

Internet of Things

Marco Zecchini

Sapienza University of Rome,
zecchini.1596071@studenti.uniroma1.it

Lecture 2: ARM Mbed OS



Agenda

Introduction to ARM Mbed OS

- Features
- Toolchains and IDE
- Architecture Diagram

Mbed OS API Examples

- Hello world example
- Echo serial example
- Driver API example
- AnalogIn example
- RTOS API example
- Power Management API

Exercise in class



ARM Mbed OS



Arm Mbed OS is a free, open-source embedded operating system designed specifically for the "things" in the Internet of Things. It includes all the features you need to develop a connected product based on an Arm Cortex-M microcontroller (32 bit).



Features

Modular - Ease of Use

With a modular library structure, the necessary underlying support for your application will be automatically included on your device. By using the Mbed OS API, your application code can remain clean, portable and simple.

End to End Security

It addresses security in device hardware, software, communication.

Open Source

<https://github.com/ARMmbed/mbed-os>

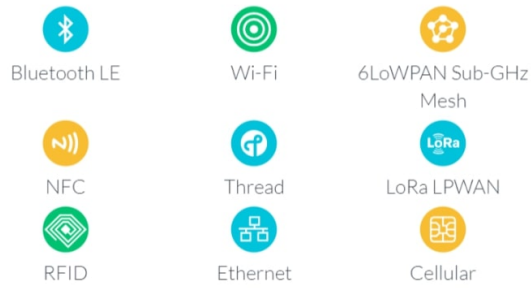
Community

Big community support and *forum*.



Features

Connectivity



Drivers and Support Libraries

Possibility to use other drivers and libraries developed by other users.



Toolchains and IDE

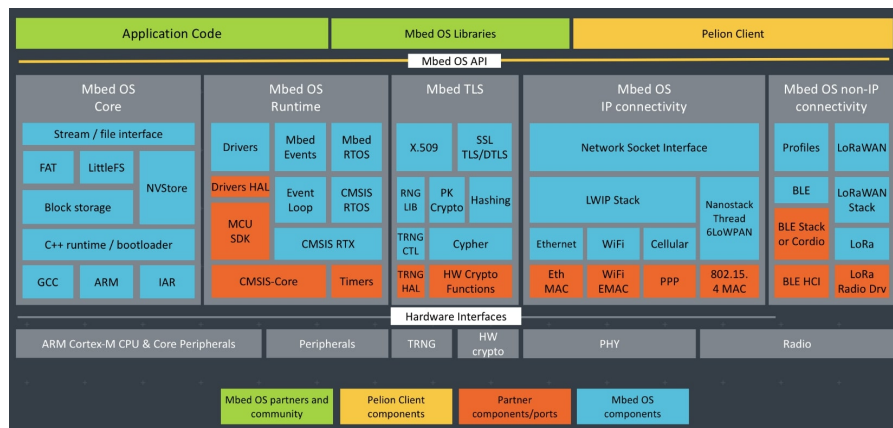
Two possibilities:

Online IDE, no setup and easiest way to get started.

Offline toolchains, (Arm Compiler 6, GCC and IAR) that support Arm Mbed CLI.



Architecture Diagram on an Mbed board



HelloWorld example

```

1 #include "mbed.h"
2
3 /*
4  * HelloWorld example
5  * Printing hello world and toggling LED1
6  */
7
8 //
9
10
11 DigitalOut led(LED1);
12
13 int main()
14 {
15     int i = 1;
16     printf("Hello World !\n");
17
18     while(1) {
19         wait(1); // 1 second
20         led = !led; // Toggle LED
21         printf("This program runs since %d seconds.\n", i++);
22     }
23 }
24
25

```

https://os.mbed.com/users/marcozecchini/code/Esempio_helloworld/



Echo serial example

```
main.cpp x
1 #include "mbed.h"
2
3 //-----
4 // Terminal configuration
5 // 9600 bauds, 8-bit data, no parity
6 //-----
7
8 Serial pc(SERIAL_TX, SERIAL_RX);
9
10 int main()
11 {
12     //pc.baudrate(115200);
13     while(1) {
14         if (pc.readable()){
15             pc.putc(pc.getc());
16         }
17     }
18 }
19
```

https://os.mbed.com/users/marcozecchini/code/Example_echo_serial/



Driver API example

```
main.cpp x
6 InterruptIn mybutton(USER_BUTTON);
7 DigitalOut myled(LED1);
8 Timer t;
9
10 float delay = 5.0; // 1 sec
11
12 void pressed()
13 {
14     t.stop();
15     printf("You pressed after %f seconds\n", t.read());
16     if (delay == 5.0)
17         delay = 0.2; // 200 ms
18     else
19         delay = 5.0; // 1 sec
20     t.reset();
21     t.start();
22 }
23
24 int main()
25 {
26     t.start();
27     mybutton.fall(&pressed);
28     while (1) {
29         myled = !myled; //toggle the led
30         wait(delay);
31     }
32 }
```

https://os.mbed.com/users/marcozecchini/code/Example_interrupt_button_timer/



AnalogIn example

```
main.cpp x
1 #include "mbed.h"
2
3 AnalogIn analog_value(A0);
4
5 DigitalOut led(LED1);
6
7 int main()
8 {
9     float meas_r;
10    float meas_v;
11
12    printf("\nAnalogIn example\n");
13
14    while(1) {
15        meas_r = analog_value.read(); // Read the analog input value (value from 0.0 to 1.0 = full ADC conversion range)
16        meas_v = meas_r * 3300; // Converts value in the 0V-3.3V range
17        // Display values
18        printf("measure = %f = %f mV\n", meas_r, meas_v);
19
20        // LED is ON is the value is below 1V
21        if (meas_v < 1000) {
22            led = 1; // LED ON
23        } else {
24            led = 0; // LED OFF
25        }
26
27        wait(1.0); // 1 second
28    }
29 }
```

https://os.mbed.com/users/marcozecchini/code/Esempio_read_analog/



RTOS API example

```
main.cpp x main.cpp x
1 #include "mbed.h"
2 #define BUF_SIZE 10
3
4 Thread thread;
5 CircularBuffer<char, BUF_SIZE> buf;
6 int bytes = 0;
7
8 void print_string(char data_stream[] = "Thread")
9 {
10    while (!buf.full()) {
11        buf.push(data_stream[bytes++ % 6]);
12    }
13 }
14
15 void print_thread()
16 {
17    while (true) {
18        wait(1);
19        print_string();
20    }
21 }
22
23 int main()
24 {
25    printf("\n\n*** RTOS - CircularBuffer basic example ***\n\n");
26    thread.start(print_thread);
27    char data;
28    while (true) {
29        printf("\n\nBuffer contents: ");
30        while (!buf.empty()) {
31            buf.pop(data);
32            printf("%c", data);
33        }
34    }
35    printf("\n\n");
36    wait(1);
37 }
38 }
```

https://os.mbed.com/users/marcozecchini/code/Example_RTOS/




Power Management API

Sleep

There is only one sleep function in Mbed OS:

```
void sleep();
```

This function invokes sleep manager, which selects the most appropriate sleep mode.

 **Note:** In most cases, you don't need to call `sleep()` directly. Mbed OS enters sleep mode automatically any time the system is idle. That is when all your threads are in a waiting state, for example waiting for an event or a timeout.

There are two available sleeps mode:

- ▶ *Sleep mode*, the system clock to the core stops. It maintains the processor, peripheral and memory state, and the peripherals continue to work and can generate interrupts.
- ▶ *DeepSleep mode*, it saves additional power by turning off the high-speed clocks but has a longer wake up time. Peripherals not relying on high-speed clocks can still work.



Power Management API

Sleep mode and *DeepSleep mode* can be locked (disabled) or unlocked (enabled) interacting with the *Sleep Manager APIs*.

https://os.mbed.com/docs/mbed-os/v5.11/mbed-os-api-doxy/group__hal__sleep__manager.html



Quiz platform - exercise in class

You have to realize, in group of 3, a platform to play a *quiz contest*. After having communicated the number of participants, through the serial communication with the laptop (where each line ends with the character "!"), each player, at its turn, receives a question and four answers. It receives a string with this form "*question ; correct answer ; wrong answer ; wrong answer ; wrong answer*" and the answers must be shown randomly. Once a question is shown the player must give the answer until 10 seconds otherwise the platform would send the string "0" as answer. If he answers, the board will read the choice on the serial channel and, then, checks its correctness. While the platform is waiting for the answer, a *thread runs in parallel toggling a led quickly*. The players have the possibility to pause the game for thirty seconds pressing the *user button* (not while the board is waiting for the answer). Once a turn ends, the score is shown.



Another exercise

With the X-NUCLEO-KS01A2 expansion board take this measurements:

- ▶ With the gyroscope/accelerator verify if the board is in the *right way* (not upside-down, not on its side,...)
- ▶ Temperature
- ▶ Humidity



