基於圖像化程式編譯工具 Blockly 之運動控制技術

Motion Control Applications Featuring Graphical Programming Approaches with Blockly

周佑儒[1*]、李宜靜[1]、陳世剛[2]、李桂銘[3]

[1] 工研院機械所 控制核心技術組 機電控制整合部 副研究員

[2] 工研院機械所 控制核心技術組 機電控制整合部 研究員

[3] 工研院機械所 控制核心技術組 機電控制整合部 產品經理

摘要：在發展多軸運動控制系統時，開發者通常需要撰寫程式的背景與邏輯概念，門檻相對高，有鑑於此，工研院機械所開發了圖像化 MCCL 運動控制函式庫，藉由 Blockly 將程式具像化為積木進行拖曳和組合，取代手動輸入程式碼，以降低初學者的程式開發門檻，增進產業的專案開發效率，並使運動控制輕易上手與快速應用；本文將介紹 MCCL 結合 Blockly 的圖像化運動控制函式庫，講述如何透過以積木程式編輯器來取代一般撰寫程式碼的方式執行運動控制命令，讓使用者可以達成快速開發運動控制系統的目的。

Abstract：Computational thinking and programming experience are required for developers when building multi-axis motion control systems, so the threshold of development is relatively high. In light of this, ITRI MMSL developed the graphical motion control command library (MCCL), which adopts Blockly to visualize programs into blocks for dragging and linking while replacing programming manually, and it can lower the educating threshold, boost the efficiency of project development among industries, and make motion control handled easier and applied faster. This article introduces the graphical motion control program of MCCL combining Blockly, explaining how to execute motion commands by stacking blocks instead of writing the general code so that users can develop motion control systems quickly.

關鍵詞：Blockly 積木程式編輯器、圖像化編譯界面、MCCL 運動控制函式庫

Keywords：Blockly, Graphical programming interface, MCCL

前言

現今開發能讓初學者上手的程式可說日益遽增，而低程式碼開發平台 (Low-code Development Platform, LCDP) 已經在產業中快速成長，在傳統的程式開發環境，如第三代程式語言 C、C++ 與 C# 等，會以程序導向語言的形式做開發，主要包含過程與函式呼叫，以典型的 C 語言為例，在程式編輯中包含宣告、邏輯運算、函數與運算符號等複雜的語法 [1]，其僵化與嚴謹的語法對初學者相對難以上手，一個開發更方便的工具勢在必行，在第四代程式語言的目標會以非程序導向語言的方向來改善其抽象化程度與軟硬體整合，而在第五代程式語言則更使用自然語言來做開發，加深人機整合與程式自動化，啟發了程式自動化和圖像化程式編輯的概念，在其概念下 Forrester 於 2014 年時提出了低程式碼平台 LCDP 的雛形，在後續的時間 LCDP 的數量以雨後春筍般的速度增加；LCDP 的目的為在當缺少專業開發人員時，依舊能達成商業資訊系統的核心目標 [2]，根據研究公司做的統計資料 [3]，其市場規模會從 2020 年時的 132 億美元，五年後將成長到至 455 億美元，成長幅度可達到 3 倍以上，原因可歸類於其特色：1. 相對快速的程式開發。2. 減少開發者

與客戶之間的認知差異，以增加系統管理的效率 [4]。LCDP 會以圖形使用者介面 (Graphical User Interface, GUI) 上建立開發環境，並使用視覺化程式設計語言 (Visual Programming Language, VPL)，而非使用複雜而艱深的文字編輯方法，低程式編輯可使開發者降低思考文法或語意，增加在功能性的驗證與開發，同時也降低除錯與建置的時間；而在開發的方法中，可分為：

1. 模型驅動的軟體開發方法。
2. 快速應用程式開發。
3. 程式碼產生自動化。
4. 視覺化程式設計語言 [5]。

在目前的 LCDP 中，Forrester 報告中提出著名且占有市場龍頭地位的 Microsoft PowerApps、以及 Mendix 與 OutSystems 等 [6-7]。

而在低程式碼的基礎上，為更便利使用，而研發出了無程式碼開發平台 (No-code Development Platform, NCDP)，其目標為使用者只需邏輯性的思考程式功能而無須程式編輯的知識即可做程式開發 [8]，為了達成去程式碼的目的，NCDP 也會以 GUI 與 VPL 來做程式開發，如 Scratch 與 Blockly 等視覺化應用程式 [9-10]，將程式碼以積木的形式做包裝，使用者可用積木組合的形式來做程式編輯 [10]，堆砌好的積木可以透過自動轉成相對應的程式語言，如 Lua (一種腳本語言)，LCDP/NCDP 其優勢在於：1. 節省時間，降低開發門檻，一般未經專業訓練的人員也可參與程式開發。2. 提高效率，人員只需專注於建置程式，無須關注語法等瑣碎的事項。LCDP/NCDP 的出現將原本複雜的程式開發注入新的活水，要如何將既有的模型、程式或平台轉換到 LCDP/NCDP 將會是一大挑戰。以 Blockly 為例，開發者需定義所需的積木方塊，其積木會透過 GUI 呈現給使用者做程式編輯 (積木組合)，定義好的積木裡應包含特定的函式與定義值等，再透過所需的程式語言轉譯成相對的程式碼，後續執行轉譯後的程式代碼；在程式的執行路徑中，會有多層的轉譯與呼叫，目的為將複雜的程式代碼包裝成簡易的 GUI 與 VPL，然而在包裝與轉換的途中，會產生多項限制，主要的缺點可分為：

1. 研發成本高，開發者需付出研發成本且使用者也需要再購買相對應的 LCDP/NCDP。
2. 客製化程度相對限制，無法將所有模組搬到 LCDP/NCDP。
3. 執行速度相對慢，因為有額外的資料須處理與複雜的程式圖像化過程。
4. 對於複雜且耗時的程式，難以或無法用 LCDP/NCDP 來達成，故無法執行相對複雜的任務。
5. 相對複雜的程式維護，多步驟的轉換將產生無法預測的錯誤，需要更多的除錯時間。
6. 複雜的整合環境，開發者需要多設計使用者介面，並將資料與運算整合在程式中 [11]。

而在 LCDP/NCDP 的產業運用中，以本文專注的運動控制系統為例，文獻指出在目前的控制系統中，其平台如用 Blockly 等 LCDP/NCDP，可以為產業創造價值與達到教育訓練目的 [12-13]，Kapila 等人 [13] 也指出如果使用 Blockly 來做運動控制與 C 語言的教育訓練，七成以上的受試者 (未經專業訓練) 可以因使用過平台而了解 C 語言與運動控制的運用，更有九成的受試者可以熟悉 LCDP/NCDP 的操作流程與圖像式程式開發。由結果可得知 LCDP/NCDP 其簡易好上手的特性在運動控制程式開發上的優勢，本文將以工研院機械所開發之運動控制函式庫 MCCL[14] 導入 LCDP/NCDP 的設計概念，以結合 Blockly 開發圖像化的運動控制函式庫，提供更友善的系統開發，以六軸運動控制卡 (Exquisite Positioning Control Inputs and Outputs, EPCIO) 為例，進一步說明如何使用圖像化的運動控制函式庫開發運動控制系統；將原本在開發環境上複雜的人機介面與運動函式庫，化繁為簡地使用腳本語言 Lua 轉換成簡易的積木，並以積木拖曳與堆疊來達成完整的運動控制，包含程式設計、運動控制卡的使用與相關軟硬體控制等，加速產業開發運動控制系統與產品多樣化。

MCCL 運動控制函式庫

MCCL 為工研院機械所開發的運動控制函式庫，目標在於簡化運動控制中複雜的運算、資料處理等步驟，使用者只需透過相對簡易的呼叫介面即可開發整合系統。在 MCCL 中，為滿足複雜的運用情境，有提供上百個函式給開發者使用，主要可分為：

1. 機構參數，可定義實際機台上的機構特性。
2. 一般運動控制，提供常用的運動命令，如直線運動、點對點運動與圓弧運動等，也包含命令的控制，如暫停與繼續等。
3. 定位控制，設定定位誤差參數等控制範圍。
4. 編碼器控制，可讀取計數值，閂鎖 (Latch)、index 觸發等訊號。
5. I/O 訊號處理，讀取接點與觸發中斷服務等。
6. D/A 轉換功能，可由控制卡上的數位類比轉換器 (Digital Analog Converter, DAC) 輸出設計電壓。
7. A/D 轉換功能，可透過控制卡上的類比數位轉換器 (Analog Digital Converter, ADC) 讀取指定電壓並做數位轉換，且 DAC 與 ADC 功能都提供中斷函式給使用者做相關程式規劃。
8. 計時器功能，使用者可設定計時時間與中斷函式，系統將依時間觸發中斷函式。
9. 看門狗功能，設定看管時間週期，系統將定時監測系統，如有軟硬體發生錯誤，將會重新設定系統以保護平台安全。

而當呼叫 MCCL 運動函式後，會將各相關命令放入運動控制命令緩衝區 (Motion Command Queue)，緩衝區為 Queue 的形式，採用先進先出演算法 (First In First Out, FIFO)，緩衝區會抓取 (Get) 運動命令來解譯與做後續的差補運算，這兩個動作 (抓取與解譯 / 差補) 為獨立動作，優點在可無視當下是否已完成運動命令而將新的命令放入緩衝區，節省等待時間，但緩衝區有 $1x10^4$ 個命令的上限。

MCCL 具有獨創的功能運動群組 (Group) 的功能，可達成多軸同動或不同動的運動控制需求，

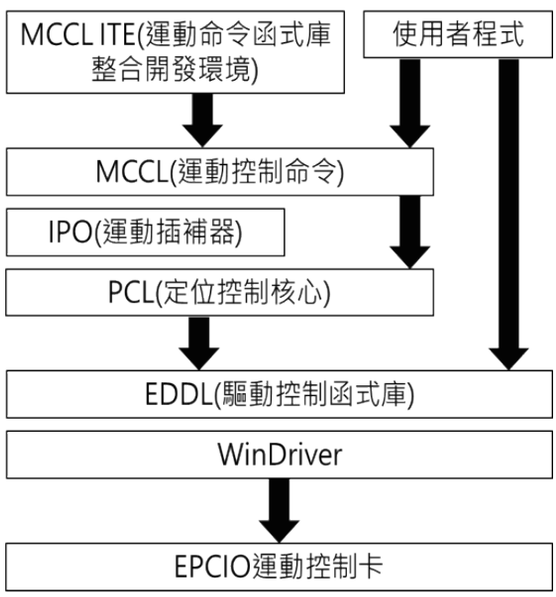


圖 1 MCCL 運動控制系統架構

使用者可規劃多個 Group，各 Group 為獨立而不影響彼此的執行，而在同 Group 中的運動軸為同動，此功能可提供了操作的彈性與降低了複雜的多軸運動控制運算 [15]，尤其是複雜的機械手臂與 IC 產業中 [16]。MCCL 系統架構圖如**圖 1** 所示，其中包含上層的使用者程式與整合開發環境，支援微軟的 Windows 等作業平台，並可使用 Visual C# 與 Visual C# NET 等開發環境；在 Visual C# 開發環境中，Lua 等腳本語言因體積小，結合性高 [17]，非常適合與 C# 整合。本文以 Blockly 程式為 MCCL 開發簡易的圖像化開發介面，將上層的使用者程式以 Blockly 完成開發介面，並將產出的積木轉譯成與 C# 的深度整合的 Lua 嵌入式語言後，再整合多功能的 MCCL 做複雜的運動控制，圖像化運動控制函式庫將會造福運動控制產業，促使工控產業的應用領域與開發者背景更加廣泛與多樣化，增加產業快速開發系統與整合，開創另一片天空。

Blocky 圖像化程式編輯工具

Blockly 為 Google 為了降低程式開發難度所研製的開源視覺化程式編輯器，以超文本標記語

言 (HyperText Markup Language, HTML)、可延伸標記式語言 (Extensible Markup Language, XML)、多範式進階直譯程式語言 (JavaScript) 等網頁程式為主體，提供程式編輯介面並使用圖文與圖像化的圖形如積木來代表傳統程式編輯中的變數與邏輯等，例如變量、邏輯判斷、迴圈等。Blockly 可建立圖像化的程式語言並回傳成使用者產出的程式代碼，將圖像化的積木匯出成多種程式語言如 JavaScript、Python 與 Lua 等。Blockly 提供客製化圖像化介面與匯出的程式語言，可將既有的平台做移植或介面優化，達成低成本、高效率的 LCDP/NCDP 的開發。Blockly 介面平台如**圖 2**，分成三大區域。

1. 程式執行工具欄，可將組合好的積木做執行、存檔、讀取與搜尋等功能。
2. 積木分類欄，將所有的積木以客制化的分類依序排序，也可用程式執行工具欄中的 Search 做網頁 (JavaScript) 的搜尋而將積木列舉出。
3. 程式積木，可將程式代碼包裝成積木，後續可依照使用者組合好的積木群自動轉譯成相對應的高階程式代碼，如 Lua。

Lua 為一種腳本語言，有簡單、輕量、高延伸與整合性的特點 [17]，在傳統的編輯流程中，以「編寫、編譯、連結、執行」為主 [18]，腳本語言為縮短流程而將指令碼直譯執行而非編譯，故效率和上手難度都有所改善，也符合為了降低程式開發難度的目的。在眾多的腳本語言中，Lua 可以做到延伸和嵌入，可讓開發者使用 C 語言等程式語言與 Lua 整合。Lua 程式大小只有不到 256 kB，並使用 ANSI C 語言編寫，無圖形介面等多餘的功能，只為擴充與嵌入 C 等語言為生，並支援物件導向程式設計 (但無繼承)，且 Lua 資料型態相對少，語法更加簡單，也支援垃圾回收等功能，並支援多種應用程式介面 (Application Programming Interface, API) 與操作平台上，舉例來說，本程式會使用 NLua 和 KeraLua 等 API 與 MCCL 所需的 C# 整合，使開發更加靈活。本文將以 Blockly 設計使用者介面，Blockly 中的積木程式將轉譯成 Lua，再使用上述的 API 相關套件與額外的動態連結函式庫 (Dynamic-Link Library, DLL) 來與 MCCL 做連接與運動控制。有了 Blockly 自動化、圖像化的介面與功能，使用者無須了解積木實際程式碼內容、Lua 轉譯的過程與 MCCL 控制等複雜的步驟，只需將心力放在將積木邏輯性與時序性的組合即可達到複雜運動控制。

Blocky 與 MCCL 的整合程式

以 Blockly 設計操作介面與所需的積木群的程式介面如**圖 3(a)**，主要以 JavaScript 網頁程式的前端來開發，介面包含積木、積木的分類與相關

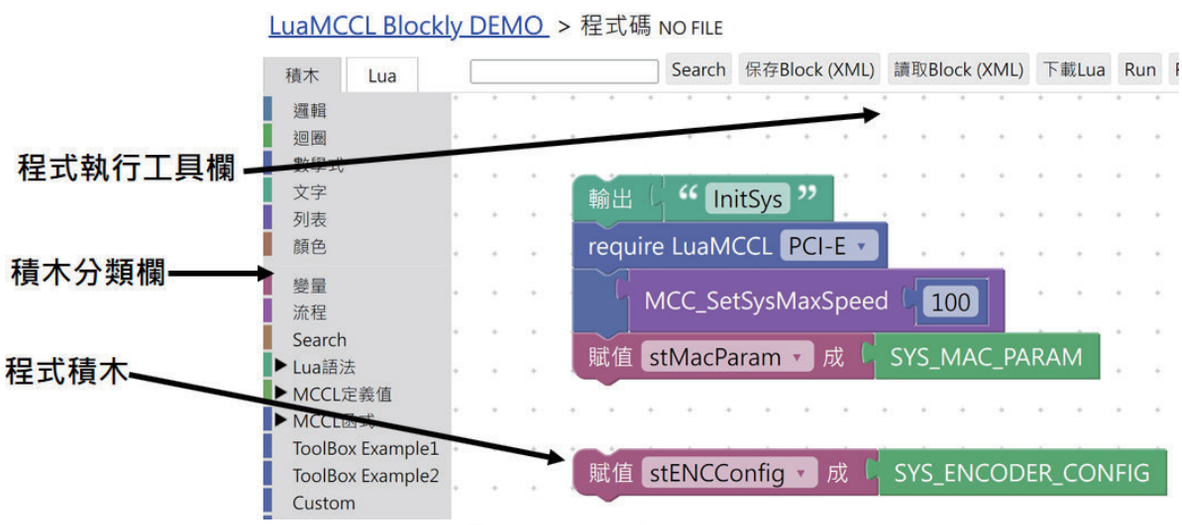
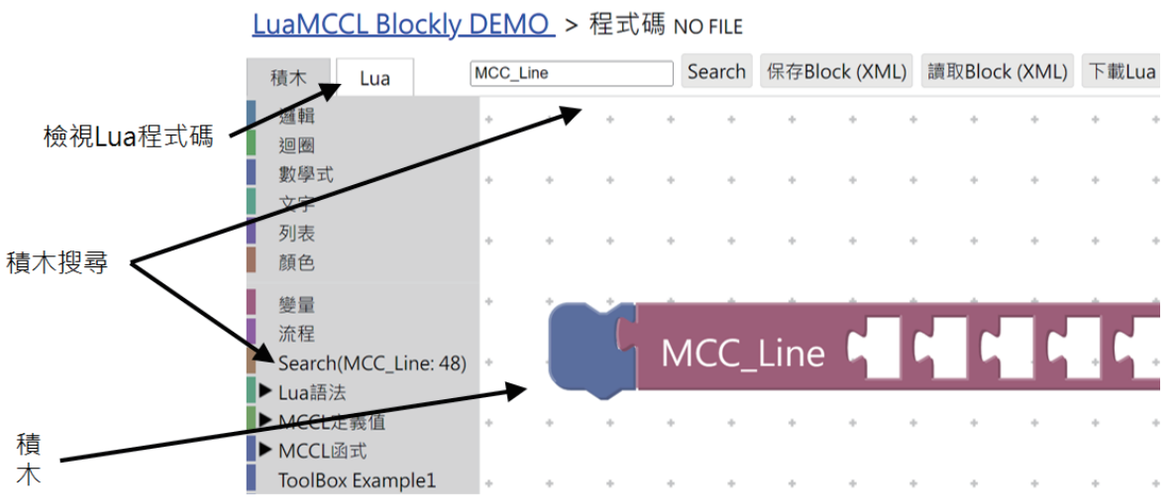


圖 2 Blockly 程式介面



(a)

(b)

圖 3 Blockly 介面與積木內容代碼，以 MCC_Line 為例 (a) 程式介面與積木外觀 (b) 積木內部的 Lua 程式代碼

執行。在積木群的創建中，分成兩個部分：1. 積木外觀。2. 積木內部代表的程式代碼；積木的外觀使用 JavaScript 語法來做定義，可定義顯示的文字、字體、積木顏色等參數；積木內的程式代碼為一段符合 Lua 語法的 JavaScript 文字檔，當上述的積木外觀與定義的 Lua 程式碼完成後，從選單拖拉積木後即可產生對應的積木外觀與程式碼如**圖 3(b)**，故當使用者在程式介面中組合積木時，也就是組合積木中的 Lua 程式碼，以達到程式編輯的目的。

除了基本的積木外，如同常見的網頁程式，還包含使用者的操作介面，包含執行的按鍵與視窗等，介面如**圖 4**，控制按鍵位於程式上方，程式中間為置放積木的空間，程式左方為分類，將所需的積木群依照 MCCL 的功能分成：1.Lua 語法。2. MCCL 定義值。3. MCCL 函式。使用者可選取分類中的積木拖曳到中間的視窗來做程式編輯，當所需的積木群組合完畢後，程式可自動轉譯成 Lua 代碼，再透過程式上方的控制按鍵進行操作，按鍵與功能說明如下：

1.Run: 執行積木程式。

2.Run Step: 數步執行積木程式，按下後可啟動 Step1、StepN，提供單步或多步執行積木程式並每當執行到該積木時，該積木會亮起。

3.Step1、StepN: 按下 Step1 可執行單一積木，若按 StepN 將出現一個對話方塊，可輸入一次想要執行幾個積木。

4.Stop: 停止執行積木程式，將按鈕狀態回復至尚未執行狀態。

5.Search: 以使用者輸入文字當成關鍵字搜尋符

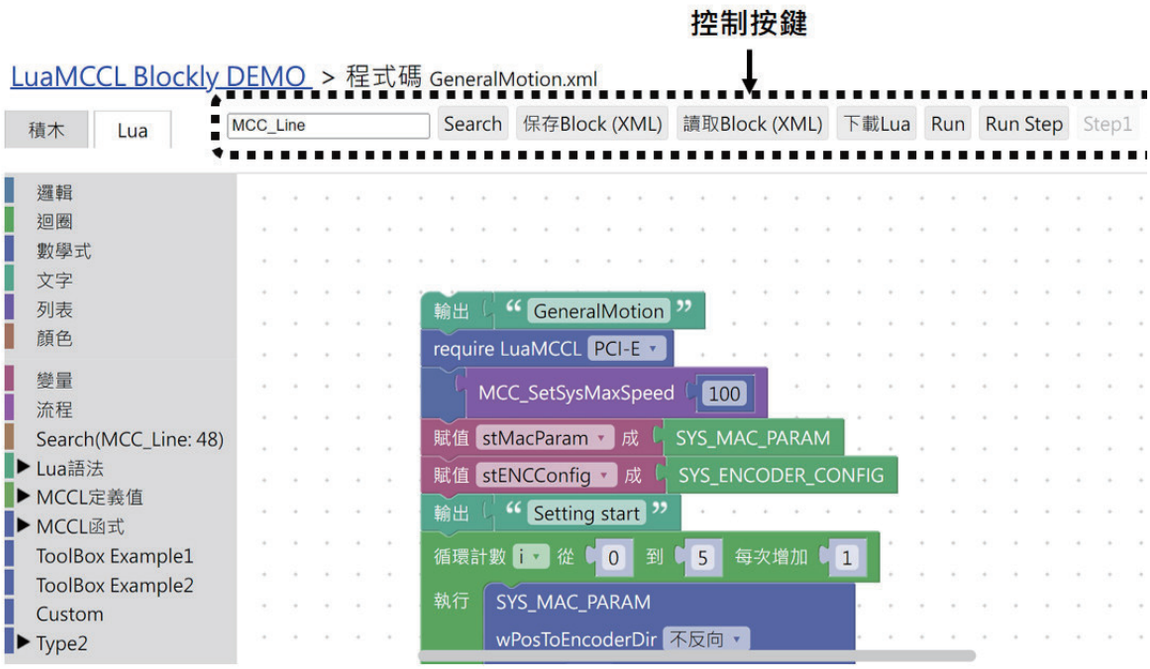


圖 4 Blockly 開發介面與相關控制按鍵

合的積木並顯示於分類中的 Search。

6. 保存 Block (XML)/ 讀取 Block (XML): 保存或讀取 .xml 的檔案。
7. 下載 Lua: 把視窗中組合好的積木轉成 Lua 後輸出並存檔。

藉由上述的介面，使用者可利用分類、積木與控制按鍵來開發運動控制程式，在傳統的運動控制程式中，完整的程式碼可依照時序性的功能分為：1. 啟動運動函式庫與設定相關參數。2. 執行控制命令。3. 關閉運動函式庫，細節如**圖 5**，在不同階段需相對應的程式碼要撰寫，其程式碼與程序邏輯需要熟練的人員輸入並有相關程式撰寫的背景，本文的程式只需使用者依照傳統的開發流程，將含有程式碼的積木依照時序與功能堆疊即可，以控制伺服馬達運行到特定位置的範例程式 (General Motion) 來說明，傳統的程式流程為：1. 啟動系統後完成 Group、機構與編碼器等參數設定。2. 開啟伺服迴路控制與設定進給速度，執行直線運動命令。3. 待命令完成後，關閉系統。1.~3. 流程中的積木群組合後如**圖 6**，執行的時序為

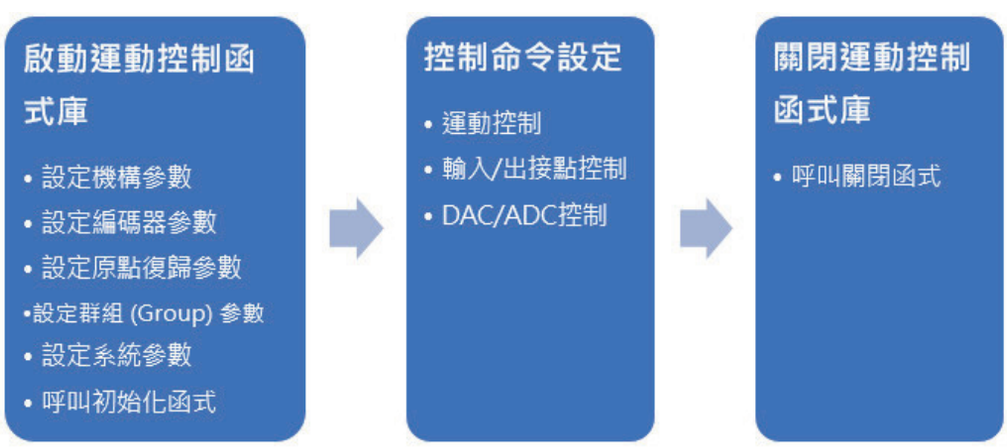


圖 5 運動控制開發程式流程

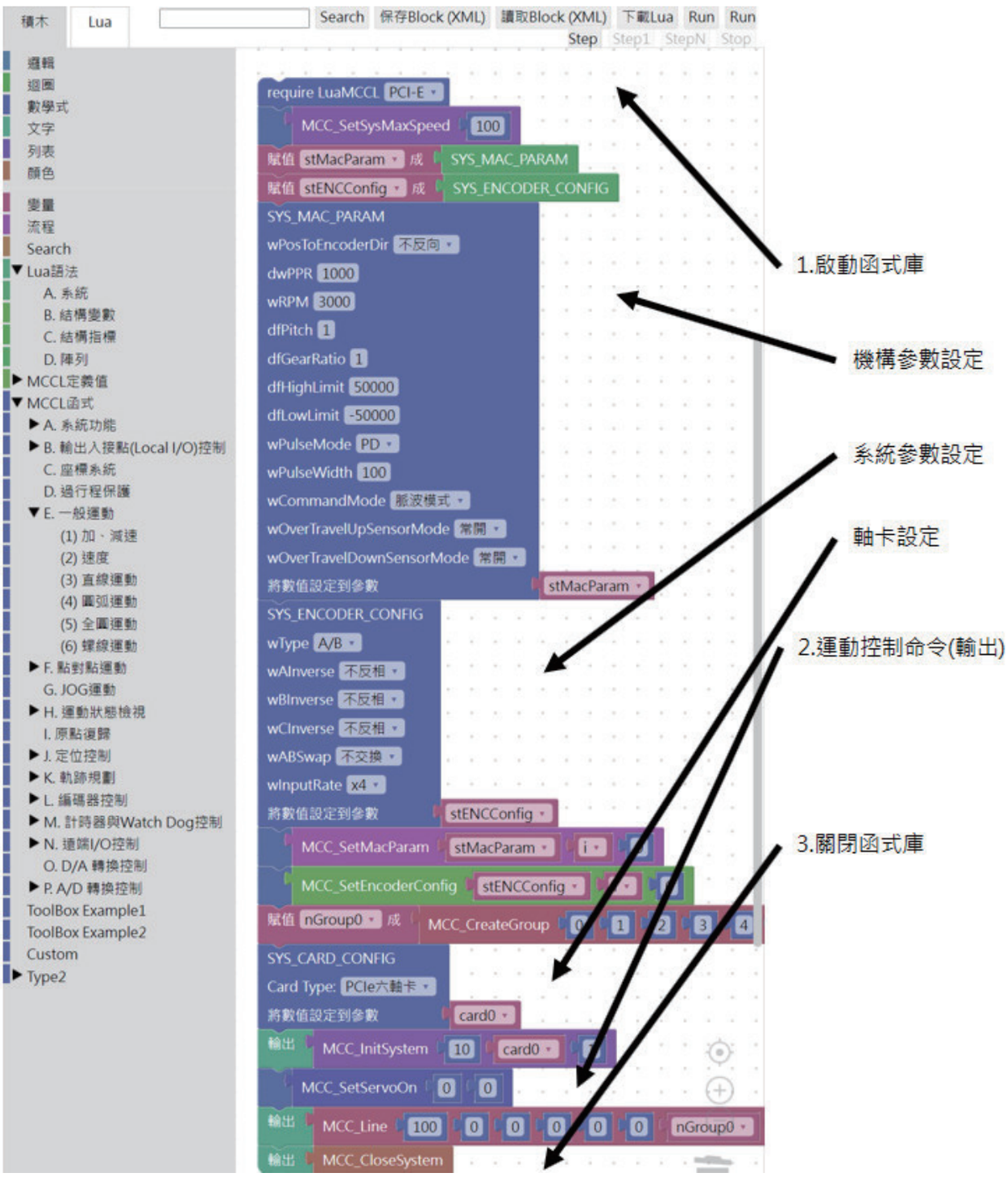
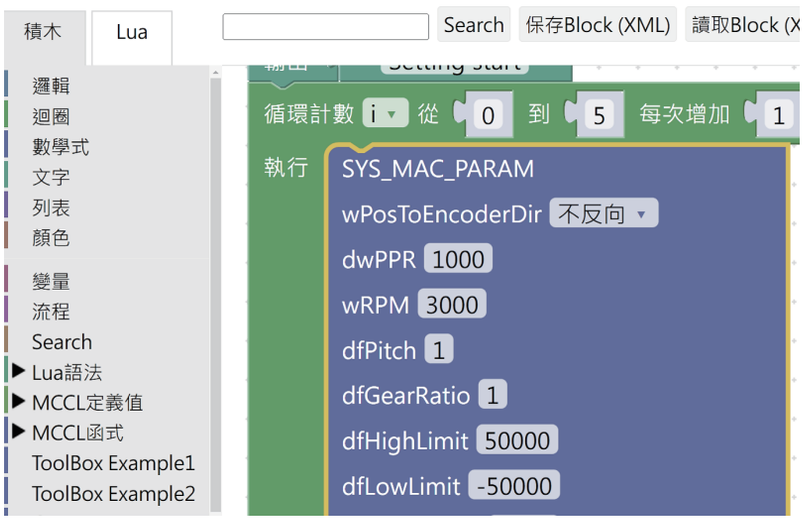
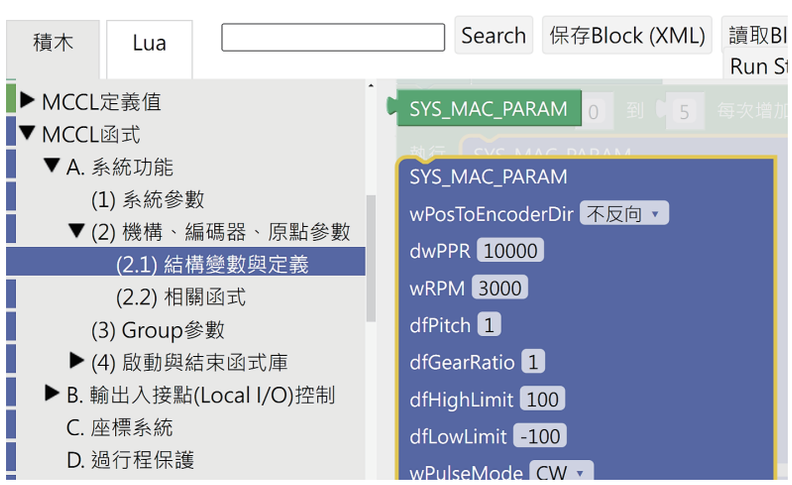


圖 6 完整積木程式用途說明

從左到右，從上到下，上述的每個流程與動作都可以從分類中找到相關的積木去做組合，如機構參數設定的積木如**圖 7(a)**，積木裡面即包含所需的程式碼，也包含下拉式選單與自填入的參數，只需從分類欄如**圖 7(b)** 中將積木拖曳到中間的編輯視窗即可完成程式編輯，其他積木也可生成與組合，當完整的積木群依照需求和正確的時序組合後如**圖 6**，就可利用控制按鍵執行完整的運動控制如**圖 4**。當使用者按下執行按鍵後，積木群會自動轉譯成 Lua 程式碼，透過使用 MCCL 的動態連結函式庫進行功能連結，達到控制運動控制卡的



(a)



(b)

(c)

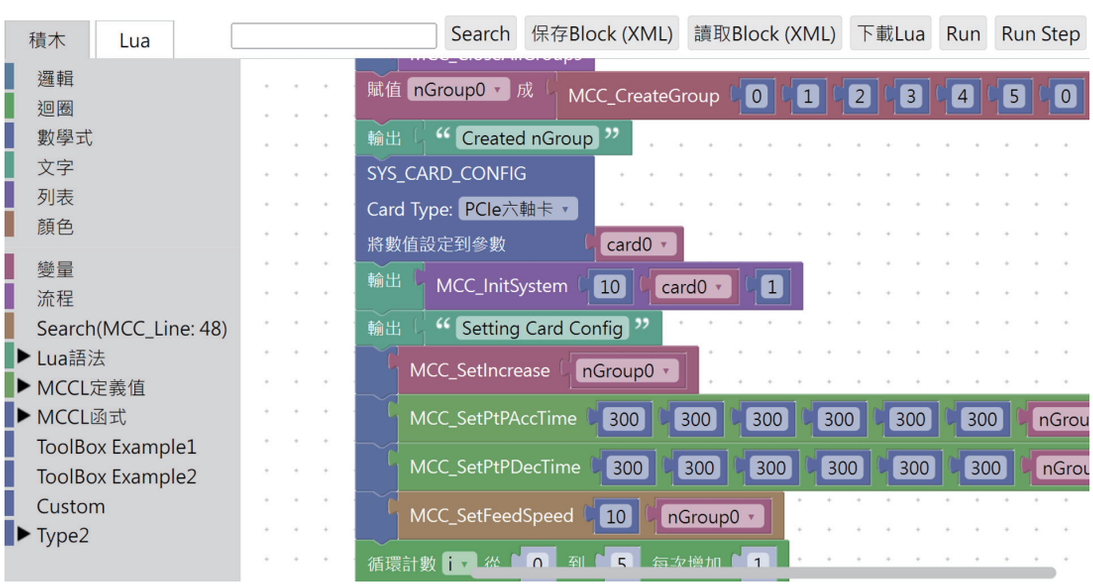
圖 7 積木程式流程與使用示意圖 (a) 用於機構參數設定之積木外觀 (b) 積木的分類選單 (c) 本程式按下上方執行命令後的執行結果

效果。**圖 7(c)** 顯示本程式可完成 EPCIO 控制卡執行直線運動命令，並將函式回傳值輸出於命令視窗，相較於原本在 Visual C# 平台上的開發環境如**圖 8(a)**，本程式的編輯介面化繁為簡成更加精簡的區域與按鍵以供使用者快速上手如**圖 8(b)**，在程式編輯上，可以從原先的文字與符號改成用更簡潔的積木方塊，並在語法與邏輯上，本程式可以自動產生邏輯、參數與副程式等，如迴圈 Loop 與中斷副程式，上手難度相對低，更適合新手做運動控制程式開發。

本文開發的程式可自動生成包含多行時序性 Lua 程式碼的檔案，稱之腳本，除了將腳本直接透過本程式的 Blockly 運行，為了改善 LCDP/NCDP 無法執行相對複雜與耗時任務的缺點 [11]，腳本

```
268         int nRet = 0;
269
270         SYS_MAC_PARAM       stAxisParam   = new SYS_MAC_PARAM();
271         SYS_ENCODER_CONFIG  stENCConfig   = new SYS_ENCODER_CONFIG();
272         SYS_GROUP_CONFIG_EX stGroupConfig = new SYS_GROUP_CONFIG_EX();
273         SYS_CARD_CONFIG     stCardConfig  = new SYS_CARD_CONFIG();
274
275         MCCL.MCC_SetSysMaxSpeed(100); // set max. feed rate
276
277         for( nChannel = 0 ; nChannel < 6 ; nChannel++)
278         {
279             // set mechanism parameters
280             stAxisParam.wPosToEncoderDir          = 0;
281             stAxisParam.dwPPR                     = 10000;
282             stAxisParam.wRPM                      = 3000;
283             stAxisParam.dfPitch                   = 1;
284             stAxisParam.dfGearRatio               = 1;
```

(a)



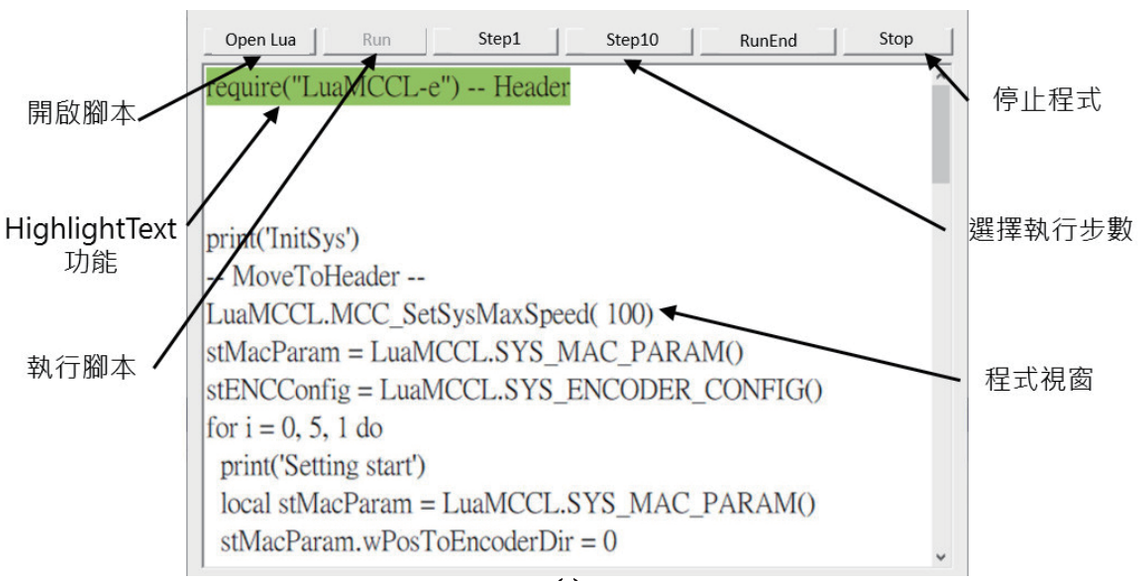
(b)

圖 8 在不同平台上開發的運動控制程式 (a) 以傳統文字編輯方法的 Visual C# 平台 (b) 本文以圖像式編輯方法的 Blockly 平台

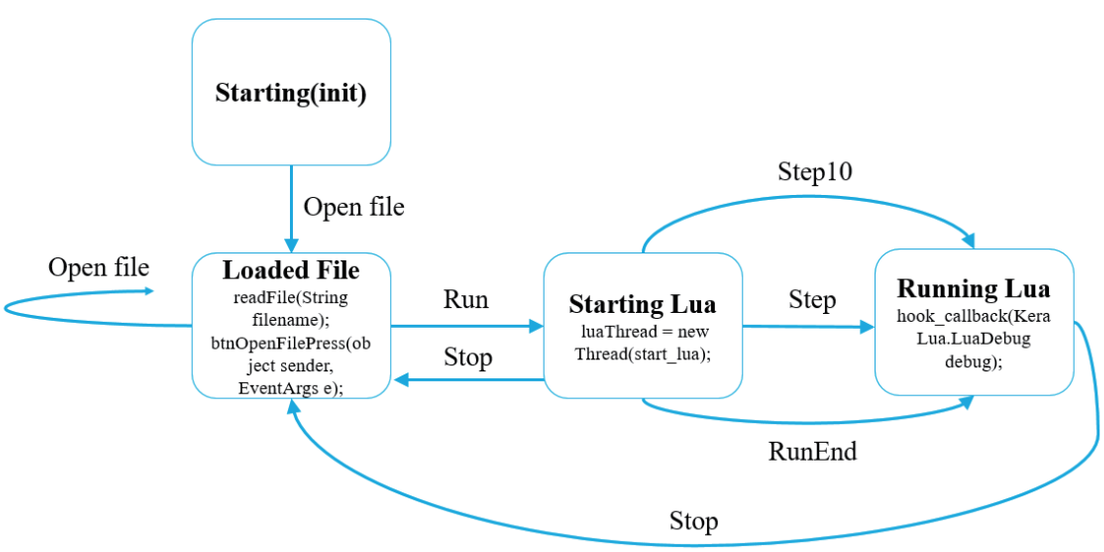
執行程式如**圖 9(a)** 可讀取與運行 Blockly 產出的腳本，並利用介面中的功能按鍵做相關的動作，流程圖如**圖 9(b)**，功能按鍵說明如下：

1. Open Lua: 可開啟腳本；如無讀取而直接執行 Run，會顯示 "Please read Lua file first"，並鎖定而無法按下其他按鍵，如：Step1、Step10、RunEnd、Stop。
2. Run、Step1、Step10、RunEnd 與 Stop: 按下 Run 可執行載入的腳本，須執行的行數可以透過按鍵決定，有單步 (Step1)、十步 (Step10) 與單步執行到最後一行 (RunEnd)。
3. Stop: 強制停止腳本的執行，也可重啟程式來確保程式正常停止。

為了提供使用者更友善的除錯與監控功能，在此介面中提供 HighlightText 功能，當使用者按下 Run 或 Step 按鍵時，在 Front panel 中的 Lua 程

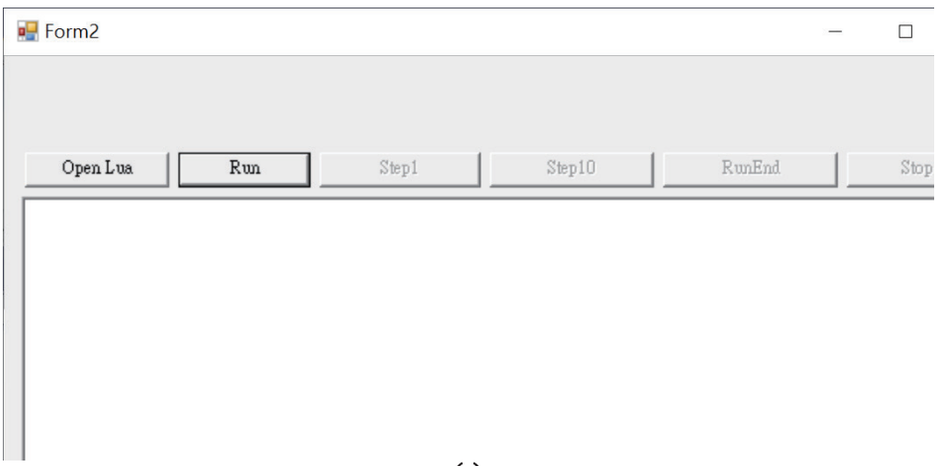


(a)

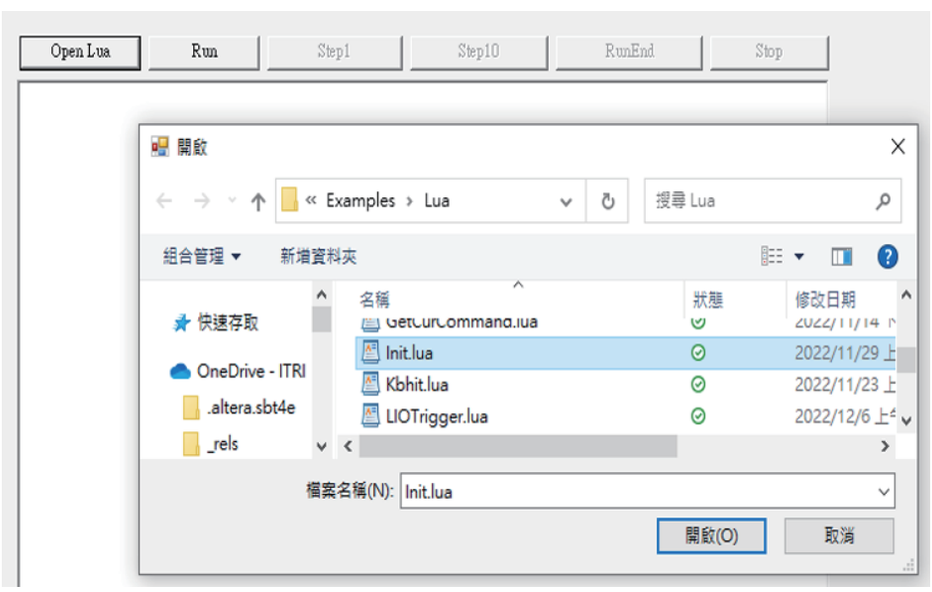


(b)

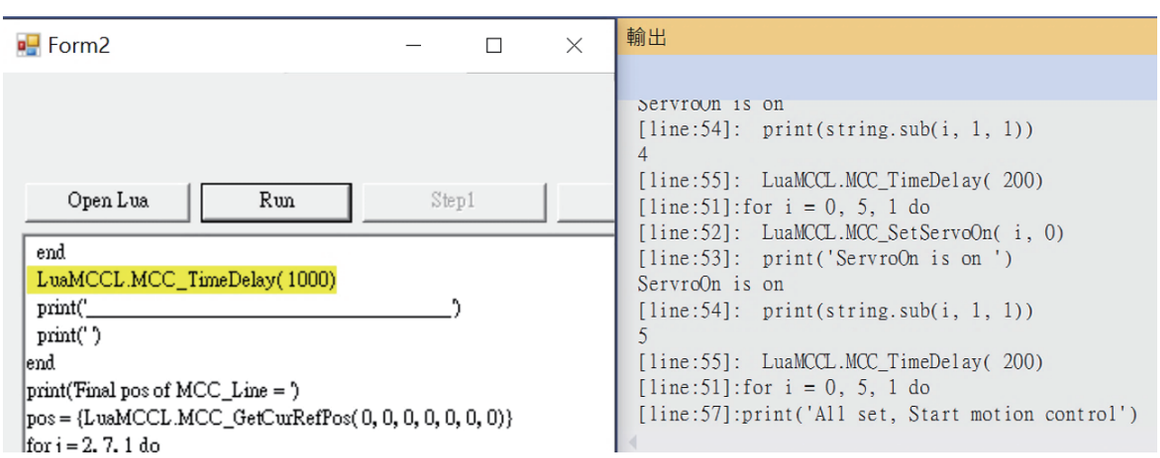
圖 9 本文的腳本執行程式 (a) 程式介面、(b) 功能按鍵流程圖



(a)



(b)



(c)

圖 10 腳本執行程式使用流程 (a) 程式基本外觀與相關按鍵 (b) 程式讀取檔案 (c) 程式執行與監控視窗顯示

式碼就會依目前執行的行數將運行的程式碼進行不同底色的顏色標示。

腳本執行程式可透過圖像化的按鍵與顯示視窗來做互動與控制如**圖 10(a)**，基本的操作流程為：

1. 按下 Open Lua 按鍵，可讀取需執行的腳本參考如**圖 10(b)**，格式限制為 .lua，其檔案可由 Blockly 程式產生。
2. 當檔案載入後，程式下方的空白區域 (監控視窗)，會自動顯示腳本中的內容，並同時將目前執行的行數變色顯示如**圖 10(c)**，使用者可藉由監控視窗檢視目前執行命令並進行除錯。
3. 按下相關執行的按鍵執行腳本；每讀取一行 Lua 程式碼，本程式自動會與 MCCL 做對應運動控制，且藉由設計的程式變色顯示機制，可讓使用者快速了解程式運行的執行情況。

在執行腳本途中，使用者可以按下 Stop 按鍵來強制停止程式執行。依照上述的流程操作並使用 EPCIO 控制卡搭配馬達箱進行實際驗證後，結果顯示腳本執行程式可正確讀取 Lua 腳本並將運動命令輸出至控制卡，能達到實體的運動控制如**圖 10(c)**。

結論

本文展示了如何將運動控制上複雜的程式開發藉由工研院機械所開發的 MCCL 運動控制函式庫，以 Blockly 圖像式介面做優化，在對應的 EPCIO 控制卡與運動平台上做驗證。以目前開發的程式與之前 C# 介面相比，我們提供：

1. 更友善的圖像化程式開發介面。
2. 複雜程式碼積木化，可整合多個程式於積木中，達成多功能的用途。
3. 可直覺地堆疊積木的方式來設計程式，更適合新手來做學習與操作。
4. 整合程式高層與底層設計，可以自動將積木轉換成對應的程式語言與其他的資源，依照程式目的做相對應的運動控制，使用者無須再做其他的程式設計。

本程式無須使用者熟悉相對複雜的程式語言，即可以圖像化的積木做程式開發，並搭配工研院機械所的運動控制卡執行運動控制命令，可降低初學者在運動控制程式開發的門檻，實務上可加速系統開發時程與降低學習時間，提升系統開發的效率。

誌謝

感謝工研院機械與機電系統研究所 (計畫編號 N353C72100) 的支持，讓本計畫案可順利推進，故在此致上感謝之意，感謝他們做出的貢獻等。

參考文獻

[1] B. W. Kernighan and D. M. Ritchie, "*The C Programming Language*" (2nd_Edition), Prentice-Hall. 1988.

[2] A.C. Bock and U. Frank, "Low-code platform," *Business & Information Systems Engineering* 63, 733-740, 2021.

[3] "Low-code development platform market by component (platform and services), application type, deployment type (cloud and on-premises), organization size (SMEs and large enterprises), industry, and region - global forecast to 2025," Marketsandmarkets, 2022.

[4] A. Sahay, A. Indamutsa, D. D. Ruscio, and A. Pierantonio, "Supporting the understanding and comparison of low-code development platforms,"*2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 171-178, 2020.

[5] R. Waszkowski, "Low-code platform for automating business processes in manufacturing," *IFAC-PapersOnLine* 52(10), 376-381, 2019.

[6] F. Gürcan and G. Taentzer, "Using microsoft powerApps, mendix and outSystems in two development scenarios: an experience report," *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and*

Systems Companion (MODELS-C), 67-72, 2021.

[7] P. Vincent, K. Iijima, M. Driver, J. Wong, and Y. Natis, “Gartner magic quadrant for enterprise low-code application platforms,” Gartner, 2021.

[8] M. Moskal, “No-code application development on the example of logotec app studio platform,” *IAPGOS* 11(1), 54-57, 2021.

[9] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, “Scratch: programming for all,” *Commun. ACM* 52(11), 60-67, 2009.

[10] N. Fraser, “Ten things we've learned from Blockly,” *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, 49-50, 2015.

[11] Y. Luo, P. Liang, C. Wang, M. Shahin, and J. Zhan, “Characteristics and challenges of low-code development: The practitioners' perspective,” *15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2021.

[12] B. Lefort and V. Costa., “Benefits of low code development environments on large scale control systems,” *Int. Conf. on Acc. and Large Exp. Physics Control Systems (17th)*, 976-981, 2019.

[13] S.P. Krishnamoorthy and V. Kapila, “Using a visual programming environment and custom robots to learn C programming and K-12 STEM concepts,” *In Proceedings of the 6th Annual Conference on Creativity and Fabrication in Education (FabLearn '16)*, 41-48, 2016.

[14] “MCCL 運動控制函式庫”, EPCIO, 2015, https://www.epcio.com.tw/product/MCCL.aspx

[15] S. S. Yeh and P. L, Hsu, “Analysis and design of integrated control for multi-axis motion systems,” *IEEE Transactions on Control Systems Technology* 11(3), 375-382, 2003.

[16] J. U. Cho, Q. N. Le, and J. W. Jeon, “An FPGA-based multiple-axis motion control chip,” *IEEE Transactions on Industrial Electronics* 56(3), 856-870, 2009.

[17] R. Ierusalimschy, L. H. Figueiredo, and W. Filho, “Lua-an extensible extension language,” *Software: Practice & Experience* 26(6), 635-652, 1996.

[18] S. Fernandes, A. Aguiar, and A. Restivo, “A Live environment to improve the refactoring experience,” *International Conference on the Art, Science, and Engineering of Programming(6th)*, 30-37, 2022.