PADO Labs

PADO Labs

Microcontroladores Aplicados a IoT

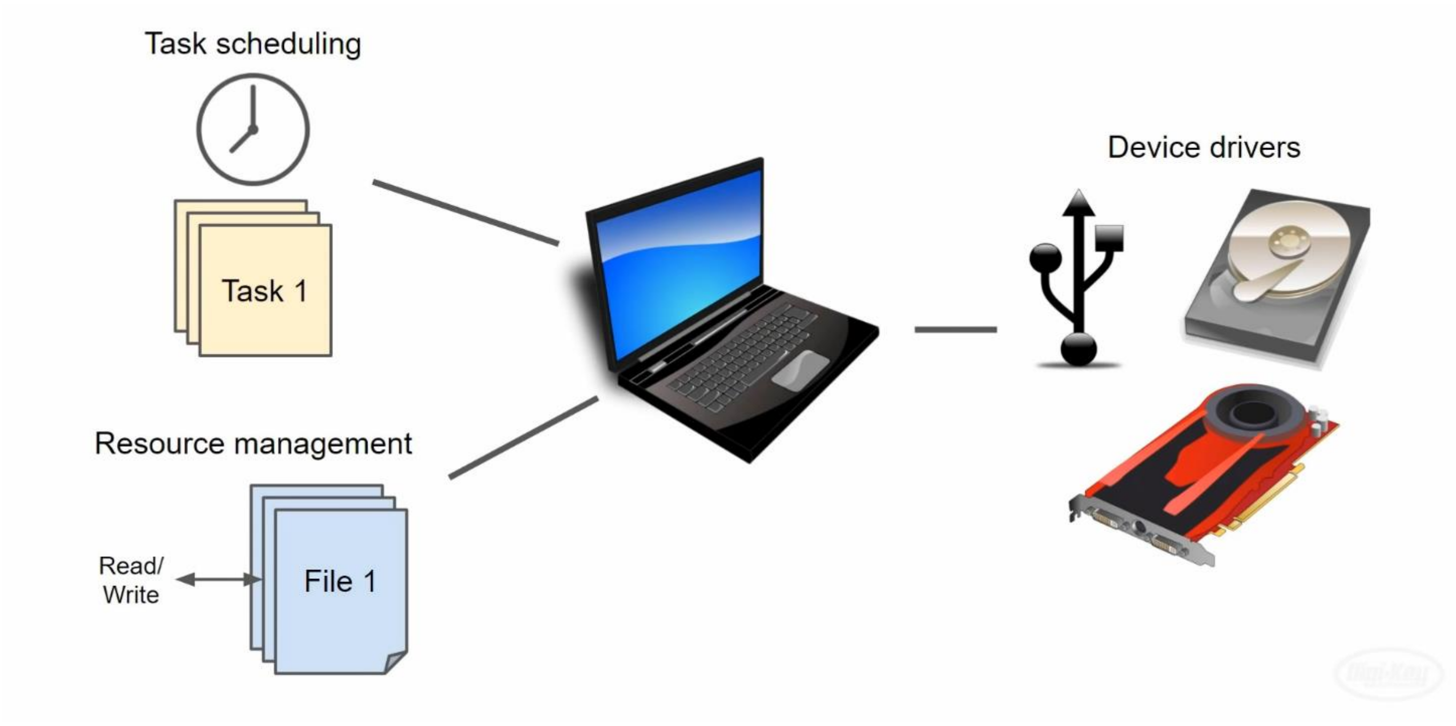
Dilson Liukiti Ito

03 – FreeRTOS: Tasks, Queues e Mutex

Microcontroladores Aplicados a IoT

GPOS

- General Purpose Operating System

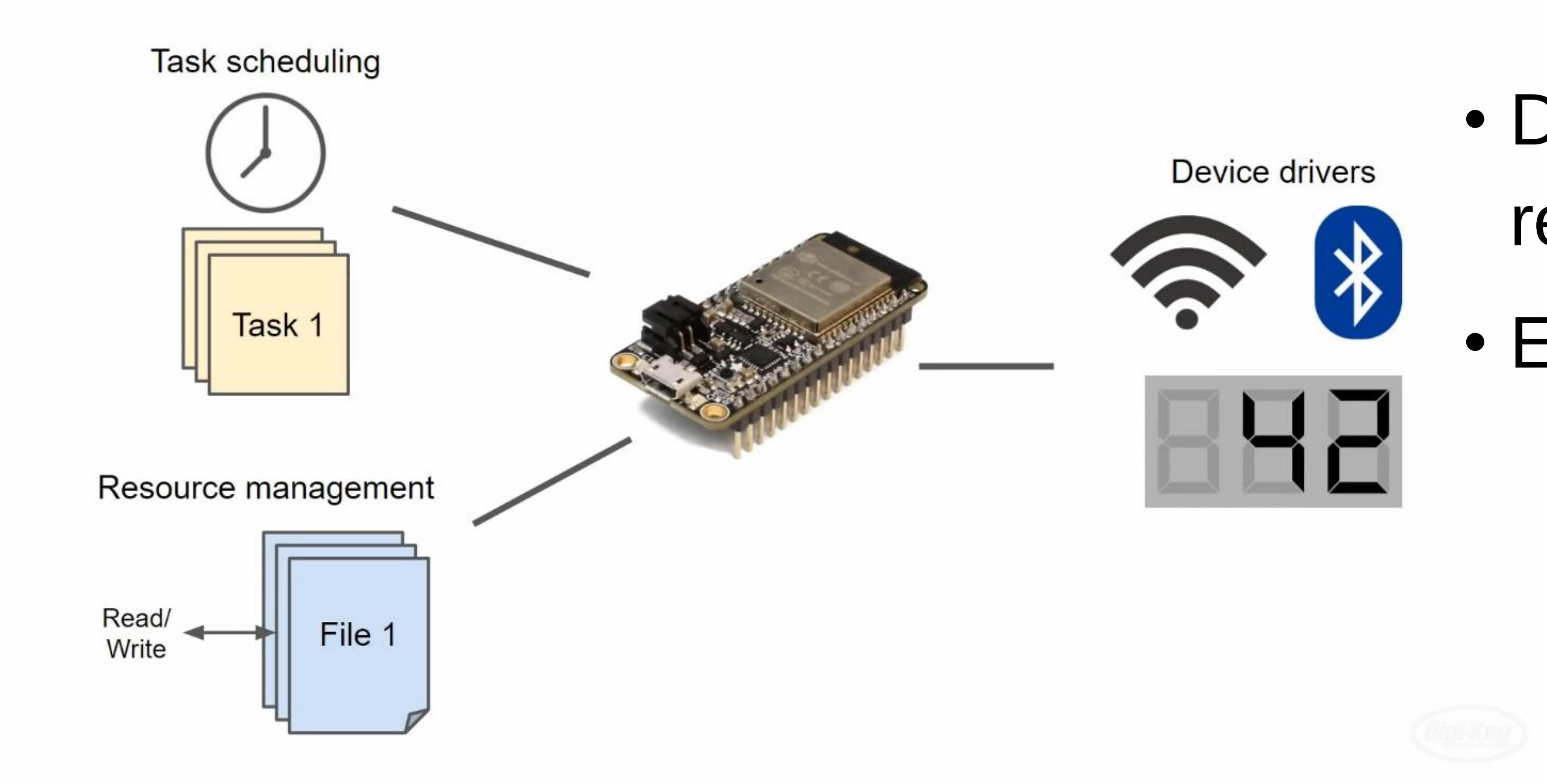


- Não Determinístico: tempo de resposta não é crítico
- Interação com humanos
- Ex.: Windows, Linux, MacOs, Android, iOS.

https://www.youtube.com/watch?v=F321087yYy4&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz

RTOS

- Real Time Operating System



- Determinístico: tempo de resposta crítico
- Ex: FreeRTOS

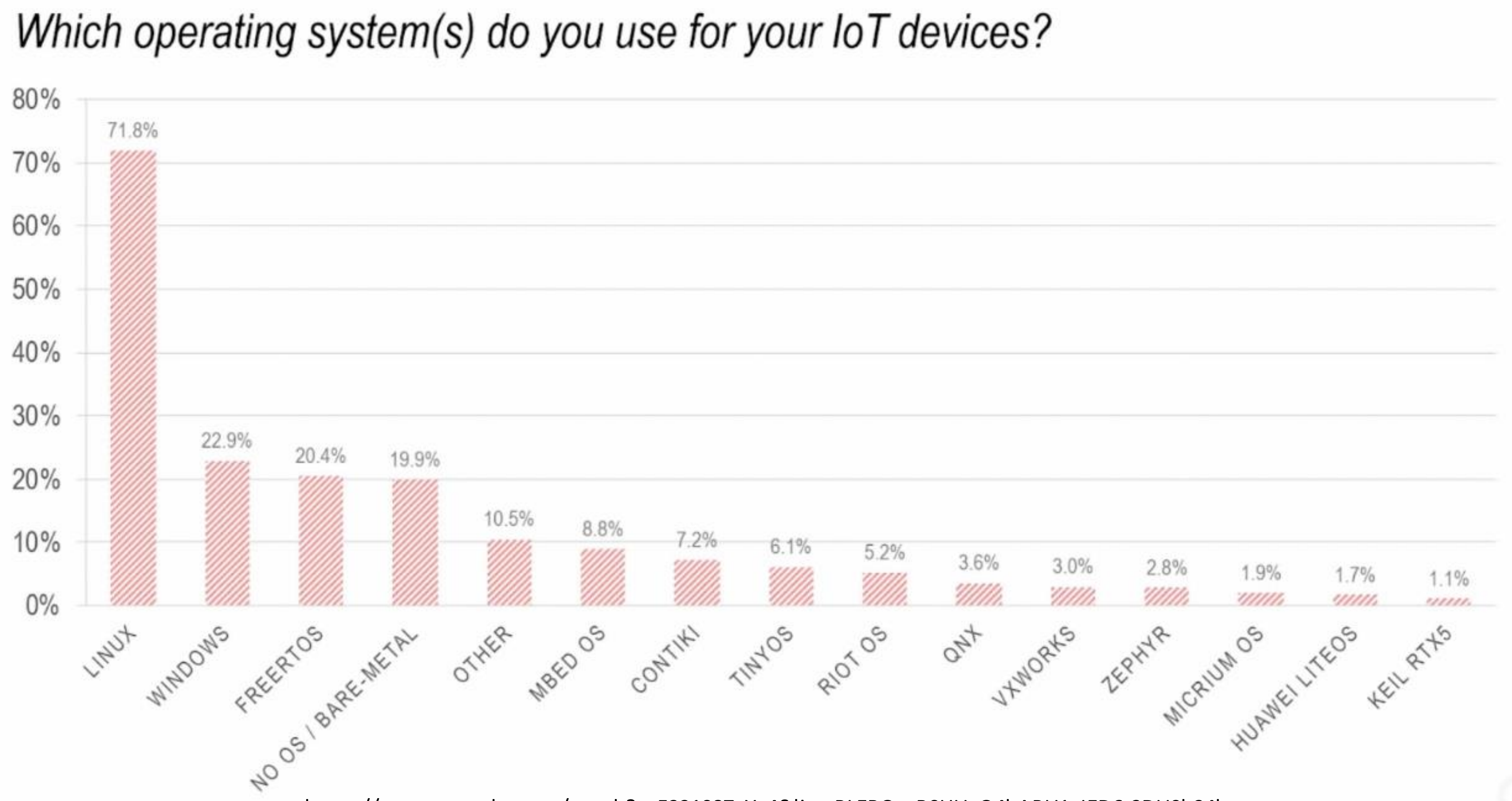
https://www.youtube.com/watch?v=F321087yYy4&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz

FreeRTOS

PADO Labs

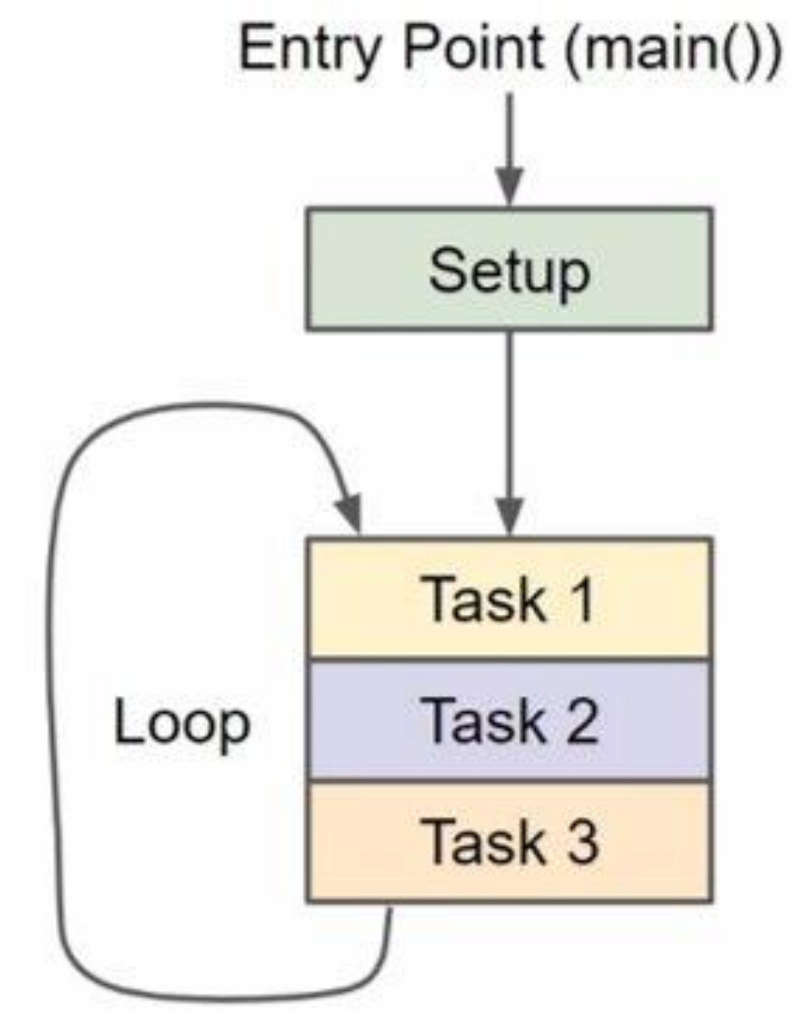
https://www.youtube.com/watch?v=EHdLvtI5DQI

IoT Operationg Systems



https://www.youtube.com/watch?v=F321087yYy4&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz

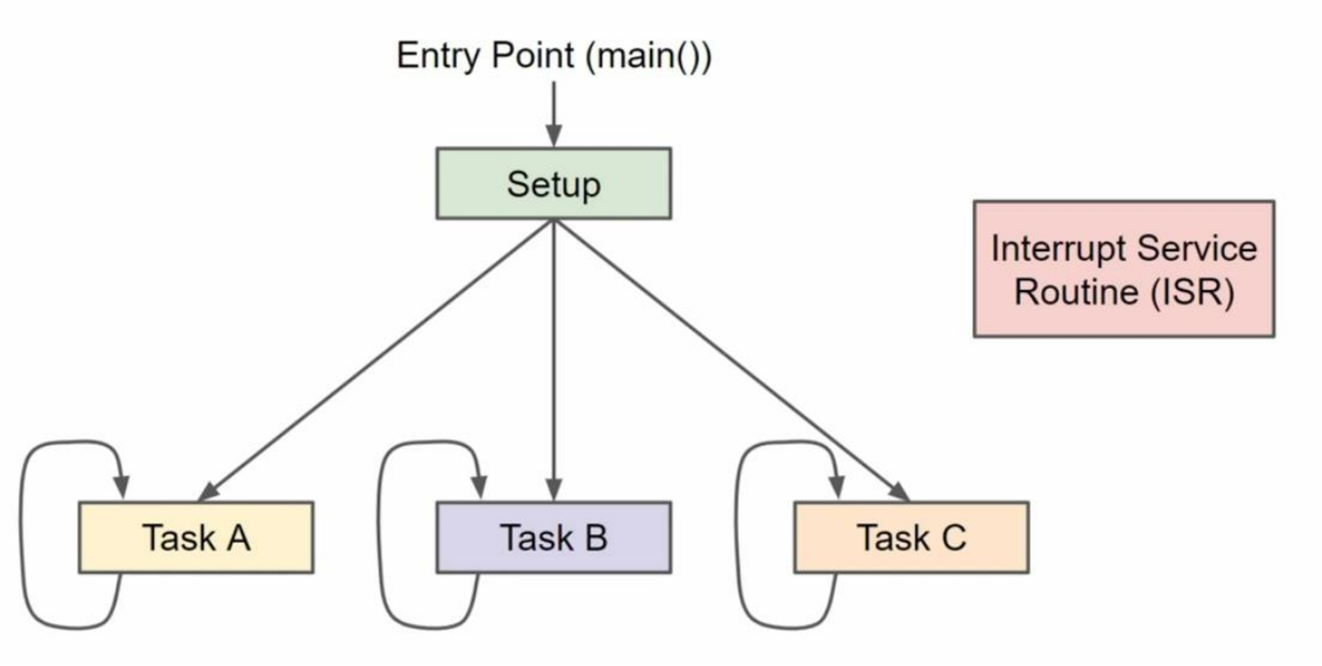
Super Loop



- Bare Metal
- Fácil implementação
- Mais fácil debugar do que sistema RTOS
- Não executa tarefas em paralelo
- Se uma tarefa tomar mais tempo mais que o normal, ocorrerão lags

https://www.youtube.com/watch?v=F321087yYy4&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz

RTOS



- Executa tarefas em paralelo
- Conceito de divisão de tempo de CPU
- Prioridades para as tarefas

https://www.youtube.com/watch?v=F321087yYy4&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz

Arduino x STM32 x ESP32

ATmega 328p
- 16 MHz
- 32 kB flash
- 2 kB RAM

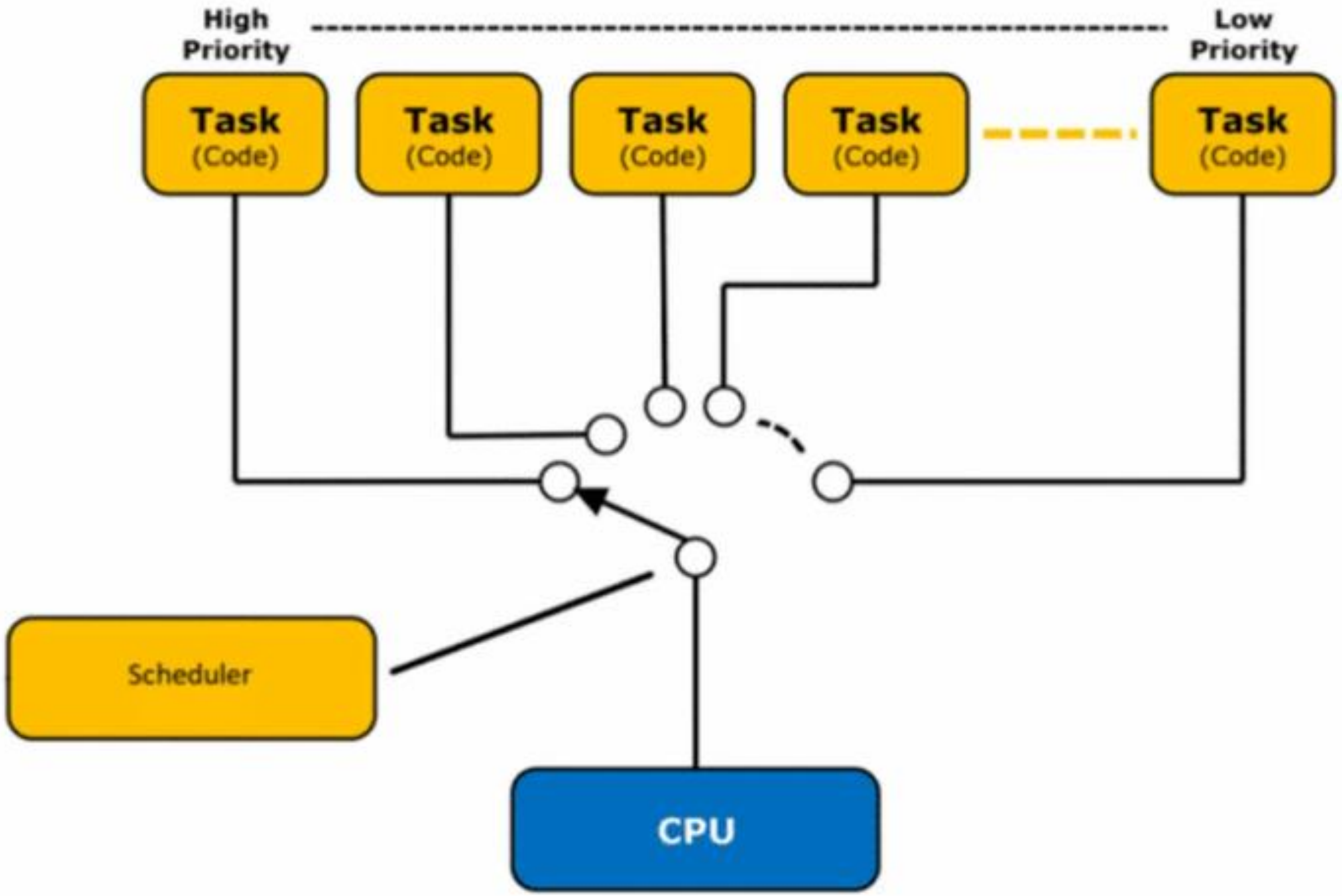
STM32L476RG
- 80 MHz
- 1 MB flash
- 128 kB RAM

ESP-WROOM-32
- 240 MHz (dual core)
- 4 MB flash
- 520 kB RAM

Super Loop → RTOS

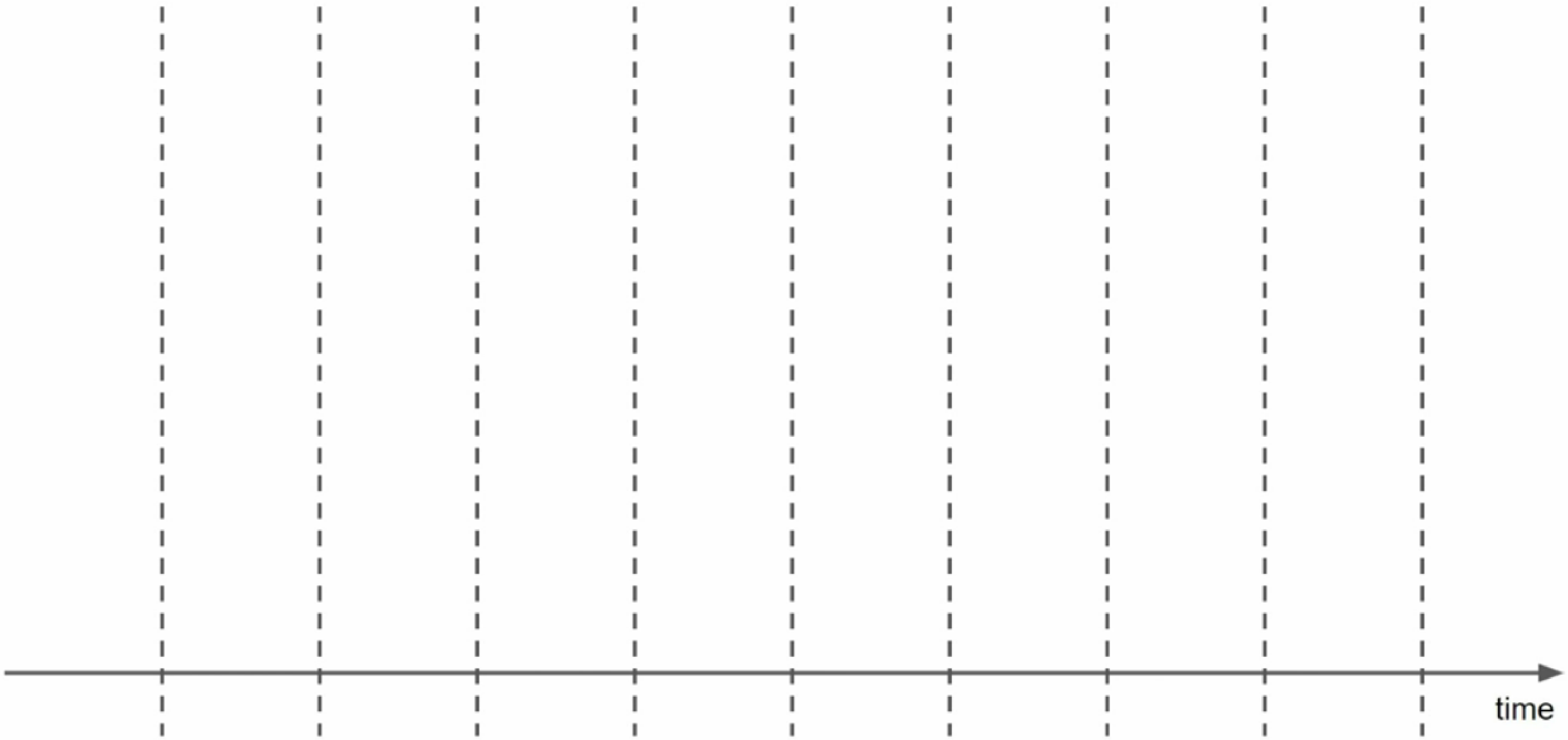
https://www.youtube.com/watch?v=F321087yYy4&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz

Scheduler



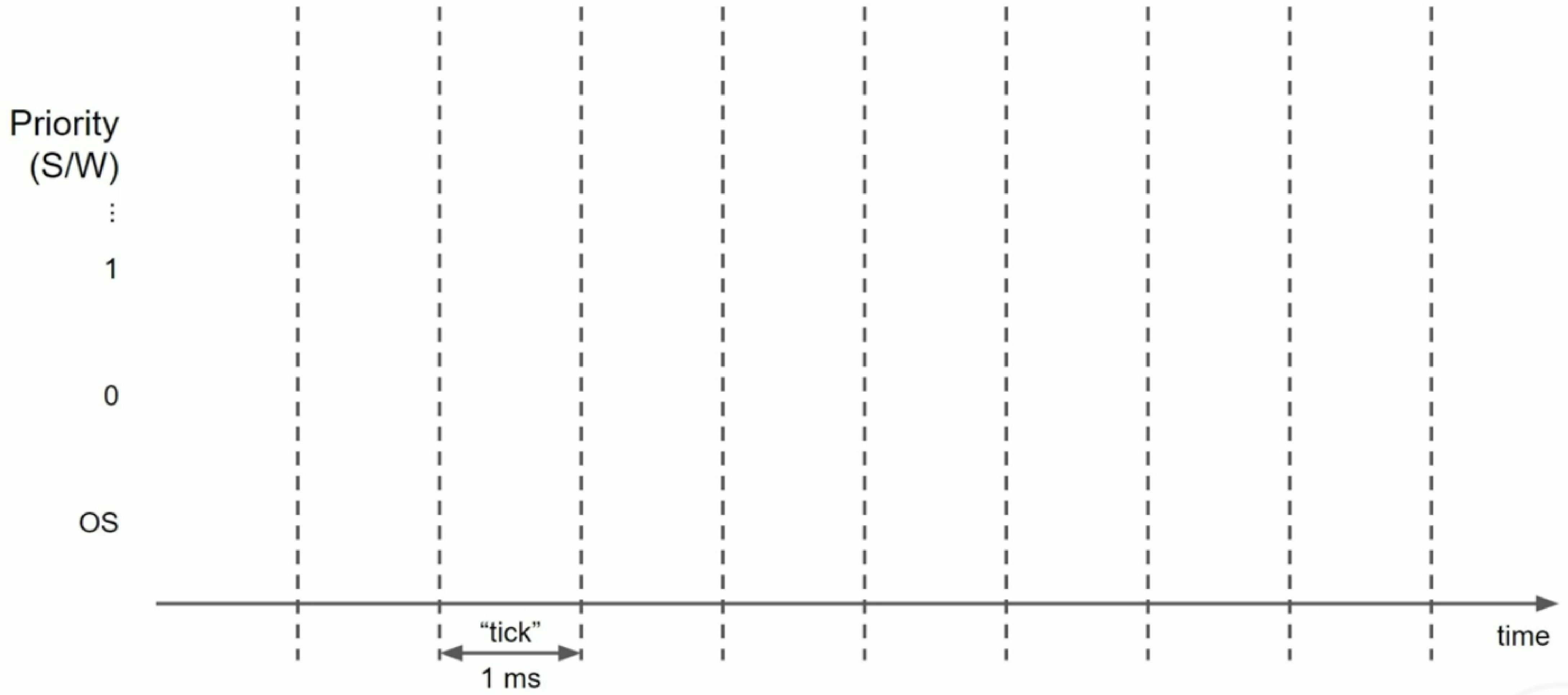
https://www.youtube.com/watch?v=EHdLvtI5DQI

Scheduler and Tasks



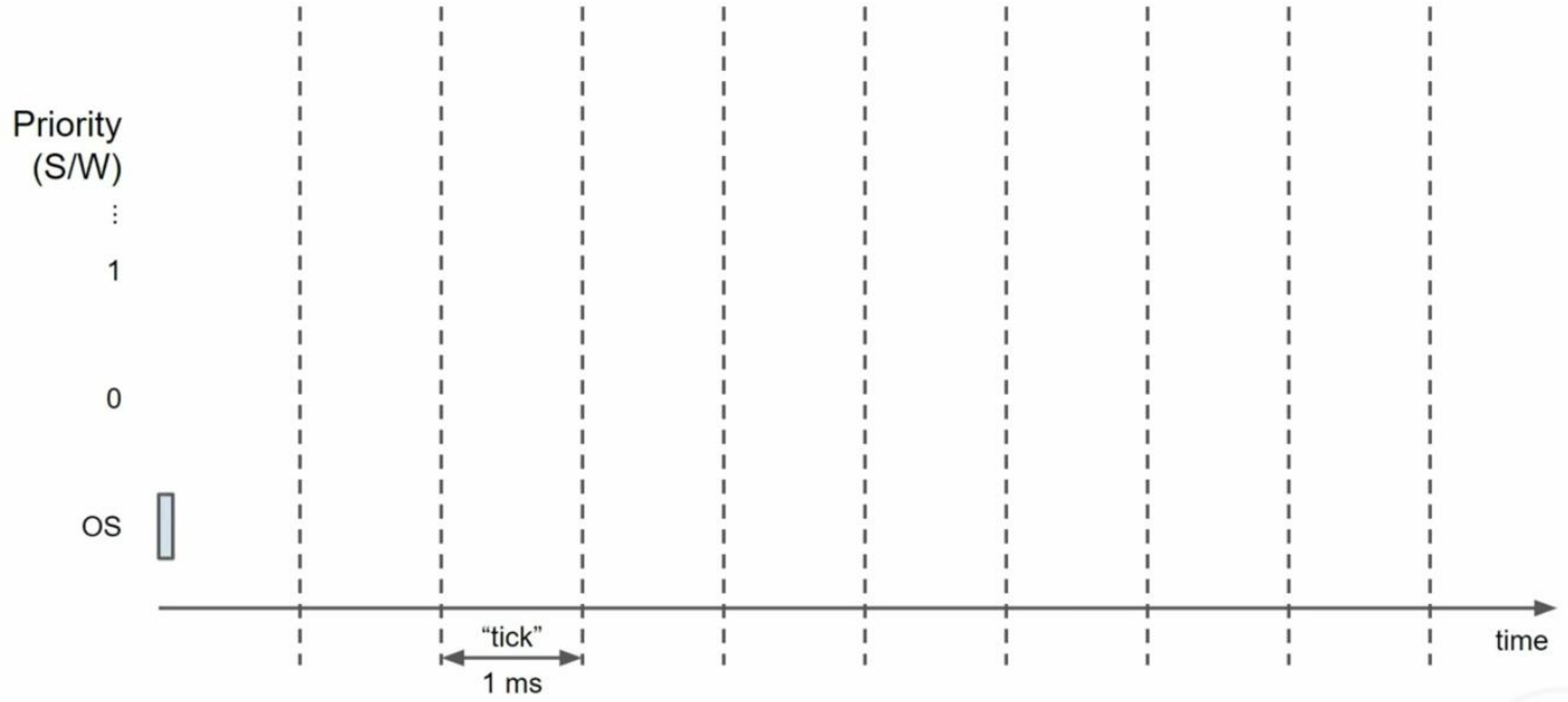
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)



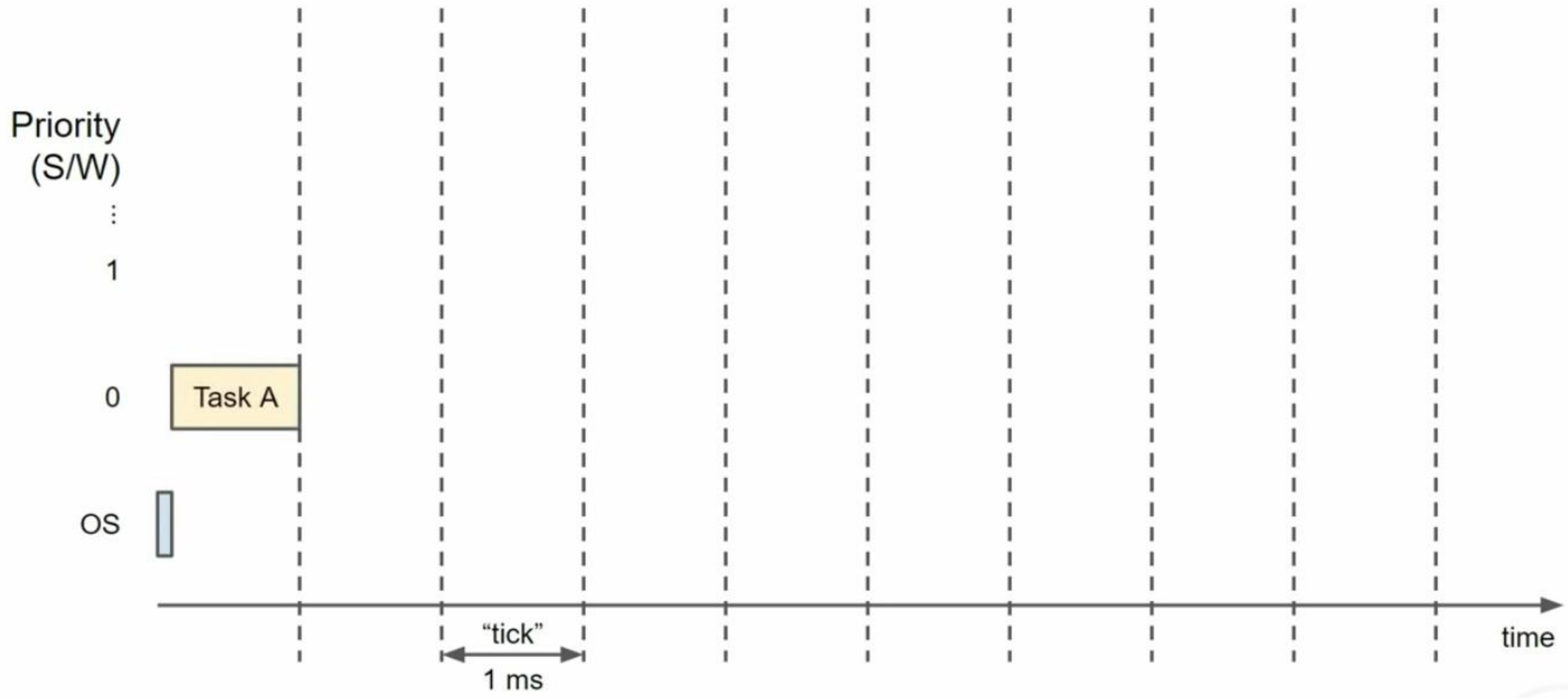
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)



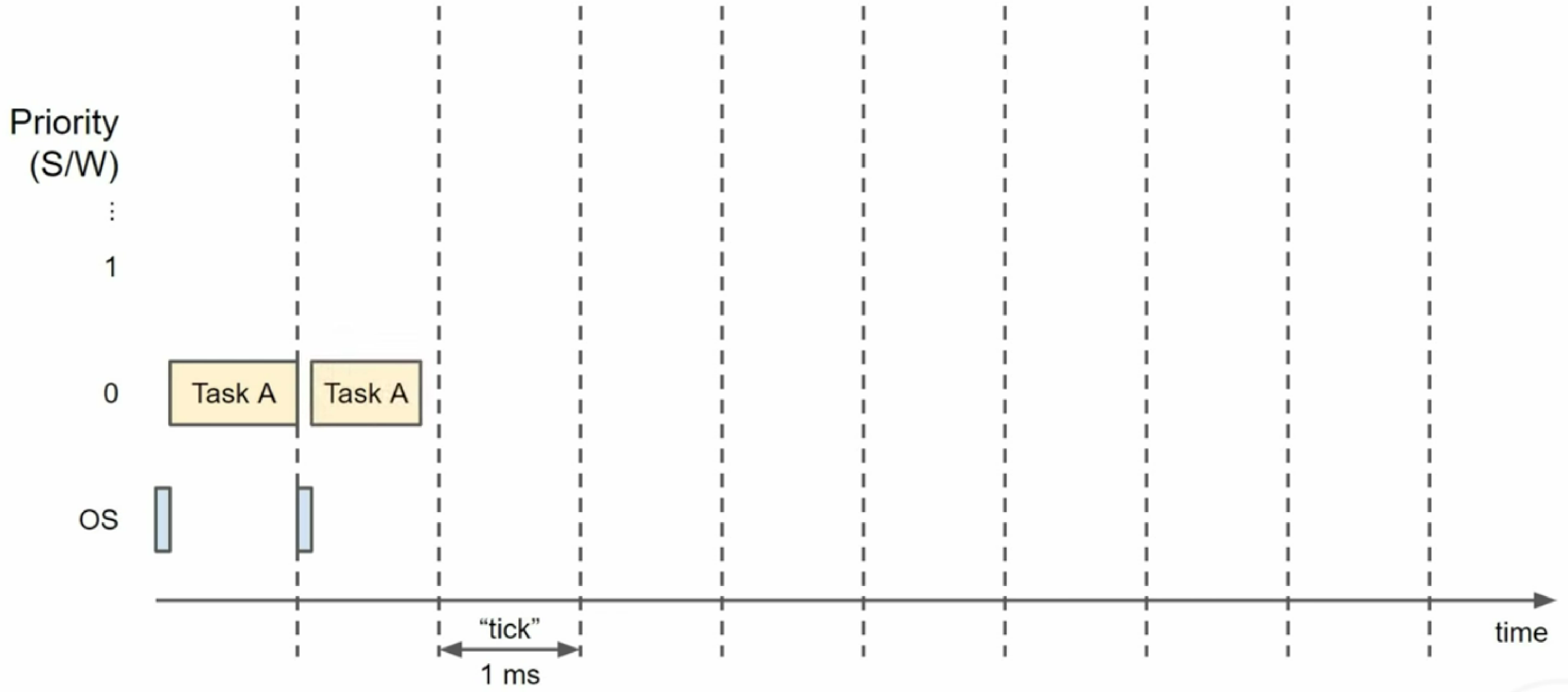
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)



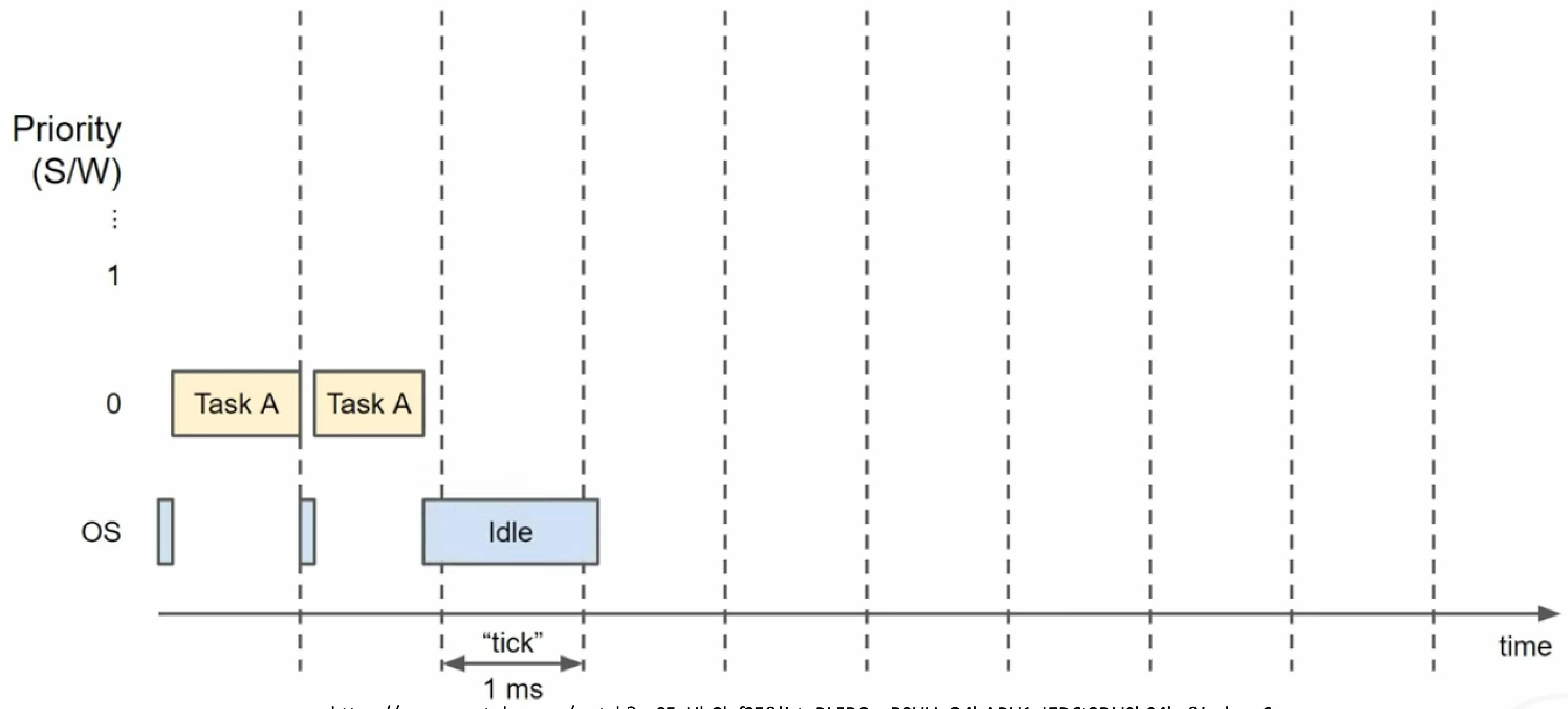
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)



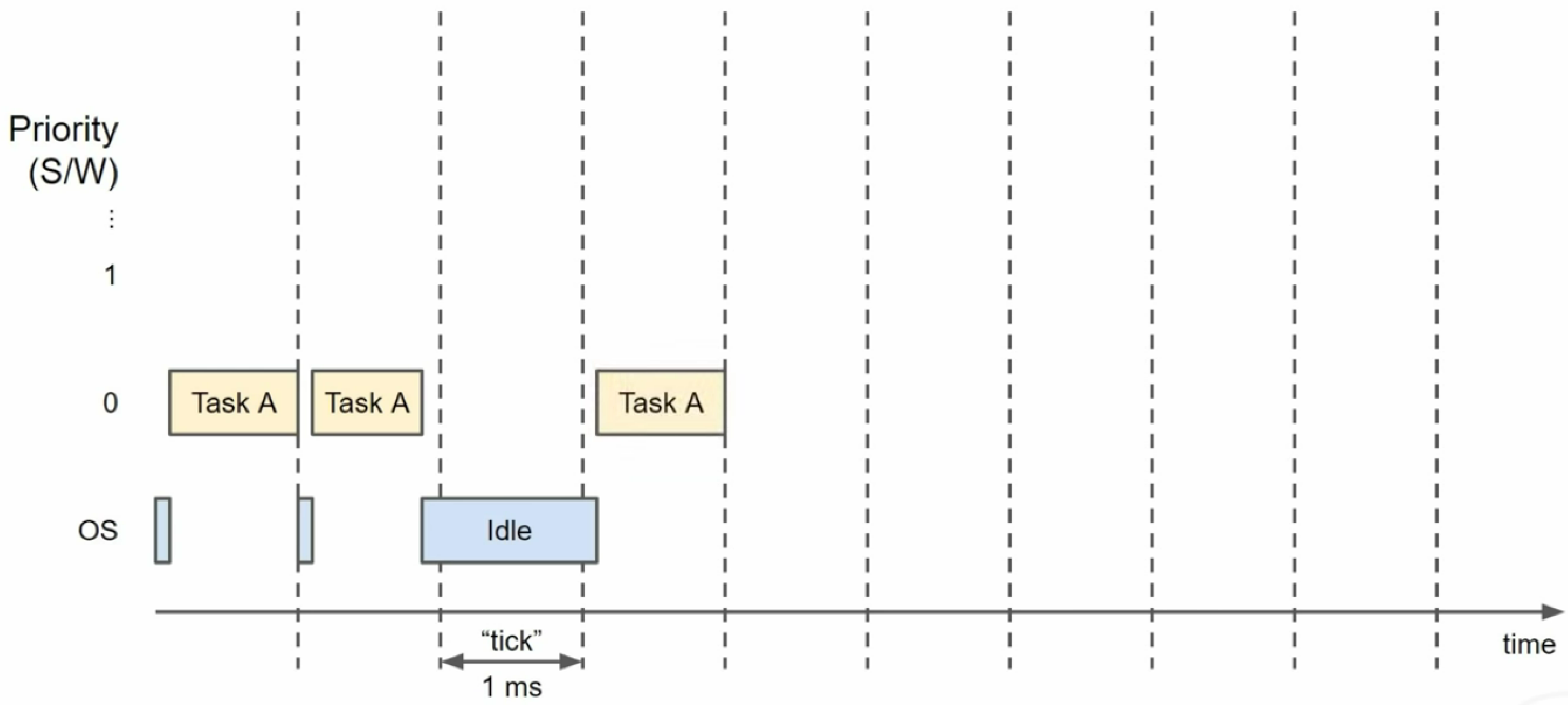
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)



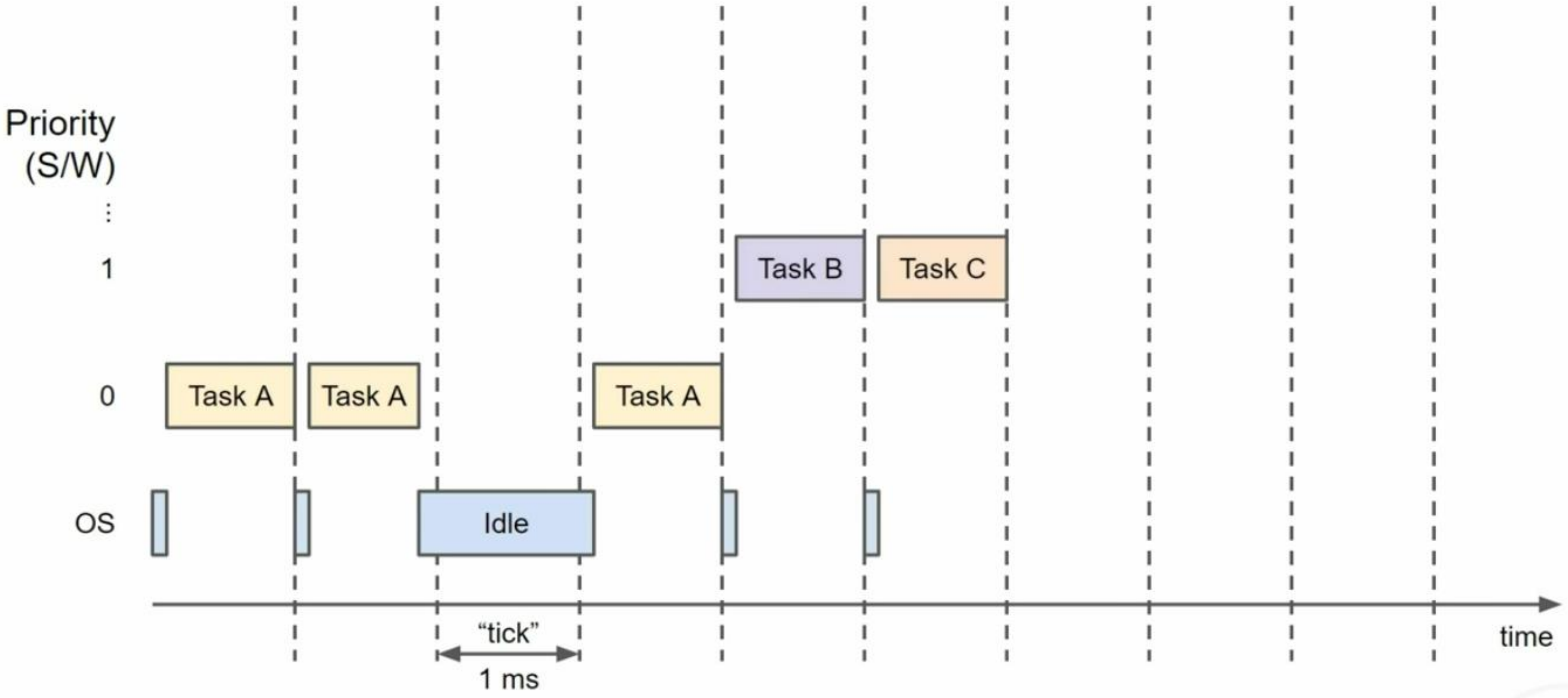
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)



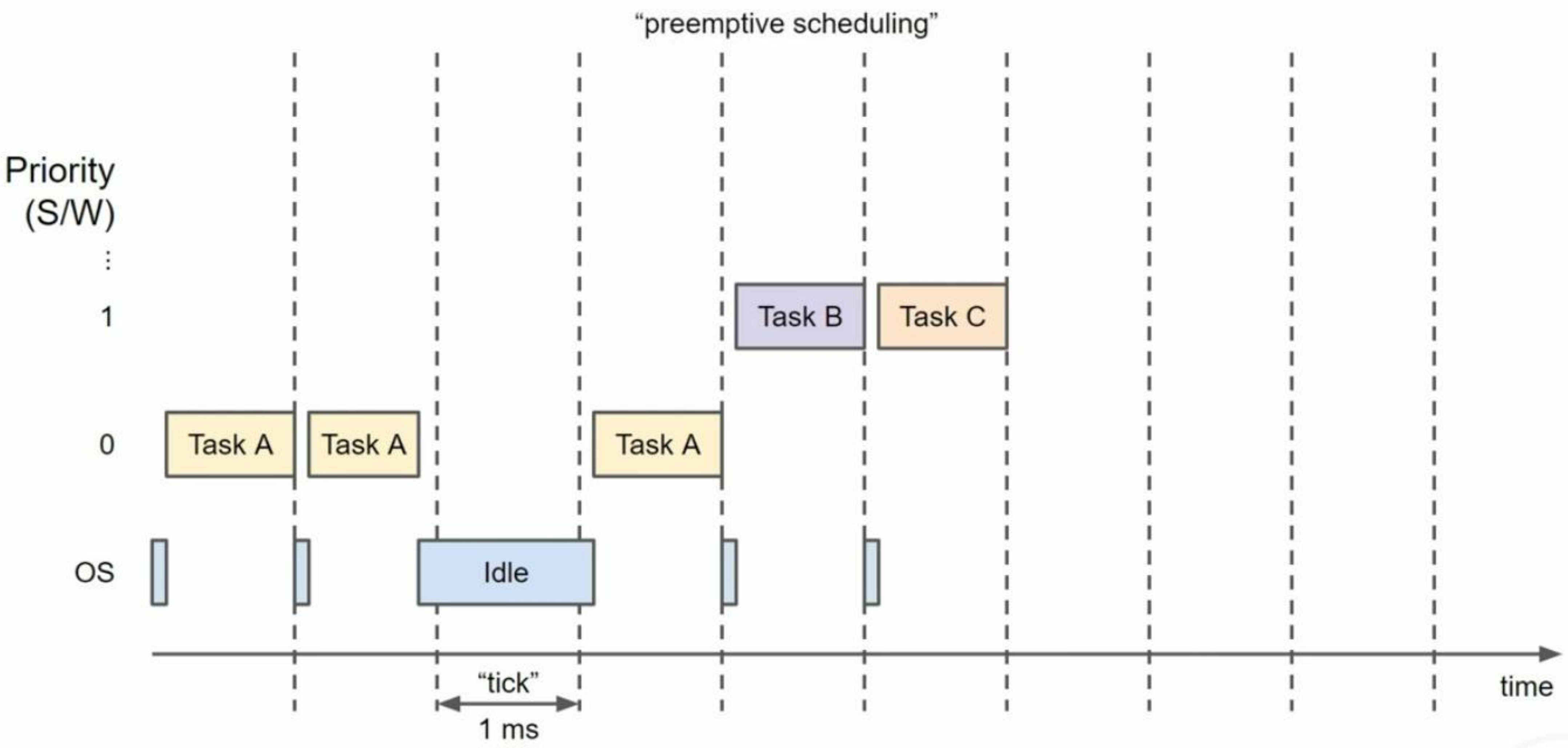
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)



https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)

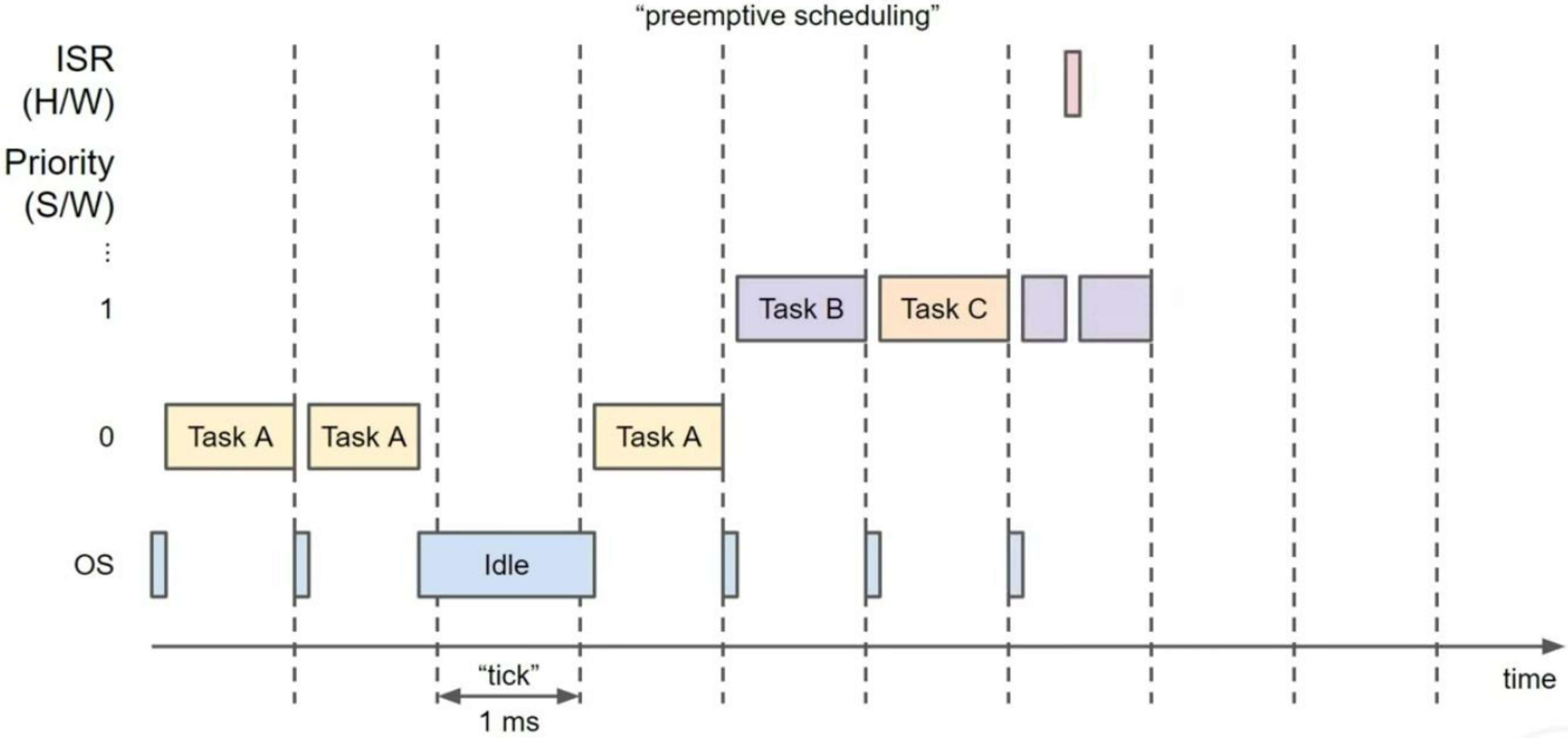


https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Preemptive scheduling

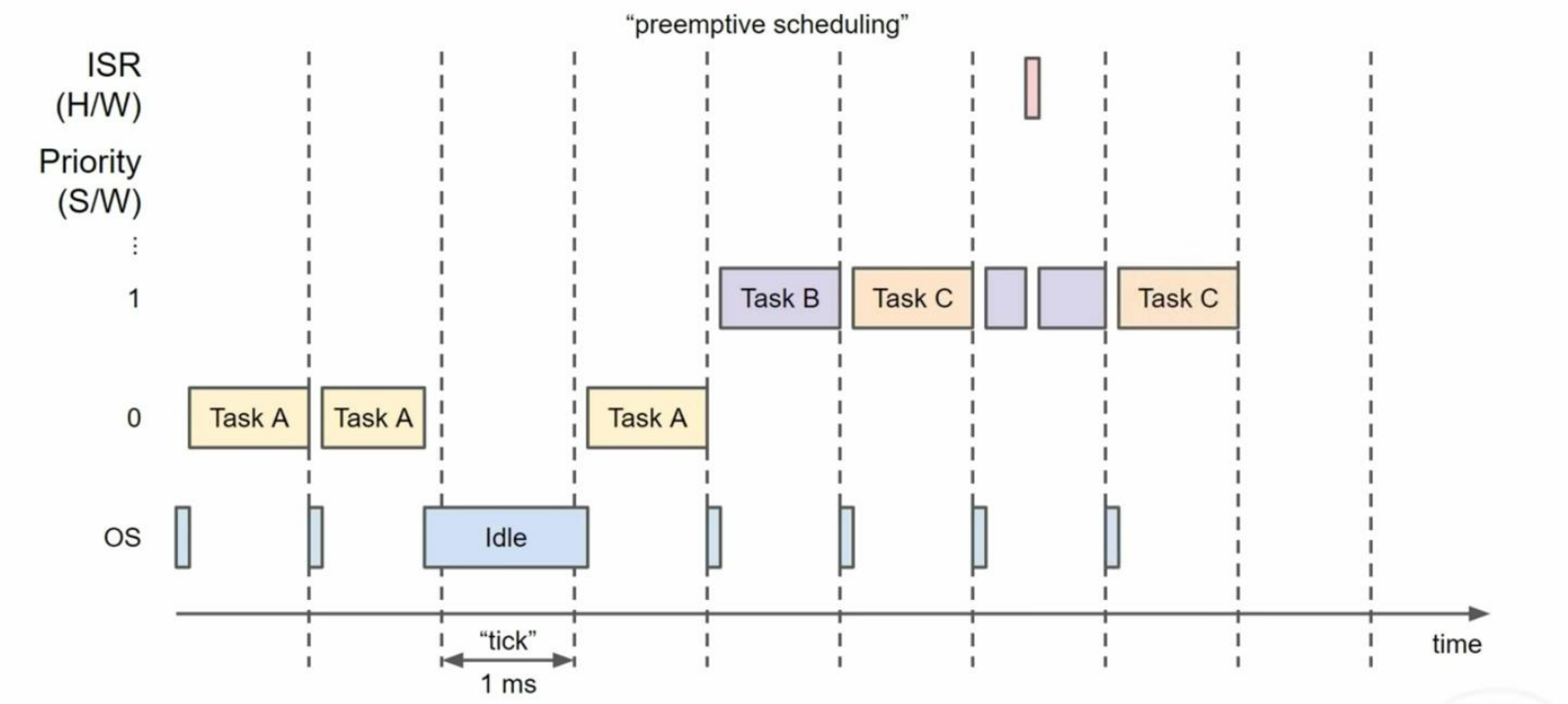
- Agendador preemptivo
- Tempo de processamento da CPU é retirado de uma task (Task A) para ser utilizado em tasks de maior prioridade (Tasks B e C);

Scheduler and Tasks (cont.)



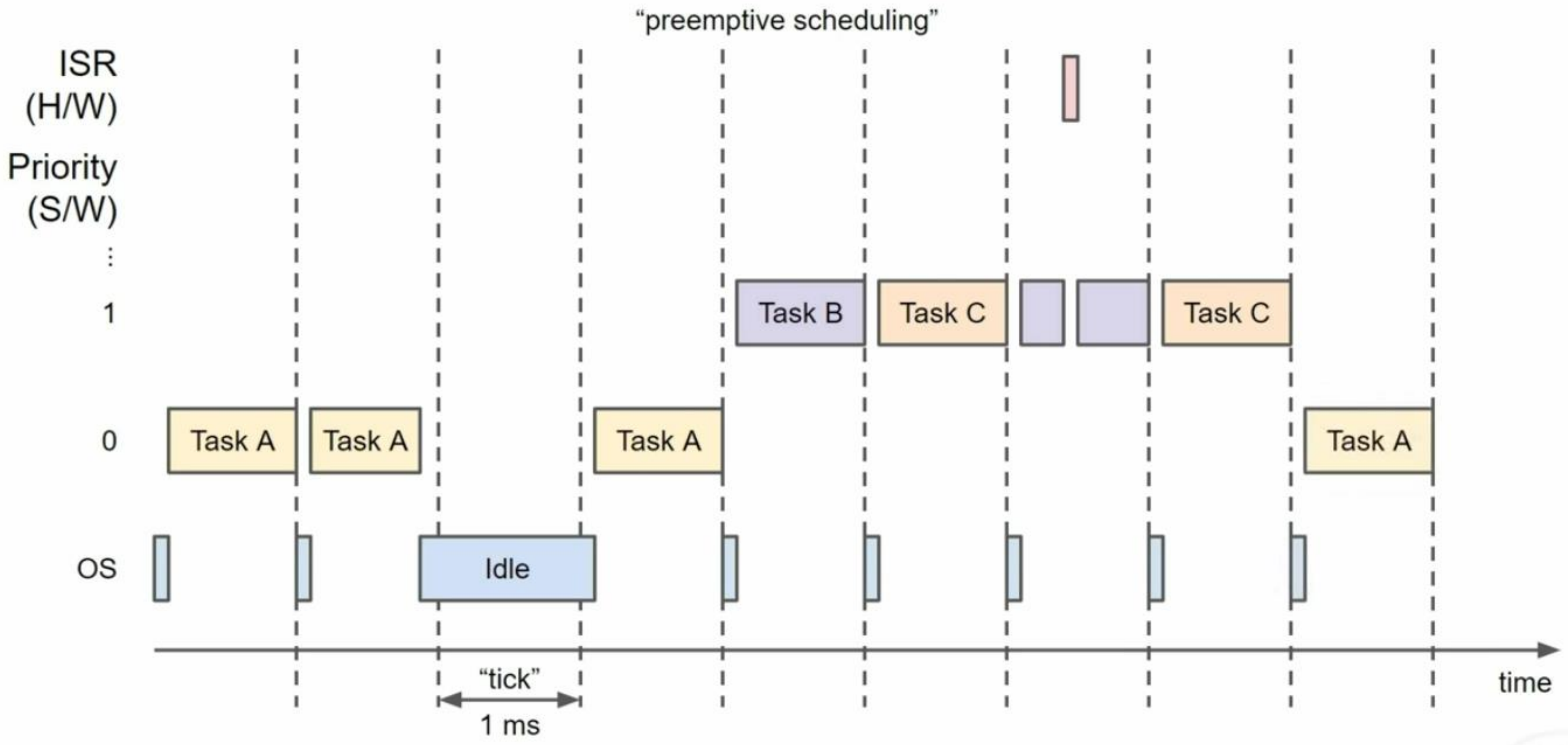
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)



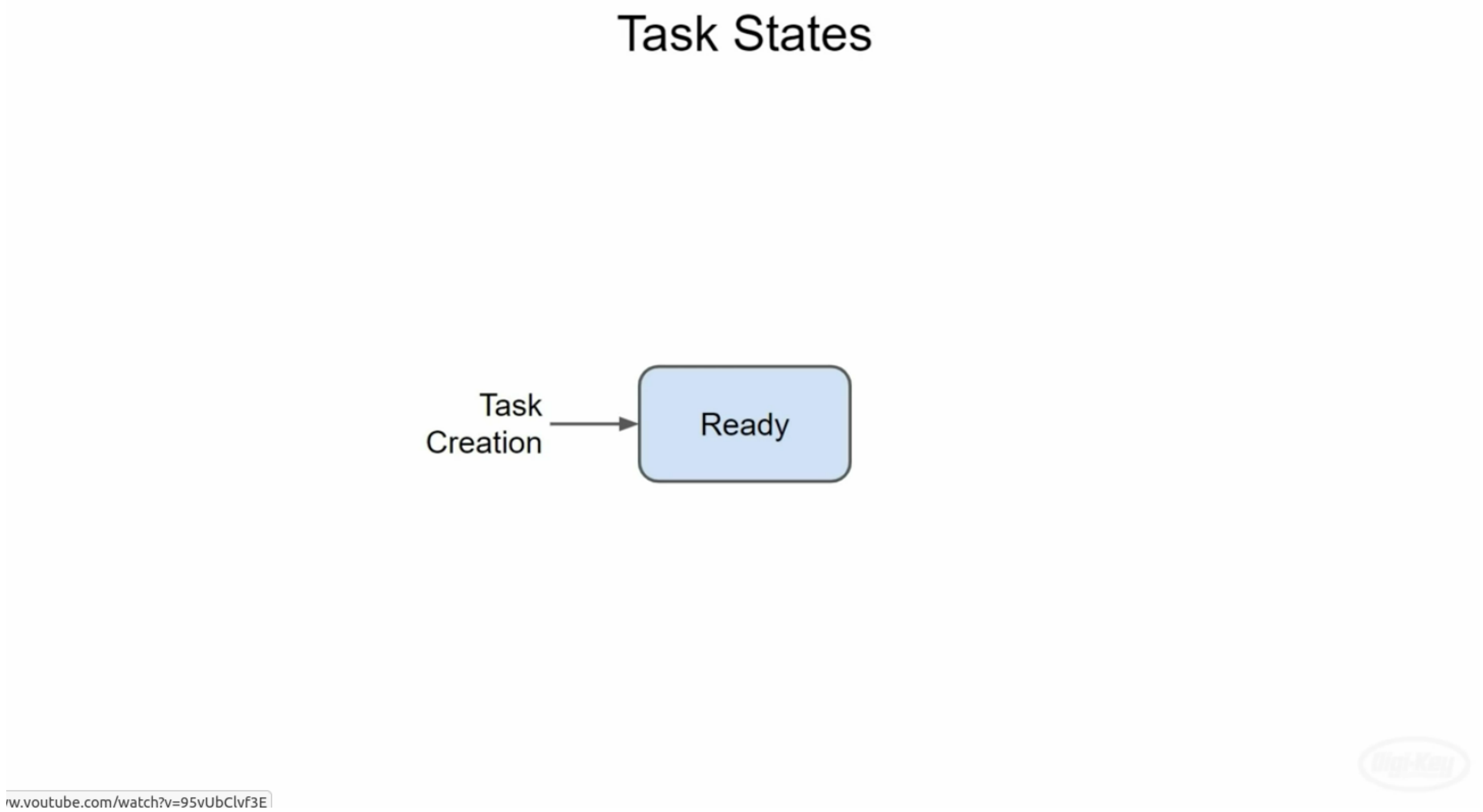
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Scheduler and Tasks (cont.)



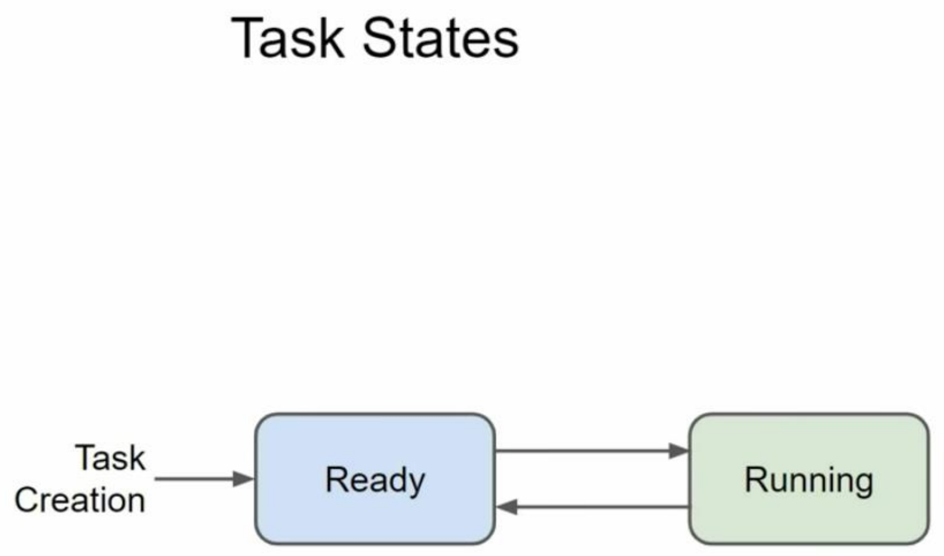
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Estados da task



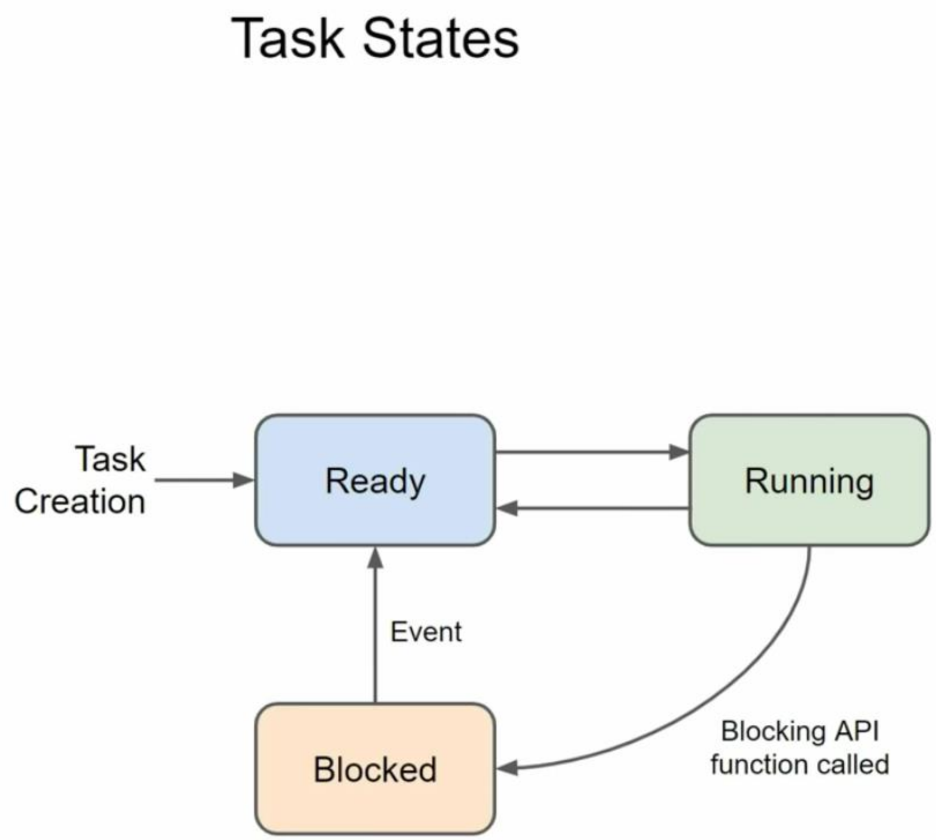
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=4

Estados da task (cont.)



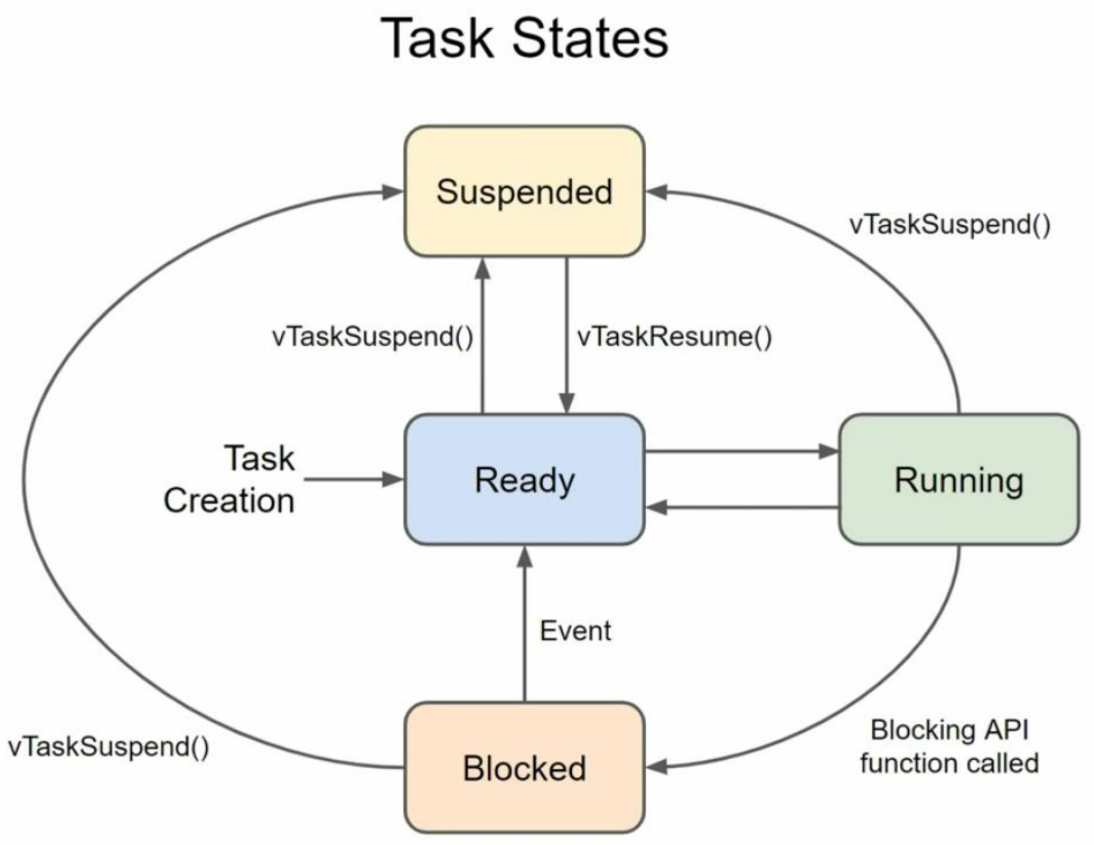
https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=4

Estados da task (cont.)



https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=4

Estados da task (cont.)



https://www.youtube.com/watch?v=95yUbClyf3E&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=4

Tasks

- Tasks é a estrutura do FreRTOS;
- Tasks são responsáveis por certas seções da aplicação
- FreeRTOS é responsável pelo agendamento das tasks, permitindo que uma ou mais tasks rodem ao mesmo tempo

Tasks

```
void task1(void *param)
```

- Não possuem retorno: tipo void
- Parâmetro do tipo ponteiro para void

```
void task1(void *param)
{
    while (true)
    {
        //Executa funções
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    vTaskDelete(NULL);
}
```

- Executam um loop infinito
- Caso não seja mais necessária, a task deve ser deletada

xTaskCreate()

```
BaseType_t xTaskCreate(
TaskFunction_t pvTaskCode,
const char * const pcName,
const uint32_t usStackDepth,
void * const pvParameters,
UBaseType_t uxPriority,
TaskHandle_t * const pvCreatedTask)
```

```
xTaskCreate(&task1, "temperature reading", 2048, NULL, 2, NULL);
```

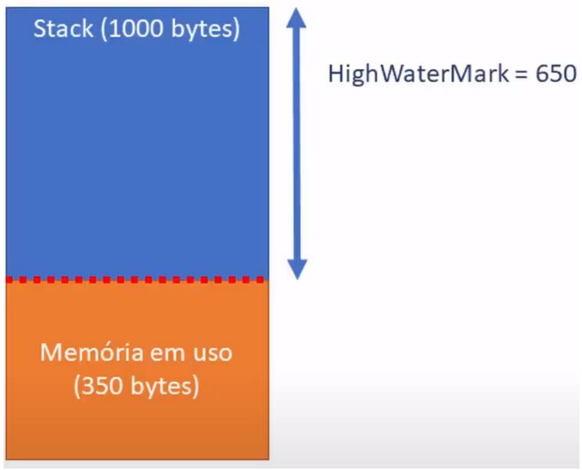
xTaskCreate() (cont.)

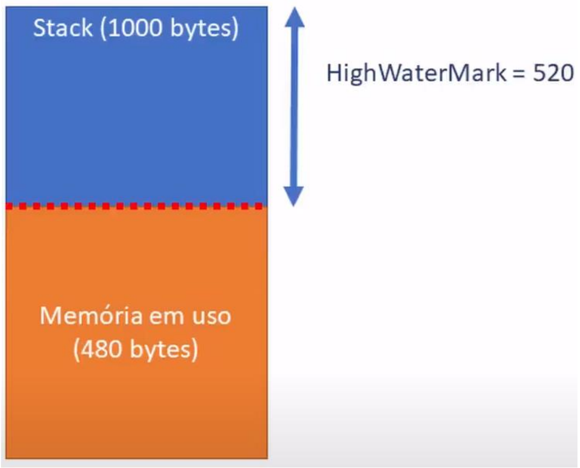
- usStackDepth Quantidade de memória ram que será alocada para a stack da task
  - A stack irá armazenar todas as variáveis locais da tarefa e o contexto da tarefa quando não estiver em execução
  - Necessário alocar memória suficiente, senão ocasionará uma stack overflow e o reset do processador
- uxPriority Tarefas possuem prioridade
  - 0 a 24: maior o número, maior a prioridade da tarefa;

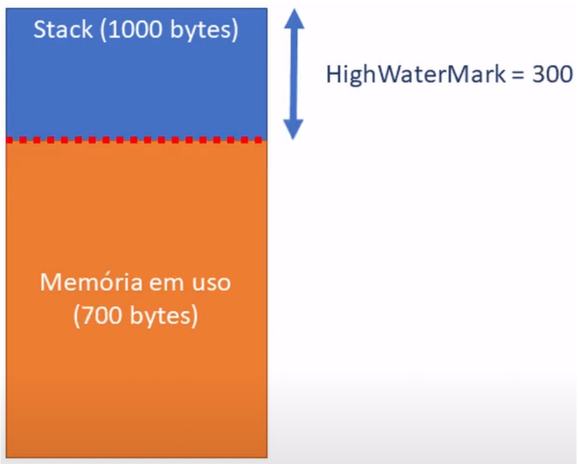
Funções task

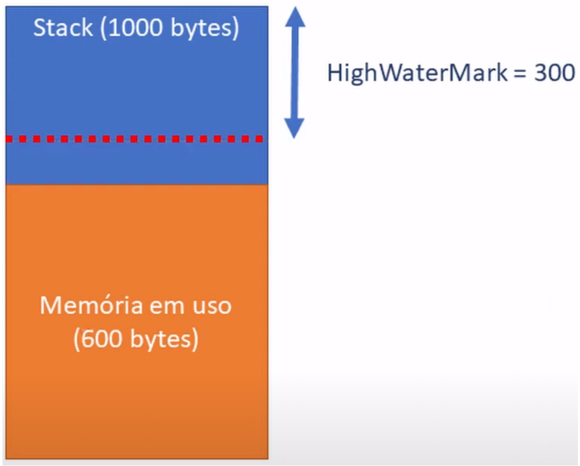
- xPortGetCoreID();
- vTaskDelete(xTaskToDelete);

- uxTaskGetStackHighWaterMark();









https://www.youtube.com/watch?v=LHuJxJl3CX4&list=PLHvD4LbjH0y0JqU3SHEGoRgLhLqo7Tlxx&index=6

Exemplo 1

```
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_log.h"

const char *TAG = "tasks.c"
```

```
void task1(void *param)
{
    while (true)
   {
      ESP_LOGI(TAG, "core %d/line %d/%s/reading temperature/%d",
      xPortGetCoreID(), __LINE__, __func__,
      uxTaskGetStackHighWaterMark(NULL));
      vTaskDelay(1000 / portTICK_PERIOD_MS);
   }
   vTaskDelete(NULL);
}
```

Exemplo 1 (cont.)

```
void task2()
{
  while (true)
  {
    ESP_LOGW(TAG, "core %d/line %d/%s/reading humidity/%d",
          xPortGetCoreID(), __LINE__, __func__,
          uxTaskGetStackHighWaterMark(NULL));
    vTaskDelay(2000 / portTICK_PERIOD_MS);
  }
  vTaskDelete(NULL);
}
```

```
void app_main(void)
{
  ESP_LOGI(TAG, "core %d/line %d/%s/starting ", xPortGetCoreID(), __LINE__, __func__);
}
```

Exemplo 1 (cont.)

```
void app_main(void)
{
    ESP_LOGI(TAG, "core %d/line %d/%s/starting ", xPortGetCoreID(), __LINE__, __func__);
    task1(NULL);
    task2(NULL);
}
```

Exemplo 1 (cont.)

```
xTaskCreate(&task1, "temperature reading", 2048, NULL, 2, NULL);
xTaskCreate(&task2, "humidity reading", 2048, NULL, 2, NULL);
```

xTaskCreatePinnedToCore()

```
xTaskCreatePinnedToCore(&task2, "humidity reading",
2048,                                   NULL, 2, NULL,
APP_CPU_NUM);
```

- Define qual o processador irá executar a task
  - #define PRO_CPU_NUM (0)
  - #define APP_CPU_NUM (1)

```
BaseType_t xTaskCreatePinnedToCore(
TaskFunction_t pvTaskCode,
const char * const pcName,
const uint32_t usStackDepth,
void * const pvParameters,
UBaseType_t uxPriority,
TaskHandle_t * const pvCreatedTask,
const BaseType_t xCoreID)
```

Exemplo 1.1

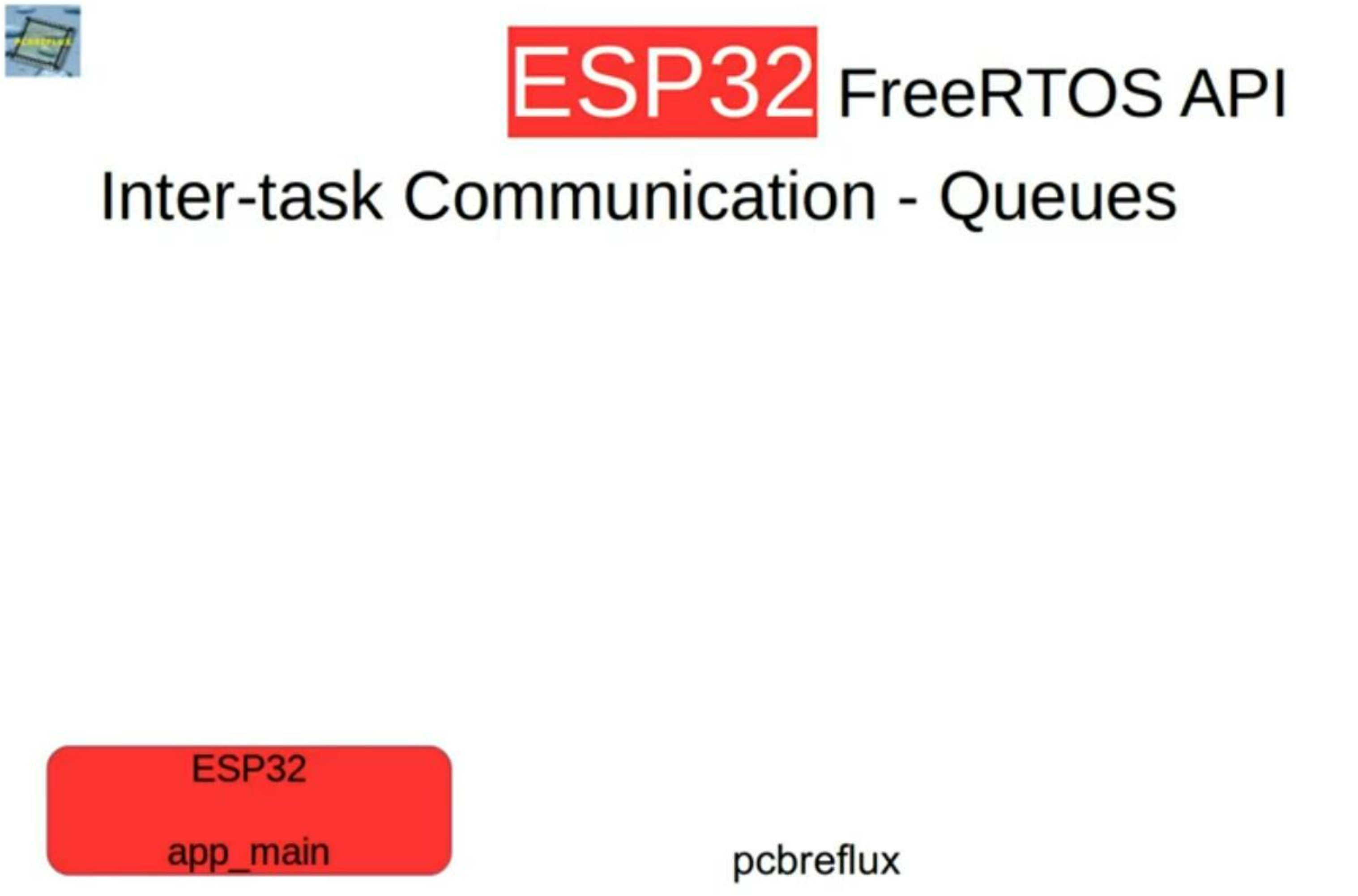
```
xTaskCreatePinnedToCore(&task1, "temperature reading", 2048, NULL, 2, NULL, 0);
xTaskCreatePinnedToCore(&task2, "humidity reading", 2048, NULL, 2, NULL, APP_CPU_NUM);
```

Proteção de recursos compartilhados e sincronização de Threads

- Queue: passagem de mensagens (dados) entre threads;
  - Modo de transferência de dados de uma task para outra

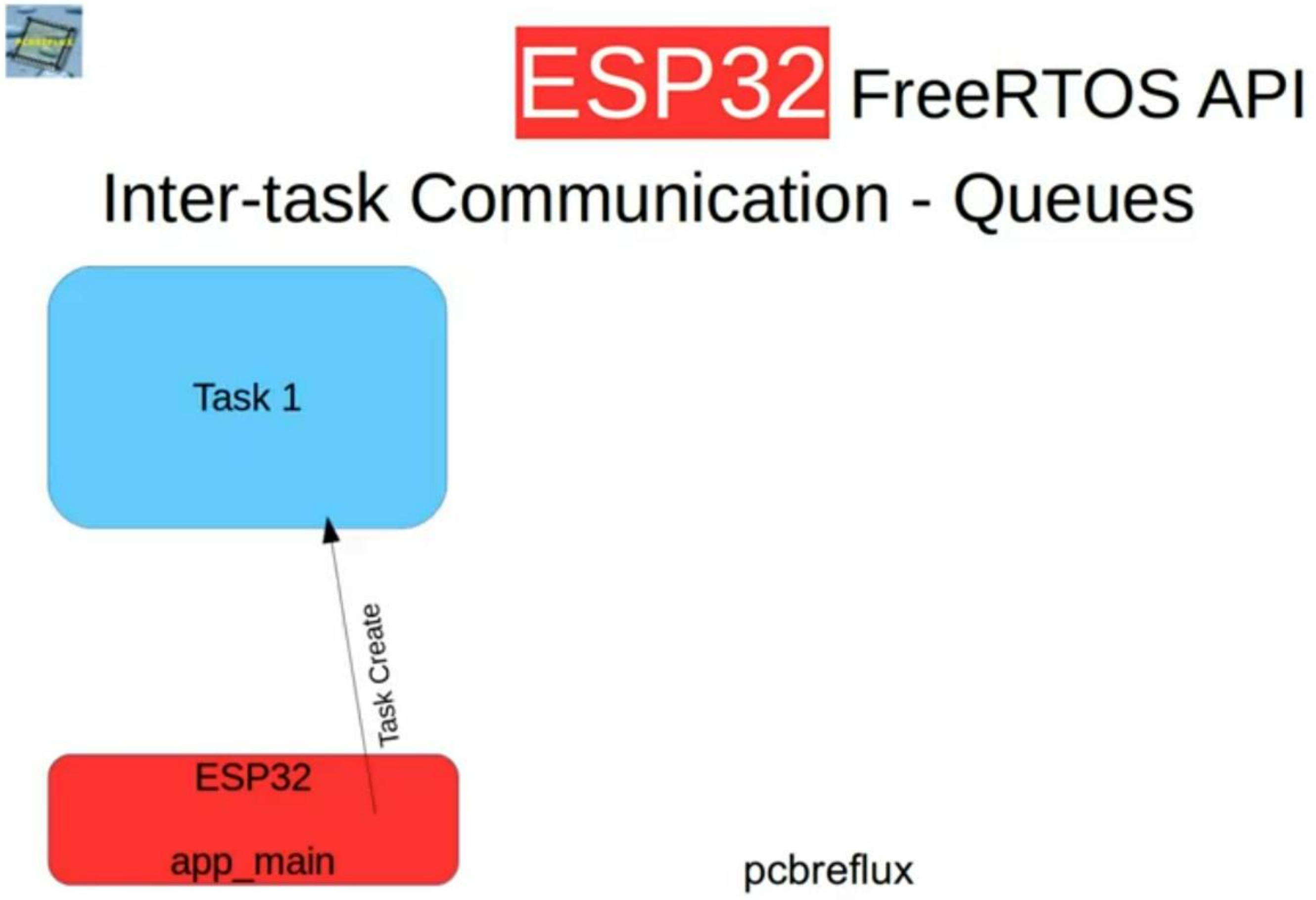
- Mutex (MUtual EXclusion); permite uma única thread a entrar em uma seção crítica do código
  - Somente a task que bloqueou pode desbloquear

Queue



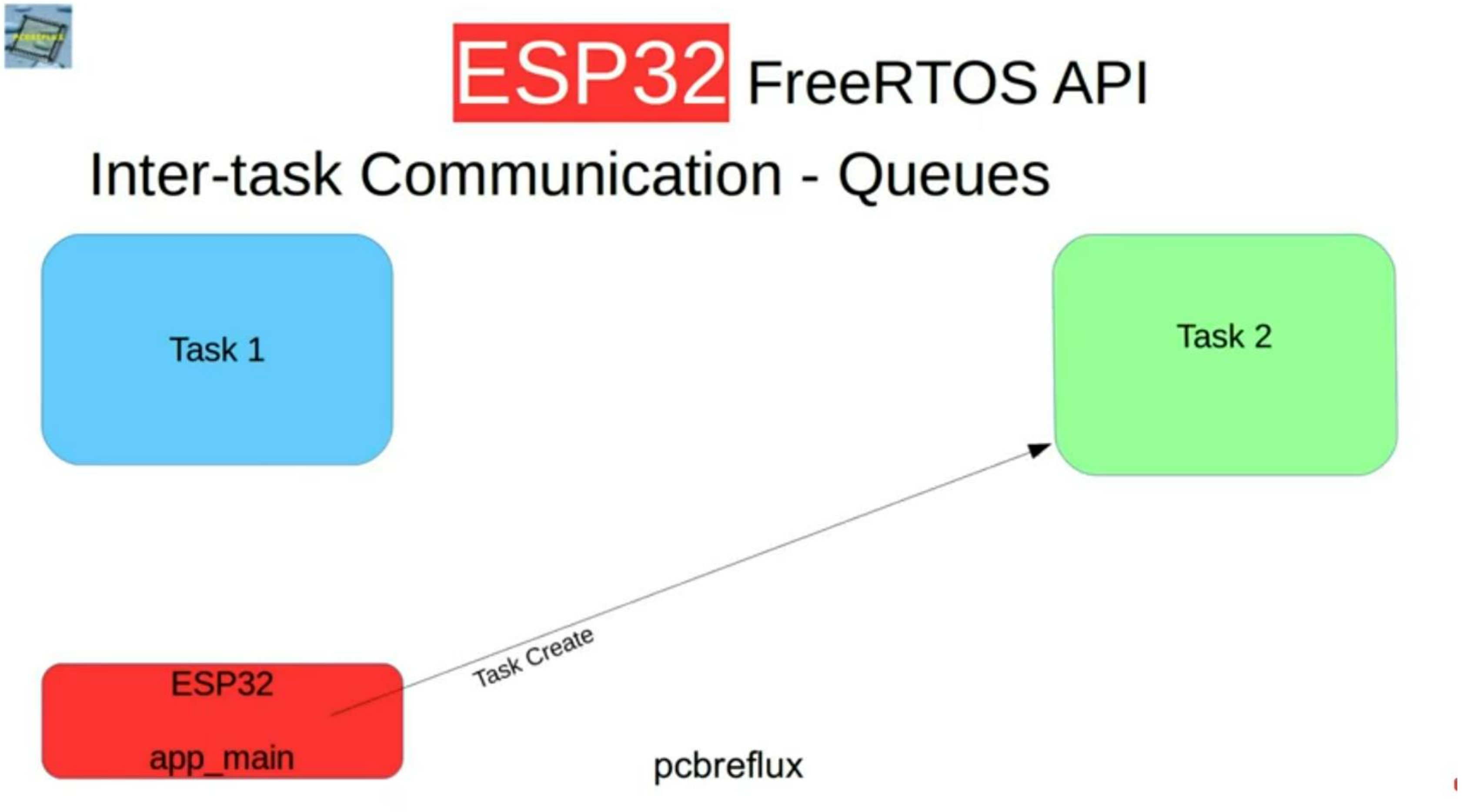
https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Queue (cont.)



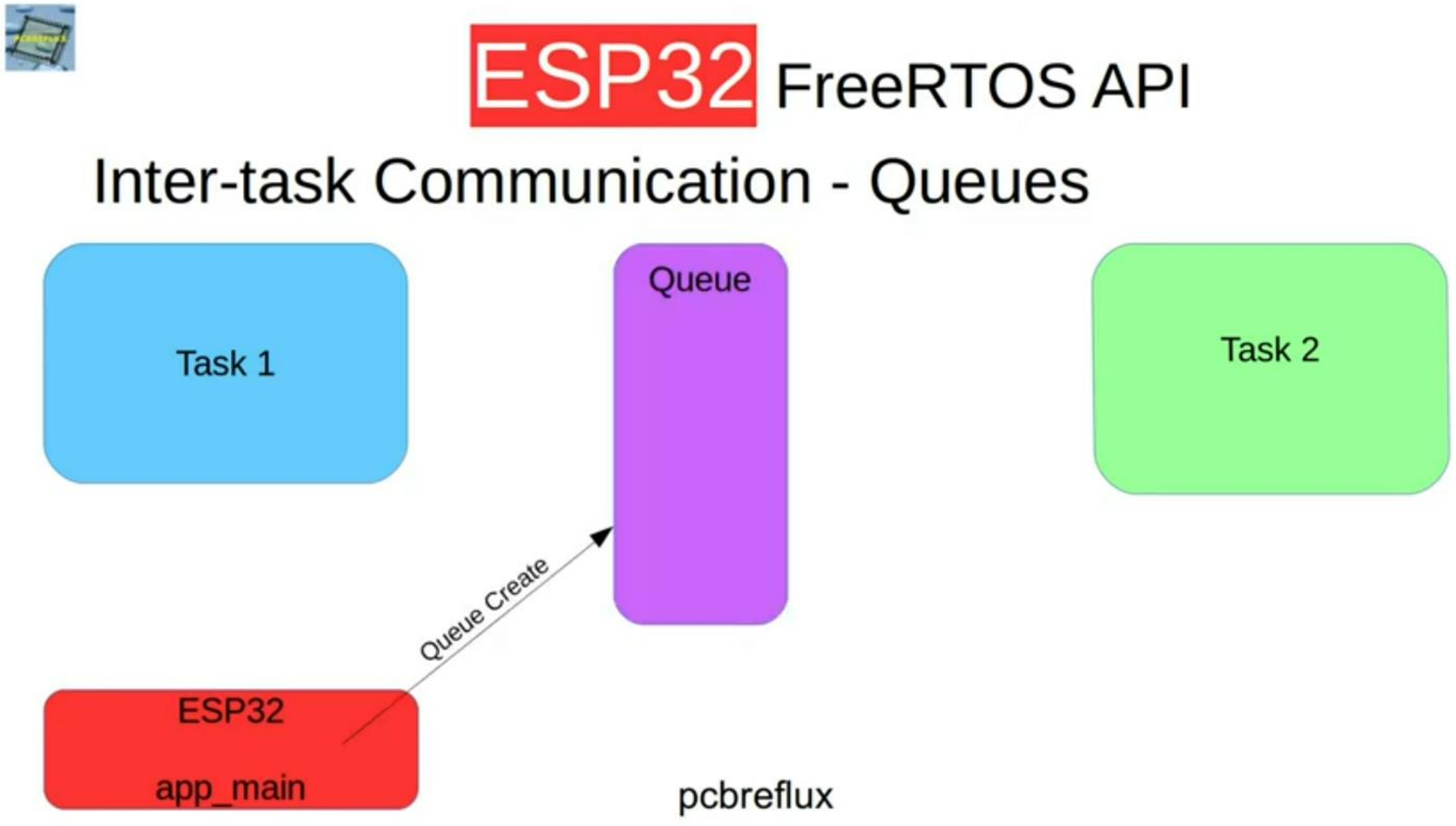
https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Queue (cont.)



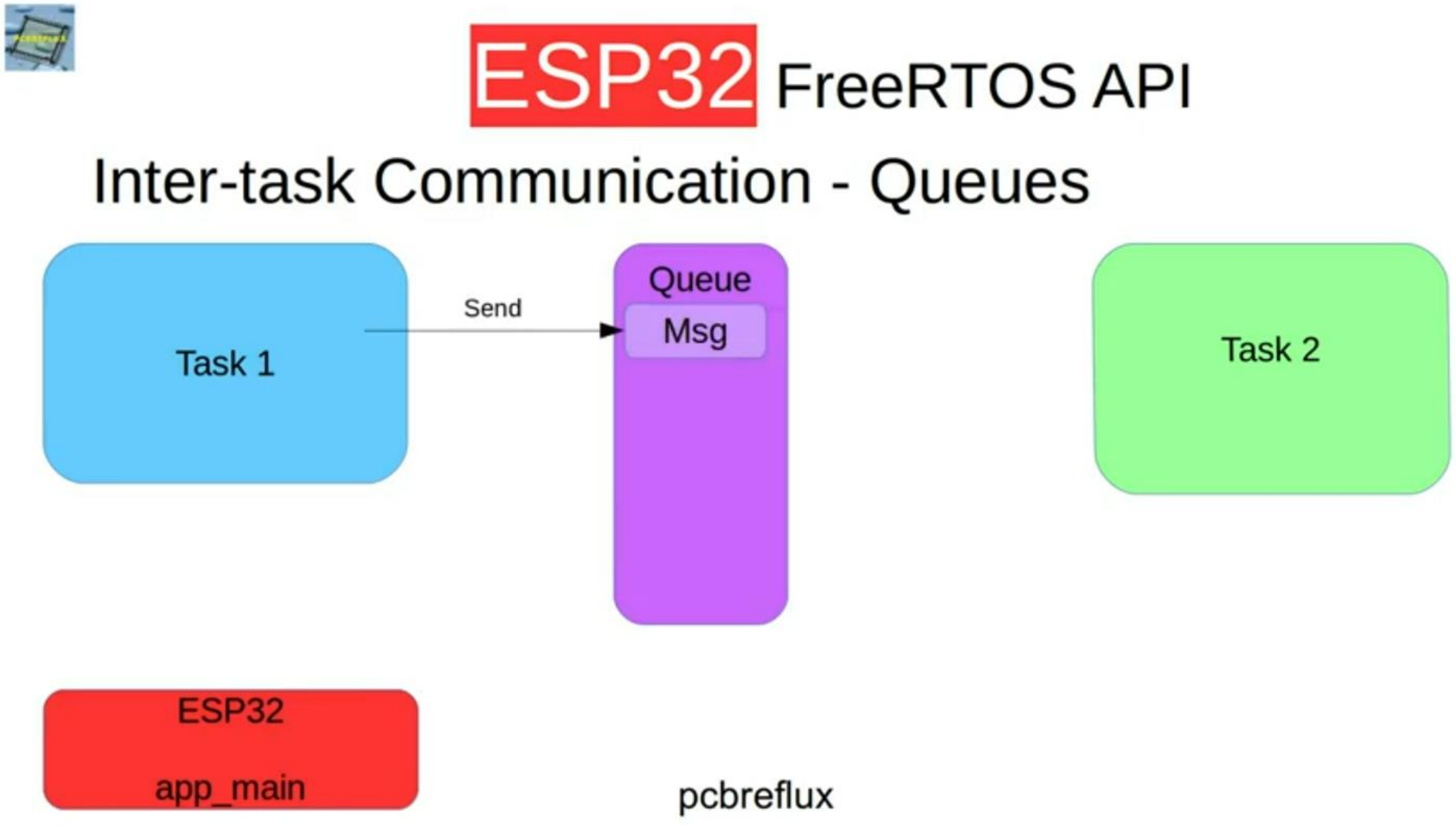
https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Queue (cont.)



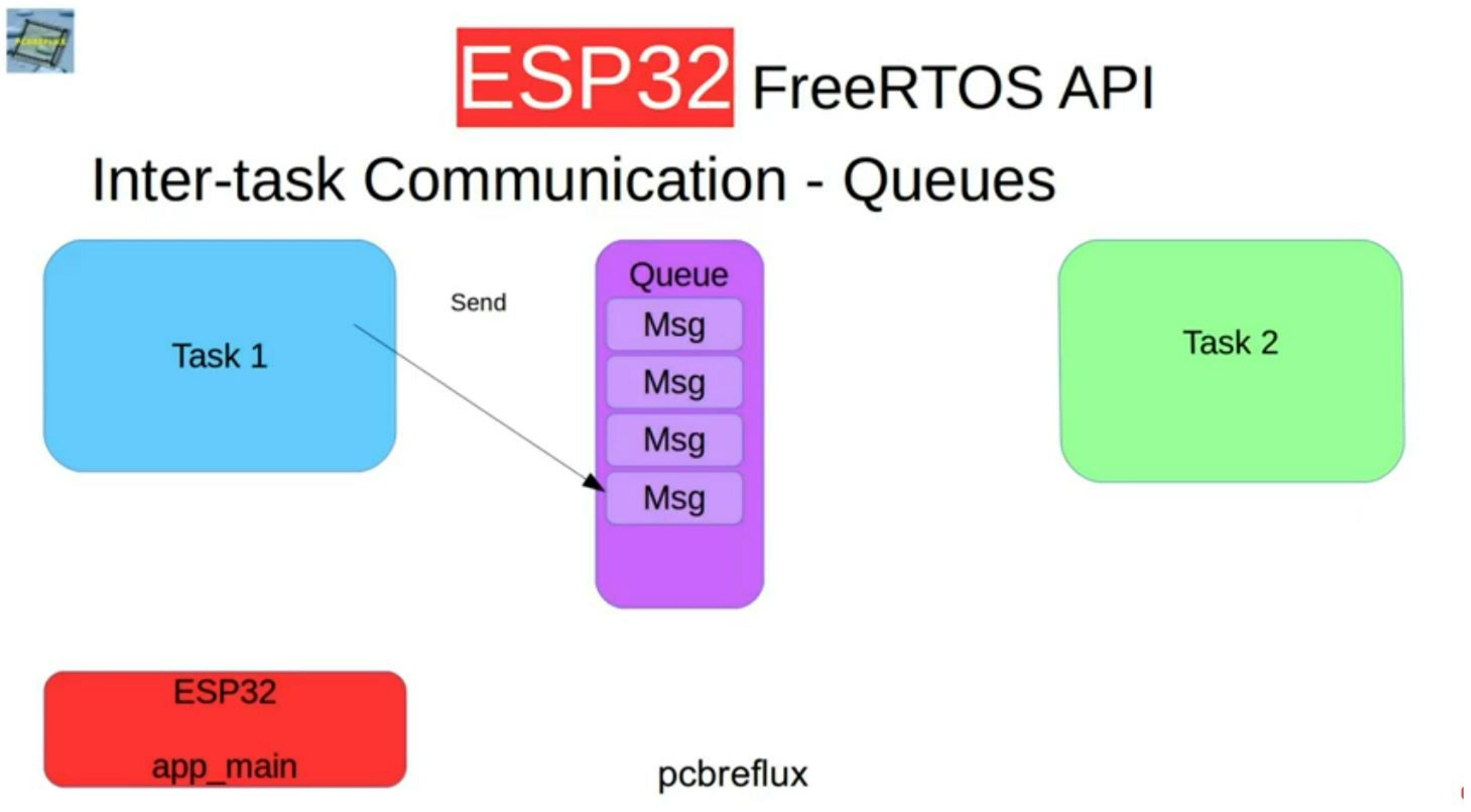
https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Queue (cont.)



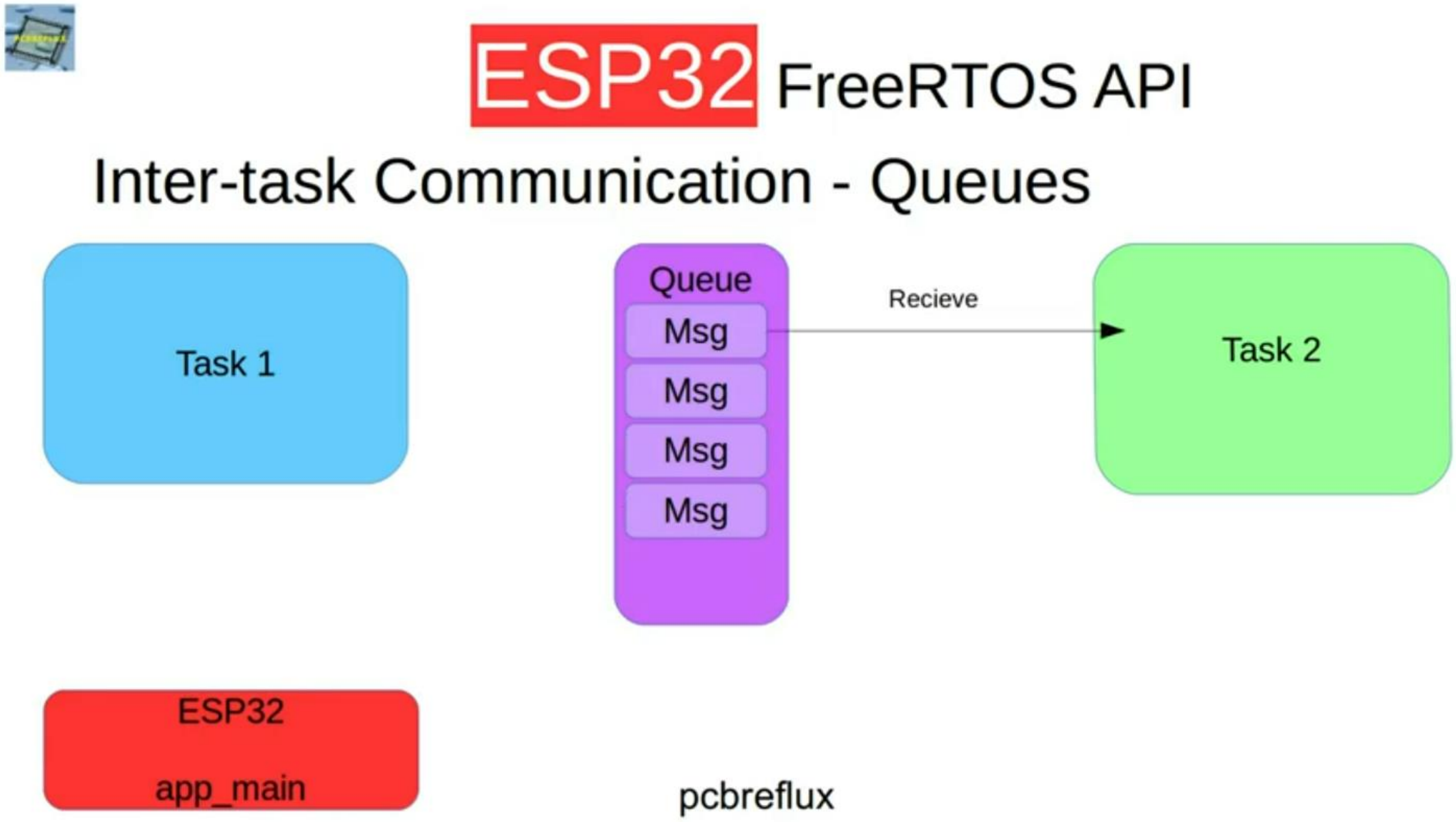
https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Queue (cont.)



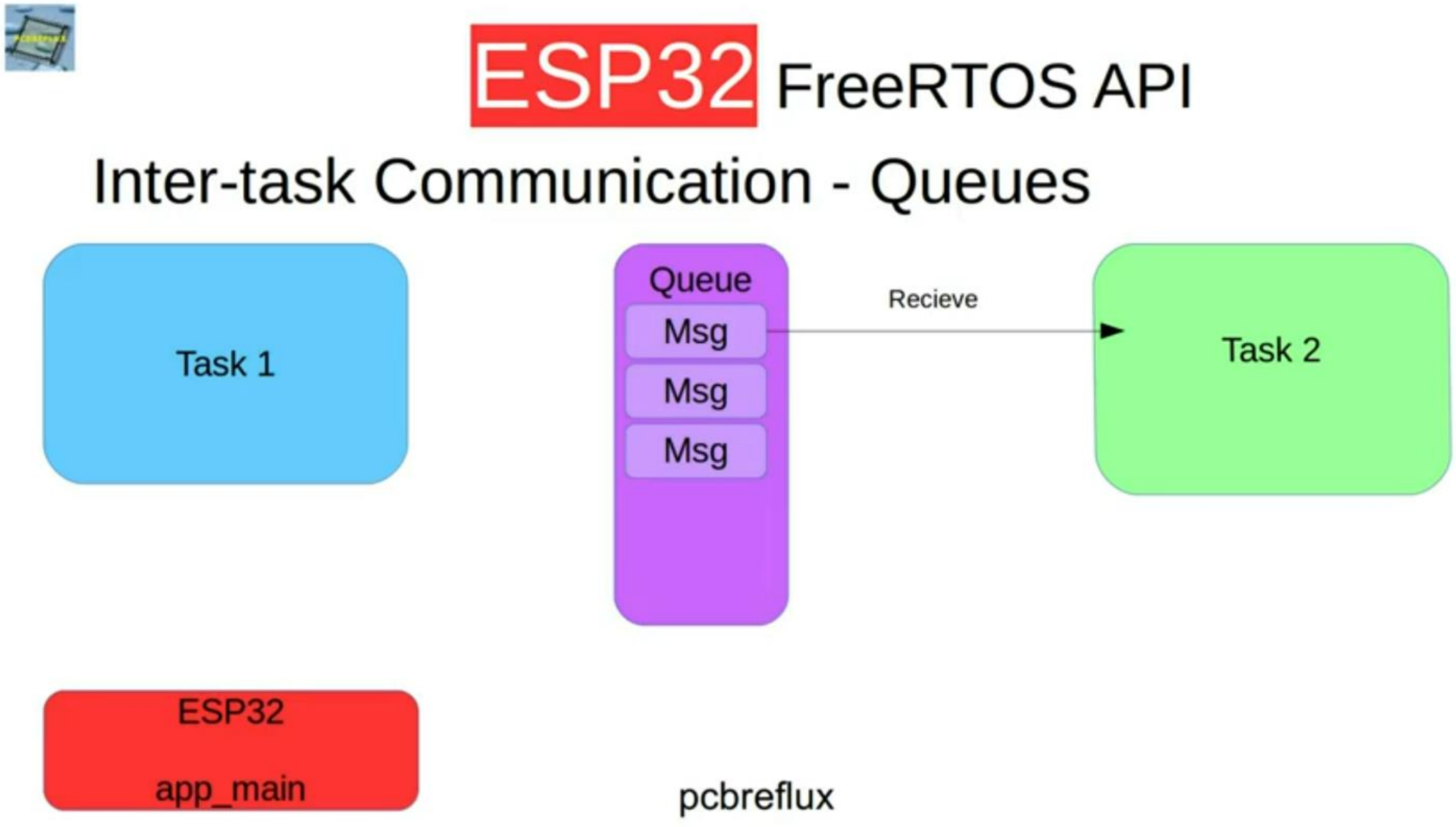
https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Queue (cont.)



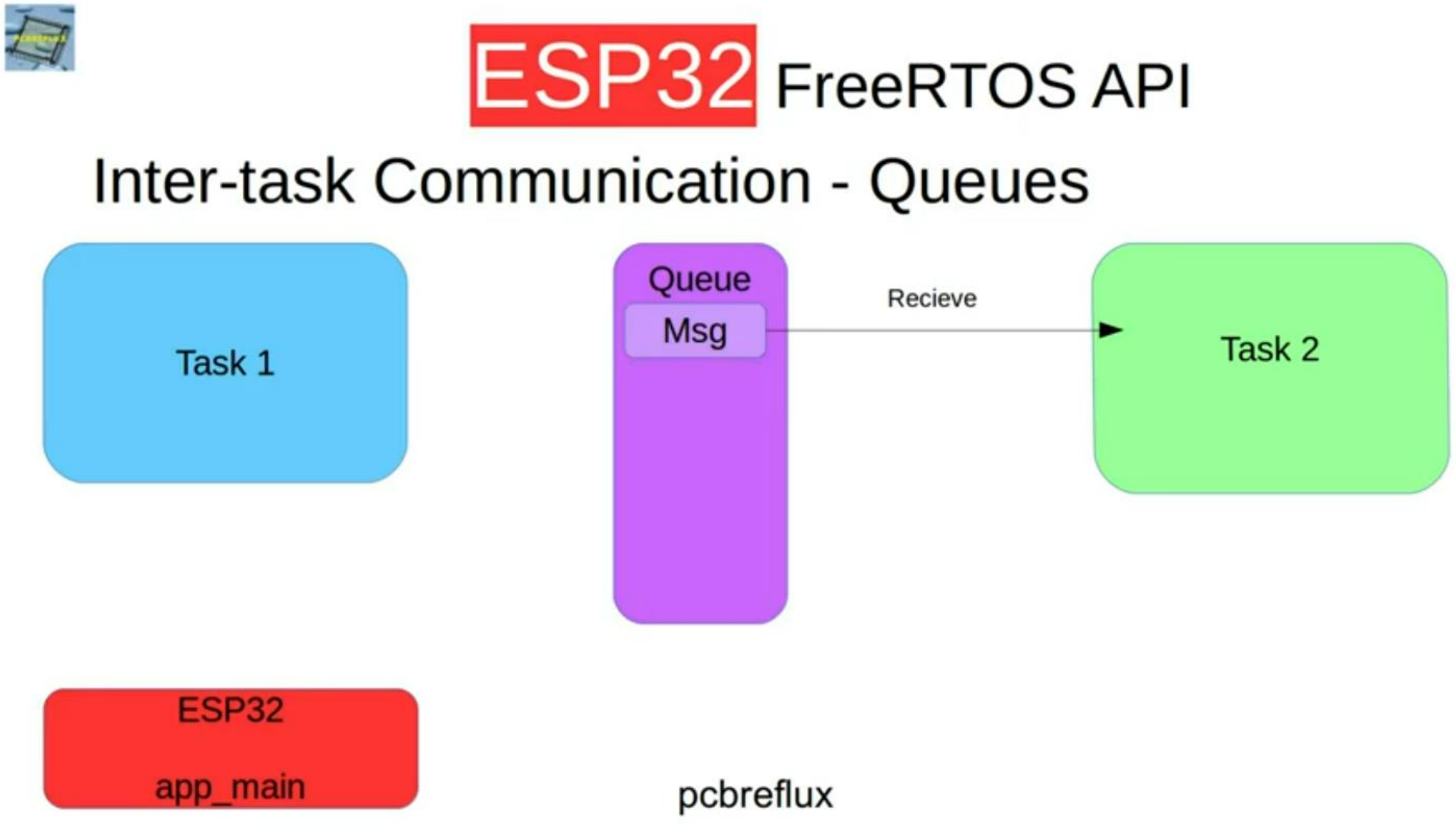
https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Queue (cont.)



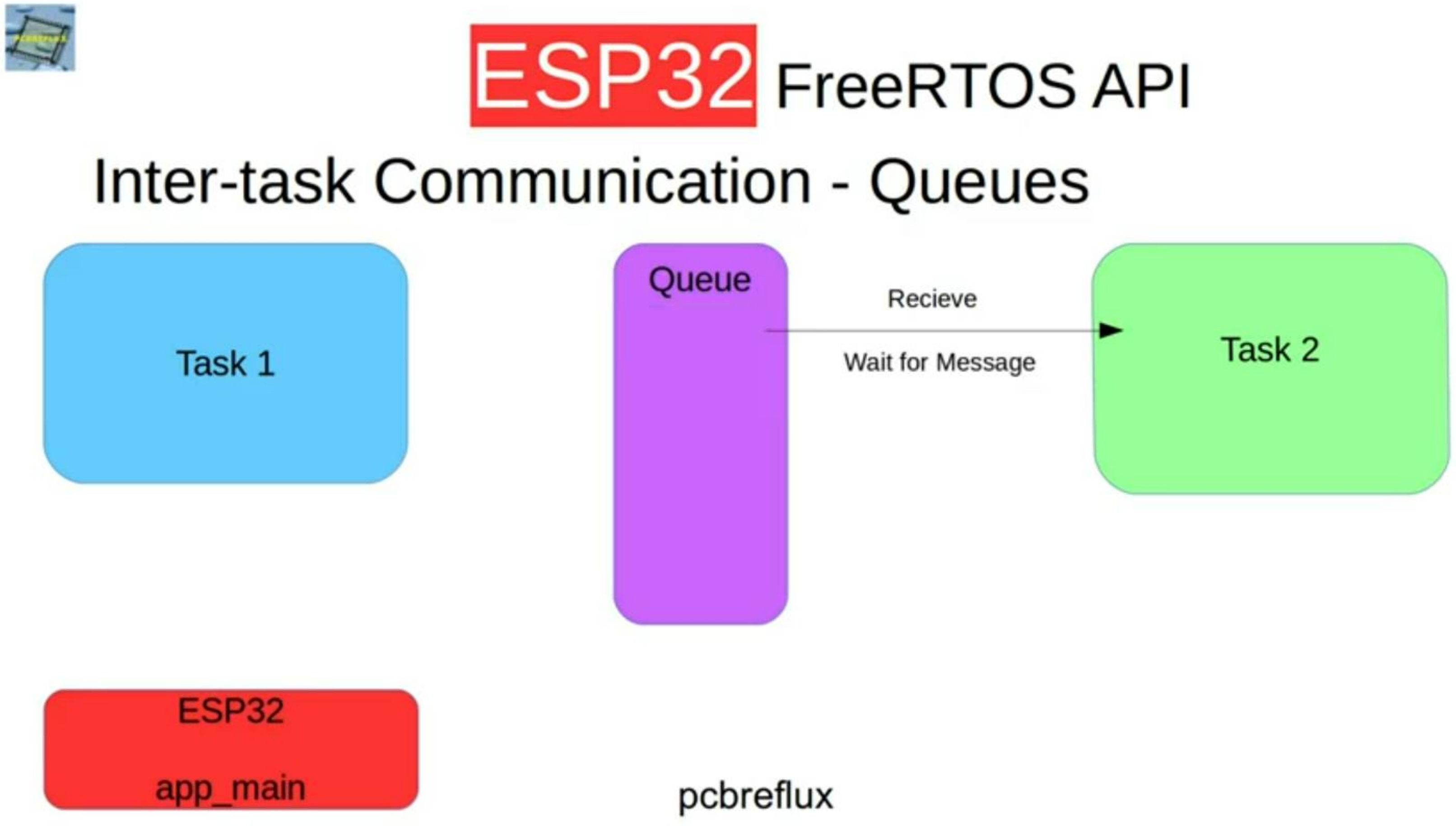
https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Queue (cont.)



https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Queue (cont.)



https://www.youtube.com/watch?v=DLqj01asDM0&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU

Exemplo 2

```
#include "freertos/queue.h"

xQueueHandle queue;
```

```
xQueueCreate(uxQueueLength,uxItemSize)
```

```
queue = xQueueCreate(3, sizeof(int));
```

Exemplo 2 (cont.)

```
xQueueSend(xQueue,pvItemToQueue,xTicksToWait)
```

```
if(xQueueSend(queue, &count, 10) == pdTRUE)
```

```
BaseType_t xQueueReceive(QueueHandle_t xQueue, void *const
pvBuffer,                                               TickType_t xTicksToWait)
```

```
if(xQueueReceive(queue, &rxInt , 0) == pdTRUE)
```

Exemplo 2 (cont.)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/queue.h"

xQueueHandle queue;
```

```
void sender(void *params)
{
    int count = 0;
   while (true)
   {
      count++;
      if(xQueueSend(queue, &count, 10) != pdTRUE)
      {
         printf("Queue FULL\n");
      }

   vTaskDelay(1000 / portTICK_PERIOD_MS);

   }
}
```

Exemplo 2 (cont.)

```c
void receiver(void *params)
{
    while (true)
    {
        int rxInt;
        if(xQueueReceive(queue, &rxInt , 0) == pdTRUE)
        {
            printf("received %d\n", rxInt);
        }
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

```c
void app_main(void)
{
    queue = xQueueCreate(3, sizeof(int));
    xTaskCreate(&sender, "send", 2048, NULL, 2, NULL);
    xTaskCreate(&receiver, "receive", 2048, NULL, 1, NULL);
}
```

Exemplo 2 (cont.)

- Altere o vTaskDelay da task sender para 500 ms;
- Veja qual o comportamento

- Altere o vTaskDelay da task sender para 1000ms;
- Altere o vTaskDelay da task receiver para 500ms;
- Printe o elemento da fila fora do if;

Mutex

- Mutual Exclusion: proteção para código crítico/compartilhado

- Duas ou mais tarefas utilizam um recurso compartilhado: variável global (flag, contador)

- Condição de corrida

- Analogia com chave de banheiro de posto;

Condição de Corrida

- Compartilhamento de recursos
- É quando o comportamento do sistema depende do tempo de eventos incontroláveis

Condição de Corrida (cont.)

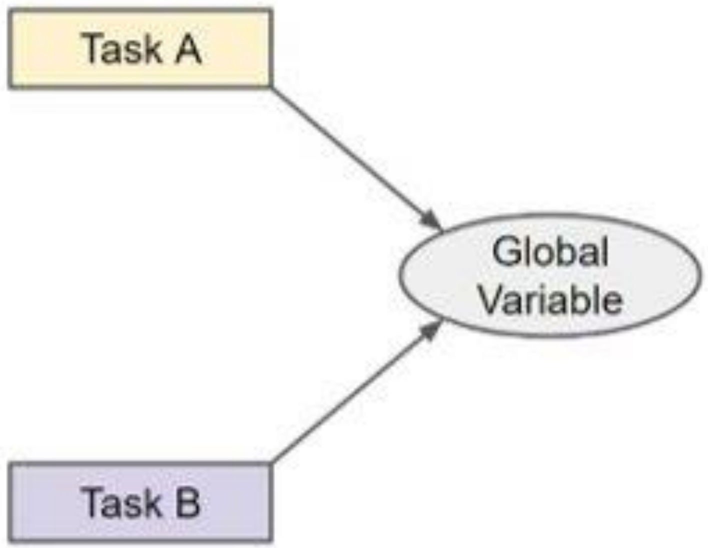
```
int global_var = 0;

void incTask(void *parameters) {
  while(1) {
    global_var++;
  }
}

void main() {
  startTask1(incTask, "Task 1");
  startTask2(incTask, "Task 2");
  sleep();
}
```

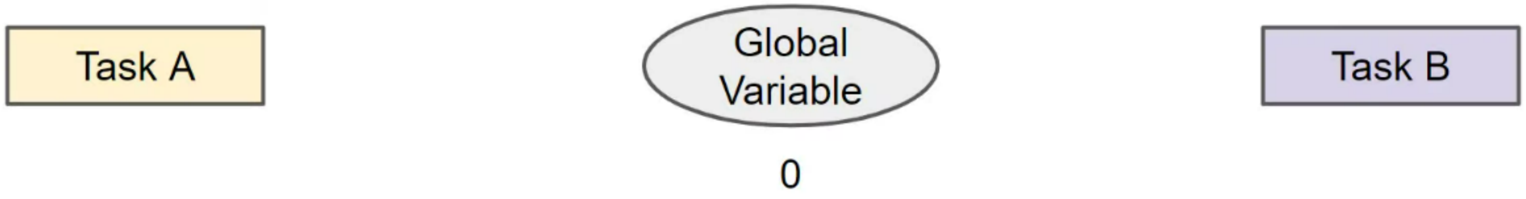
global_var increment can take several instruction cycles!

global_var:
0
1
2
3
3
4
5
5



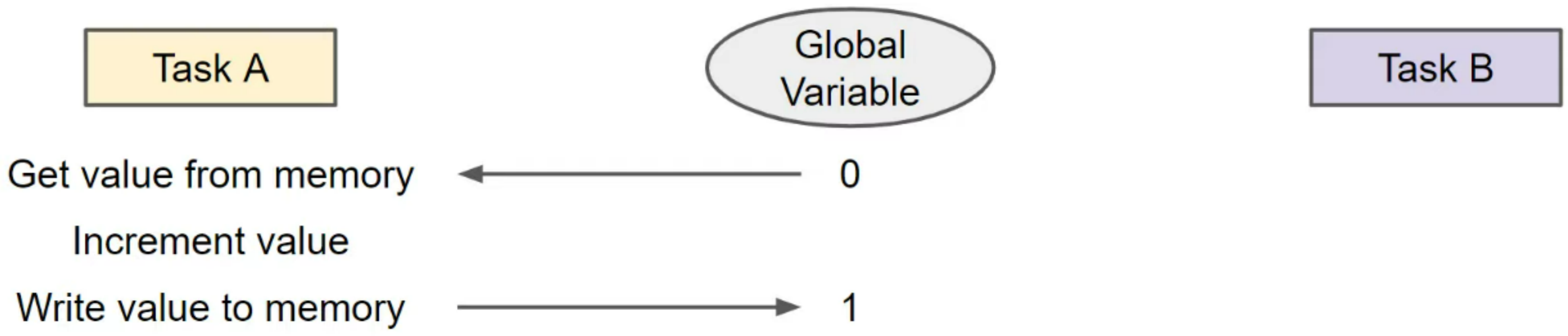
https://www.youtube.com/watch?v=I55auRpbiTs&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Condição de Corrida (cont.)



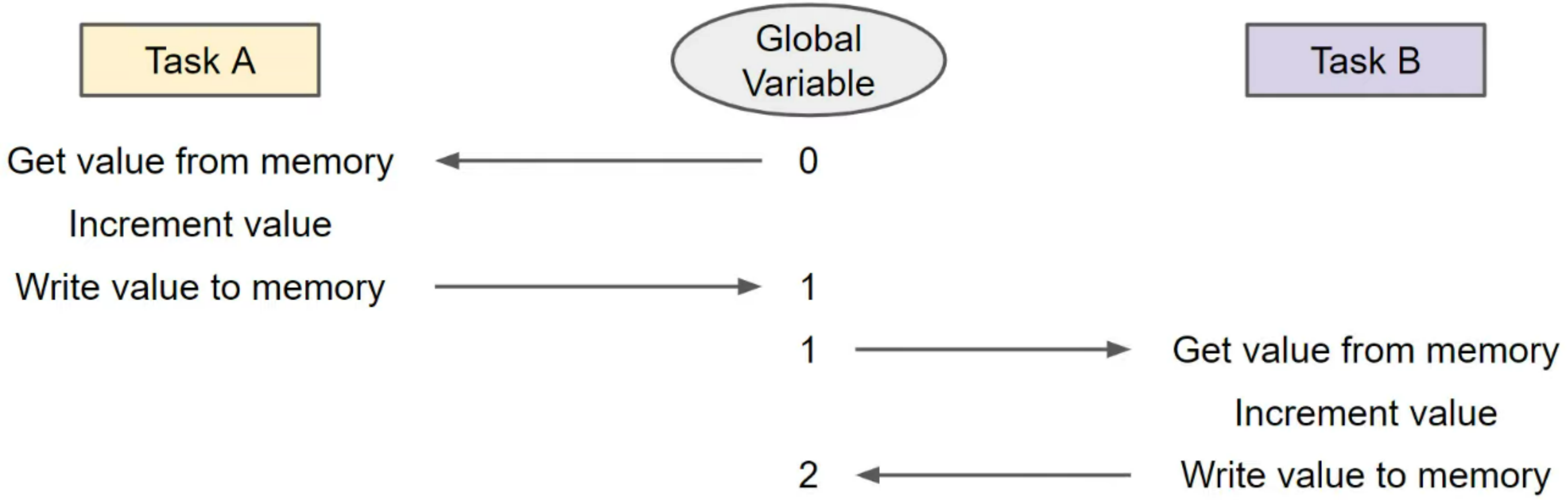
https://www.youtube.com/watch?v=I55auRpbiTs&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Condição de Corrida (cont.)



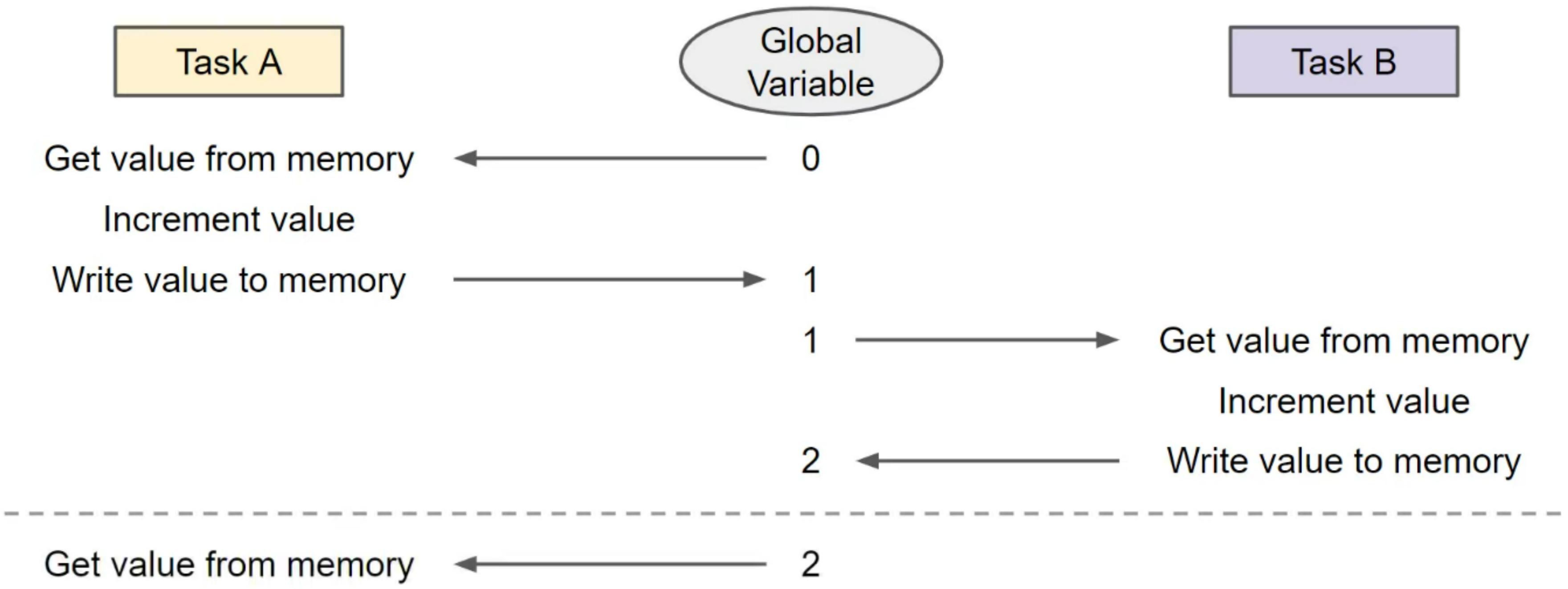
https://www.youtube.com/watch?v=I55auRpbiTs&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Condição de Corrida (cont.)



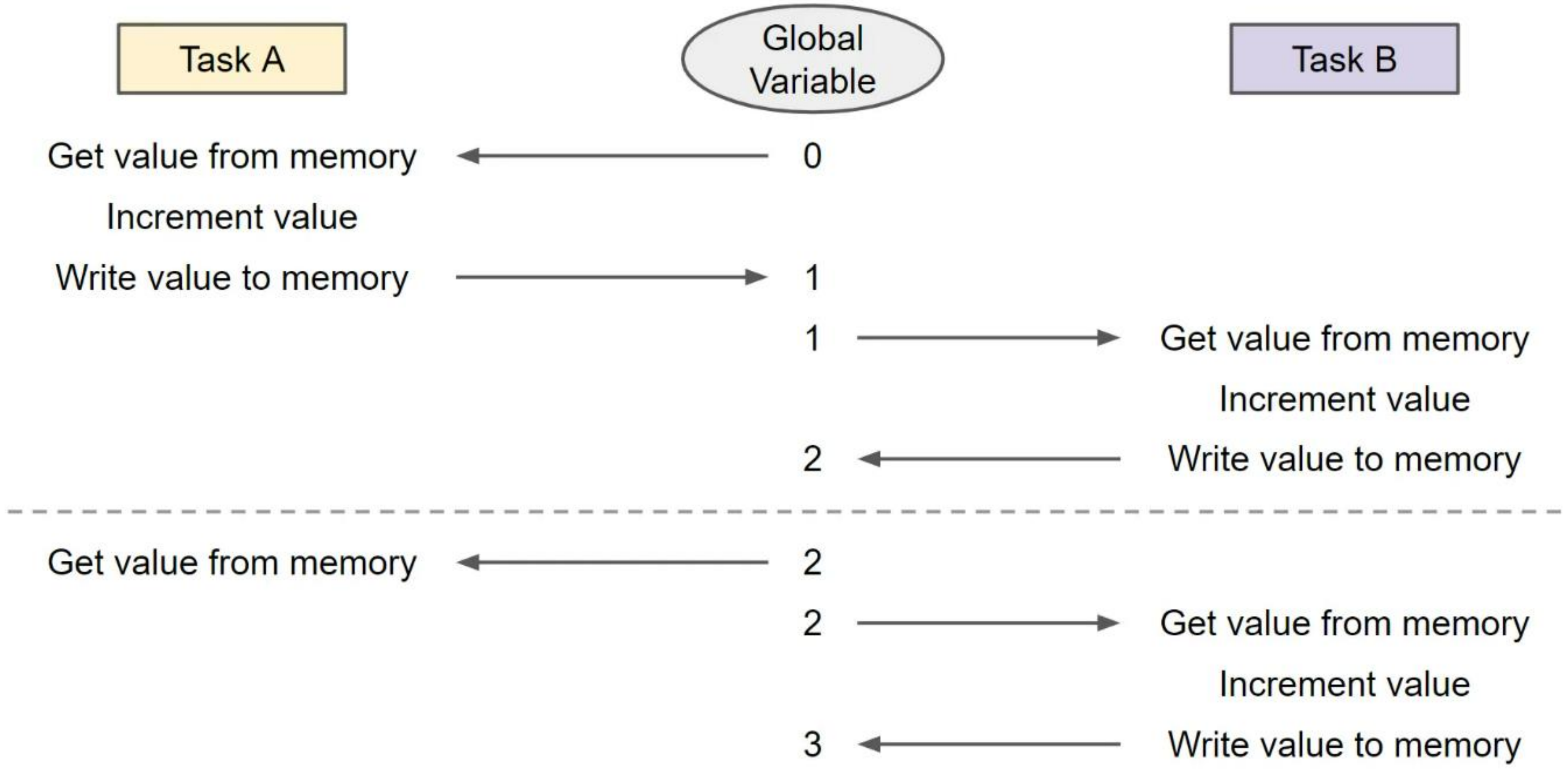
https://www.youtube.com/watch?v=I55auRpbiTs&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Condição de Corrida (cont.)



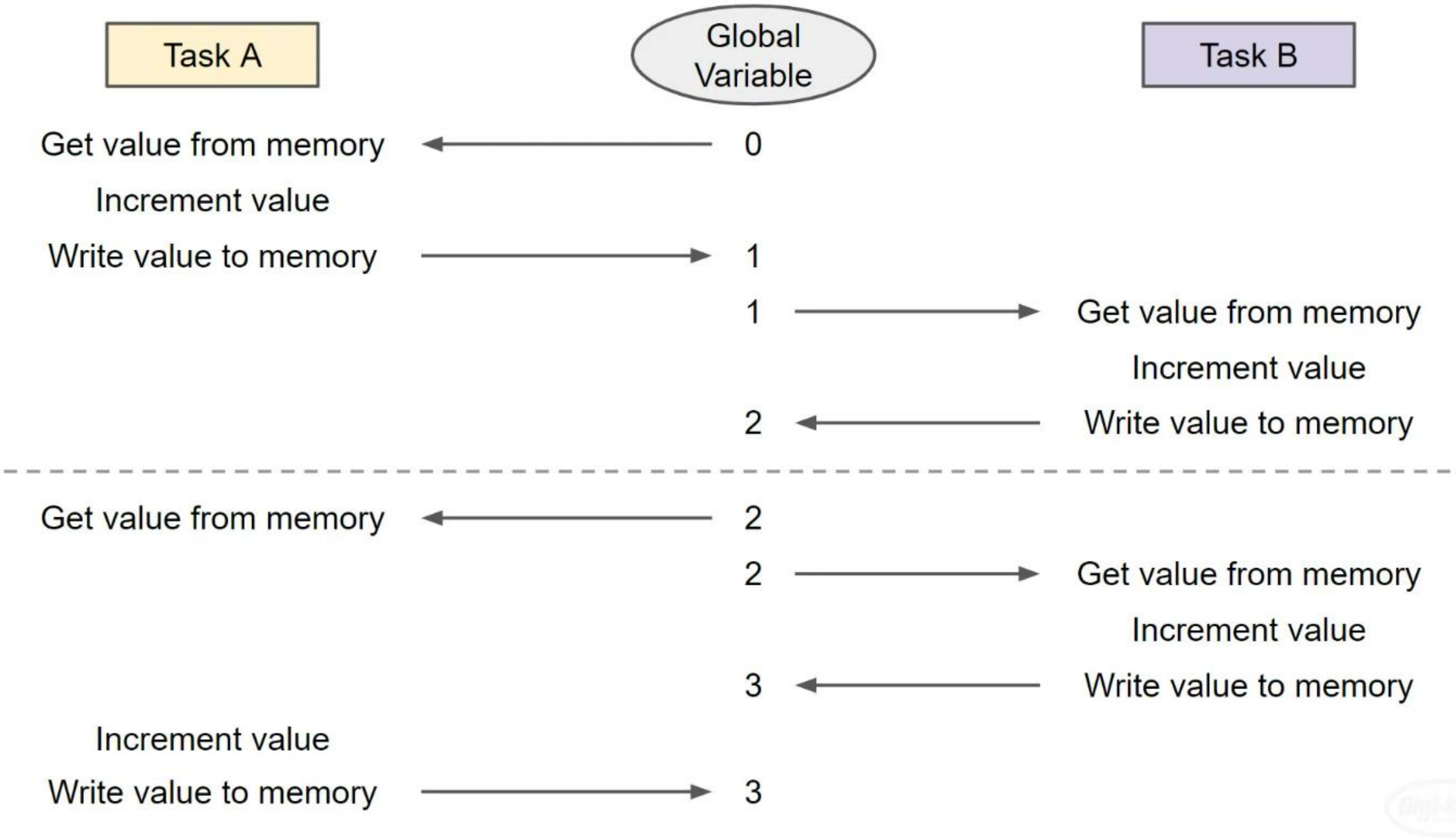
https://www.youtube.com/watch?v=I55auRpbiTs&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Condição de Corrida (cont.)



https://www.youtube.com/watch?v=I55auRpbiTs&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Condição de Corrida (cont.)



https://www.youtube.com/watch?v=I55auRpbiTs&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz&index=6

Exemplo 3

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_log.h"

const char *TAG = "mutex.c";
```

```
void displayMessage(char *message)
{
    for (int i = 0; i < strlen(message); i++)
    {
      printf("%c", message[i]);
      for (long i = 0; i < 1000000; i++) {}
    }
    printf("\n");
}
```

Exemplo 3 (cont.)

```
void task1(void *param)
{
    while (true)
    {
      displayMessage("temperature is 25c\0");
      vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    vTaskDelete(NULL);
}
```

```
void task2(void *param)
{
    while (true)
    {
      displayMessage("humidity is 50\0");
      vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    vTaskDelete(NULL);
}
```

Exemplo 3 (cont.)

```
void app_main(void)
{
    ESP_LOGI(TAG, "core %d/line %d/%s/starting ", xPortGetCoreID(), __LINE__, __func__);

   xTaskCreatePinnedToCore(&task1, "temperature reading", 2048, NULL, 2, NULL,0);
   xTaskCreatePinnedToCore(&task2, "humidity reading", 2048, NULL, 2, NULL, 1);
}
```

Exemplo 3.1

- Adicionando mutex

```c
#include "freertos/semphr.h"

xSemaphoreHandle mutexBus;
```

```c
void app_main(void)
{
    ...
    mutexBus = xSemaphoreCreateMutex();
    ...
}
```

Exemplo 3.1 (cont.)

```
void task1(void *param)
{
   while (true)
   {
     if (xSemaphoreTake(mutexBus, 1000))
     {
       displayMessage("temperature is 25c\0");
       xSemaphoreGive(mutexBus);
     }
     vTaskDelay(1000 / portTICK_PERIOD_MS);
   }
   vTaskDelete(NULL);
}
```

```
void task2(void *param)
{
   while (true)
   {
     if (xSemaphoreTake(mutexBus, 1000))
     {
       displayMessage("humidity is 50\0");
       xSemaphoreGive(mutexBus);
     }
     vTaskDelay(1000 / portTICK_PERIOD_MS);
   }
   vTaskDelete(NULL);
}
```

Exercícios

1. Faça um programa, utilizando tasks, que ao manter o botão pressionado o LED pisca em uma frequência de 10 Hz e quando solto em 2 Hz

2. Faça um programa, utilizando tasks, que ao pressionar o botão o led pisca e pressionar novamente o led fica ligado

Exercícios (cont.)

3. Faça um programa, utilizando tasks, que a medida que o botão é pressionado aumente a intensidade do LED ao atingir o valor máximo volte a zero. Adicione 5 passos.

4. O ESP32 possui um sensor hall integrado. Descubra como fazer leituras dele e apresente os dados a cada 500ms, utilizando tasks.

Exercícios (cont.)

5. Faça um monitor de bateria de Li Íon, utilizando tasks.

- Crie um circuito com um potenciômetro que simule os valores da bateria: 2,8V a 4,2V.
- Com o ESP32 faça a leitura desses valores e acenda os leds de acordo com a tabela

LED Ligado	Faixa de Tensão
Verde	4,2V à 3,75V
Amarelo	3,75 à 3,4V
Vermelho	3,4 à 2,8V

Recomendados

- Introduction to RTOS
    - https://www.youtube.com/watch?v=JIr7Xm_riRs&list=PLEBQazB0HUyQ4hAPU1cJED6t3DU0h34bz
- Mastering the FreeRTOS Real Time Kernel – A Hands-On Tutorial Guide
    - https://www.freertos.org/fr-content-src/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf

Recomendados (cont.)

- Curso ESP32 - INPE
  - https://www.youtube.com/playlist?list=PLHvD4LbjH0y0JqU3SHEGoRgLhLqo7TIxx
- pcbreflux ESP32 playlist
  - https://www.youtube.com/watch?v=iunw7qd5Wr4&list=PLxJ8_KSR8bp5-F4HVG4QOm4Kt6wQhzsjU
- The Internet of Things with ESP32
  - http://esp32.net/

PADO
Labs