BoardOS Project

Master's Degree in Computer Engineering

Cybersecurity

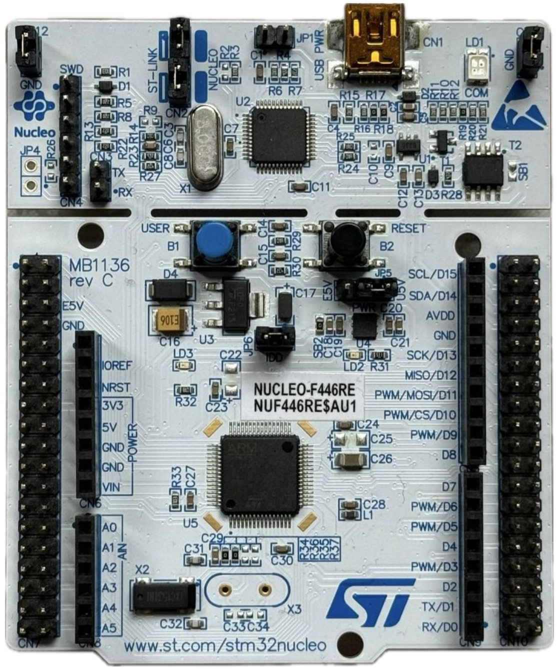
Presented by:
Jacopo Coniglio
Francesca Coriale
Sibilla Merlo
Giulia Lydia Perini
Ankesh Porwal

A.Y. 2023/2024

Politecnico di Torino

The project required the usage of a STM32 Nucleo-64 Board paired with STM32F446RE microcontroller.

We used FreeRTOS for the implementation of exercises.



Index

- Point 1: Installation and usage procedures
- Point 2: Examples illustrating the functionalities of the board
- Point 3: Customization of the operating system
- Point 4: Benchmarks of the results obtained in point 3

Index

Point 1: Installation and usage procedures

Requirements

- Device running Windows OS (7, 8, or 10), Linux 64-bit, or macOS
- USB Type-A to Mini-B cable to connect the board to the device

Point 1: Installation and usage procedures

Initial steps

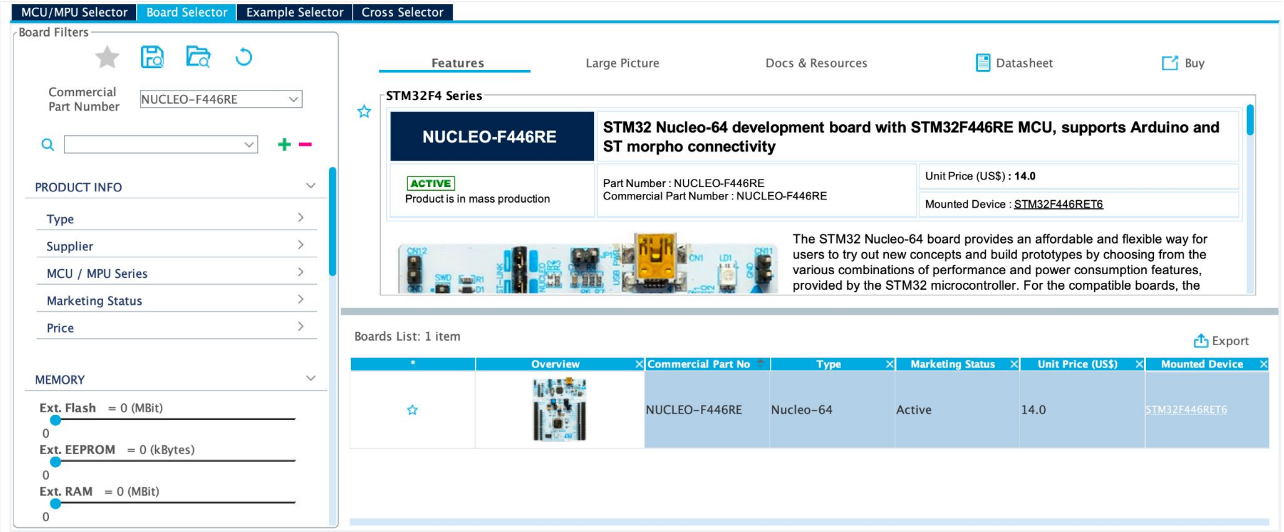
- Users must register on the official ST website at www.st.com
- Download STM32CubeIDE-xxx
- Connect the board to the device
- Launch the application

	Part Number	General Description	Latest version	Download	All versions
+	STM32CubeIDE-DEB	STM32CubeIDE Debian Linux Installer	1.14.0	Get latest	Select version
+	STM32CubeIDE-Lnx	STM32CubeIDE Generic Linux Installer	1.14.0	Get latest	Select version
+	STM32CubeIDE-Mac	STM32CubeIDE macOS Installer	1.14.0	Get latest	Select version
+	STM32CubeIDE-RPM	STM32CubeIDE RPM Linux Installer	1.14.0	Get latest	Select version
+	STM32CubeIDE-Win	STM32CubeIDE Windows Installer	1.14.0	Get latest	Select version

Point 1: Installation and usage procedures

Creating a project

- Define a workspace
- Select the specific MCU or MPU or choose the board



Index

Point 2: Examples illustrating the functionalities of the board

Point 2: Examples

Example 0: Basic Scheduling

- **Aim**: explanation of how scheduling works in FreeRTOS
- Three tasks: High, Medium and Low priority
- After *oSKernelStart()* operation the scheduler starts

```
/* Definitions for Task1 */
osThreadId_t Task1Handle;
const osThreadAttr_t Task1_attributes = {
  .name = "Task1",
  .stack_size = 128 * 4,
  .priority = (osPriority_t) osPriorityHigh,
};
```

Point 2: Examples

Example 0: Basic Scheduling

- Each task has an infinite loop that makes it to execute 3 times
- Then terminate
- First task chosen is High task, then Medium and then Low

```
void StartTask1(void *argument)
{
  /* USER CODE BEGIN 5 */
  /* Infinite loop */
  for(;;)
  {
          int c;
          c = Task1_Profiler++;
          printf("Task-1 %d \n", c);
         if (c >= 3) {
         osThreadExit();
          }
  }
  /* USER CODE END 5 */
}
```

Point 2: Examples

Example 1: LCD1602

- **Aim**: initialization and use of an LCD1602 screen to display a message "Time left" and a countdown from 600 to 1
- Writes the string "Time left" to the LCD1602 display at position (0,3)
- The *HAL_Delay* function creates a time interval between iterations of the for loop, thus producing a visual countdown effect on the display

```
Lcd_cursor(&lcd, 0,3);
Lcd_string(&lcd, "Time left");
  for ( int x = 600; x >= 1 ; x-- )
  {
    Lcd_cursor(&lcd, 1,7);
    Lcd_int(&lcd, x);
    HAL_Delay (1000);
  }
```

Point 2: Examples

Example 1: LCD1602

```
lcd = Lcd_create(ports, pins, GPIOB, GPIO_PIN_5, GPIOB, GPIO_PIN_4, LCD_4_BIT_MODE);
```

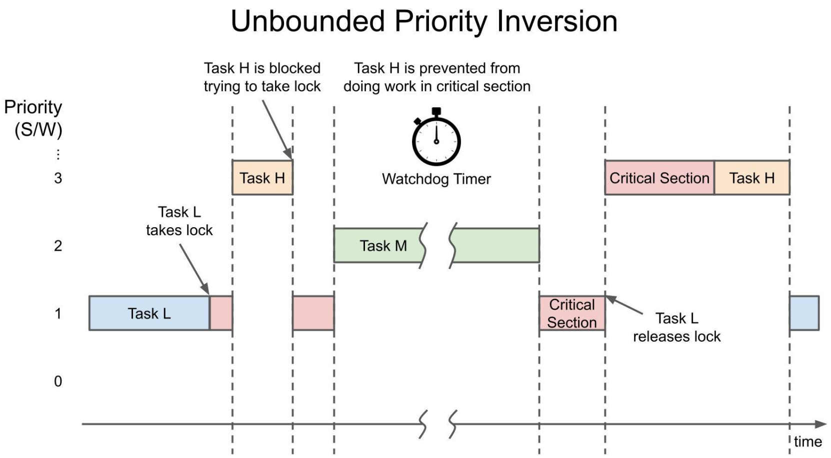
- ports: an array of GPIO port configurations. This array specifies the GPIO ports used for communication with the LCD: GPIOC, GPIOB, GPIOA are being passed.
- pins: an array of GPIO pin configurations. This array specifies the specific pins within the GPIO ports used for communication with the LCD: GPIO_PIN_7, GPIO_PIN_6 are being passed.
- lcd_4_bit_mode: indicates that the LCD is configured to use a 4-bit data bus mode, a common configuration for interfacing with LCDs.
- gpio_pin_4: this is used for the EN function.

Point 2: Examples

Example 3: Priority inversion

- **Aim**: develop an example with the priority inversion bug and then fix it
- Three task with different priorities: High, Medium, Low

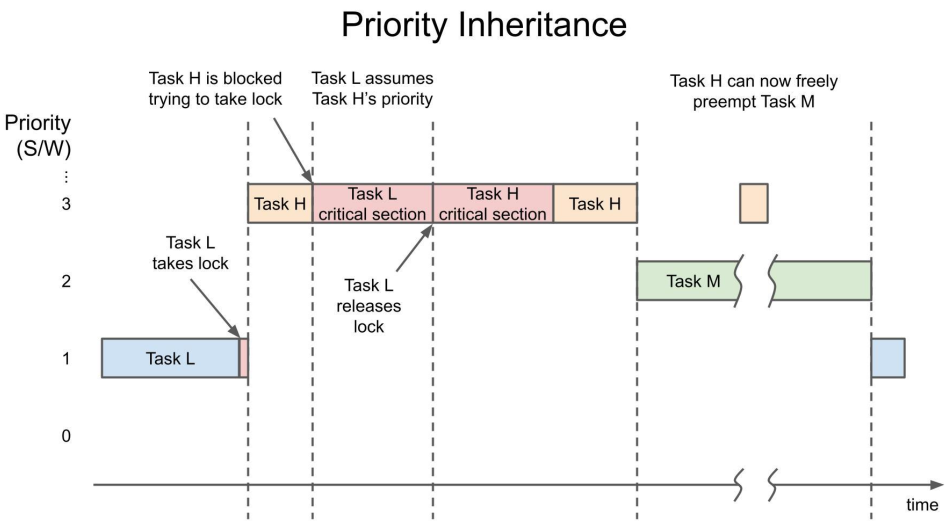
- High task binary semaphore
- Low semaphore binary semaphore but button to release it
- Medium task linear execution



Point 2: Examples

Example 3: Priority inversion

- mutexes in FreeRTOS implement priority inheritance
- priority of low task temporarily raised



Point 2: Examples

Example 2: Led with interrupt

- **Aim**: update the status of the green LED using the Blue Button

- counter = 0, then it blinks with a period of 0.5 seconds;
- counter = 1, then it remains lit;
- counter = 2, then it remains off;
- counter = 3, we start again the cycle by setting the counter =0

```
for(;;)
{
        /* Blue button not pressed: */
        if(counter == 0){
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
            osDelay(500);
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
            osDelay(500);
        }
        /* Blue button pressed 1 time: */
        else if(counter == 1){
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
        }
        /* Blue button pressed 2 times: */
        else if(counter == 2){
            HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
        }
        else{
            counter = 0;
        }
    }
```

Point 2: Examples

Example 2: Led with interrupt

```
/* EXTI interrupt init*/
HAL_NVIC_SetPriority(EXTI15_10_IRQn, 5, 0);
HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
```

- Callback function to update the counter when the interrupt occurs

```
/* Function to handle the interrupt */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
    if(GPIO_Pin == GPIO_PIN_13){
        counter++;
    }
}
```

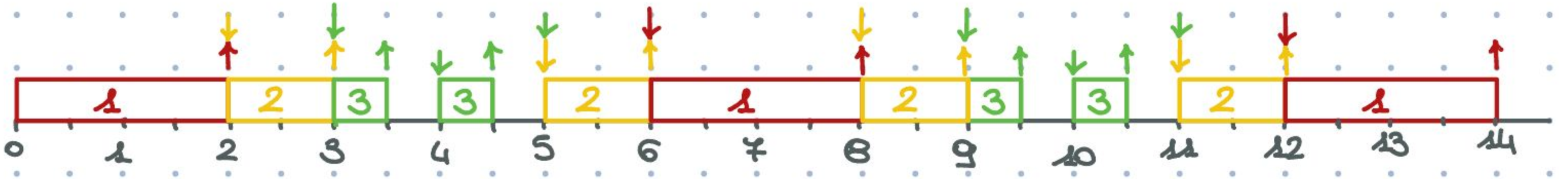
Index

Point 3: Customization of the operating system

Point 3: Customization

Customization of the operating system

- Task 1 with priority = 3, an execution time of 2s and then stopped for 4s;
- Task 2 with priority = 2, an execution time of 1s and then stopped for 2s;
- Task 3 with priority = 1, an execution time of 0.5s and then stopped for 0.5s;



Point 3: Customization

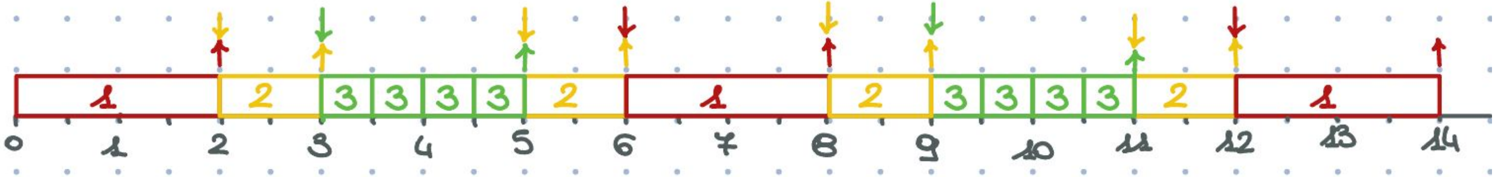
Customization of the operating system

```
TickType_t xCurrentTime = xTaskGetTickCount(); //current time
TCB_t *pxTCB;

//Initial control: two tasks in the block list
if (listCURRENT_LIST_LENGTH(pxDelayedTaskList) >= 2)
{
    //get the first element of the list
    ( pxTCB ) = listGET_OWNER_OF_HEAD_ENTRY(pxDelayedTaskList);

    //get the remaining ticks of the delay time of the first element of the block list
    xNextTaskUnblockTime = listGET_LIST_ITEM_VALUE( &( ( pxTCB )->xStateListItem ) );

    TickType_t xCurrentTaskUnblockTime = xCurrentTime + xTicksToDelay;
    //compare the delay ticks of the current task with the delay ticks of the first element
    if(xCurrentTaskUnblockTime <= xNextTaskUnblockTime)
        return;
}
```



Index

Point 4: Benchmarks of the results obtained in point 3

Point 4: Benchmarks

Benchmarks of the implementation

- define the function *convertTicksToTime()*

```
/* function to convert tick in time */
void convertTicksToTime(TickType_t ticks, unsigned int *seconds, unsigned int *milliseconds) {
    // Calcola il tempo totale in millisecondi
    uint32_t totalMilliseconds = (ticks * 1000) / TICKS_PER_SECOND;

    // Calcola i secondi e i millisecondi
    *seconds = totalMilliseconds / 1000;
    *milliseconds = totalMilliseconds % 1000;
}
```

- timestamp for every task the tick of beginning and the tick of ending, convert them into seconds and milliseconds and then print them on the screen.

Point 4: Benchmarks

Benchmarks of the implementation

- **Final result:**

```
Port 0 ×
Start scheduling...
 Start Task 1...  0 seconds, 0 milliseconds
 End Task 1...  2 seconds, 0 milliseconds
 Start Task 2...  2 seconds, 0 milliseconds
 End Task 2...  3 seconds, 0 milliseconds
 Start Task 3...  3 seconds, 0 milliseconds
 End Task 3...  3 seconds, 500 milliseconds
 Start Task 3...  3 seconds, 500 milliseconds
 End Task 3...  4 seconds, 0 milliseconds
 Start Task 3...  4 seconds, 0 milliseconds
 End Task 3...  4 seconds, 500 milliseconds
 Start Task 3...  4 seconds, 500 milliseconds
 End Task 3...  5 seconds, 0 milliseconds
 Start Task 2...  5 seconds, 0 milliseconds
 End Task 2...  6 seconds, 0 milliseconds
 Start Task 1...  6 seconds, 0 milliseconds
 End Task 1...  8 seconds, 0 milliseconds
 Start Task 2...  8 seconds, 0 milliseconds
 End Task 2...  9 seconds, 0 milliseconds
 Start Task 3...  9 seconds, 0 milliseconds
 End Task 3...  9 seconds, 500 milliseconds
 Start Task 3...  9 seconds, 500 milliseconds
 End Task 3...  10 seconds, 0 milliseconds
 Start Task 3...  10 seconds, 0 milliseconds
```

Thanks for your attention!