

# 2018 Synopsys ARC Contest

競賽題目：  
應用於溫度感測器校正之機器學習方法

參賽單位：國立交通大學  
隊伍名稱：317 戰隊  
指導老師：張錫嘉 老師/教授  
參賽隊員：劉力瑋, 宋季轅, 郭緯謙  
完成時間：2018 年 05 月 18 日

## 基本資料表

隊伍名稱	317 戰隊		學校名稱	國立交通大學	
作品主題	應用於溫度感測器校正之 機器學習方法				
項目負責人	劉力瑋		E-Mail	willvegapunk@gmail.com	
電話	Private Data		學校科系級別	電子研究所	
指導教授	張錫嘉		教授 E-Mail	hcchang@mail.nctu.edu.tw	
參賽隊員	姓名	在籍學歷 (含系級)	身分證號碼	專業	分工
	劉力瑋	博士班一 年級	Private Data	電子工程	C code 移植 硬體平台建 置
	宋季轅	大學部四 年級	Private Data	電子工程	C code 移植 硬體平台建 置
	郭緯謙	碩士班 3 年 級	Private Data	電子工程	機器學習演算法開發
隊伍簡介	本團隊成員都來自國立交通大學電子研究所 OASIS 實驗室，本實驗室研究 方向為利用演算法與電路架構針對特定功能系統進行最佳化。本次比賽擬 藉由機器學習方法偵對溫度感測晶片的感測精準度來進行優化。				
參賽項目	2018 Synopsys ARC 盃電子設計大賽				
曾獲獎紀錄	無				
研究專長	IC 設計(Verilog) 演算法模擬(Python、MATLAB)				

## 摘要

在本競賽中，我們提出了一個機率估計的機器學習模型，並應用於溫度感測電路精準度校正。我們預計使用的方法其運算複雜度適合在 Embedded ARC Starter Kit 開發平台上實現。藉此克服了晶片實踐後因雜訊干擾而影響的精準度問題，使溫度量測更為準確，期望能更廣泛的運用於各式感測器。

實作的感測端為兩組電阻式感測電路，一個用於偵測晶片外部的溫度，也就是室溫，另一個則是偵測晶片本身的溫度，這個感測電路能將感測端偵測到的電阻差值轉換成數位碼輸出。利用這兩組的溫度輸出，透過機器學習的方法，針對個別溫度的輸出訓練出高斯模型，再藉由機率估計的方法，提高晶片的精準度。校正後能夠克服外界環境所產生的干擾，在溫度量測情境中提供一個更精確的結果。

系統平台為 新思科技提供的 Embedded ARC Starter Kit(EMSK) 嵌入式開發套件，此平台將進行感測溫度的資料讀取，將蒐集的數據透過 桌上型電腦 訓練機器學習模型的參數，最後將參數與機器學習模型透過 C code 實現於 EMSK，提升溫度感測精準度，並顯示即時溫度資訊。讓 EMSK 可以運行機器學習模型，讓物聯網的溫度感測器更聰明，更準確。

實驗結果顯示，我們所提出的方法不僅能夠抗外界雜訊的干擾，也能將溫度感測器的準確度提高至原本的 200%，也就是精準度從 1 °C 到至少 0.5 °C。

## ABSTRACT

**Keywords:** Machine Learning, ARC EM Starter Kit(EMSK)

In this contest, we aim to develop a machine learning method to calibrate the thermal sensor and to avoid the interference from the environment for higher accuracy level in temperature measurement. The computation complexity of our proposed method would be suitable for porting on to the ARC Embedded Starter Kit platform.

The sensing circuit is divided into two parts. One circuit is for the external temperature as known as room temperature, while the other is for the die temperature. This sensing circuits can translate the differential resistance from the sensing- ends into digital code. By using those two thermal outputs, we will train a multivariable linear regression model to obtain higher accuracy. After calibration, we can avoid the interference and get higher accuracy in temperature measurement.

The demonstration platform includes a sensor chip that is fabricated in the process of UMC 0.18 $\mu$ m CMOS-MEMS technology and the ARC EM Starter Kit (provided by Synopsys) to achieve raw data collection , machine learning model inference and real-time temperature display. Make our IOT device more intelligent and accurate.

The measurement results show that the method is effective in approving the temperature accuracy by 200%, that is from 1 degree Celsius to 0.5 degree Celsius.

**注意事項：**

- 參賽者同意主辦單位得將其參賽作品予以公開發表、重製、公開播送、公開展示、重新編輯、出版等非商業用途之實施，且參賽者不得對於上述實施要求任何形式之報償。
- 參賽者擔保為參賽作品之著作權人，並同意主辦單位擁有該參賽作品之公開發表、重製、公開播送、公開展示、重新編輯與出版等使用於學術或推廣教育之權利。若有因該參賽作品而引起智慧財產糾紛、訴訟等，均由參賽者全權負責。
- 參賽者同意主辦單位得將其個人資料及其相關參賽作品納為「通訊大賽創意機制媒合人才資料庫」之用。

# 目 錄

摘 要	3
<b>ABSTRACT</b>	4
<b>目 錄</b>	6
第一章 方案論證	9
1.1 專案概述	9
1.2 資源評估	10
1.3 預期結果	11
1.4 專案實施評估	11
第二章 新作品難點與創新	13
2.1 作品難點分析	13
2.2 創新性分析	13
2.3 小結	14
第三章 系統結構與硬體實現	15
3.1 系統原理分析	15
3.2 系統結構	15
3.3 硬體實現	17

---

3.4 小結	18
第四章 軟體設計流程及實現	20
4.1 軟體設計流程	20
1. 讓 ARC Board 可以進行 GPIO 進行溫度資料讀取	20
2. 訓練溫度感測模型 (Train Machine Learning Model)	20
3. 將 Model 實現在 EMSK 上面	20
4.2 軟體實現	21
4.2.2 演算法二:Multiple Linear Regression	26
4.3 小結	28
第五章 系統測試與分析	29
5.1 系統測試單位	29
5.2 測試環境	29
5.2.1 驗證開發平臺	29
5.2.2 測試方案	30
5.3 測試結果	30
5.3.1 功能測試	30
5.3 結果分析	31
第六章 總 結	31
參考文獻	33





# 第一章 方案論證

## 1.1 專案概述

本專案 透過 **Synopsys ARC** 嵌入式開發平台 運行 機器學習模型 進行溫度感測電路校正與顯示，透過本團隊的方案能夠將溫度感測晶片的精度有效提升至 0.5 度 C。全部專案將分成兩個階段

第一階段；

藉由麵包版、運算放大器、Pmod 線材 實現資料讀取電路，ARC 開發平台將透過此電路讀取 溫度感測晶片的溫度資訊(精確度低)，所有的資料將透過 UART 介面顯示在 終端機上。由於建構機器學習的統計模型需要足夠的數據，因此我們會將溫度感測晶片與讀取電路放置於恆溫恆濕試驗機中，藉此獲取不同溫度的大量數據，這些數據將會是統計模型的訓練資料與驗證資料。

第二階段

利用這些不同溫度區間的訓練資料，在 PC 端使用 MATLAB 去建立機器學習模型，並訓練模型參數。

第三階段:

將機器學習的演算法透過 C 語言去實現，在 ARC EMSK 平台以 baremetal 的方式開發。當即時溫度已經能夠透過 UART 介面正確顯示於終端機上，我們將更進一步利用無線傳輸模組將溫度資訊呈現於使用者介面。

## 1.2 資源評估

資料讀取電路的部分，本團隊會利用麵包版，穩壓 IC 與 Pmod 線材實作資料讀取電路。

溫度感測晶片為 DIP32 封裝，其中晶片有 14 個輸入腳位 5 個輸出腳位，為了使 IO 訊號穩定，我們將會在晶片的 IO 利用 7404 晶片作為 BUFFER。並且讓 ARC Embedded Starter Kit 提供 3.3V 作為溫度感測晶片的電源。外部溫度感測元件的選擇，將有三種選擇: (1) 熱電偶(Thermocouple) (2) 電阻溫度感測器(RTD) (3) 溫敏電阻(Thermistor)

以下為三種溫度感測元件的特性比較表

Attribute	Thermocouple	RTD	Thermistor
Cost	Low	High	Low
Temperature Range	Very wide	Wide	Medium
Accuracy	Medium	High	Medium
Sensitivity	Low	Medium	Very high
Linearity	Fair	Good	Poor

考量到成本與溫度靈敏度的特性，溫敏電阻(Thermistor)比較適合用於本專案中的溫度感測系統的外部溫度感測元件。

溫度展示的部分有以下三種方案進行評估，在實作過程我們會選擇一種方式去執行

方案一:

溫度資訊透過 Wifi 方式傳輸到指定的網頁伺服器，透過網頁做為使用者介面顯示目前的溫度。此方案需要新思科技另外提供 無線網路模組。

方案二:

透過藍芽將溫度資訊傳輸到手機應用程式，藉由手機 app 作為使用者介面。此方案需要新思科技另外提供 藍芽模組。

### 方案三

直接透過 UART 介面 將溫度資訊直接傳輸到展示用電腦，直接利用終端機顯示目前溫度。

綜合以上評估，在 EMSK 平台 IO 腳位的部分的需求如下

1)溫度資料讀取電路的 IO 部分需要 19 PIN 預計佔用 3 組 PMOD

2)無線傳輸的部分無論是 Bluetooth 或是 Wifi 都會占用 1 組 PMOD

Bluetooth 與 Wifi 模組的部分需要 Synopsys 提供

## 1.3 預期結果

我們希望透過此方案展現，即使是運算功能有限的嵌入式開發環境，仍能夠運行機器學習演算法以提升感測器精確度，一方面提升產品的良率，並減少報廢所造成的環境負擔，另一方面，透過定期的校正可以延長產品的使用壽命。此外，藉由 ARC Embedded Starter Kit 可擴充性的特點，將溫度資訊即時準確地傳達給使用者以實現落地應用。

## 1.4 專案實施評估

在資料讀取電路的建置，需要有一個適當的硬體除錯方案，確保溫度感測晶片有符合預期的狀況下工作。在數據蒐集的階段務必確定每個溫度區間的在較大的容忍範圍內(低精度)狀況下運作。

由於在運算能力有限的嵌入式平台開發，因此在程式碼的撰寫上需要盡量

輕量化，盡量保留足夠的空間存放訓練好的機器學習參數。

無線傳輸部分，除了了解模組使用的方式以外，仍需要考量無線傳輸對於整體系統運算能力是否能夠承受這另外的負載。在成功建置無線傳輸後，仍需要另外開發對應的使用者環境介面。如果是 Wifi 傳輸，則必須撰寫網頁服務，如果是藍芽傳輸 則必須撰寫 手機應用程式。

專案時間規劃如下表所示:

開發者工具鍊的熟悉與架設	3/15
範例測試(LED GPIO) 完成資料讀取電路並開始蒐集數據	3/31
機器學習模型建置與移植	4/8
範例測試(WIFI 藍芽)	4/15
整合無線傳輸與溫度讀取平台	4/22
使用者介面開發	4/30
準備簡報與技術文件	5/15

## 第二章 新作品難點與創新

### 2.1 作品難點分析

在建置資料讀取電路時，如果我們要让溫度感測晶片有在預期的狀況下工作，那就要找出一個有效的硬體的除錯方案，目前的方法就是測試每個腳位是否正常運作，不過要花費的時間成本過高，因此，我們須找出另外的除錯方案，增加蒐集資料的效率。

因為 訓練好的機器學習參數資料量 相當龐大，在運算能力有限且儲存空間較小的開發板上，放上訓練好的機器學習參數以及程式碼後，能否順利執行，目前能以得知，需在拿到板子後方能進行評估。

\*\*\*\*\*

### 2.2 創新性分析

我們在 EMSK 上執行機器學習模型，儘管開發板的資源有限，如果開發板上能夠運行機器學習模型，那未來也可能在穿戴式裝置實行機器學習模型。

原本的溫度感測晶片精確度約  $0.7^{\circ}\text{C}$ ，在經過機器學習演算法運算後，精確度能達到  $0.5^{\circ}\text{C}$ ，讓此溫度感測晶片可以應用於更多的裝置上。

\*\*\*\*\*

---

## 2.3 小結

我們須找到更好的硬體除錯方案，讓我們可以有效地獲取數據資料，並且評估開發板的運算能力及儲存空間，使機器學習模型能在開發板上順利執行。

我們在 EMSK 開發平台上實行機器學習模型，增加開發板未來更多的可用性，並加強溫度感測晶片的精確度，使其能使用於更多應用裝置上。

---

## 第三章 系統結構與硬體實現

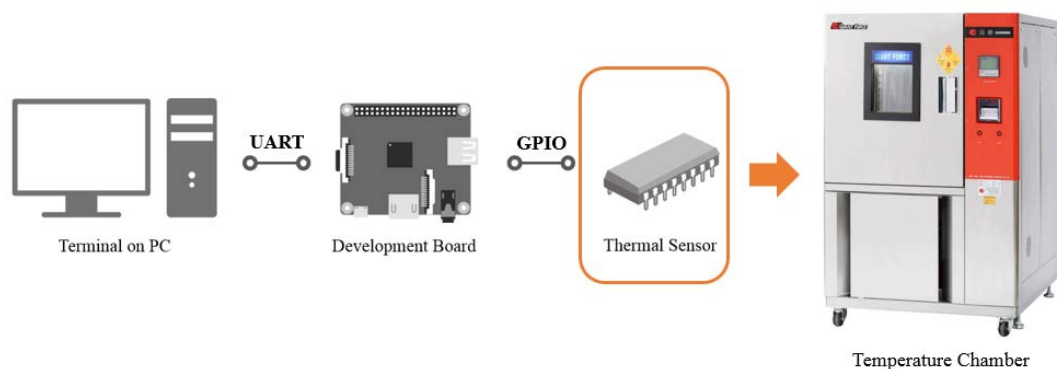
### 3.1 系統原理分析

本團隊的應用於溫度感測器校正之機器學習方法，基本原理是利用 EMSK 與 溫度感測讀取電路 將晶片在個溫度區間的讀出值(Readout Value)蒐集為一個訓練資料集，將這些數據利用不同機器學習方法進行訓練，不同模型的參數。有這些參數後，我們再將適當的機器學習模型以 C 語言去實現並移植到 EMSK 平台中。最後在使用對應的 Wifi 或是 Bluetooth 模組，將精確的溫度資訊傳輸到使用者介面。

\*\*\*\*\*

### 3.2 系統結構

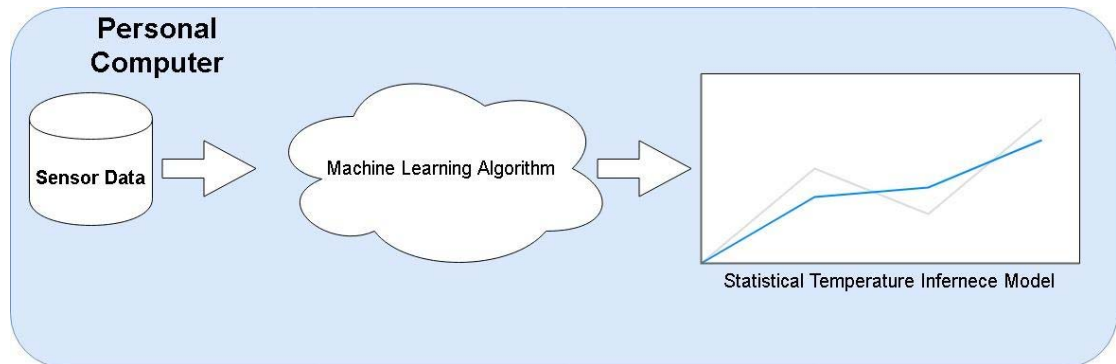
#### 第一階段



第一階段是利用 開發版、讀取電路與感測晶片 建構一個溫度感測系統並

放置於恆溫恆濕試驗箱進行溫度數據的蒐集。這些溫度數據將透過 UART 傳輸至 PC 上儲存。

## 第二階段



第二階段透過 MATLAB 與 訓練數據 基於機器學習演算法建立一個 統計溫度的推測模型，並記錄這些訓練好的參數。

機器學習訓練的 PC 環境如下:

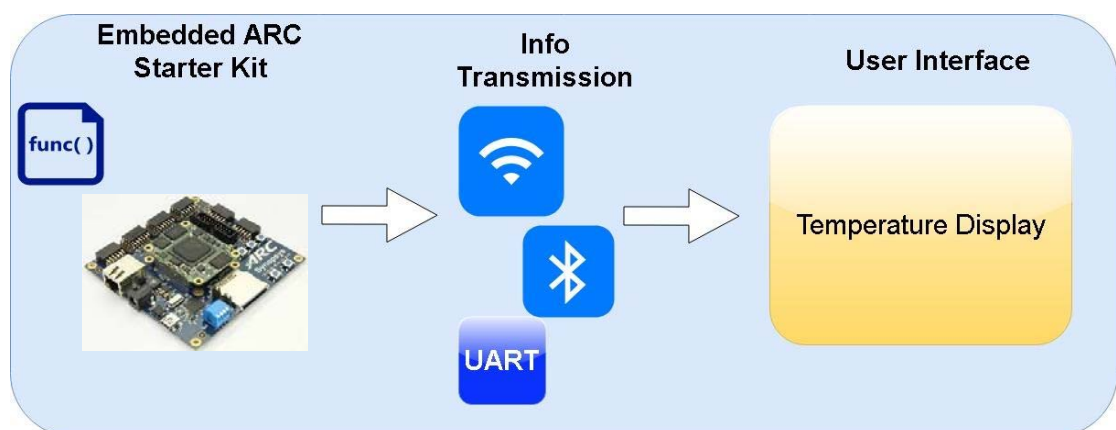
Intel Core i5-4570S CPU @ 2.90GHz (4 核心)

RAM: 12GB

Windows 7 64-bit Operating System

MATLAB2016a

## 第三階段



第三階段的硬體架構如上圖所示，我們將會依照上述所提的三個方案去實現一種，利用 EMSK 擴充 Wifi 或是 藍芽進行 資料傳輸，將溫度資訊呈現給使



用者。

### 3.3 硬體實現

第一階段感測讀取電路所需材料如下圖所示:

下圖為溫度感測晶片的腳位圖

ROUT3		1		32	ROUT4
VIN_TOUT2	#	2		31	CHIPGND
3.3IOVDD		3		30	TINOUT_SIPO_OUT
ROUT2		4		29	IOGND
1.8CHIPVDD		5		28	5 IOCLK
ROUT1		6		27	9 TINOUT_VLD
VIN_TOUT1	#	7		26	12 VIN_R2
RST		1	8	25	11 VIN_R1
VLD		3	9	24	1.8CHIPVDD
W	#	10		23	10 IN_DATA_T
CHIPGND		11		22	4 IN_DATA_R
OUT_TEX		12		21	2 VIN_R
IOGND		13		20	PISO_OUT_R
DCO_SIPO_OUT		14		19	1.8IOVDD
1.8CHIPVDD		15		18	8 DCO_IN_DATA
DCO_VLD	7	16		17	CK16

熱敏電阻: 採用 NTC 熱敏電阻器(2M)外部電阻



緩衝器(Buffer):為了確保晶片的輸出訊號穩定，我們將使用

7404 兩個 Inverter 來實現緩衝器。





利用以上材料與符合晶片規格的電阻與電容，我們將實現感測讀出電路(Sensor

Read Out Circuit)。

第三階段硬體實現所需材料:

### Wifi & Bluetooth Module

	<p>Digilent® PmodWiFi - 802.11g WiFi Interface Adds IEEE 802.11G Wi-Fi communication support for embedded applications.</p>
	<p>Bluetooth 2.1/2.0/1.2/1.0 compatible Add wireless capability with this low power, Class 2 Bluetooth radio Supports HID profile for making accessories such as pointing devices, etc. Secure communications, 128-bit encryption</p>



Synopsys ARC Embedded Starter Kit 開發整合平台

\*\*\*\*\*

## 3.4 小結

準備完上述材料之後，先進行第一階段資料讀取電路與 EMSK 系統的資料讀取功能整合，藉由 EMSK 的 UART 接面將即實溫度數據蒐集為 機器學習訓練資料集。

第三階段我們將上述的 WIFI 或 藍芽模組 與 EMSK 進行整合，透過無線方式將及時溫度資訊傳遞給使用者。

\*\*\*\*\*

## 第四章 軟體設計流程及實現

### 4.1 軟體設計流程

#### 1. 讓 ARC Board 可以進行 GPIO 進行溫度資料讀取

- 必須完成 GPIO 的 Example
- 找到 Timer Interrupt 的 Example 產生固定的 Clock Cycle
- 進行電路的連接，確定腳位圖與外部電路
- 進行電路的連接，確定腳位圖與外部電路
- 用 Pmod 的 IO，將訊號打入晶片中
- 測試感測晶片是否正常運作（數值是否隨溫度變化）
- 進行 raw data 的擷取

#### 2. 訓練溫度感測模型（Train Machine Learning Model）

- 將原始資料切割成資訊
- 資料預處理
- 隨機產生訓練資料與測試資料
- 觀察資料特性，提出適當的機器學習演算法
- 利用測試資料與測試資料驗證 機器學習演算法的準確度。

#### 3. 將 Model 實現在 EMSK 上面

- 在溫度擷取的環境中再次進行測試

\*\*\*\*\*

## 4.2 軟體實現

### GPIO:

首先，在資料讀取階段之前，我們必須能夠控制好 GPIO。由於目前 embarc\_osp (2017.12)所提供的範例中，並沒有提供 gpio 的範例。感謝 user forum support team 的建議，我們從舊版本的 embarc\_osp 的 GPIO 範例去觀察與修改，最後能夠成功的控制 PMOD 的輸入輸出。

由於 embarc 會預設將 PMOD 的 UART、I2C、SPI 設定為 enable，因此當我們需要使用特定的 PMOD 作為 GPIO 時，我們是去修改 mux.c 的內容(如下圖所示)，修改的值可以參考 mux.h。

```
set_pmod_mux(PM1_UR_UART_0 | PM1_LR_SPI_S \
| PM2_I2C_HRI \
| PM3_GPIO_AC \
| PM4_I2C_GPIO_D \
| PM5_UR_SPI_M1 | PM5_LR_GPIO_A \
| PM6_UR_SPI_M0 | PM6_LR_GPIO_A );
*/
// VEGA Config for PMOD GPIO
set_pmod_mux(PM1_UR_GPIO_C | PM1_LR_GPIO_A \
| PM2_GPIO_AC \
| PM3_GPIO_AC \
| PM4_GPIO_AC \
| PM5_UR_GPIO_C | PM5_LR_GPIO_A \
| PM6_UR_GPIO_C | PM6_LR_GPIO_A );
```

如此一來，在初始化後，所有的 PMOD 都可以由我們的 GPIO 去控制。

以下為我們程式碼中 GPIO 初始化的部分。

```
port_output = gpio_get_dev(DW_GPIO_PORT_C);
port_output -> gpio_open(0x00ffff00);
port_output -> gpio_control(GPIO_CMD_SET_BIT_DIR_OUTPUT, (void *)0x00ffff00);
port_output -> gpio_control(GPIO_CMD_DIS_BIT_INT, (void *)0x00ffff00);
```

首先，先參考 EMSK2.2 document AppendixA: Pmod Pin Configuration，去找我們所需 PMOD 對應的 GPIO Port Name (例如:DW\_GPIO\_PORT\_C)。再來用 masking 的方式 (0x00ffff00 因為我需要 4 個 PMOD 的 PIN1 到 PIN4)，DW\_GPIO\_PORT\_C 打開，並設定為 OUTPUT，為訊號輸出腳位，將訊號輸入至封裝晶片中。最後，記得必須要把 disable bit interrupt。如此一來，我們就可以藉由 gpio\_write()來控制輸出訊號。

```
port_output-> gpio_write(PATTERN_EXCEL[count]<<PMOD_OFFSET0, MASK_GPIO0);
```

[hint: 建議用 led 皆在 PMOD output 來 debug]

## Timer Interrupt:

由於我們需要在特定的 frequency 範圍內，將訊號輸出至封裝晶片中，因此我們必須用 Timer Interrupt 的方式，產生固定 Clock Cycle 的訊號。

```
int_disable(INTNO_TIMER0);
timer_stop(INTNO_TIMER0); // stop timer for setting handler
int_handler_install(INTNO_TIMER0, timer0_isr);
int_enable(INTNO_TIMER0);
timer_start(TIMER_0, TIMER_CTRL_IE, BOARD_CPU_CLOCK/1000);
```

int\_handler\_install 的部分要將 user defined 的 function name (timer0\_isr)，置入 int\_handler 中。因此每當 interrupt 發生時，就會去呼叫 timer0\_isr 這個 function。

Enable 之後，最後設定 TIMER\_CTRL\_IE (如此一來，當 count 到 limit value 就會去觸發 interrupt，limit value 為 BOARD\_CPU\_CLOCK/1000)。因為 emsk 2.2 arcml1d 的預設 CPU Clock Rate 為 20 MHz，因此 Interrupt Cycle 相當於 1 ms。

## Interrupt Handler Function:

```
port_output-> gpio_write(PATTERN_EXCEL[count]<<PMOD_OFFSET0, MASK_GPIO0);
```

我們每次的 interrupt 會把對應的 rom code 輸出到 gpio。我們可以藉由 count variable 去了解在哪些特定的 interrupt cycle 要做甚麼事情。(例如讀取封裝晶片

的訊號，或是將目前讀取的數值透過 UART 輸出到 Terminal )。

讀取 gpio 訊號，並儲存於 variable 中。

```
// read data bit in J4[10:9]
port_input->gpio_read(&bit0,0x40000000) ;
port_input->gpio_read(&bit1,0x80000000) ;
bit0 = bit0 >> 30;
bit1 = bit1 >> 31;

positive=count%2;          //      positive=clk & 0x0001;

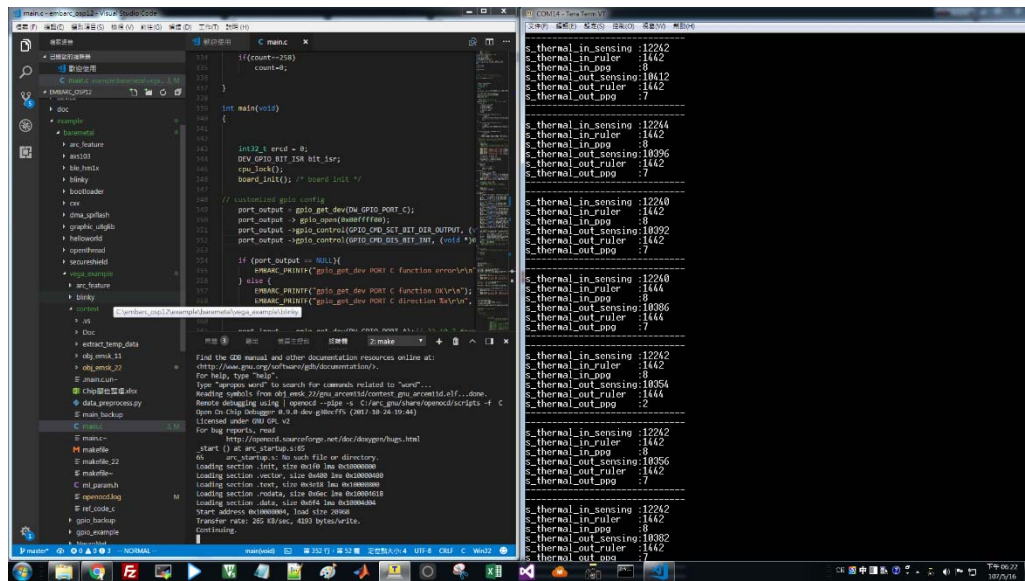
if(count==0){
    s_thermal_in_sensing=0;
    s_thermal_in_ruler=0;
    s_thermal_in_ppg=0;
    s_thermal_out_sensing=0;
    s_thermal_out_ruler=0;
    s_thermal_out_ppg=0;
    s_in=0;
    s_ex=0;
}
if(count>=134 && count<=167 && positive==1){
    if(count==135)
        idx=0;
    s_thermal_in_sensing=s_thermal_in_sensing|(bit0<<(17-idx));
    s_thermal_out_sensing=s_thermal_out_sensing|(bit1<<(17-idx));
    idx++;
}
}
```

輸出 internal 與 external raw data 到 terminal output

```
if(count==237){
    //Original
    EMBARC_PRINTF("-----\n");
    EMBARC_PRINTF("s_thermal_in_sensing:  %5d\n",s_thermal_in_sensing);
    EMBARC_PRINTF("s_thermal_out_sensing:  %5d\n",s_thermal_out_sensing);
}
```

以下圖片，為在 terminal 輸出的狀況





```

// C: main.c
111 // (count=256)
112 count;
113
114 int main(void)
115 {
116     // Initialize GPIO
117     int32_t errd = 0;
118     DEV_GPIO_BIT_ISR bit_isr;
119     gpio_init(&bit_isr);
120     board_init(); // board init
121
122     // customized gpio config
123     port_output = gpio_get_dev(DM_GPIO_PORT_C);
124     port_output -> gpio_set_dir(DIR_OUT);
125     port_output -> gpio_set_dir(DIR_OUT);
126     port_output -> gpio_set_dir(DIR_OUT);
127     if (port_output == NULL) {
128         fprintf(stderr, "gpio_get_dev PORT C function error\n");
129         return -1;
130     }
131     fprintf(stderr, "gpio_get_dev PORT C direction %d\n",
132             port_output->gpio_get_dir(DIR_OUT));
133
134     // Find the GPIO manual and other documentation resources online at:
135     // http://www.gnu.org/software/gnu/documentation/
136     // For help, type "help"
137     type "gpio_get_dev" to search for commands related to "gpio"...
138     Loading section: from objdump: /usr/bin/objdump: gpio_get_dev:elf...done.
139     Remote debugging using | opened -> /usr/bin/objdump: /usr/bin/objdump: scripts -f C
140     Open on chip debugger: 8.0.0 dev: 8.0.0 (2017.10.24.10.04)
141     Licensed under GNU GPL v3
142     For bug reports, read
143     http://sourceware.org/gdb/bug-reports.html
144     start() at arc_start:0x105
145     arc_start:0x105: No such file or directory.
146     Loading section: vector, size 0x100000000
147     Loading section: .text, size 0x100000000
148     Loading section: .rodata, size 0x100000000
149     Loading section: .data, size 0x100000000
150     Start address 0x10000000, load size 20000
151     Transfer rate: 205 KByte/s, 400 bytes/write.
152     Continuing.
    
```

```

s_thermal_in_sensing: 12262
s_thermal_in_ruler: 1442
s_thermal_in_ppg: 8
s_thermal_out_sensing: 10412
s_thermal_out_ruler: 1442
s_thermal_out_ppg: 7

s_thermal_in_sensing: 12260
s_thermal_in_ruler: 1442
s_thermal_in_ppg: 8
s_thermal_out_sensing: 10392
s_thermal_out_ruler: 1442
s_thermal_out_ppg: 7

s_thermal_in_sensing: 12260
s_thermal_in_ruler: 1444
s_thermal_in_ppg: 8
s_thermal_out_sensing: 10386
s_thermal_out_ruler: 1444
s_thermal_out_ppg: 7

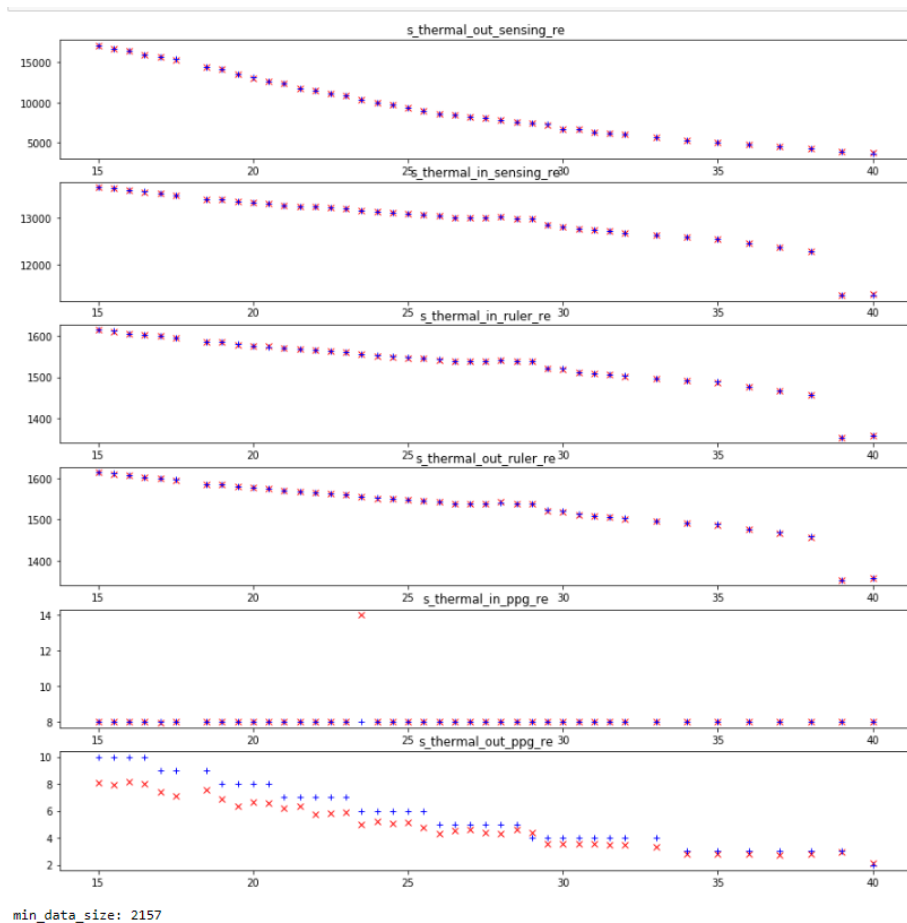
s_thermal_in_sensing: 12262
s_thermal_in_ruler: 1442
s_thermal_in_ppg: 8
s_thermal_out_sensing: 10354
s_thermal_out_ruler: 1444
s_thermal_out_ppg: 7

s_thermal_in_sensing: 12262
s_thermal_in_ruler: 1442
s_thermal_in_ppg: 8
s_thermal_out_sensing: 10356
s_thermal_out_ruler: 1442
s_thermal_out_ppg: 7

s_thermal_in_sensing: 12262
s_thermal_in_ruler: 1442
s_thermal_in_ppg: 8
s_thermal_out_sensing: 10382
s_thermal_out_ruler: 1442
s_thermal_out_ppg: 7
    
```

擷取下來的資料，利用 Regular Expression 將所需要的資料爬出來，再用 SciPy 套件 Matplot 去做資料視覺化。

原始資料分析(Raw Data Analysis)

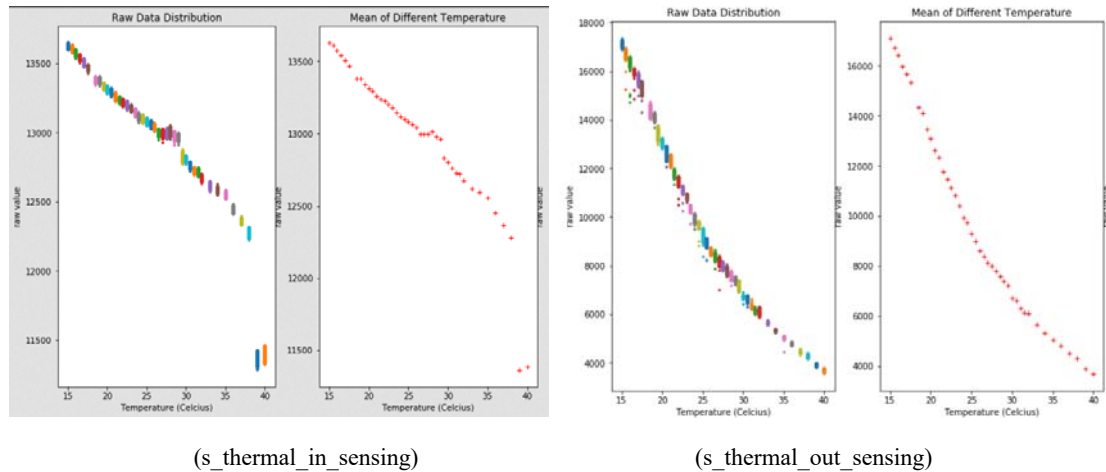




藍色:median 紅色:mean value

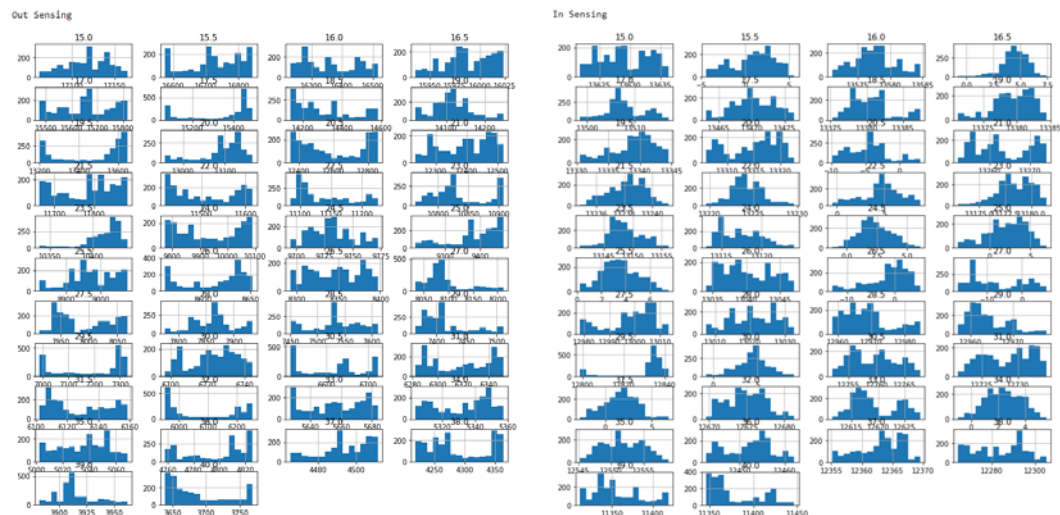
由上圖可知，s\_thermal\_out\_sensing 與 s\_thermal\_in\_sensing 與溫度的相關性比較大。因此，我們將會用這兩種資料來建立機器學習模型。

接下來針對這兩種資料特性去做更深入的分析，



由上圖可以看出，右邊原始資料比較線性，比較有機會用線性回歸的方式來建模。左邊的資料，比較適合用來輔助。

另外，我們可以觀察每個溫度內的資料分布狀況，如下圖 histogram 所示。

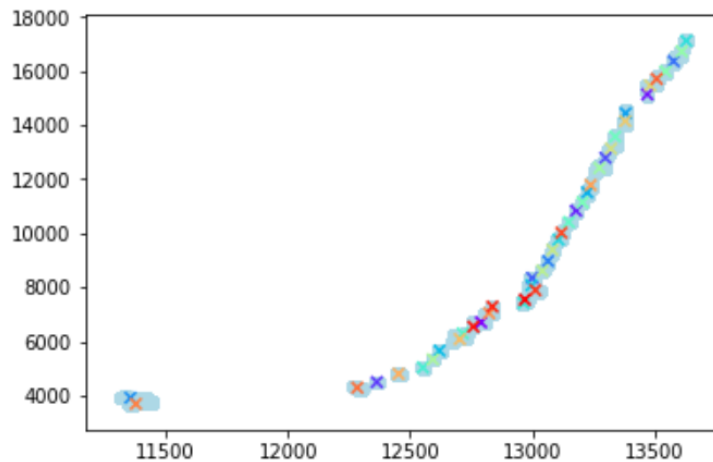


由上圖可以知道，每個溫度區間的資料分布並非都成高斯分布，為了讓每次的資料分布均勻，因此我們針對每筆資料做 moving average，這樣一來處理過的資料可以減少因為突然出現雜訊出現的影響。

因此，我們打算用以下兩種演算法來建模 - K-Means Algorithm 與 Multiple Linear Regression。

#### 4.2.1 演算法一:K-mean Clustering

k-平均演算法，源於訊號處理中的一種向量量化方法，現在則更多地作為一種聚類分析方法流行於資料探勘領域。k-平均聚類的目的是：把  $n$  個點（可以是樣本的一次觀察或一個例項）劃分到  $k$  個聚類中，使得每個點都屬於離他最近的均值（此即 clustering center）對應的 clustering。



(x: in\_sensing y:out\_sensing)

上圖 x 代表經過 K-Mean Clustering 計算出的 Clustering Center 位置，從上圖可知，有許多的 clustering center 非常接近，因此在 accuracy 的表現的部分只有 25% 誤差落於  $\pm 0.5$  度 C 之間。代表有許多溫度區間會互相重和，因此 clustering 並不適合用來處理這類型的溫度資料。

\*\*\*\*\*  
\*\*\*\*\*

#### 4.2.2 演算法二:Multiple Linear Regression

Multiple Linear Regression 主要是用多變量分析，利用資料相關性建立回歸模型。由剛剛的原始資料分析，我們可以得知 out\_sensing 比較線性，因此我們可以用相對 in\_sensing 較高 dimension 去擬和溫度分布曲線，而 in\_sensing 資料利用它跟 out\_sensing 溫度相關性進行輔助。

在建立模型之前，為了讓 in\_sensing data 與 out\_sensing data 避免因為變異範圍落差，產生對模型輸出有不同的變異，我們會先將兩筆訓練資料進行 資料正規劃(Normalization)。正規化之後，我們將資料切割成訓練資料(training set)與驗

證資料(testing set)。接下來，就可以開始建立我們的回歸模型。

我們開始用 Multiple Linear Regression 的方式去建立模型，如以下所示。

$$\beta_0 out^3 + \beta_1 out^2 + \beta_2 out^1 + \beta_3 in^2 + \beta_4 in^1 + \beta_5 = \text{Estimated Temperature}$$

我們利用 minimum mean square error 的方式去訓練我們所需要的 weight

$$L = \sum_{i=1}^n \varepsilon_i^2 = \varepsilon' \varepsilon = (y - X\beta)'(y - X\beta)$$

$$\frac{\partial L}{\partial \beta} = 0 \Rightarrow X'X \hat{\beta} = X'y$$

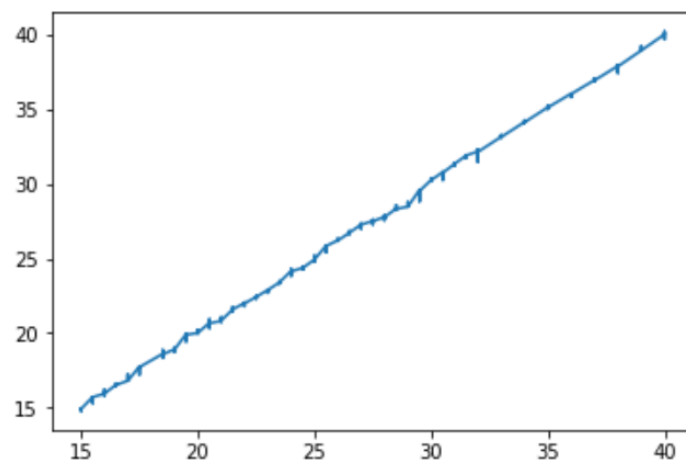
L: square error

y: true temperature

$\beta$ : weight

X:input data matrix

建立完模型之後，以下圖片為在驗證資料及的狀況



Accuracy: 0.974880952381

(x: Ground Truth Temperature y: Predict Temperature)

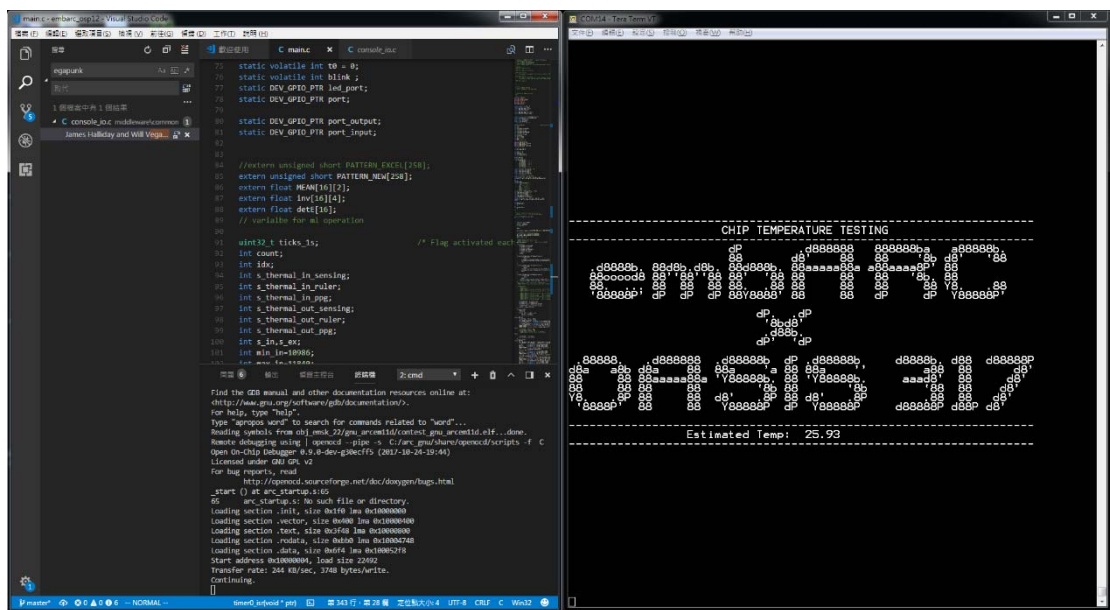
因此在測試資料集有 97.48%的 accuracy 落在±0.5 度 C

\*\*\*\*\*

\*\*\*\*\*

由以上 2 種演算法中，使用 Multilinear Regression，對於硬體的需求最少且準確度比較高，因此我們會將此 Inference Model 用 C-code 去實現，置入 ARC emsk 平台運行。

以下為在 EMSK 平台運行結果的圖片



### 4.3 小結

我們從最基本的建立資料擷取的環境的建置 GPIO Timer Interrupt、原始資料擷取、原始資料分析，經過機器學習演算法的分析與選擇，最後我們選擇使用 Multiple Linear Regression Method 的 model，將 Inference Model 在 EMSK 平台的實現。

\*\*\*\*\*  
\*\*\*\*\*

## 第五章 系統測試與分析

### 5.1 系統測試單位

系統測試單位分為以下幾點:

1. GPIO 單元測試
2. Timer Interrupt 測試
3. EMSK 與封裝晶片 電路平台 溫度擷取功能測試
4. 機器學習演算法測試
5. EMSK 平台 在 恆溫恆濕機 輸出結果測試

\*\*\*\*\*  
\*\*\*\*\*

### 5.2 測試環境

\*\*\*\*\*

#### 5.2.1 驗證開發平臺

- GPIO 與 Timer Interrupt 測試 :主要使用 LED
- 溫度相關測試環境主要仰賴於恆溫恆濕機
- 機器學習演算法測試則藉由 PC 或是筆電進行資料分析與演算法驗證

\*\*\*\*\*

### 5.2.2 測試方案

- GPIO 與 Timer Interrupt 測試方式，主要藉由 LED 接在預備輸出的 PMOD 外，確認輸出的時序是否符合預期的結果。
- EMSK 與 感測溫度晶片測試平台的功能測試，主要藉由示波器與邏輯分析儀，檢測封裝晶片的輸入輸出是否符合預期，另外藉由恆溫恆濕機，測試我們的感測晶片是否隨著溫度變化。
- 機器學習算法測試，主要利用資料切割出來的 testing set 針對不同 Machine Learning Model 撰寫 accuracy 的 function 去評估在 $\pm 0.5$  度 C 的正確率。

\*\*\*\*\*

## 5.3 測試結果

### 5.3.1 功能測試

- EMSK 平台的 GPIO 與 Timer Interrupt 確實能夠運作
- EMSK 與封裝晶片電路平台 顯示的 raw data 確實會隨溫度變化
- 機器學習演算法正確率( $\pm 0.5$  度 C):  
K-Means :  
Multiple Linear Regression :
- EMSK 平台 在 恆溫恆濕機 輸出結果測試  
顯示結果絕大部分都落於 $\pm 0.5$  度 C 的區間內

\*\*\*\*\*

## 5.3 結果分析

透過以上測試的方式，確保每個單元功能皆能運作，使系統感測溫度在測試設備中具有 $\pm 0.5$  度 C 的精準度。

\*\*\*\*\*

## 第六章 總 結

這個作品的目的是克服溫度感測晶片因雜訊干擾而產生的精確度問題。原先的精準度大約為 1 度 C，我們希望感測晶片在校正之後的精確度能達到 0.5 度 C。

我們使用了原先的感測晶片電路，從擷取出的初始資料分析資料特性，取兩個輸出來實現機器學習演算法以達成我們的目標。我們使用了兩種演算法—K-mean clustering 以及 multiple linear regression，來訓練我們的資料並產生溫度判斷模型。最終，我們選擇了 multiple linear regression 這個方法來實現我們的溫度感測晶片的精確度校正。他有兩個特點。首先，multiple linear regression 適合用來訓練線性的多變量資料。再者，他對於硬體的需求最少且精確度比較高。

有著以上的特點，multiple linear regression 模型可以將精確度由原先的 1 度 C，提升至 0.5 度 C。

\*\*\*\*\*





## 參考文獻

embARC 官方文件

[http://embarc.org/embarc\\_osp/doc/embARC\\_Document/html/index.html](http://embarc.org/embarc_osp/doc/embARC_Document/html/index.html)

embARC OSP Github

[https://github.com/foss-for-synopsys-dwc-arc-processors/embarc\\_osp](https://github.com/foss-for-synopsys-dwc-arc-processors/embarc_osp)

numpy tutorial

<https://www.numpy.org/devdocs/user/quickstart.html>

pandas documentation

<https://pandas.pydata.org/pandas-docs/stable/>

scikit learn tutorial

<http://scikit-learn.org/stable/tutorial/index.html>

embARC User Forum

<https://forums.embarc.org/>

\*\*\*\*\*