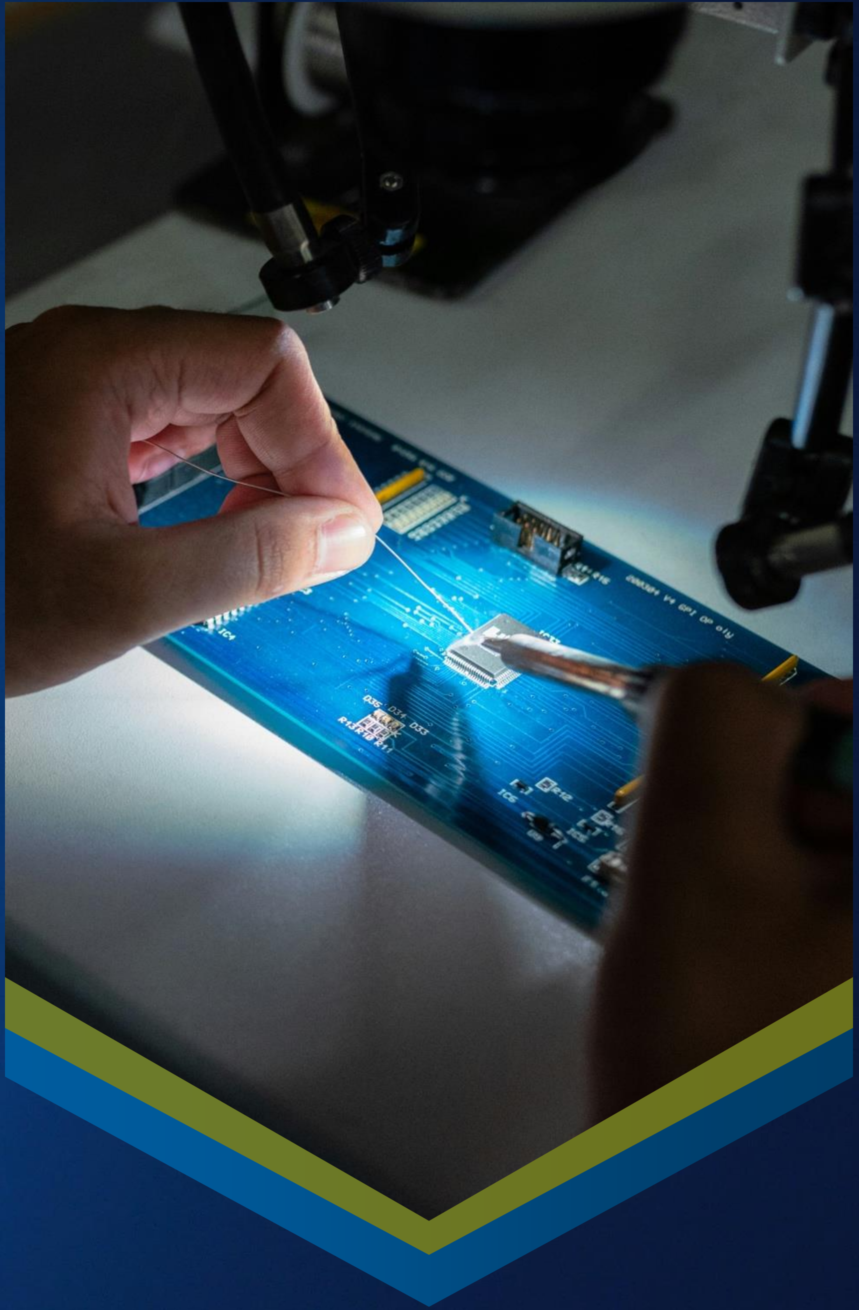
Qt
TINY ML
boost C++ LIBRARIES
MicroPython
C++
TensorFlow
SQLite
MISRA
unity
THREADX
EDGE IMPULSE
free RTOS

Testing & Quality

googletest
Google C++ Testing Framework
XRAY
Se Selenium
NATIONAL INSTRUMENTS™
VECTOR
iSAQB®
Playwright
ISO 26262
ISTQB®
appium
Azure DevOps
ISO
ISO 13485:2016 CERTIFIED QMS

Provisioning

Google Home
amazon alexa
aws
Azure
docker



AGENDA

1 Get to know each other

2 KAS, Yocto, QEMU, Risc-V

3 C++ immediate functions

4 Interactive demo

KAS, YOCTO, QEMU, RISC-V

OUR TOOLS FOR TODAY

KAS

YOCTO PROJECT

QEMU

RISC-V

KAS, YOCTO, QEMU, RISC-V

OUR TOOLS FOR TODAY

KAS

- KAS is a configuration and build automation tool primarily used with the Yocto Project to streamline the setup of complex embedded Linux builds.
- It simplifies the management of multiple layers and configurations by using a single YAML file to define repositories, layers, machine targets, and other build parameters.

YOCTO PROJECT

QEMU

RISC-V

KAS, YOCTO, QEMU, RISC-V

OUR TOOLS FOR TODAY

KAS

- > KAS is a configuration and build automation tool primarily used with the Yocto Project to streamline the setup of complex embedded Linux builds.
- > It simplifies the management of multiple layers and configurations by using a single YAML file to define repositories, layers, machine targets, and other build parameters.

YOCTO PROJECT

- > The Yocto Project is an open-source collaboration project that provides tools, templates, and methods to create custom Linux-based systems for embedded devices.
- > It uses a powerful build system called BitBake along with metadata (recipes and layers) to define how software components are fetched, configured, compiled, and packaged.

QEMU

RISC-V

KAS, YOCTO, QEMU, RISC-V

OUR TOOLS FOR TODAY

KAS

> KAS is a configuration and build automation tool primarily used with the Yocto Project to streamline the setup of complex embedded Linux builds.

> It simplifies the management of multiple layers and configurations by using a single YAML file to define repositories, layers, machine targets, and other build parameters.

YOCTO PROJECT

> The Yocto Project is an open-source collaboration project that provides tools, templates, and methods to create custom Linux-based systems for embedded devices.

> It uses a powerful build system called BitBake along with metadata (recipes and layers) to define how software components are fetched, configured, compiled, and packaged.

QEMU

> QEMU is an open-source machine emulator and virtualizer that allows users to run operating systems and applications for different hardware platforms on a host system.

> It supports full system emulation and user-mode emulation, making it useful for development, testing, and debugging without access to physical hardware.

RISC-V

KAS, YOCTO, QEMU, RISC-V

OUR TOOLS FOR TODAY

KAS

- KAS is a configuration and build automation tool primarily used with the Yocto Project to streamline the setup of complex embedded Linux builds.
- It simplifies the management of multiple layers and configurations by using a single YAML file to define repositories, layers, machine targets, and other build parameters.

YOCTO PROJECT

- The Yocto Project is an open-source collaboration project that provides tools, templates, and methods to create custom Linux-based systems for embedded devices.
- It uses a powerful build system called BitBake along with metadata (recipes and layers) to define how software components are fetched, configured, compiled, and packaged.

QEMU

- QEMU is an open-source machine emulator and virtualizer that allows users to run operating systems and applications for different hardware platforms on a host system.
- It supports full system emulation and user-mode emulation, making it useful for development, testing, and debugging without access to physical hardware.

RISC-V

- RISC-V is an open, royalty-free instruction set architecture (ISA) based on established reduced instruction set computing (RISC) principles.
- Its modular design enables customization, allowing developers to implement only the features needed for a specific application, making it ideal for everything from microcontrollers to high-performance CPUs.

KAS, YOCTO, QEMU, RISC-V

OUR TOOLS FOR TODAY

```
kas shell bare-metal.yml
devtool modify baremetal-helloworld
bitbake -c build baremetal-helloworld
runqemu nographic slirp
```

KAS SHELL BARE-METAL.YML

OUR TOOLS FOR TODAY

```yaml
header:
  version: 10

machine: qemuriscv32
distro: poky
target: baremetal-helloworld

repos:
  poky:
    url: https://git.yoctoproject.org/git/poky
    branch: kirkstone
    layers:
      meta:
      meta-poky:
      meta-yocto-bsp:
      meta-skeleton:

local_conf_header:
  meta-custom: |
    TMPDIR = "/opt/tmp"
    TCLIBC = "baremetal"
```

DEVTOOL MODIFY BAREMETAL-HELLOWORLD

OUR TOOLS FOR TODAY

```
BBLAYERS ?= " \
  ${TOPDIR}/../poky/meta \
  ${TOPDIR}/../poky/meta-poky \
  ${TOPDIR}/../poky/meta-skeleton \
  ${TOPDIR}/../poky/meta-yocto-bsp \
  /workspaces/ecc25_workspace/build/workspace \
"
BBPATH ?= "${TOPDIR}"
BBFILES ??= ""
```

second time you run the command:
ERROR: recipe baremetal-helloworld is already in your workspace

not an error

BITBAKE -C BUILD BAREMETAL-HELLOWORLD

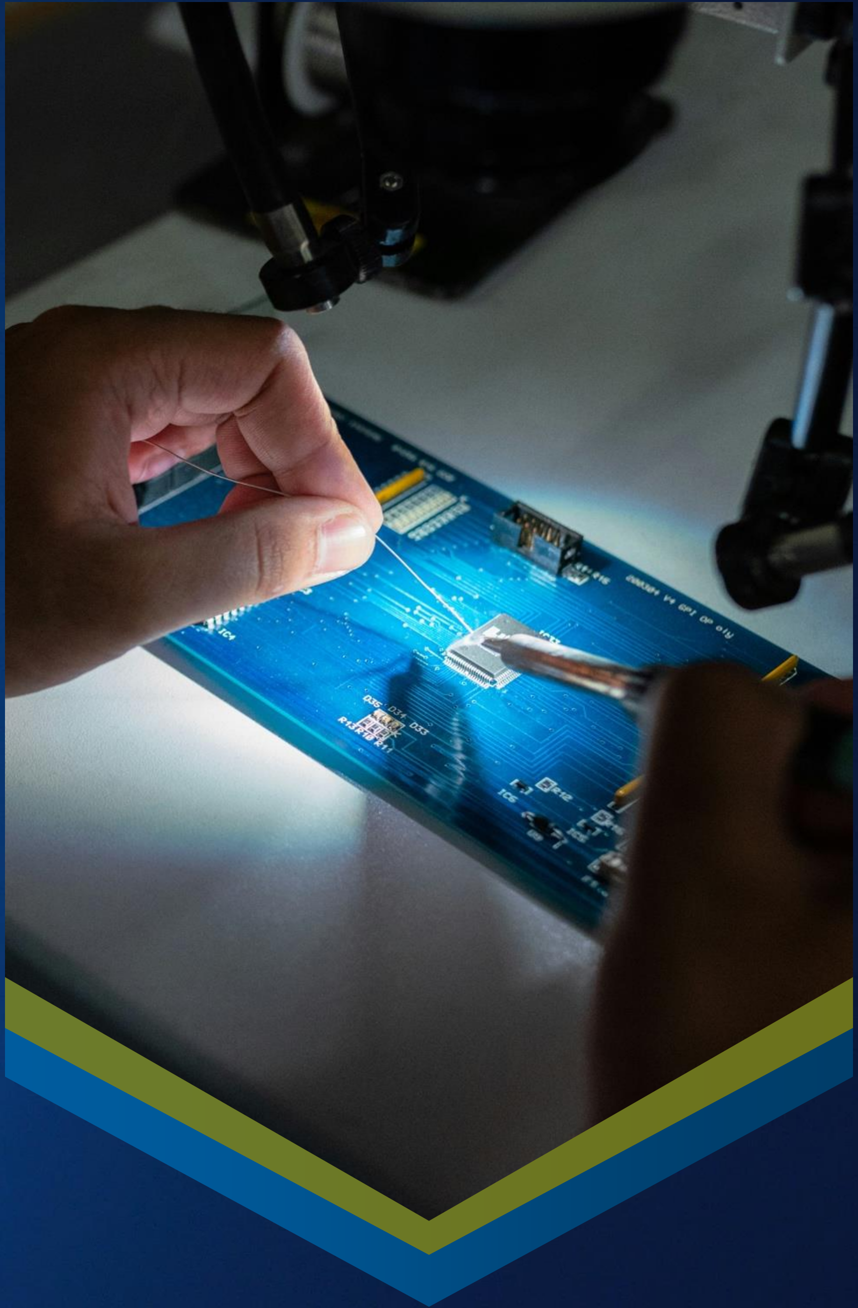
OUR TOOLS FOR TODAY

```
Build Configuration:
BB_VERSION = "2.0.0"
BUILD_SYS = "aarch64-linux"
NATIVELSBSTRING = "universal"
TARGET_SYS = "riscv32-poky-elf"
MACHINE = "qemuriscv32"
DISTRO = "poky"
DISTRO_VERSION = "4.0.27"
TUNE_FEATURES = "riscv32"
meta
meta-poky
meta-skeleton
meta-yocto-bsp = "kirkstone:ab9a994a8cd8e06b519a693db444030999d273b7"
workspace = "master:9d6361449182bf9986690a889d49f1a793731a10"
```

RUNQEMU NOGRAPHIC SLIRP

OUR TOOLS FOR TODAY

```
runqemu - INFO - Host uptime: 79362.64
Hello, world!
```



AGENDA

1 Get to know each other

2 KAS, Yocto, QEMU, Risc-V

3 C++ immediate functions

4 Interactive demo

C++ IMMEDIATE FUNCTIONS

OUR TOOLS FOR TODAY

consteval - specifies that a function is an *immediate function*, that is, every call to the function must produce a compile-time constant

https://en.cppreference.com/w/cpp/language/consteval

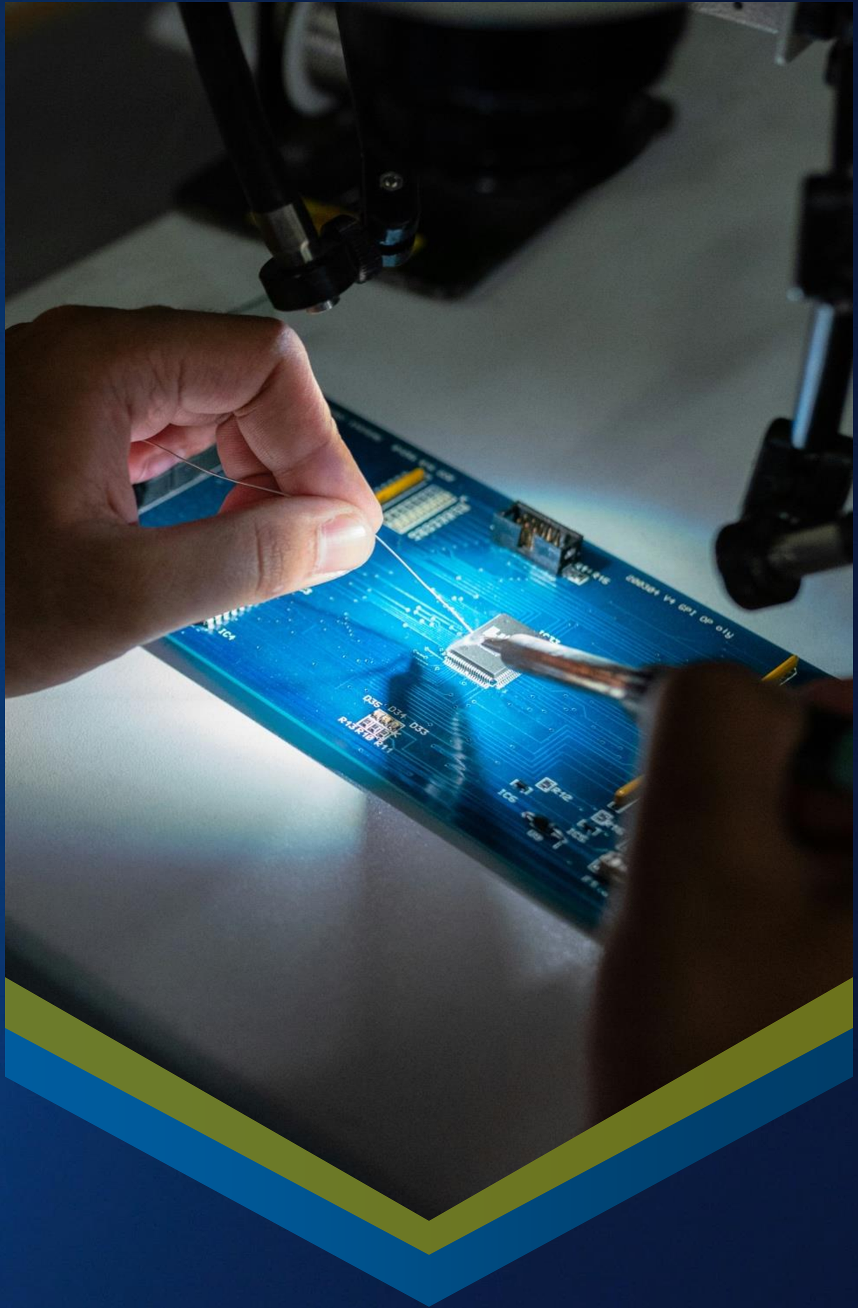
C++ IMMEDIATE FUNCTIONS

EXAMPLE

```
// This function might be evaluated at compile-time, if the input
// is known at compile-time. Otherwise, it is executed at run-time.
constexpr unsigned factorial(unsigned n) {
  return n < 2 ? 1 : n * factorial(n - 1);
}
// With consteval we enforce that the function will be evaluated at compile-time.
consteval unsigned combination(unsigned m, unsigned n)
{
  return factorial(n) / factorial(m) / factorial(n - m);
}
static_assert(factorial(6) == 720);
static_assert(combination(4, 8) == 70);

int main(int argc, const char*[])
{
  …
//  unsigned z = combination(argc, 7); // error: 'argc' is not a constant expression
}
```

https://en.cppreference.com/w/cpp/language/consteval



AGENDA

1 Get to know each other

2 KAS, Yocto, QEMU, Risc-V

3 C++ immediate functions

4 Interactive demo

INTERACTIVE DEMO

STEP 0: STATUS QUO

```
#define VIRT_UART0 0x10000000

volatile unsigned int * const UART0DR = (unsigned int *)VIRT_UART0;

/* Until we reach to the end of the string, put each char on UART0 */
void print_uart0(const char *str) {
  while(*str != '\0') {
    *UART0DR = (unsigned int) *str;
    str++;
  }
}

extern "C" void c_entry() {
  print_uart0("Hello, world!\n");
  while(1);
}
```

INTERACTIVE DEMO

STEP 0: STATUS QUO

```
#define VIRT_UART0 0x10000000

volatile unsigned int * const UART0DR = (unsigned int *)VIRT_UART0;

/* Until we reach to the end of the string, put each char on UART0 */
void print_uart0(const char *str) {
  while(*str != '\0') {
    *UART0DR = (unsigned int) *str;
    str++;
  }
}

extern "C" void c_entry() {
  print_uart0("Hello, world!\n");
  while(1);
}
```

INTERACTIVE DEMO

STEP 0: STATUS QUO

```
#define VIRT_UART0 0x10000000

volatile unsigned int * const UART0DR = (unsigned int *)VIRT_UART0;

/* Until we reach to the end of the string, put each char on UART0 */
void print_uart0(const char *str) {
  while(*str != '\0') {
    *UART0DR = (unsigned int) *str;
    str++;
  }
}

extern "C" void c_entry() {
  print_uart0("Hello, world!\n");
  while(1);
}
```

any issue ?

INTERACTIVE DEMO

STEP 1: ENABLE TESTING

```
#define VIRT_UART0 0x10000000

constexpr void print(volatile unsigned int* const uart, const char * str) {
    while(*str != '\0’) {
    *uart = (unsigned int) *str;
    str++;
  }
}

extern "C" void c_entry() {
  volatile unsigned int * const uart0 = (volatile unsigned int*) VIRT_UART0;
  print(uart0, "Hello, world!\n");
  while(1);
}
```

INTERACTIVE DEMO

STEP 1: ENABLE TESTING

```
#define VIRT_UART0 0x10000000

constexpr void print(volatile unsigned int* const uart, const char * str) {
    while(*str != '\0’) {
    *uart = (unsigned int) *str;
    str++;
  }
}

extern "C" void c_entry() {
  volatile unsigned int * const uart0 = (volatile unsigned int*) VIRT_UART0;
  print(uart0, "Hello, world!\n");
  while(1);
}
```

INTERACTIVE DEMO

STEP 1: ENABLE TESTING

```
#define VIRT_UART0 0x10000000

constexpr void print(volatile unsigned int* const uart, const char * str) {
    while(*str != '\0’) {
    *uart = (unsigned int) *str;
    str++;
  }
}

extern "C" void c_entry() {
  volatile unsigned int * const uart0 = (volatile unsigned int*) VIRT_UART0;
  print(uart0, "Hello, world!\n");
  while(1);
}
```

INTERACTIVE DEMO

STEP 2: MOCKING STRUCT

```
struct mock_uart {};
consteval bool test_dereferece_uart() {
  auto uart = mock_uart();
  (void)*uart;
  return true;
}
static_assert(test_dereferece_uart(),
  "test_dereferece_uart() failed");

consteval bool test_assignment_uart() {
  auto uart = mock_uart();
  *uart = (unsigned int) 'H’;
  return true;
}
static_assert(test_assignment_uart(),
  "test_assignment_uart() failed");
```

INTERACTIVE DEMO

STEP 2: MOCKING STRUCT

```
struct mock_uart {};
consteval bool test_dereferece_uart() {
  auto uart = mock_uart();
  (void)*uart;
  return true;
}
static_assert(test_dereferece_uart(),
  "test_dereferece_uart() failed");

consteval bool test_assignment_uart() {
  auto uart = mock_uart();
  *uart = (unsigned int) 'H’;
  return true;
}
static_assert(test_assignment_uart(),
  "test_assignment_uart() failed");
```

any issue ?

INTERACTIVE DEMO

STEP 2: MOCKING STRUCT

```
struct mock_uart {};
consteval bool test_dereferece_uart() {
  auto uart = mock_uart();
  (void)*uart;
  return true;
}
static_assert(test_dereferece_uart(),
  "test_dereferece_uart() failed");

consteval bool test_assignment_uart() {
  auto uart = mock_uart();
  *uart = (unsigned int) 'H';
  return true;
}
static_assert(test_assignment_uart(),
  "test_assignment_uart() failed");
```

1

2

```
| step2.hpp:23:11: error: no match for 'operator*' (operand type is 'mock_uart')
|    23 |      (void)*uart;
| step2.hpp:32:28: error: no match for 'operator=' (operand types are 'mock_uart' and 'unsigned int')
|    32 |      *uart = (unsigned int) 'H';
```

INTERACTIVE DEMO

STEP 2: MOCKING STRUCT

```
struct mock_uart {
  constexpr mock_uart() noexcept {}
  constexpr mock_uart& operator*() noexcept { return *this; }
  constexpr void operator=(unsigned int val) noexcept {}
};
```

```
consteval bool test_dereferece_uart() {
  auto uart = mock_uart();
  (void)*uart;
  return true;
}
static_assert(test_dereferece_uart(),
  "test_dereferece_uart() failed");

consteval bool test_assignment_uart() {
  auto uart = mock_uart();
  *uart = (unsigned int) 'H’;
  return true;
}
static_assert(test_assignment_uart(),
  "test_assignment_uart() failed");
```

INTERACTIVE DEMO

STEP 3: GENERALIZE FUNCTION PRINT

```
consteval bool test_print() {
  auto uart = mock_uart();
  print(uart, "Hello, world!");
  return true;
}
static_assert(test_print(),
  "test_print() failed");
```

INTERACTIVE DEMO

STEP 3: GENERALIZE FUNCTION PRINT

```
consteval bool test_print() {
  auto uart = mock_uart();
  print(uart, "Hello, world!");
  return true;
}
static_assert(test_print(),
  "test_print() failed");
```

any issue ?

INTERACTIVE DEMO

STEP 3: GENERALIZE FUNCTION PRINT

```
consteval bool test_print() {
  auto uart = mock_uart();
  print(uart, "Hello, world!");
  return true;
}
static_assert(test_print(),
  "test_print() failed");
```

1

2

```
| step3.hpp:41:11: error: cannot convert 'mock_uart' to 'volatile unsigned int*'
|    41 |     print(uart, "Hello, world!");
```

INTERACTIVE DEMO

STEP 3: GENERALIZE FUNCTION PRINT

```
template <typename UART>
constexpr void print(UART&& uart, const char * str) {
  while(*str != '\0’) {
    *uart = (unsigned int) *str;
    str++;
  }
}
```

```
consteval bool test_print() {
  auto uart = mock_uart();
  print(uart, "Hello, world!");
  return true;
}
static_assert(test_print(),
  "test_print() failed");
```

INTERACTIVE DEMO

STEP 4: OUR FIRST REAL COMPILE-TIME TEST

```
struct mock_uart {
...
constexpr void operator=(unsigned int val) noexcept {
  if (buffer && index < N) {
    buffer[index++] = (char) val;
}}...};

consteval bool test_print_content() {
  // arrange
  const unsigned int STRING_SIZE = 15;
  char actual[STRING_SIZE] = {0};
  const char expected[STRING_SIZE] = "Hello, world!\n";

  // act
  auto uart = mock_uart(actual, STRING_SIZE);
  print(uart, "Hello, world!\n");

  // assert
  return strcmp(expected, actual);
}
static_assert(test_print_content(),
  "test_print_content() failed");
```

INTERACTIVE DEMO

STEP 4: OUR FIRST REAL COMPILE-TIME TEST

```
struct mock_uart {
…
constexpr void operator=(unsigned int val) noexcept {
  if (buffer && index < N) {
    buffer[index++] = (char) val;
}}…};

consteval bool test_print_content() {
  // arrange
  const unsigned int STRING_SIZE = 15;
  char actual[STRING_SIZE] = {0};
  const char expected[STRING_SIZE] = "Hello, world!\n";

  // act
  auto uart = mock_uart(actual, STRING_SIZE);
  print(uart, "Hello, world!\n");

  // assert
  return strcmp(expected, actual);
}
static_assert(test_print_content(),
  "test_print_content() failed");
```

any issue ?

INTERACTIVE DEMO

STEP 4: OUR FIRST REAL COMPILE-TIME TEST

```
struct mock_uart {
…
constexpr void operator=(unsigned int val) noexcept {
  if (buffer && index < N) {
    buffer[index++] = (char) val;
}}…};

consteval bool test_print_content() {
  // arrange
  const unsigned int STRING_SIZE = 15;
  char actual[STRING_SIZE] = {0};
  const char expected[STRING_SIZE] = "Hello, world!\n";

  // act
  auto uart = mock_uart(actual, STRING_SIZE);
  print(uart, "Hello, world!\n");

  // assert
  return strcmp(expected, actual);
}
static_assert(test_print_content(),
  "test_print_content() failed");
```

INTERACTIVE DEMO

RUN-TIME

```
runqemu - INFO - Host uptime: 79362.64
Hello, world!
```

25-Oct-25

Stefano Fiorentino

IoT | Embedded Systems | Low-Latency

Stefano Fiorentino
stefano.fiorentino@adesso.ch
M +41 79 135 98 51

adesso Schweiz AG
Vulkanstrasse 106
3011 Zürich

adesso Svizzera SA
Corso E. Pestalozzi 9
6900 Lugano

THANK YOU!