وظيفة نظم الزمن الحقيقي

اسماء الطلاب :

رهف دريد خزندار

عمار الياس الأترم

عبدالله عبد الرزاق زعرور

أحمد محمد الخضر الحسن

اشراف الآنسة :

ميساء

شرح الكود الأول :

أولاً استخدمنا هذه التعليمة لنقوم بتضمين مكتبة Arduino FreeRTOS في البرنامج:

#include <Arduino\_FreeRTOSS.h>

ثم دخلنا في ()void setup :

   - هذه الدالة تعد جزءًا من البرنامج وتستخدم لتهيئة البيئة قبل بدء تنفيذ البرنامج الرئيسي.

   - في مثالنا، تقوم ببدء الاتصال مع Serial وطباعة "In Setup function".

 وصلنا لتعليمة :

xTaskCreate(MyTask1, "Task1", 100, NULL, 1, NULL)

تعمل هذه التعليمة على إنشاء مهمة باسم MyTask1، بأولوية 1 وحجم ذاكرة 100.

تقوم بتحديد الدالة التي سيتم تنفيذها كـMyTask1.

 أما تعليمة :

xTaskCreate(MyTask2, "Task2", 100, NULL, 2, NULL)

فهي تعمل على إنشاء مهمة باسم MyTask2، بأولوية 2 وحجم ذاكرة 100.

و تقوم بتحديد الدالة التي سيتم تنفيذها كـMyTask2.

 وأيضاً :

xTaskCreate(MyIdleTask, "IdleTask", 100, NULL, 0, NULL)

تقوم بإنشاء مهمة خاصة بالفراغ (Idle Task) باسم MyIdleTask، بأولوية 0 وحجم ذاكرة 100.

و تحدد الدالة التي سيتم تنفيذها كـMyIdleTask.

 دخلنا في حلقة ()void loop :

هذه الدالة تعد الجزء الحلقي الرئيسي لدينا حيث يتم تنفيذ الأوامر بشكل متكرر في جملة while(1).

 قمنا بتعريف الدالة التي ستُنفّذ كـMyTask1، والتي تقوم بطباعة "Task1" بانتظام وتأخير لفترة معينة

static void MyTask1(void\* pvParameters)..وايضاً قمنا بتعريف الدالة التي ستُنفّذ كـMyTask2، والتي تقوم بطباعة "Task2" بانتظام وتأخير لفترة أطول من MyTask1

static void MyTask2(void\* pvParameters)

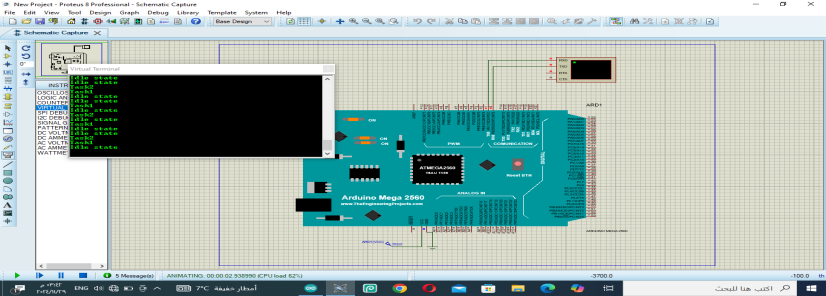
 وأخيراً عرفنا الدالة التي ستُنفّذ كـMyIdleTask، والتي تقوم بطباعة "Idle state" بانتظام وتأخير لفترة قصيرة.

static void MyIdleTask(void\* pvParameters)

 نتائج تنفيذ الكود ستكون:

- سترى رسائل "Task1" و "Task2" بانتظام، تمثل كل من MyTask1 و MyTask2 على Serial.

- إذا لم تكن هناك مهام أخرى قيد التشغيل، ستظهر رسالة "Idle state" بانتظام من قبل MyIdleTask.



شرح الكود الثاني :

 كالعادة اولاً قمنا بتضمين مكتبة FreeRTOS باستخدام التعليمة :

#include <Arduino\_FreeRTOSS.h>

ثم دخلنا في ()void setup : دالة setup() هنا تُستخدم لتهيئة البيئة قبل تشغيل البرنامج الرئيسي.

ثم تقوم ببدء الاتصال مع Serial وطباعة "In Setup function".

و تقوم بإنشاء مهمتين بأولويات 1 و 2 على التوالي باستخدام xTaskCreate.

ومن بعدها دخلنا بحلقة () void loop :

في مثالنا هنا  يتم طباعة "Loop function" بانتظام وتأخير لمدة 50 ميلي ثانية بين كل تكرار.

هنا قمنا بتعريف MyTask1، وهي المهمة التي تقوم بطباعة "Task1" بانتظام كل 100 ميلي ثانية باستخدام

static void MyTask1(void\* pvParameters)

وايضاً عرفنا MyTask2، المهمة التي تقوم بطباعة "Task2" بانتظام كل 150 ميلي ثانية

static void MyTask2(void\* pvParameters)

نتيجة التنفيذ :

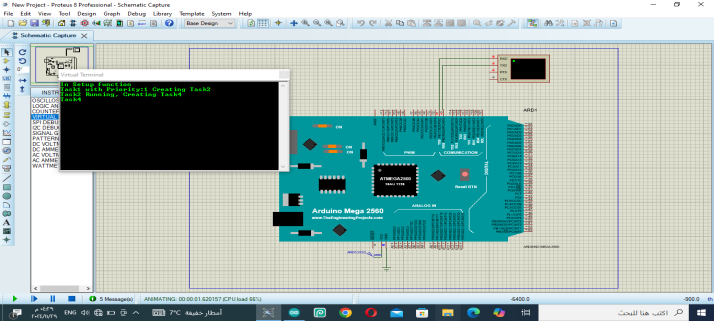
أولاً ستبدأ البرنامج باستدعاء الدالة setup()، وتتم طباعة "In Setup function" على Serial.

ثم سيتم إنشاء مهمتين: MyTask1 و MyTask2 بأولويات 1 و 2 على التوالي.

- في الوضع العادي ( ()loop )، سيتم طباعة "Loop function" بانتظام وتأخير 50 ميلي ثانية بين التكرارات.

- MyTask1 سيطبع "Task1" بانتظام كل 100 ميلي ثانية.

- MyTask2 سيطبع "Task2" بانتظام كل 150 ميلي ثانية.



شرح الكود الثالث :

قمنا بالتضمين كما سابقاً :

#include <Arduino\_FreeRTOSS.h>

ثم قمنا بتعريف متغيرات لتخزين مقابض (handles) لكل مهمة. يُستخدم هذا المتغير لحذف المهمة بواسطته.

TaskHandlet TaskHandle1

 TaskHandlet TaskHandle2

TaskHandlet TaskHandle3

 ثم دخلنا في ()void setup:

هنا دالة setup() تُستخدم لتهيئة البيئة قبل بدء تشغيل البرنامج الرئيسي،و تبدأ ببدء الاتصال مع Serial وطباعة "In Setup function"،و تُنشئ ثلاث مهام بأولويات 1 و 2 و 3 على التوالي، وتحفظ تفاصيل المهمة في المقابض المناسبة.

الآن دخلنا بحلقة ()void loop:

دالة ()loop في مثالنا تطبع "Loop function" بانتظام كل دورة مع تأخير مقداره 50 ميلي ثانية.

ثم لدينا مهمة تُسمى MyTask1 تُنفّذ بأولوية 1، تطبع "Task1 Running" بانتظام ثم تحذف نفسها بعد دورتها.

static void MyTask1(void\* pvParameters)

  ولدينا مهمة تُسمى MyTask2 تُنفّذ بأولوية 2، تطبع "Task2 Running" بانتظام ثم تحذف نفسها.

static void MyTask2(void\* pvParameters):

وأخيراً مهمة تُسمى MyTask3 تُنفّذ بأولوية 3، تطبع "Task3 Running" بانتظام ثم تحذف نفسها.

static void MyTask3(void\* pvParameters)

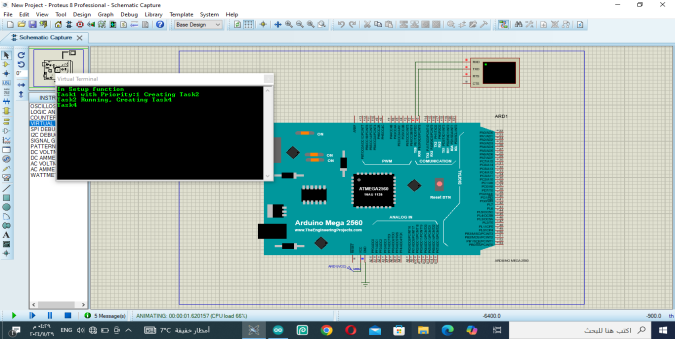
نتيجة التنفيذ :

- بعد بدء تشغيل البرنامج، ستظهر "In Setup function" على Serial،ستقوم ثلاث مهام بالتنفيذ، كل منها بتأخير محدد.

- MyTask1 ستطبع "Task1 Running" بانتظام ثم تحذف نفسها باستخدام TaskHandle\_1.

- MyTask2 ستطبع "Task2 Running" بانتظام ثم تحذف نفسها بنفسها.

- MyTask3 ستطبع "Task3 Running" بانتظام ثم تحذف نفسها باستخدام TaskHandle\_3.



  شرح الكود الرابع :

اولاً التضمين :

#include <Arduino\_FreeRTOSS.h>

   ثم عرفنا متغيرات لتخزين مقابض (handles) لكل مهمة. تُستخدم هذه المتغيرات لحذف المهام.

TaskHandlet TaskHandle1;

TaskHandlet TaskHandle2;

TaskHandlet TaskHandle3;

TaskHandlet TaskHandle4;

ودخلنا في ()void setup:

يتم بدء الاتصال مع Serial وطباعة "In Setup function"،ثم إنشاء مهمتين بأولويات 1 و 3 على التوالي، وتخزين تفاصيل المهمة في المقابض المناسبة.

ودخلنا بحلقة ()void loop:

لتقوم بتنفيذ دوري يُطبع "Loop function" بانتظام كل دورة مع تأخير 50 ميلي ثانية.

  عرفنا عن مهمة تُسمى MyTask1 تُنفّذ بأولوية 1، تطبع "Task1 Running" ثم تحذف نفسها باستخدام TaskHandle\_1.:

static void MyTask1(void\* pvParameters)

   ولدينا المهمة تُسمى MyTask2 - هنا تعريف خاطئ لأن المهمة غير موجودة في الكود

static void MyTask2(void\* pvParameters)

ولدينا المهمة تُسمى MyTask3 تُنفّذ بأولوية 3، تطبع "Task3 Running" وتنشئ مهمتين Task2 و Task4 ثم تحذف نفسها باستخدام TaskHandle\_3 :

static void MyTask3(void\* pvParameters)

وأخيراً مهمة تُسمى MyTask4 تُنفّذ بأولوية 4، تطبع "Task4 Running" وتحذف نفسها.

static void MyTask4(void\* pvParameters)

نتيجة التنفيذ :

- بعد بدء تشغيل البرنامج، سيُظهر "In Setup function" على Serial،سيتم إنشاء مهمتين: MyTask1 بأولوية 1 و MyTask3 بأولوية 3.

- MyTask1 ستطبع "Task1 Running and About to delete itself" ثم تحذف نفسها باستخدام TaskHandle\_1.

- MyTask3 ستطبع "Task3 Running, Creating Task2 and Task4"، ثم ستنشئ MyTask2 بأولوية 2 و MyTask4 بأولوية 4، وبعد ذلك تحذف نفسها باستخدام TaskHandle\_3.

