二、研究計畫內容（以10頁為限）：

（一）摘要

混合關鍵性即時系統(Mixed-Criticality Real-Time Systems，MCRTS)目前已廣泛地應用在許多安全攸關(Safety-Critical)領域，成為備受重視的研究領域。MCRTS研究領域源自於即時系統(Real-Time Systems)，其所考慮的工作不但具有嚴格的時效性要求，還必須滿足工作在不同的關鍵層級(Criticality Levels)下的各項要求。自Steve Vestal於2007年提出了著名的混合關鍵性工作模型後[1]，已有許多的工作排程方法被提出[2]，本計畫擬設計與實作一套可用以模擬MCRTS的平台，針對既有的排程方法進行模擬驗證，讓開發人員可以在建置相關系統前，先行得知工作的可排程性並據以進行相關的修正。本計畫將以屏東大學即時與嵌入式系統實驗室所發展的多核心即時系統工作排程模擬工具(MCRTsim)為基礎[3]，將其修改並延伸支援MCRTS的工作排程；此外，我們也將針對著名的混合關鍵性工作的排程方法，例如SMC[4]、AMC[5]以及AMC$^+$[6, 7]等進行實作。本計畫的成果預計將可以提供相關的研究團隊與開發人員，做為驗證工作可排程性的模擬環境，並且可以將此模擬工具作為新工作排程方法的研發平台，為了此目的，我們將提供相關的API協助開發人員實作新的工作排程方法。

本計畫具體的工作項目如下：(1) *SimART*實作、(2)建立MCRTS的工作排程模擬環境、(3)實作關鍵性及時效性的相關參數及觸發條件、(4)增加在混合關鍵性層級工作排程方法模擬、(5)使用動態電壓調整(Dynamic Voltage Frequency Scaling, DVFS)來設計處理器並應用在MCRTS的工作排程。詳情請參閱本計畫書後續說明。

（二）研究動機及研究目的

2.1 研究動機

考慮到混合關鍵性即時系統(Mixed-Criticality Real-Time System，MCRTS)在學理和實際應用上的日益重要性，其相關研究已成為一個極為重要的課題。MCRTS不同於傳統應用系統之處，主要有以下兩個特點：

1. 時效性(Timeless)：指的是工作必須在特定時間限制內完成，否則可能導致無法彌補的後果。例如在車輛上使用雷達來測量前方車輛距離並控制車速以維持安全距離的車距調節工作，其執行頻率被設定為每秒5次，每次必須在200毫秒內完成。這是為了避免可能導致交通事故的後果。
2. 關鍵性(Criticality)：指的是對安全性的特定要求。當MCRTS執行時，可以切換不同安全性要求的關鍵性層級(Criticality Level)，例如一個用於輔助車輛駕駛的車用MCRTS系統可以運行在低與高兩個關鍵性層級，當車輛開始行駛時，預設以低關鍵性層級運行；然而，當車輛加速為高速行駛時 (例如時速達到每小時100公里以上)，則將會切換至高關鍵性層級運行。關鍵性層級反映的是工作的執行結果與人身安全的相關性，愈高的關鍵性就代表工作的執行結果應該更為精確，或者必須增加工作執行的頻率。例如維持安全距離的車距調節工作在低關鍵性層級運行時，其使用雷達來測量前方車輛距離的精確度只需要到小數點後2位數；但當運行在高關鍵性層級時，為了確保安全性，則必須要求精

確度至小數點後5位數。另一方面，此一車距調節工作在低關鍵性層級運行時，其執行頻率被設定為每秒5次，每次必須在200毫秒內完成；但當車輛處於高速駕駛的狀態時，200毫秒內車輛與前車間的相對距離可能已發生極大的變化，為安全起見，當使工作運行在高關鍵性層級時，則改變其工作頻率被設定為每秒10次，以提升安全性。

由於時效性與關鍵性的關係，包含即時系統在內的傳統工作排程方法並不適用於MCRTS的環境。例如，在即時系統領域著名的Earliest Deadline First (EDF)[8] 排程方法，儘管是動態優先權的最佳(Optimal)排程方法，然而當應用在MCRTS時，每個工作除了各自具有不同的時效要求外並未考慮到工作的關鍵性，所以只能應用於所有工作都具有相同的關鍵性要求的情況 — 因此不符合MCRTS的要求。為了應對此一挑戰，目前學術界已提出許多MCRTS的工作排程方法及理論，詳細的相關研究成果亦可參考Burns教授的Survey論文[2]。

2.2 研究目的

本計畫的主要研究目的在於開發一套可用於混合關鍵性即時系統(Mixed-Criticality Real-Time System，MCRTS)的工作排程方法驗證的模擬環境 — 我們將其命名為 A Simulation Environment for Mixed-CriticAlity Real-Time Task Scheduling Algorithms (*SimART*)。*SimART*的主要目的是為MCRTS的開發人員提供了一個實驗的環境(Sandbox)，用以幫助MCRTS的開發人員在實際進行MCRTS開發前，使用*SimART*進行系統架構與特性的定義，並且可以挑選使用數個著名的MCRTS工作排程方法進行工作的排程。透過*SimART*所提供的模擬環境，開發人員只要針對所要模擬的系統與工作進行相關的組態設定，就可以進行MCRTS工作的排程模擬。此外，根據*SimART*所先行得到的工作排程結果，可以用來評估不同的排程策略在各種條件下的效能表現，進一步確保系統在實際應用中能夠有效地滿足時效性和關鍵性的要求，並在進行系統開發前進行全面的測試與驗證。本計畫所擬開發的*SimART*將以屏東大學即時與嵌入式系統實驗室過往所開發的工作排程模擬平台MCRTsim[3]為基礎，將其修改並延伸支援混合關鍵性即時系統的工作排程方法的模擬驗證。

目前除了屏東大學的研究團隊有使用MCRTsim進行工作排程方法的模擬與驗證外[9, 10]，國際上亦有包含埃及海爾溫大學 (Helwan University)[11, 12]、土耳其梅爾辛大學(Mersin Üniversitesi)[13]等研究團隊使用MCRTsim來進行相關的排程方法實驗；由於MCRTsim已經許久未持續更新，國內外的研究團隊在使用上必須先建置較舊版本的作業系統以及JDK 8，才能夠用以進行工作排程的模擬與驗證。有鑑於此，本計畫預計修改軟體的架構並採用最新版本的JDK 21進行大幅度改寫，增加包含關鍵性層級管理在內的模組，使其能支援MCRTS的工作排程模擬。具體來說*SimART*將具有以下的特點：

*SimART*支援多種工作模型如週期性即時工作模型(Periodic Real-Time Task Model)[8]、偶發性即時工作模型(Sporadic Real-Time Task Model)[14](以上兩個工作模型是承襲自MCRTsim的成果)外，還將支援具有時效性與關鍵性的混合關鍵性即時工作模型(Mixed-Criticality Real-Time Task Model)。透過多種工作模型可幫助開發人員進行各式應用相關的排程模擬。除了上述的工作模型外，*SimART*同時提供多種工作排程方法，如實作單核心處理器架構之即時系統的EDF[8]與RMS(Rate-Monotonic Scheduling)[8]排程方法，也有實作多核心處理器排程方面的Partion Scheduling架構排程方法，例如P-RMS[15]、P-EDF[16]; 以及採用Global Scheduling架構的G-RMS、G-EDF等。支援多種架構與多種排程方法對於開發人員而言，將能夠提供各種

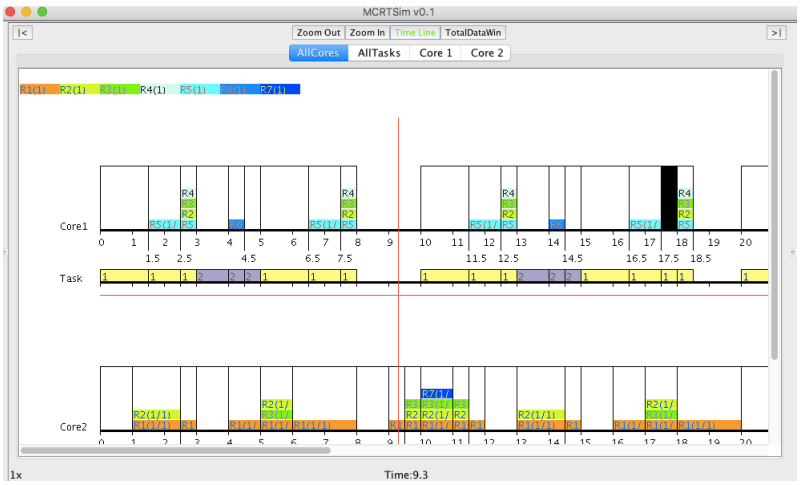

圖 1： MCRTsim的工作排程結果檢視畫面截圖

應用情境的模擬之需。除了上述的排程方法外，本計畫還預計支援包含SMC[4]AMC[5]以及AMC$^+$[6, 7]等著名的混合關鍵性即時工作排程方法。

同時*SimART*提供視覺化的排程結果檢視，藉由*SimART*內建的圖形化使用者介面(Graphic User Interface, GUI)可以幫助開發人員可以更簡易地使用此模擬環境進行實驗外，*SimART*還針對模擬工作排程的結果，提供詳盡的數據資料並且採用視覺化方式呈現，使開發人員能夠了解複雜的工作排程結果，包含系統的運行狀態和性能表現，以及工作分配在多核心處理器的相關資訊等，實際執行結果可參考圖 1的MCRTsim的執行畫面截圖結圖。在設定方面，*SimART*提供豐富的參數設定，允許開發人員根據具體的MCRTS需求，調整不同任務的關鍵層級、執行週期等參數；這些與所要模擬的系統環境以及工作定義相關的參數，都可以使用組態檔的方式進行詳細的參數設定。具體來說，*SimART*可讓開發人員選擇使用GUI的方式，設定系統或工作的相關參數，後續再自動產生相關的組態檔案。

(三)研究背景與相關作品

3.1 研究背景

由於混合關鍵性即時系統源自即時系統，因此本節將先就即時系統簡介其相關學理與工作排程方法，後續再針對混合關鍵性即時系統加以說明。

3.1.1 即時系統

即時系統(Real-Time System)重視工作執行的時效性，其工作的執行除了運算結果必須正確外，還要求必須要求工作必須在特定的截止時間(Deadline)內完成，若能在截限時間內完成的工作稱為滿足截限時間(Meet Deadline)，無法在截限時間內完成的工作則稱為錯失截限時間(Miss Deadline)。一般而言，即時系統依據工作是否滿足或違反截限時間，還可區分為「硬即時系統(Hard Real-Time System)」和「軟即時系統(Soft Real-Time System)」兩類。在硬即時系統方面，工作必須嚴格遵循其預定的時間限制(例如截限時間)，完全不允許工作違反時間限制，否則將有可能造成嚴重的後果，適用於對時間要求非常嚴格且不能容忍任何遲延的應用領域。另一方面，軟即時系統仍然重視工作的時間限制，但可接受工作違反時間限制，在條

件允許的情況下，仍追求儘可能地在截限時間內完成；若已超出截限時間的工作，則可以視情況將其捨棄或是繼續加以執行。

不論是硬即時或軟即時系統，都需要有設計良好的工作排程方法來決定工作執行的先後順序，並且要確保工作能夠滿足特定的時間限制。目前大部份應用在即時系統的工作排程方法，都是採用可搶先(Preemptive)的優先權驅動(Priority-Driven)的方法為之，並可依據優先權給定的方法再區分為「固態優先權(Fixed Priority)」和「動態優先權(Dynamic Priority)」兩類。固定優先權的排程方法是在系統運行前，事先給定每個工作τ_i一個固定的優先權P_i。當系統開始運行後，所有τ_i依據其週期定期產生的工作實例$\tau_{i,j}$都會使用此固定的、相同的優先權P_i。至於在動態優先權方面，在系統運行時，依據工作實例$\tau_{i,j}$產生時系統當時的況狀來決定其優先權，同一個工作τ_i的任意兩個不同工作實例$\tau_{i,j}$與$\tau_{i,k}$將可能以不同的優先權運作。

相關的工作排程方法，還可以針對單核心及多核心處理器環境分別加以介紹。首先在單核心處理器方面，目前最為著名的是固定優先權的Rate-Monotonic Scheduling (RMS)[8]以及動態優先權的Earliest Deadline First (EDF)[8]，其中RMS方法是根據工作的週期賦予優先權(Priority)的方法，週期愈短的工作將給予愈高的優先權。至於EDF則是基於工作的截限時間(Deadline)，優先選擇最接近截限時間的工作加以執行，確保了系統內最緊急的工作能夠優先地執行，有助於減少工作錯失截限時間(Miss Deadline)的機會。另一方面，在多核心處理器環境下的排程方法，可概分為兩種常見的做法，分割排程(Partitioned Scheduling)與全域(Global Scheduling)。在分割排程方法方面，每個工作在系統運行前被指派給特定的核心執行，並且通常禁止工作遷移(Task Migration)[1]。當系統開始運行後，每個核心擁有自己區域的就緒佇列(Local Ready Queue)，並使用原先為單核心處理器設計的工作排程方法(每個核心可以使用相同或不同的排程方法)來對其就緒佇列中的工作進行排程。此類的分割排程方法，在實務上相當常見，例如Oh與Baker分別提出的P-RMS[15]與P-EDF方法[16]；其中P-RMS是在每個核心上都使用固定優先權的RMS方法對其區域的就緒佇列進行排程，而P-EDF方法則是在每個核心上都使用動態優先權的EDF方法。至於在全域排程方法方面，則是考慮讓所有到達系統的工作都進入一個全域的就緒佇列(Global Ready Queue) ，並由排程方法分派到核心上執行，通常允許工作遷移。對於一個擁有K個核心的處理器而言，在任何時間點，系統最多只能有K個工作同時被執行。在此類的排程方法中，以Global EDF(G-EDF)[16] 和Proportionate Fair(Pfair)[17]兩個排程方法最為著名。

3.1.2 混合關鍵性即時系統

混合關鍵性即時系統(Mixed-Criticality Real-Time System，MCRTS)研究領域起源至即時系統，除了原本就要求工作必須滿足特定時間限制的**時效性**(Timeliness)要求外，還額外考慮了工作執行時的**關鍵性**(Criticality)。因此原本適用於即時系統的工作排程方法並不能直接應用於MCRTS。以下本節將就工作的關鍵性要求與適用於MCRTS的工作排程方法加以介紹。

首先，工作的「關鍵性」就表示對「安全性」的特定要求。當一套MCRTS在執行時，可以視工作的執行與安全性的關聯性切換不同的關鍵性層級，指的是如果工作與安全性的關聯愈高則其關鍵性層級也會隨之提昇，相反時則愈低。假設有一個在車用電腦系統有低與高兩個關鍵性層級，並且會視行車速度的高低切換不同的關鍵性層級運行 — 時速低於80公里時

[1]工作僅能在其被指派的核心上執行，不允許在系統運行時「遷移」到其他的核心執行。

系統運行在低關鍵性層級，時速高於80公里時則切換運行在高關鍵性層級。考慮一個負責計算車輛轉向角度修正的工作，當它執行在低速度的低關鍵性層級時，此工作在計算修正角度時只需要精確到小數點後第2位數。當車輛速度提昇到80公里以上時，此工作將切換至高關鍵性層級運行，在高速行駛的情況下，若是此工作所計算的修正角度不夠精確或是計算錯誤將有可能導致車禍事故發生。因此，此轉向角度修正的工作運行在高關鍵性層級時，將會花上更多時間進行運算去取得精確到小數點後5位數的運算結果，同時還要反複進行兩次運算確認結果一致，如此才能確保車輛行駛在高速度下的安全要求 — 這就是我們所說的「**工作與安全性的關聯愈高則其關鍵性層級也會隨之提昇**」。這種概念亦可視為是工作的執行時間與其關鍵性層級的高低成正比，最早是由Steve Vestal於2007年IEEE RTSS會議中所提出[1]。

由於MCRTS的工作，除了即時系統所要求的時效性以外，還額外有關鍵性的要求，因此包含即時系統在內的許多傳統工作排程方法並不適用於MCRTS的環境。為了因應此一挑戰，目前學術界已提出許多MCRTS的工作排程方法及理論(完整的相關研究成果可參考Burns教授的Survey論文[2])，其中以SMC[4]、AMC[5]以及AMC$^+$[6, 7]等方法較為著名。SMC的全名為Static Mixed-Criticality Scheduling[4]，它針對只擁有L_{low}與L_{high}兩個關鍵層級的系統，採用捨棄逾時的低關鍵層級工作的策略(當一個低關鍵層級的工作T_i已執行其在低關鍵層級時所應執行的時間後，若l仍未完成則會被加以捨棄)，讓高關鍵層級的工作能夠有更多的機會使用處理器資源，以增進系統整體(特別是高關鍵層級工作)的可排程性。至於AMC方法，其全名為Adaptive Mixed-Criticality Scheduling[5]，它在系統初始時設定使用低關鍵性層級執行，此時接受所有關鍵層級的工作並依其優先權加以執行；但是如果有一個高關鍵層級的工作τ_h已執行超過其在低關鍵層級時的執行時間但仍未完成時，系統就會切換為高關鍵性層級執行，並且不再接受任何低關鍵層級的工作執行，只會從高關鍵層級的工作中挑選優先權最高者執行。AMC方法比起SMC方法更有彈性，但卻也有當切換到高關鍵性層級執行後，就不允許系統降回至低關鍵層級運行，導致低關鍵層級的工作再也無法被執行的缺點。因此AMC$^+$方法(全名為Adaptive Mixed-Criticality Scheduling Plus)[6, 7]為原始的AMC方法增加了贖回協定(Bailout Protocol)，允許當系統切換到高關鍵性層級執行後，還可在未來視情況(例如當處理器閒置時)再切換回低關鍵性層級執行的方法，讓低關鍵層級的工作能得到更多使用處理器的機會。

雖然已有這些設計良好的MCRTS排程方法，但由於不同MCRTS在硬體架構和軟體設計上差異性極大，不一定能將實務上現有的學理與方法套用在各種應用情境。有鑑於此，我們可以透過*SimART*在進行系統開發前，透過一個可定義系統組態與工作特性的模擬環境，事先將各種排程方法進行效能的模擬，掌握其真實運行的適用性。另一方面，我們也可以在*SimART*裡實作新的排程方法，並透過模擬來比較與現有方法的優劣，所以*SimART*對於新排程方法的研發亦有很大的助益。

3.2 國內外相關作品

本節彙整國內外相關的即時系統排程模擬的工具或平台如下：

- **Cheddar**[18]：Cheddar是由法國西布列塔尼大學(Université de Bretagne Occidentale)的F. Singhoff及其研究團隊於2002年所提出的一個用於即時系統的工作排程工具，Cheddar模擬了多核心處理器的結構，同時考慮了資源共享的問題。在處理工作排程的同時，Cheddar還關注週期性工作、非週期性工作和零散工作，增加了模擬情境的豐富性。除此

之外，Cheddar允許使用者自行設計排程方法，並以視覺化的方式呈現模擬結果。

- **Simso**[19]：是由法國的LAAS-CNRS實驗室的Maxime Chéramy、Pierre-Emmanuel Hladik、Anne-Marie Deplanche等人用Python設計並在2014年提出的即時系統任務排程的模擬器，它針對了多核心處理器的架構進行設計。Simso會將每個事件發生的模擬情形彙整成數據再通過分析、統計並以圖形化與表格的方式顯示出來。除此之外，Simso也能讓使用者自行設計工作排程方法。後續Maxime Chéramy等人也在2015年提出了SimSo的網頁版，稱為SimSoWeb，它具有簡單明瞭的使用者介面，讓開發人員可以直接透過瀏覽器進行即時工作排程的模擬以及直接設計工作排程方法的程式碼。
- **YARTISS**[20]：是由法國國立巴黎東部馬恩河谷大學(University Paris-Est Marne-La-Vallee)(UPEM)的Younès Chandarli等人於2014年提出的以Java程式語言為基礎所開發出來的即時系統排程模擬工具，因此它能透過Java虛擬機(Java Virtual Machine, JVM)在不同的作業系統環境直接執行。YARTISS支援單、多核心處理器的排程架構，同時也具備與架構相對應的多種排程方法。除了可以讓使用者自定義即時系統工作的相關參數之外，還具備工作量產生器，在確保可排程性之下，根據使用者設定的參數範圍隨機產生即時系統工作模型。使用者可以設定處理器的相關能耗參數，讓YARTISS統計出能耗的相關資訊。YARTISS具有模擬、分析以及將資訊繪製各種統計圖表的功能。
- **PERTS**[23]：是由美國伊利諾大學厄巴納(University of Illinois Urbana)的Jane W. S. Liu, Kwei-Jay Lin and C. L. Liu於1991年所開發的即時系統的環境模擬，主要目標是希望建立一個雛型開發環境和框架來使開發人員能夠在模擬的環境中測試所欲開發的雛型系統；其雛型系統可經由PERTS內的可排程性分析器(Schedulability Analyzer)確保工作之可排程性，進而建構出可在平行與分散式的硬體平台上運行的即時系統。PERTS也提供使用圖形編輯器來指定任務及資源的方法，並且可以手動或自動的方式測試工作在不同週期的情況下之可排程性分析，以進一步幫助開發人員設計適切的工作。但可惜的是，PERTS後續並沒有持續更新，亦無法透過擴展套件來增加功能。
- **MCRTsim**[3] 是由國立屏東大學吳卓俊教授所帶領的即時嵌入實驗室研究團隊，於2017年所發表的一套多核心處理器環境的即時系統模擬平台，除了支援即時系統的工作排程與同步方法的模擬外，還具備動態電壓調整技術的即時系統，可進行相關節能方法的模擬並提供系統整體耗能的資訊。MCRTsim亦支援視覺化方式呈現排程結果，讓開發人員更易瞭解與分析排程結果，如圖 1所示。本計畫所欲開發的$SimART$已經得到吳教授的同意，預計以MCRTsim為基礎，進行混合關鍵性即時系統模擬工具的實作。
- **MC**^{2}[21] 是多核心關鍵性即時系統平台(Mixed-Criticality on Multicore)，由美國北卡羅來納大學教堂山分校(UNC)的研究團隊於2014年所發表，主要參與者為Namhoon Kim、Jeremy P. Erickson 和 James H. Anderson。MC^{2}的實作是以同一研究團隊所發表的LITMUS^{RT}[22]為基礎[2]，其假設系統具備5個關鍵性層級，且分別為每個關鍵性層級在每一個處理器核心上都配置一個獨立的伺服器，每個處理器核心上都分別建立了五個負責不同關鍵性層級工作的伺服器，並且可以針對不同關鍵性層級使用不同的排程方法，不但為系統內最高關鍵層級的工作提供最高的可排程性保證，還兼顧了低關鍵性層級工作的效率。

綜上所述，在MCRTS工作排程模擬的研發仍處於起步階段，只有非常少數的作品，目前較為

[2]LITMUS^{RT}是一個專為多核心處理器的即時工作排程方法(Scheduling Algorithms)和同步協定(Synchronization Protocols)研究而設計的實驗性平台。

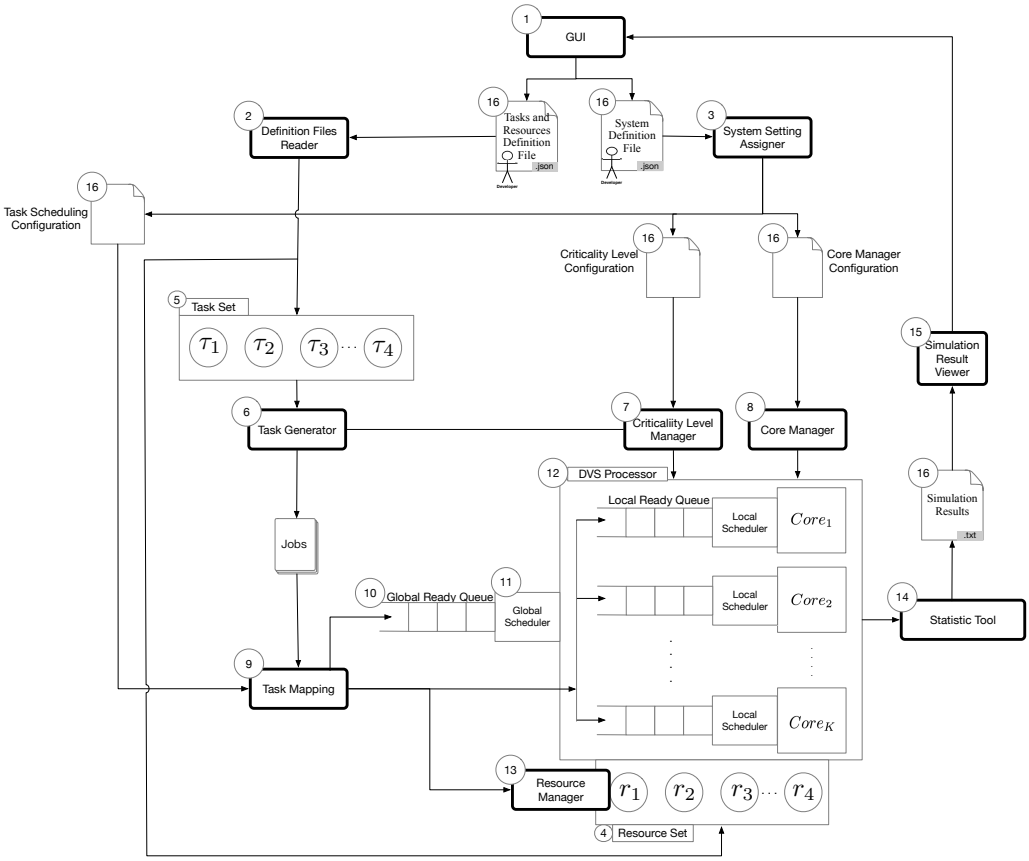

圖 2： *SimART*架構及模組圖

著名的MCRTS工作排程模擬的研發成果僅有MC[2]一項，有待學界投入更多的關注。

(四)研究方法及實作步驟

本研究的目的在於開發一套有關混合關鍵性即時系統(Mixed-Criticality Real-Time System，MCRTS)的工作排程模擬環境，我們將其命名為*SimART* (A Simulation Environment for Mixed-CriticAlity Real-Time Task Scheduling Algorithms)。此模擬環境的開發將以實驗室於2017年所開發的即時工作模擬平台MCRTsim[3]為基礎，由於MCRTsim至今已有7年沒有更新，因此我們計劃進行大幅度的升級與擴充並且就MCRTsim原有的架構及功能進行修改，以確保模擬結果的正確性。具體來說，我們將採用Eclipes 2023-12 R與JDK 21(或以上版本)以Java語言進行*SimART*的開發。另一方面，*SimART*將針對MCRTS提供工作排程模擬的環境，因此我們預計將加入新的關鍵性相關的模組，並支援著名的SMC[4]、AMC[5]、AMC$^+$[6, 7]等MCRTS的工作排程方法。此外，*SimART*也將提供混合關鍵性即時工作排程的統計工具及分析套件，視覺化的排程結果檢視，並提供圖形化使用者介面讓開發人員可用以輸入參數、選擇所欲使用的工作排程方法等相關設定。

4.1 系統架構及模組

關於我們所預計開發的*SimART*的系統架構及模組可參考圖 2，詳細的模組內容說明可參考4.2節，本節在此先就模組的功能提供簡要的說明(為清楚說明起見，以下提到相關模組時

將會加上在圖 2裡的圓形數字標籤)。

開發人員在使用*SimART*進行MCRTS的工作排程模擬時，首先可以透過①GUI進行④Task Set、⑤Resource Set以及系統環境的參數設定，當設定完成後會產生兩份組態文件分別給②Definition Files Reader以及③System Setting Assigner進行任務及資源集合以及系統上的設定如(⑦Criticality Level Manager、⑧Core Manager、⑨Task Mapping等模組)，等④Task Set建立好後會通過⑥Task Generator以及⑦Criticality Level Manager來產生具有關鍵層級優先權的Job，之後會經由原先使用者所選定使用的排程方法在⑨Task Mapping上進行工作上的分配及映射並且將Job區分為在多核心環境下的排程工作或是在單核心的環境下的排程工作，等到分類完畢後Jobs會在⑩Global Ready Queue以及⑫DVFS Processor中的Local Ready Queue中等待⑪Global Scheduler以及Local Scheduler進行相關的排程作業，在DVFS Processor運行的過程中，⑬Resource Manager會將處理器的資源做系統上的分配並由Lock(鎖住)及Unlock(解鎖)的方式來確保資源使用上的最高效率，最後等DVFS Processor將排程模擬運行完後會經由⑭Statistic Tool以及⑮Simulation Result Viewer來進行分析和統整工作排程的過程，並最後在①GUI進行顯示。

4.2 相關模組說明

此小節將針對系統架構中的模組進行說明，請參照圖2的系統架構圖來檢視下列模組。

- ① Graphic User Interface，GUI(圖形化使用者介面)：在GUI上，我們會建置出一個主視窗，並且在主視窗中提供工作量和資源量、 Task Generator、Criticality Level Manager與DVFS Processor以及系統等相關參數欄位讓開發人員可以設定並建立模型，並且提供一個可以讓Simulation Result Viewer的圖形化結果進行顯示的頁面。
- ② Definition Files Reader(定義文件閱讀器)：功能為讀取開發人員所輸入的Tasks and Resources Definition File並各自產生Task Set和Resource Set。
- ③ System Setting Assigner(系統設定指派器)：根據從開發人員所輸入的System Definition File，產生Criticality Level Manager、Core Manager和Global Scheduler等模組的組態檔案並將各自的組態檔案傳送至所需要使用的模組進行使用。
- ④ Resource Set(資源集合)＆⑤ Task Set(工作集合)：依照開發人員所輸入的工作量及資源總數，在系統內部產生資源模型$R = \{r_1, r_2, \ldots, r_n\}$以及週期性工作模型$T = \{\tau_1, \tau_2, \ldots, \tau_n\}$。
- ⑥ Task Generator(工作產生器)：Task Generator會結合Task Set的週期性工作模型及Criticality Level Manager設定所產生的Jobs為哪種關鍵層級。
- ⑦ Criticality Level Manager(關鍵層級管理器)：讀取Criticality Level Configuration後可以依照使用者所設定的Primitive Condition(優先條件)直接從DVFS Processor進行Switch(調度)及設定所需產生的Job。
- ⑧ Core Manager(核心管理器)：核心管理器會讀取Core Manager Configuration來設定DVFS Processor擁有幾個Core，和每個Core在不同的關鍵性層級上的執行速度。
- ⑨ Task Mapping(工作映射)：當Job生成後，Task Mapping會根據Task Scheduling Configuration來讀取所選定的排程方法，像是G-EDF,SMC,AMC方法等，如果該排程方法需要以多核心處理器環境來進行工作排序如G-EDF，則該Job會被分配到Global Ready Queue進行儲存；如果是使用單核心處理器環境來進行工作排序如EDF、RMS等工作排程，則會自動

分配Job到未執行的核心上的Local Ready Queue等待執行，同時Job會依照優先權的高低進行排列並儲存在等待佇列中。

- ⑩ Global Ready Queue(全域等待佇列)＆⑪ Global Scheduler(全域排程器)：這兩個模組僅在全域多核心處理器架構下運作。Global Ready Queue根據開發人員所選擇的排程方法和特定條件下的優先順序進行排序並在Job執行前都存放在等待佇列中。之後Global Scheduler根據開發人員所選擇的排程方法，將每項Job分配到不同的Core執行。
- ⑫ DVFS Processor(動態電壓調整處理器)：此模組是以Dynamic Voltage Frequency Scaling, DVFS(動態電壓調整)的處理器架構來進行設計，開發人員可以按照所選擇的動態電壓調整方法以及Core Manager去設定於DVFS Processor中擁有幾個Core和每個Core個別執行系統工作的速度，Processor內部組成為Local Ready Queue、Local Scheduler和Core，可以依照Task Mapping的要求來執行工作排程，同時如果該Job是由Global Scheduler所進行排序的，則會將各自不同的Job分配給不同的Core讓工作排程能夠Optimal(最佳化)運行。
- ⑬ Resource Manager(資源管理器)：Resource Manager是以鎖為基礎去鎖定資源的概念進行架構的設計。舉例來說，使用Exclusive(排他鎖)與LockShared Lock(共享鎖)來避免多個執行緒同時獲得鎖，並依照開發人員所選定的工作排程方法和現有的Resource Set去協調Resource的鎖定及解鎖。
- ⑭ Statistic Tool(統計工具)＆⑮ Simulation Result Viewer(模擬結果觀察器)：Statistic Tool按照從DVFS Processor所產生的模擬數據進行統計、歸納並生成Simulation Result File，Simulation Result Viewer提取檔案內的數據並繪製圖表將圖檔顯示在GUI的視窗上。
- ⑯ 各模組之相關定義檔及組態檔(Definition and Configuration Files)：Definition Files有Tasks and Resources和System，為開發人員於GUI輸入之工作量及系統參數所產生以json格式進行儲存之檔案；Configuration Files則有Core Manager、Criticality Level 和Task Scheduling，各自提供開發人員所設定之參數給相關模組所使用；Simulation Results則使用txt格式來儲存檔案，並提供給Simulation Result Viewer繪製圖檔使用。

(五)預期結果

在本計畫的預期結果部分，我們將以MCRTsim為基礎，開發符合混合式關鍵性即時系統(MCRTS)的排程模擬環境。除了更新MCRTsim套件之外，我們還將新增了關鍵性和時效性之條件參數，以及多個與MCRTS相關的工作排程方法，使其可以在*SimART*上運行並分析結果。未來我們將開發相關的數據監控及收集工具程式，針對以 Linux 為基礎的嵌入式系統，在系統運行期間週期性地收集電池剩餘電量與CPU使用率等相關資訊，這將為後續的分析及能耗模型建立提供重要的資料來源。這一階段的工作將使用實驗室現有的設備來進行相關的研究及實作，以及建立可以運行的混合關鍵性即時的工作排程環境，讓開發人員在不增加成本的情況下預測所建立系統使用的工作排程方法在處理器上的效率。包括增加關鍵性及時效性的相關參數及觸發條件，使得工作集合可以被系統讀取及排序，同時新增資源管理器模組來管理並建立相關的資源集合，使得資源在使用上更有效率。我們將增加在不同關鍵性層級下的工作排程模擬選項，如 SMC[4]、AMC[5]、AMC$^+$[6, 7] 等，並新增核心管理器、關鍵層級管理器和工作映射等模組，讓 *SimART* 的模組比起MCRTsim更加多元化。我們將使用json檔案格式儲存定義檔，使得定義檔更加簡潔以減輕模擬系統的負擔，同時提升此檔案的可讀

性。這些結果將為MCRTS的開發和測試提供重要的工具和支持。

(六)需要指導教授指導內容

本計畫的指導教授為吳卓俊老師，老師無論在教學的時候或是在指導大專生研究計畫時，在許多層面上都能夠耐心回應我所詢問的問題，讓我的研究計畫在寫作上能夠更加的完整且嚴謹。在與老師的許多次會議中能夠感受到他對學術研究的熱忱與對學生的支持，因此我總是能夠收穫滿滿。非常感謝老師願意指導我來撰寫大專生研究計畫，我一定會認真地學習並努力地執行此計畫。本計畫需要老師的指導與協助，包含系統設計與模組實作，也希望老師在未來能夠督促我，使我能夠在實作計畫和專題的過程中更加的充實和進步。本計畫需要老師指導的部分包含：(1)混合關鍵性系統的相關學理及知識並運用在$SimART$；(2)各項工作排程方法的設計及實作；(3)關鍵性層級在工作上的優先權；(4)架構上的調整和修正；(5)實作動態電壓調節功能的處理器並且能夠連結模組來模擬工作排程以及進行可行性分析。

(七)參考文獻

References

[1] S. Vestal. Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance. In: ***28th IEEE International Real-Time Systems Symposium***. Tucson, Arizona, USA, Dec. 3 – 6, 2007, 239 – 243.

[2] A. Burns and R. I. Davis. ***Mixed Criticality Systems - A Review (13th Edition)***. Feb. 2022.

[3] J. Wu and Y.-C. Huang. MCRTsim: A Simulation Tool for Multi-core Real-Time Systems. In: ***IEEE ICASI***. May 2017.

[4] S. Baruah and S. Vestal. Schedulability Analysis of Sporadic Tasks with Multiple Criticality Specifications. In: ***20th Euromicro Conference on Real-Time Systems (ECRTS)***. July 2008, 147 – 155.

[5] S. K. Baruah, A. Burns, and R. I. Davis. Response-Time Analysis for Mixed Criticality Systems. In: ***32nd IEEE RTSS***. 2011, 34 – 43.

[6] I. Bate, A. Burns, and R. I. Davis. A Bailout Protocol for Mixed Criticality Systems. In: ***27th ECRTS***. 2015, 259 – 268.

[7] I. Bate, A. Burns, and R. I. Davis. An Enhanced Bailout Protocol for Mixed Criticality Embedded Software. ***IEEE Transactions on Software Engineering***, 43(4):298 – 320, 2017.

[8] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. ***Journal of the Association for Computing Machinery (JACM)***, 20(1):46 – 61, Jan. 1973.

[9] J. Wu. Energy Efficient Dual Execution Mode Scheduling for Real-Time Tasks with Shared Resources. ***Computer Systems Science and Engineering***, 31:239 – 253, May 2016.

[10] J. Wu and X.-J. Hong. Energy-Efficient Task Scheduling & Synchronization for Multicore Real-Time Systems. In: ***3rd HPSC***. 2017.

[11] M. A. El Sayed, E. S. M. Saad, R. F. Aly, and S. M. Habashy. Energy-Efficient Task Partitioning for Real-Time Scheduling on Multi-Core Platforms. ***Computers***, 10(1), 2021.

[12] M. A. El Sayed, M. S. El Sayed, S. M. Habashy, and R. F. Aly. Online DVFS Scheduling Algorithm for Sporadic Tasks with Energy-Conscious in Hard Real-Time Systems. In: ***16th International Conference on Computer Engineering and Systems***. 2021, 1 – 7.

[13] S. Palamut, T. Gönültas, A. Elewi, and E. Avaroglu. Task Scheduling Algorithms and Resource Access Protocols in Real Time Systems. In: ***2019 International Artificial Intelligence and Data Processing Symposium (IDAP)***. Sept. 2019, 1 – 6.

[14] A. K. Mok. Fundamental Design Problems for the Hard Real-Time Environment. Ph.D. Dissertation. MIT, 1983.

[15] D.-I. Oh and T. Baker. Utilization Bounds for N-Processor Rate Monotone Scheduling with Static Processor Assignment. ***Real-Time Systems***, 15(2), 1998.

[16] T. P. Baker. ***A Comparison of Global and Partitioned EDF Schedulability Tests for Multiprocessors***. Tech. rep. TR-051101. Department of Computer Science, Florida State University, 2005.

[17] S. K. Baruash, N. K. Cohen, C. G. Planton, and D. A. Varvel. Proportionate Progress: A Notion of Fairness in Resource Allocation. ***Journal of Algorithmica***, 15(6):600 – 625, 1996.

[18] F. Singhoff, J. Legrand, L. Nana, and L. Marcé. Cheddar: a flexible real time scheduling framework. In: ***SIGAda Conference***. 2004.

[19] M. Chéramy, P.-E. Hladik, and A.-M. Déplanche. SimSo: A Simulation Tool to Evaluate Real-Time Multiprocessor Scheduling Algorithms. In: ***5th WATERS***. 2014.

[20] Y. Chandarli, M. Qamhieh, F. Fauberteau, and D. Masson. YARTISS: A Generic, Modular and Energy-Aware Scheduling Simulator for Real-Time Multiprocessor Systems. In: ***3rd WATERS***. 2014.

[21] N. Kim, J. P. Erickson, and J. H. Anderson. Mixed-Criticality on Multicore (MC^2): A Status Report. In: ***International Workshop on Operating System Platforms for Embedded Real-Time Applications***. 2014.

[22] J. M. Calandrino, H. Leontyev, A. Block, U. C. Devi, and J. H. Anderson. LITMUSRT: A Testbed for Empirically Comparing Real-Time Multiprocessor Schedulers. In: ***27th IEEE Real-Time Systems Symposium (RTSS)***. Dec. 2006, 111 – 123.

[23] J. Liu, J. Redondo, Z. Deng, T. Tia, R. Bettati, A. Silberman, M. Storch, R. Ha, and W. Shih. PERTS: A Prototyping Environment for Real-Time Systems. In: ***1993 Real-Time Systems Symposium***. 1993, 184 – 188.