ERTS Final Assessment – Task 2

QINGYU ZHANG and vn22984
SHURAN YANG and rw22242
HAIBO LIAN and tb22111
Group 04

Outline

1) **Practical element**
   I. Overview of the system
   II. The process of specific tasks (Six, one for interrupt)
   III. The process of specific timer (Two)
   IV. LED instruction
   V. Implementation details
2) **Video display**
3) **QA**
   I. Task / Scheduling
   II. Resource control
   III. Suspend and delete

Practical element

Overview of the system

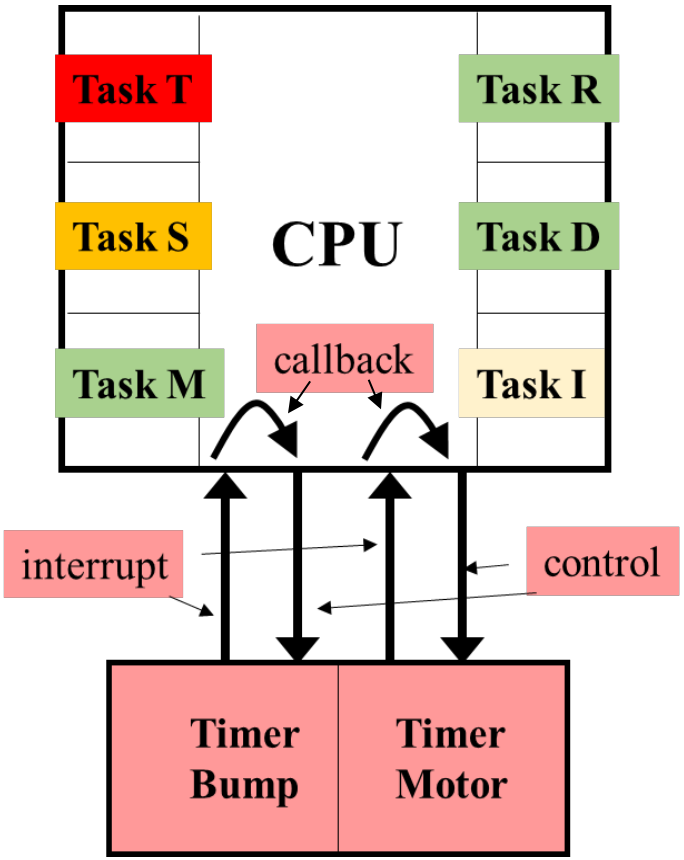


Figure System concept diagram

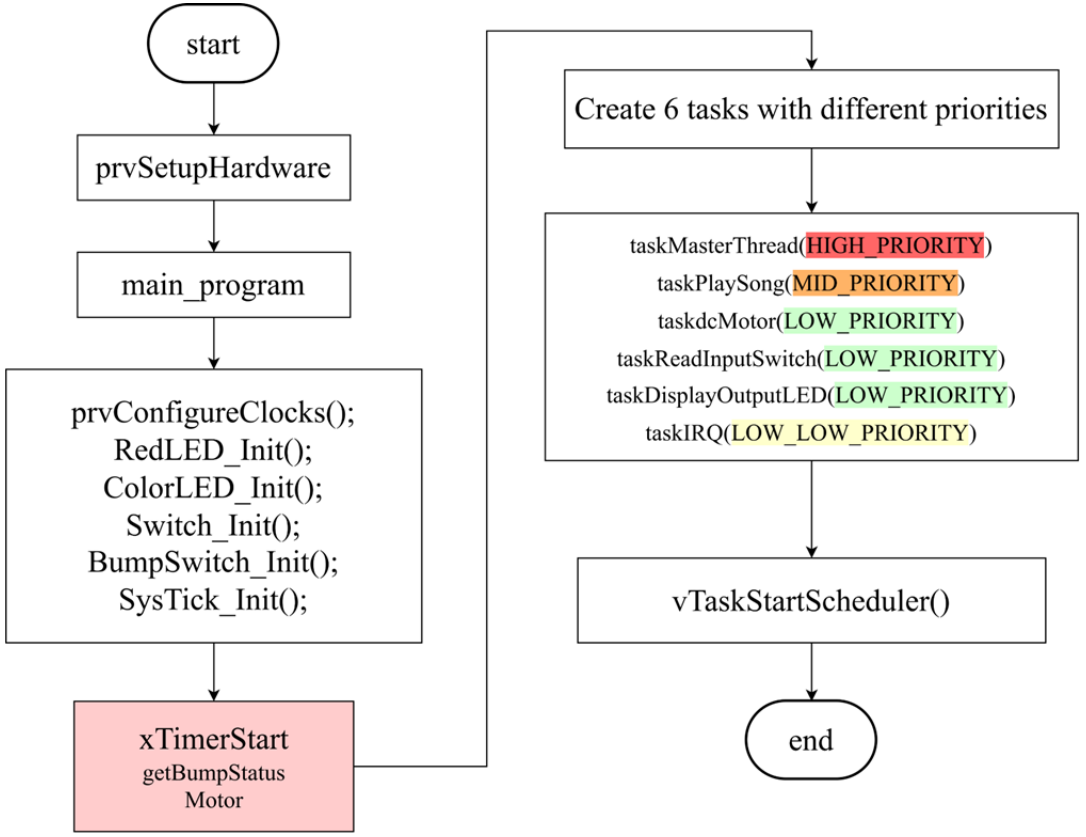


Figure Main function diagram

Practical element

The process of specific loop tasks

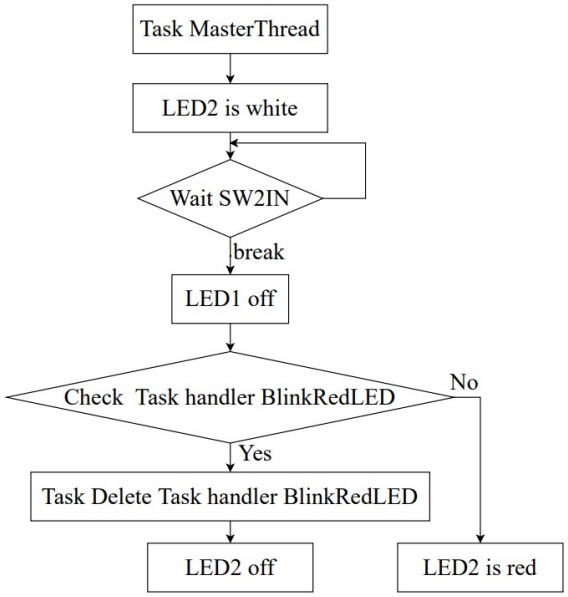


Figure Task MasterThread diagram

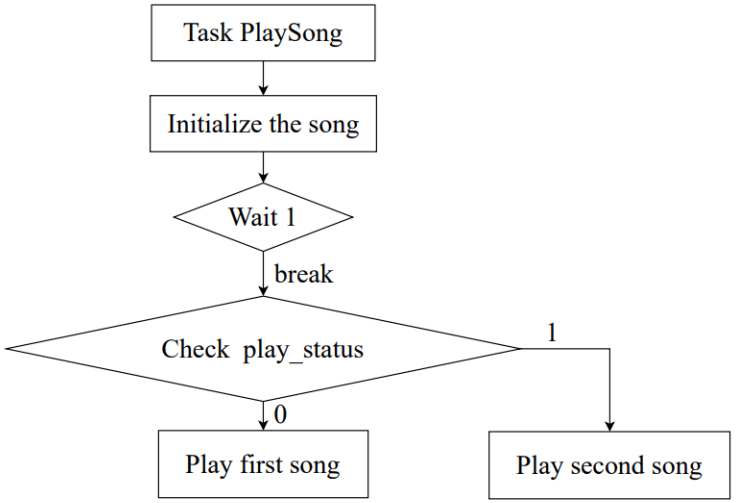


Figure Task PlaySong diagram

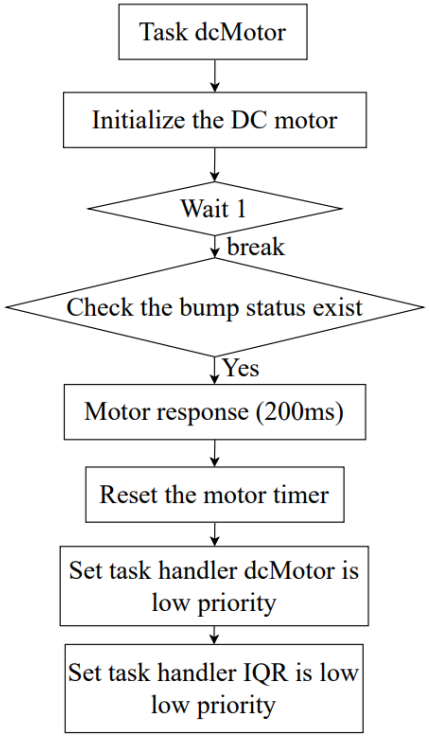


Figure Task dcMotor diagram

Practical element

The process of specific loop tasks

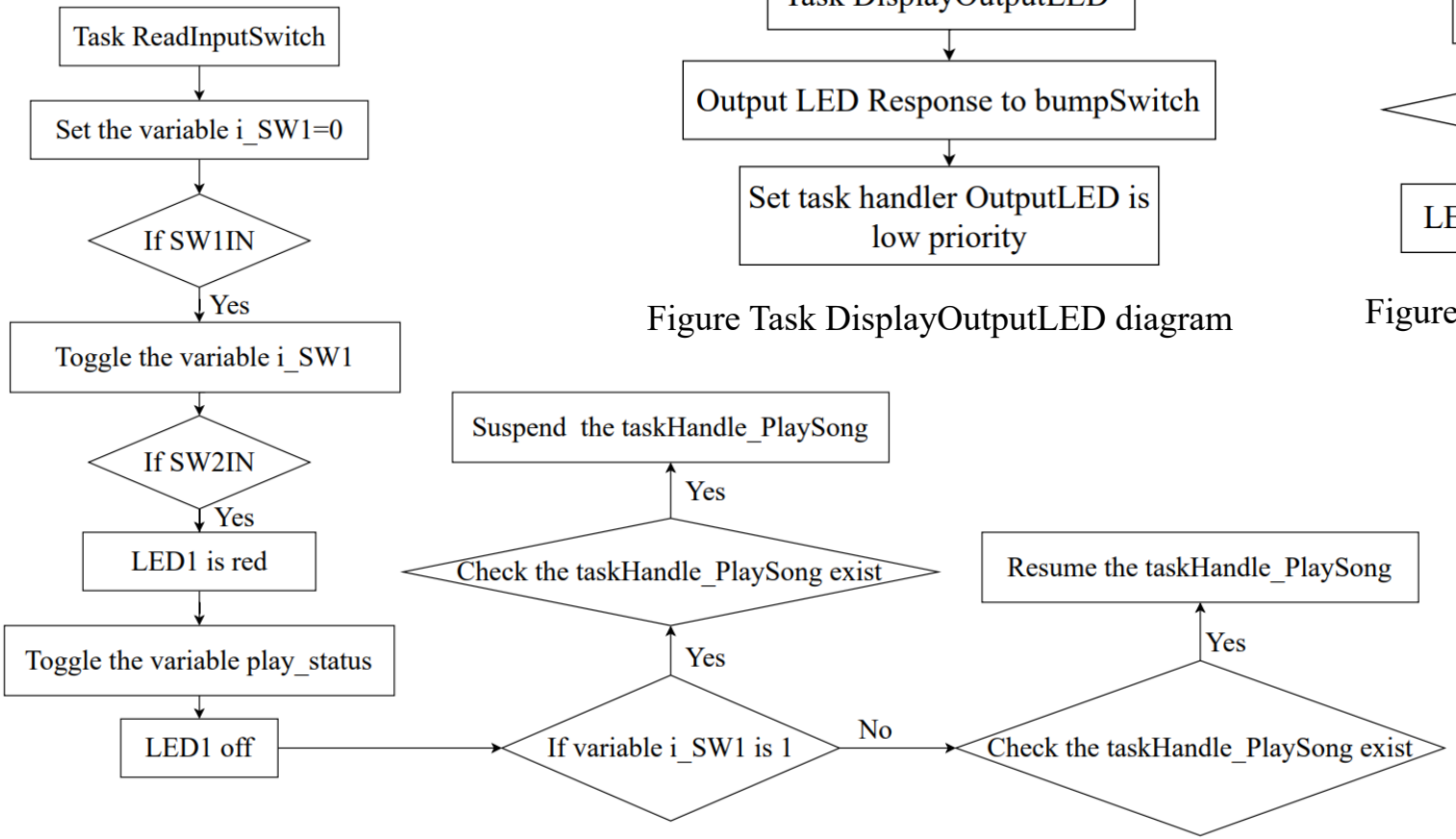


Figure Task ReadInputSwitch diagram

Figure Task DisplayOutputLED diagram

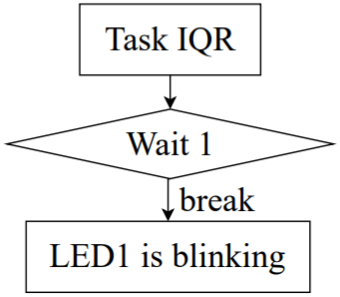


Figure Task IQR diagram

Practical element

The process of specific timer tasks (Two)

This design uses the principle of time-sharing multiplexing to effectively reduce the CPU resource usage through CPU callbacks to **two external timers**.

- vTimerCallback_bumpStatus: Checking the bump status every 100ms by callback.
- vTimerCallback_Motor: Realising the PWM movement to achieve normal formal moving by callback.

*: For the 'vTimerCallback_Motor', the timer will pause when the bump occurs (response bump switch), and execution will only continue after the response has finished.

Practical element

LED instruction

- **1.** In the initial state, the LED2 will be bright white. After pressing switch 2, the LED2 will be off, and no the LED is on at this time. In this state, by pressing switch 2, the LED1 will flash red and switch songs.
- **2.** The robot will back up and turn its direction after being hit by different bump switches. In this process, the LED2 will flash different colors after different bump switches are bumped. Meanwhile, the LED1 will keep blinking until the robot continues to go straight.

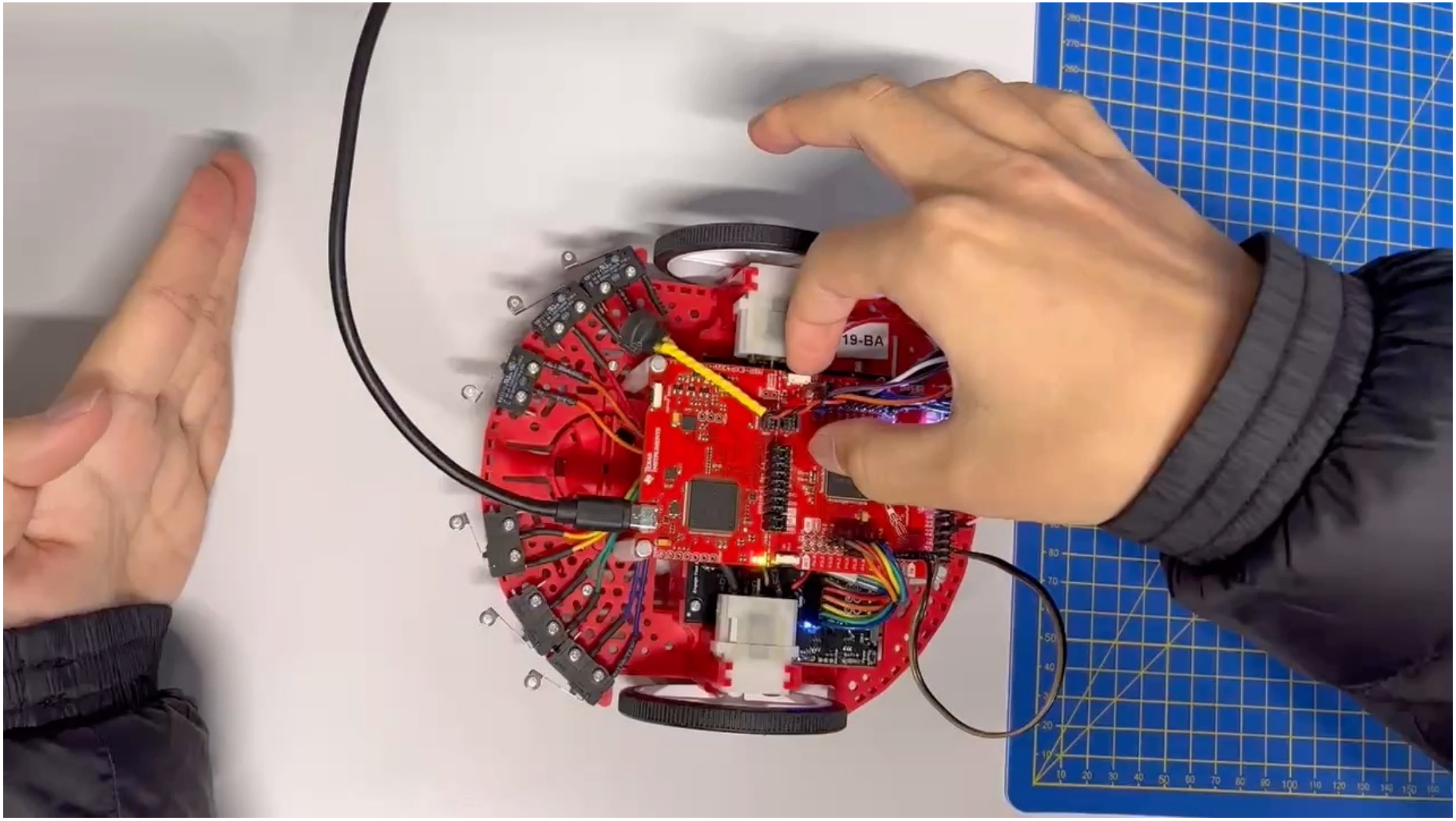
	Reset	Switch1	Switch2	Bump1	Bump2	Bump3	Bump4	Bump5	Bump6
LED1 (red)			Red	Red	Red	Red	Red	Red	Red
LED2 (colour)				Skyblue	Red	Pink	Yellow	Green	Blue

Practical element

Implementation details

- The initial state of the robot is stationary, and it will enter the motion mode after pressing switch 2. In this mode, first the robot will run forward and start playing the song at the same time. For songs, pressing switch 2 can realize the switching of song and pressing switch 1 can pause or continue to play song.
- In addition, if any of the bump switches is hit during the movement, the robot will reverse and turn. Next, go ahead and run. Finally, the state of the robot can be reset using switch reset.
- There is an IRQ thread during the implementation, when the bump switch is bumped the IRQ thread will increase the priority, so the LED1 will blink.

Video display



QA

Tasks/Scheduling

What tasks have you created?

In this project, there are 6 tasks with 4 different priorities.:

1. taskMasterThread:High priority, used for the task of blinking red LED.
2. taskPlaySong: Mid priority, used to complete the task of playing the song.
3. taskdcMotor: Low priority, used for motor-driven task.
4. taskReadInputSwitch : Low priority, used to detect bump switches.
5. taskDisplayOutputLED : Low priority, used to control the signal light display in the event of a collision.
6. taskIRQ: Low and low priority, used as an interrupt to complete a song switch.

QA

Tasks/Scheduling

What scheduling algorithm have you used?

We use an algorithm and a new functional function to coordinate the work between the various tasks to achieve more complex functions.

1. The scheduling algorithm

We adjust the priority of the tasks corresponding to the two responses and one interrupt function according to the status of the collision and subsequently reset the priority of the response and interrupt tasks after the response is completed. This makes the responses and interrupts could react very fast.

2. The Timer function

We use an external hardware resource, a timer, to reduce the utilisation of the CPU. A callback function is executed ted whenever the external timer expires to detect collision status and normal motor operation.

QA

Resource control

How have you controlled access to any shared data/hardware?

We try to use the “volatile” keyword to declare a variable in the source code. This keyword informs the compiler that the value of the corresponding variable can be changed at will without the need to give the relevant task in the code. This makes the safety of the shared data.

As this is a single-threaded CPU, the hardware resources are left to the system FreeRtos to schedule, and the user does not have to think about the simultaneous use of hardware.

From the point of view of the underlying AHB design of a single-threaded chip, the corresponding data storage changes are completed before the next task is carried out, so there are no memory conflicts or variables that are unsafe or are being used at the same time. However, in a multicore design, this issue needs to be considered.

QA

Suspend and delete

What is the difference between suspend and delete task and what are their uses ?

"vTaskSuspend" means that the execution of the tasks was hung on in RAM. These tasks could be released at any time by the function "vTaskResume". However, "vTaskDelete" means that I remove the task from RAM completely. The tasks execution status, dispatch and pointer, are also released. If we need to recreate the task, we should re-execute "vTaskCreate" to create a new task and reschedule this task.

Thank you

Group 04