

111 學年度上學期
微處理機實習期末報告

題目：萬用遙控器

組員：

資訊三乙 陳俊瑋

資訊三乙 吳承翰

111 年 12 月 23 日

目錄

專題簡介與介紹.....	1
專題目標簡介.....	1
測試環境	1
前置背景知識.....	2
關於遙控器的紅外線發訊原理.....	2
實驗流程	3
實驗一：運用 PWM_IRDA sample code 獲得紅外線發射的訊號.....	3
實驗二：運用 UART_TX, UART_RX 接收紅外線訊號.....	5
針對訊號的分析，關於 NEC 協定.....	7
實驗三：運用電位變化的 upper edge 和 falling 得到訊號時間差.....	9
實驗四：實驗三改進，藉由修正訊號正確性，發送邏輯 0 和邏輯 1，並且修正誤差.....	13
實驗五：通過 NEC 推導出訊號，並且直接通過 NEC sample code 中 SendNEC 發送訊號.....	14
專題總結	16

專題簡介與介紹

本次專題題目為萬用遙控器，我們將會基於 NUC 140 上另外外掛兩個外接模組，分別為 IR transmitter 和 IR receiver，我們將使用 IR receiver 讀取要學習的訊號，並且將該訊號由 IR transmitter 發送，我們可以紀錄多組訊號，並且使用者可以通過 keypad 和 LCD 選擇先前學習過的設備訊號，並對該設備實現控制，而這也是萬用遙控器所希望達成的目標。

專題目標簡介

在萬用遙控器專案中，我們將實現以下功能：

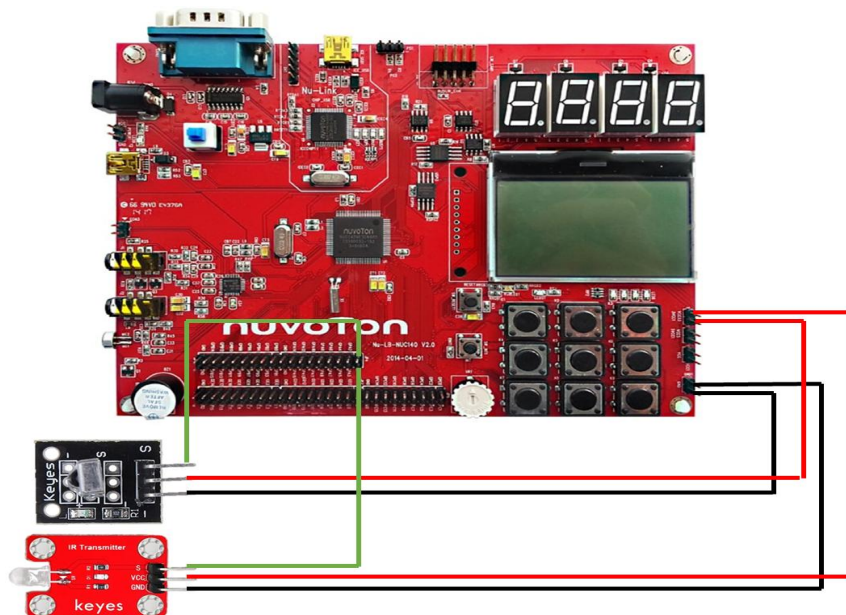
- 選單功能：LCD 上面將顯示我們先前紀錄過的設備，並且可以通過 keypad 選取設備。
- 接收訊號功能：當 IR receiver 接收到訊號，將會在 LCD 上跳出訊息，詢問使用者是否紀錄該訊號，如果選擇紀錄，該訊號將會被儲存並且顯示在 LCD 上的選單。
- 發送訊號功能：當我們通過 keypad 以及 LCD 選擇特定設備後，我們將可以通過 keypad 選擇發送該設備對應到的訊號，實現對於該設備的控制。

測試環境

NuvoTon NU_LB_002 Rev 2.1

Keys IR Transmitter Sensor

Keys Infrared Sensor Receiver Module Ky-022



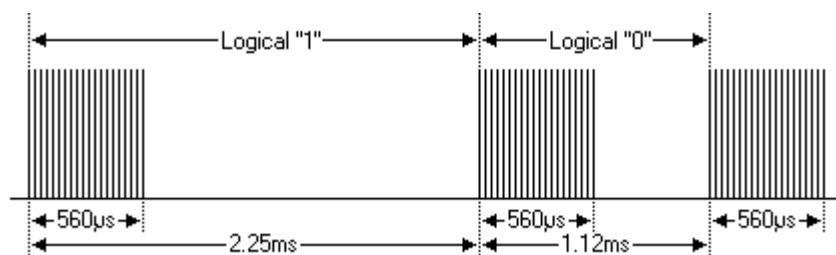
圖(一) NUC 140 學習開發版與 IR receiver 和 transmitter 的線路圖

前置背景知識

關於遙控器的紅外線發訊原理

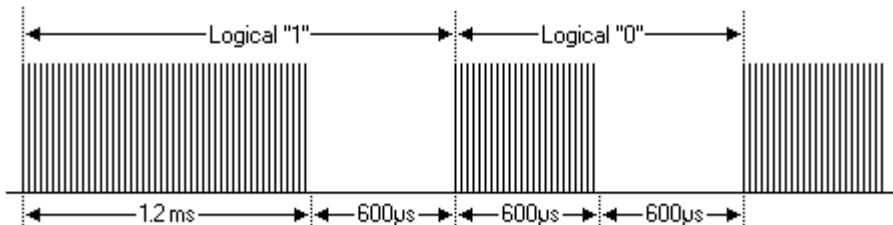
對於紅外線發訊，訊息的本質是一串 0 和 1 交織的訊號包，不同的遙控器會有不同的 0 和 1 出現的時間以及訊號長短，這一些會被紅外線發通訊協定所規範，依據廠家的不同，會有不同的通訊協定，常見的通訊協定為 NEC, SONY 等等，而我們持有的三把遙控器，其中兩把為 NEC 協定，另外一把則為私有協定。

以下為 NEC 協定的訊號圖（邏輯 0 與邏輯 1 訊號圖）



圖(二) NEC 邏輯 0 與邏輯 1 訊號示意圖

以下為 SONY 協定的訊號圖（邏輯 0 與邏輯 1 訊號圖）



圖(三) SONY 邏輯 0 與邏輯 1 訊號示意圖

從訊號圖中我們可以快速地了解到，邏輯 0 和邏輯 1 高電位和低電位之間的時間差是不同的，而在 sample code 中，是針對於 NEC 協定進行設計，因此判斷 sample code 會在解析 SONY 等其他協定出現無法正確解析訊號的錯誤，因為邏輯 0 和邏輯 1 的構造方式不同，而為了實現能夠對應到多種協議的萬用遙控器，解析多種協議的訊號，我們將會記錄高電位和低電位之間的時間差，藉此判斷協定已達成萬用遙控器，針對不同協定個別進行解析。

實驗流程

目前我們持有有關於紅外線訊號處理的 sample code 有三個，分別為 PWM_IrDA_NEC, UART_TX, UART_RX。

實驗一：運用 PWM_IRDA sample code 獲得紅外線發射的訊號

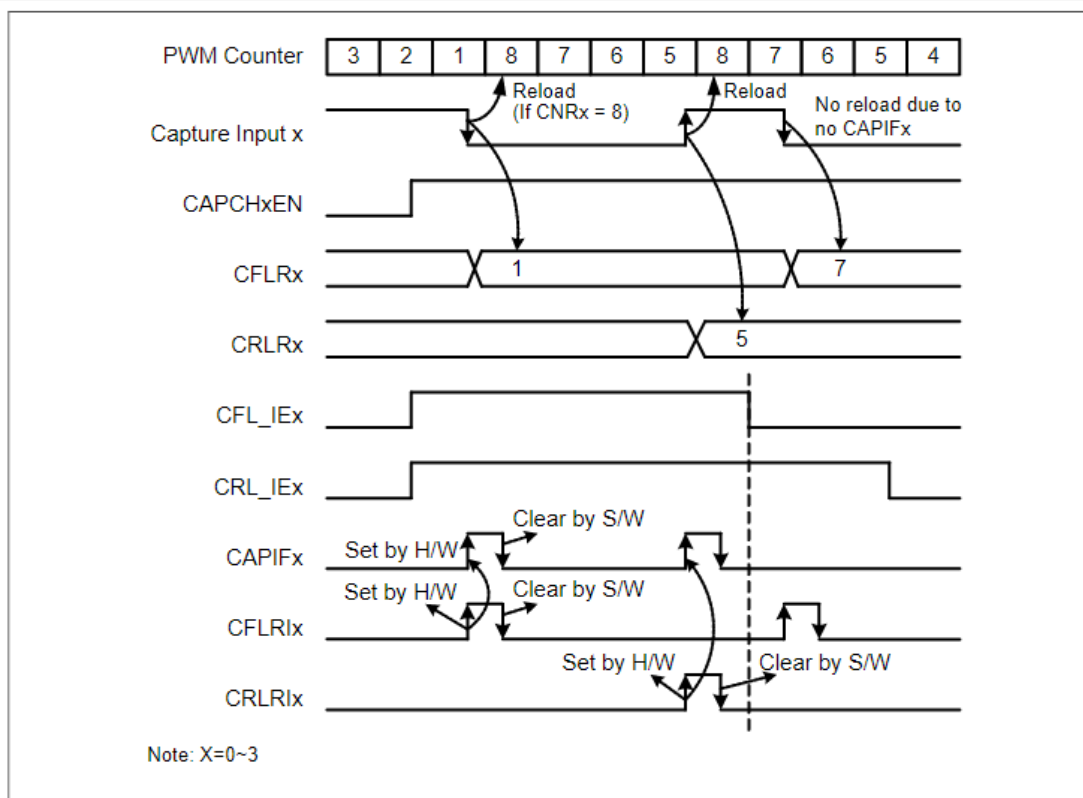
預期結果以及目標：成功將紅外線發射的訊號儲存到一個 buffer 中。

PWM_IRDA 我們的發現為在接收到訊號時，並沒有進入到 IRQ，通過 debugger 追蹤巨集定義後，發現有 IRQ 定義的錯誤，UART0 對應到的 IRQ 應該是 PWMA 的 IRQ，但是在 sample code 中卻是 PWM0 的 IRQ，目前暫且推測是 sample code 的問題，經過修正後成功在接收訊號之後，進入到 IRQ。

而在進入到 IRQ 後，我們查看 PWM_IRQ 的程式碼，理論上在進入到 IRQ 後，會由 TDR_tmp 陣列接收紅外線的發送訊號，但實際上卻沒有如預期發生。

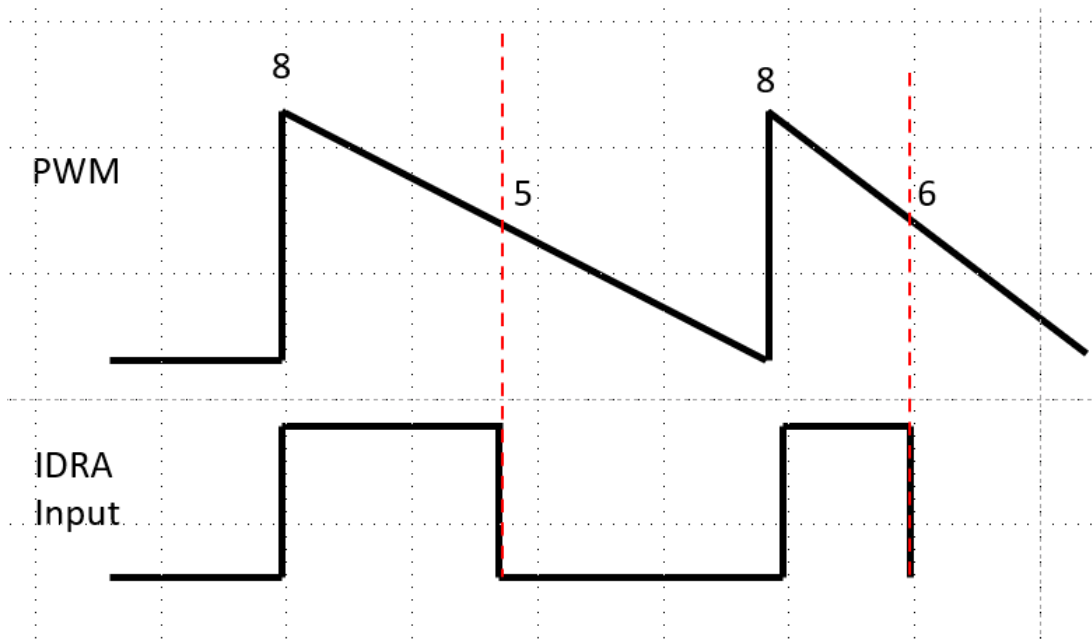
以下為 PWM_IRQ 部分程式碼

```
TDR_tmp = MaxValue - PWM_GET_CAPTURE_FALLING_DATA(PWM0, PWM_CH1);
```



圖(三) PWM Counter 與訊號解析的關係

以下為 PWM 接收輸入的示意圖，可以看到我們有一個 counter，從 8 開始向下數，當我們捕捉到信號時，重製該 counter。



圖(四) PWM 與紅外線訊號之間的解析關係

我們在訊號 upper edge 時進入到中斷，進入到中斷時重製 counter，並且開始往下數，當收到下一個 falling edge，記錄從 8 到該 counter 的時間差，藉此我們可以得到一個電位完整變化的時間長，由此推測邏輯 0 和邏輯 1，而下一個訊號將會在下一個 upper edge 進入到中斷，重製 counter 到 8，接著繼續。

以上為 PWM 接收紅外線訊號解析的邏輯，但是並沒有成功記錄到 TDR_tmp。

實驗一小結：使用 PWM_IRDA sample code 沒有成功將紅外線訊號儲存到 TDR_tmp 中。

實驗二：運用 UART_TX, UART_RX 接收紅外線訊號

預期結果以及目標：成功收到紅外線訊號，並且得到紅外線訊號的數值。

由於 PWM 失敗，因此我們決定嘗試 UART_TX, UART_RX，我們成功接收到訊息，但是接收到的訊息全部都是 255，以下為猜想

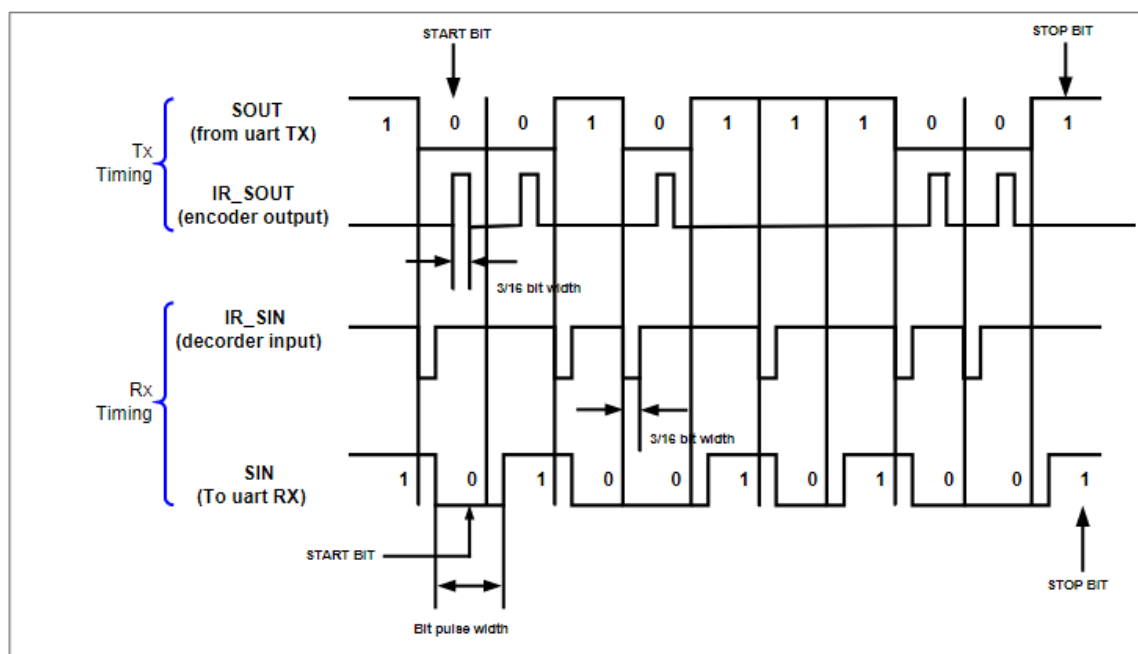
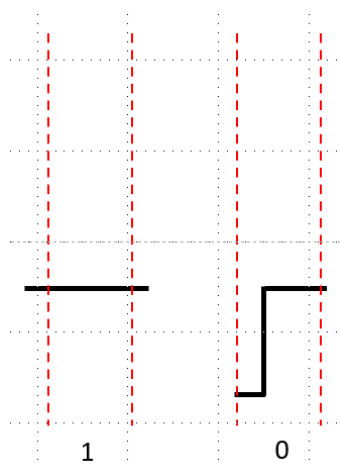


Figure 5-73 IrDA TX/RX Timing Diagram

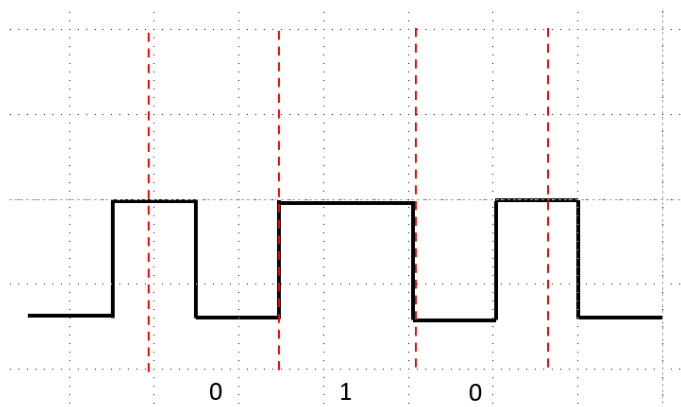
圖(五) IrDA TX/RX 與 Bit plus width 之間的關係以及解析出的數值

上面為 IrDA 的解析邏輯，我們可以看一下上面解析的邏輯，我們可以看到有許多條線將訊號分割，而只要一個分割的單位中出現 0，就表示為 0，反之則為 1，以下範例：

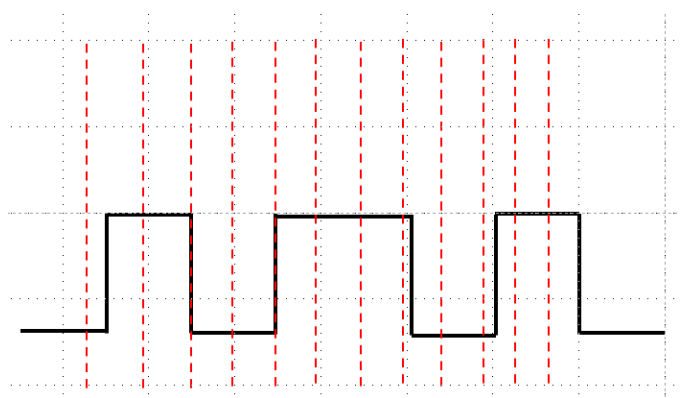


圖(六) 電位變化與邏輯 0 和邏輯 1 之間的關係

一個分割我們使用一個 frame 進行稱呼，假設目前 baudrate 為 x ，則下面大概是 frame 分割的情況

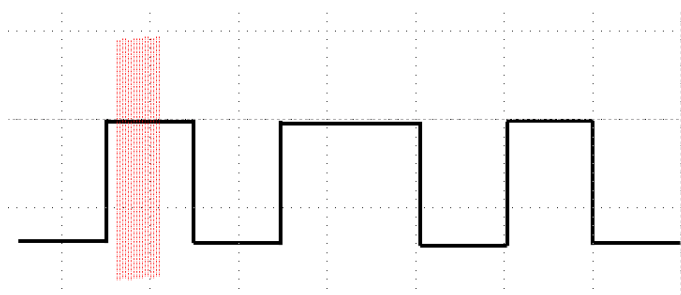


圖(七) 在固定 baudrate 為 x 時的訊號解析示意圖



圖(八) 將 baudrate 提升到 $5x$ 時的訊號解析示意圖

而我們嘗試解釋讀取到全部都是 1 的情況，假設讀取的 buffer 長度為 16，而我們的 baudrate 設置極大，則可能會出現以下情況

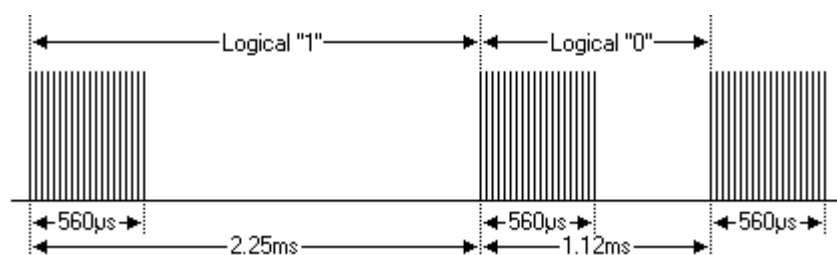


圖(九) 當將 baudrate 設置到一個極大值時，所產生的訊號解析示意圖

我們可以發現每一個 frame 解析出來都是 1，而假設 buffer 長度有限，則就會發生我們讀取到都是 1 的情況，反之，假設我們 baudrate 設置超級小，則我們應該要得到 0，但事實上並非如此。

由於我們無法成功解析，且上面這樣的解析方式會有問題，無法正確讀取紅外線訊號。

我們嘗試去閱讀紅外線訊號的相關協議，發現邏輯 0 和邏輯 1 的長度不同，如下圖所示



圖(十) NEC 邏輯 0 和邏輯 1 的訊號示意圖

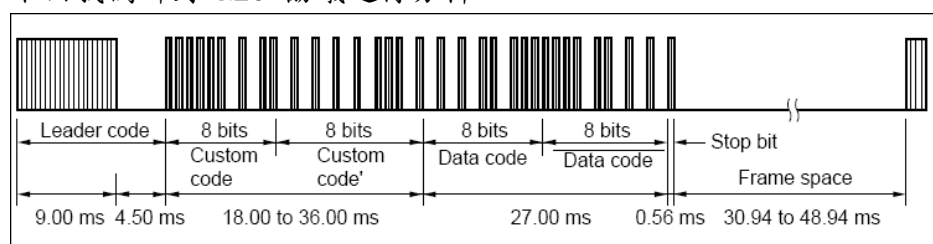
不同協定中，邏輯 0 和邏輯 1 的 frame 長度並不一定相同，如 sony, Philip, RC6 或是我們電風扇所使用的 NEC 協定中，邏輯 0 和邏輯 1 的長度也是不同的，因此使用 UART 上面這樣的策略是有問題的。

針對訊號的分析，關於 NEC 協定

IrDA 為紅外線數據協會，建立了統一的紅外線通訊標準，發表的第一版規範為 IrDA 1.0，而紅外線通訊技術包含了許多規格，諸如 IrPHY, IrLAP 等等

IrPHY 為紅外線協議中物理層協議的部分，制定了傳輸距離，傳輸速率等等，而對於傳輸速率，分成 SIR, MIR, FIR 依序為低速率，中速率，高速率，在我們的實現中，我們使用的為 SIR，低速率紅外線傳輸

下面我們針對 NEC 協議進行分析



圖(十一) NEC 協議下的封包示意圖

從上面這一張圖，我們知道在開頭的地方會出現 9.00 ms 的高電位訊號以及 4.5 ms 的低電位訊號，在 NEC 協定中，這樣表示一個紅外線訊號的開頭，接下來的內容便是紅外線訊號發送的訊息內容，這邊值得注意的是，由於 IR Receiver 接收到高電位訊號時會輸出低電位，因此實際上接收到的波型會跟圖片中的相反。

接下來會有 8 bit 的 data frame，作用為用來辨識裝置，可以將這個 data frame 稱為 address，接著後面 8 bit 的 data frame 為反向的 address，用於和前面的 data frame 進行比較，目的是為了進行錯誤校驗。

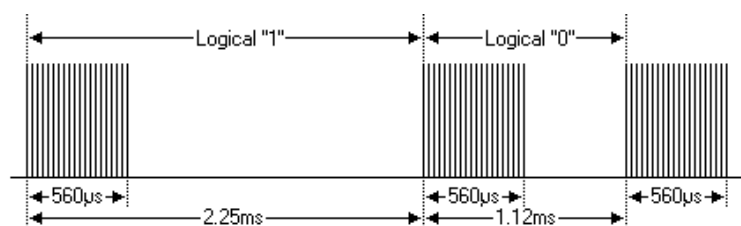
接下來為實際控制指令的 data frame，通過改變這個 data frame 我們可以控制目標裝置的行為，如開機，或是調整溫度等等，而後面接下來 8 bit 的 data frame 為前面 data frame 的反向，目的是為了進行錯誤校驗。

最後會有一個 stop bit，表示資料的結尾（概念上為 C 的 `\0`），而多筆資料之間，會有一個 Frame space，表示兩個訊號之間的間隔。

對於訊號的間隔，有兩種情況，一種為按了兩次按鈕，發送了兩次訊號，而在這兩次訊號之間會存在 Frame space。

另外一種情況為一筆長訊號，假設一個訊號發送稱為 Data，而發送訊號可能會變成兩筆 subData，我們以 subData1 和 subData2 進行表示，在 subData1 和 subData2 之間會有一個 Frame space。

回顧 NEC 的邏輯 0 和邏輯 1 的表示法



圖(十二) NEC 邏輯 0 和邏輯 1 的訊號示意圖

上面為 NEC 協定中 0 和 1 的表示法，可以看到以 0 來說，總共有 560 µs，而空白的區域為 1690 µs，所以，只要我們收到一筆具有 560 µs 高電位的訊號，且具有 1690 µs 的低電位，就可以判斷這是 1 的訊號。

實驗二小節：由於 baudrate 解析訊號的 frame 長度固定，因此無法正確的解析 NEC 等通訊協議的訊號。

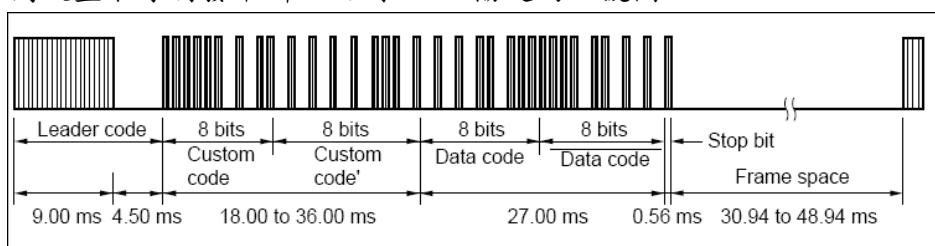
實驗三：運用電位變化的 upper edge 和 falling 得到訊號時間差

預期結果：得到訊號時間差以及電壓差，將這個訊號儲存並且直接發送。

由於我們無法通過固定的 frame 抓取訊號，因此這邊我們有了一個想法，我們通過捕捉 upper edge 和 falling 的相對時間差，我們就可以藉由時間的變化，得到一個 frame 的時間長度，以及高低電位變化，得到這一些資訊後，我們就能夠直接構造一個訊號，理論上我們構造了一模一樣的訊號並且發送，我們就能夠達到收到遙控器訊號，並且經過學習版，經過紅外線發送器發送相同的訊號達到控制的目的了。但是我們失敗了。

以下為我們使用 GPIO 的實作細節：

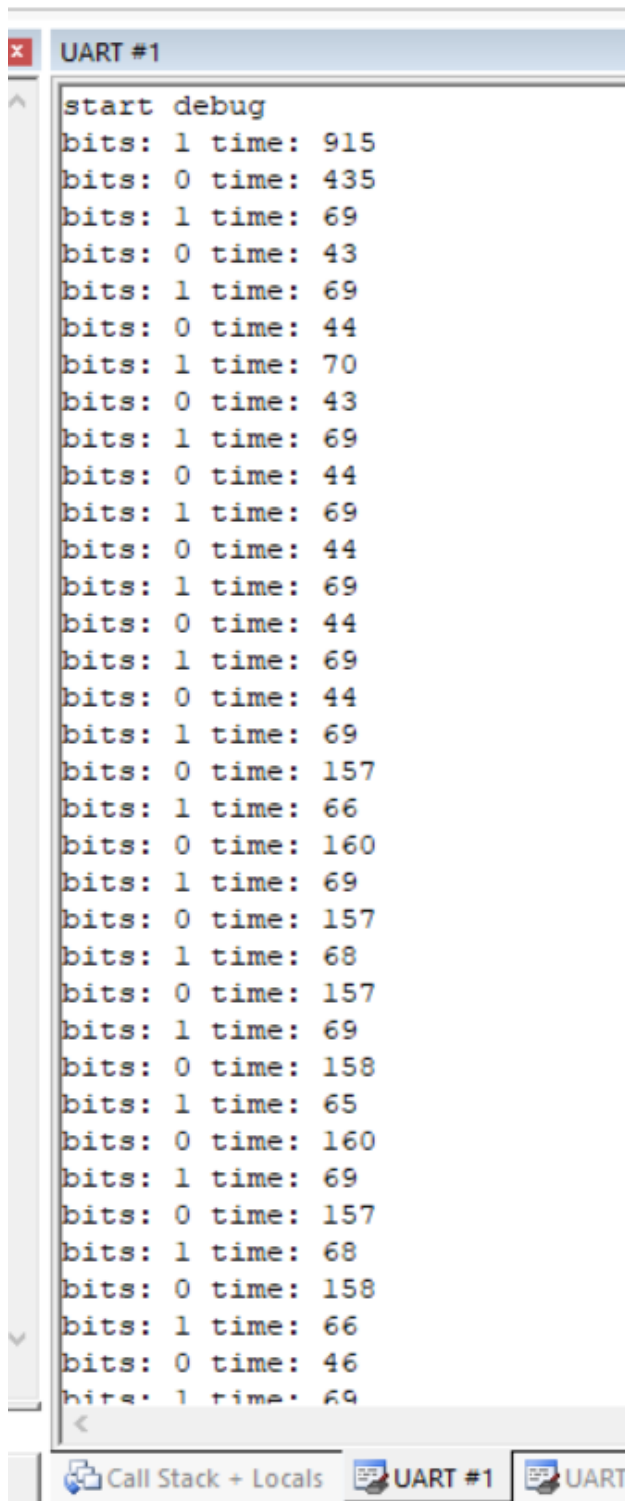
我們通過 GPIO 配合 timer 進行讀取，GPIO 可以自行設置觸發中斷的條件，這邊我們設置下緣觸發中斷，回到 NEC 協定的訊號圖



圖(十三) NEC 協議下的封包示意圖

如果我們在 GPIO 中斷時將一 counter 加一，我們可以預期將對觸發總共 $32 + 1 + 1$ 次中斷，32 表示 Customer + Data 的部分，而開頭 Leader code 和 stop bit 各有一次下緣觸發，因此總共是 34 次，我們可以驗證，如果輸入了一個 NEC 協定的訊號，而最終 counter 的值為 34，表示我們成功收到了一個 NEC 協定的訊號。

我們也嘗試將每一個觸發中斷的時間差給印出來，由於我們是由在某一個電壓變化緣會處發到中斷。因此我們可以通過比較上一個中斷觸發的時間，知道兩個電位差的時間，檢視電位差時間，我們可以去判定該遙控器協議，如果該遙控器使用 NEC 協議，我們可以判斷第一個負緣到第二個負緣，依序時間差為 $9.0\mu s$ 和 $4.5\mu s$ ，由下面 UART1 的輸出我們可以看到



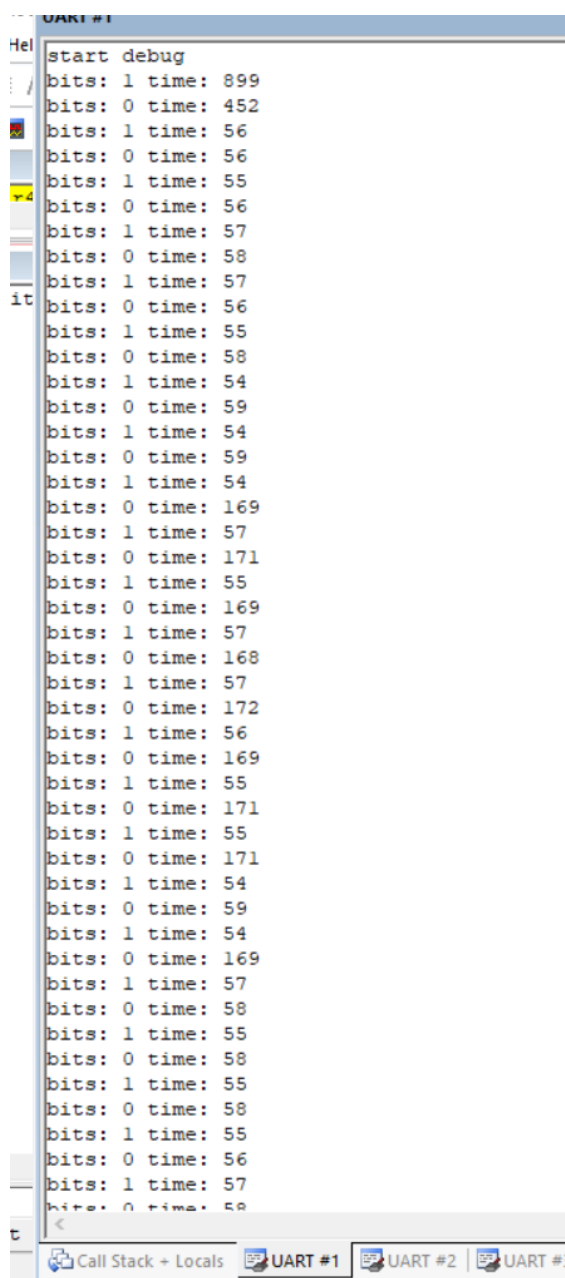
```
UART #1
start debug
bits: 1 time: 915
bits: 0 time: 435
bits: 1 time: 69
bits: 0 time: 43
bits: 1 time: 69
bits: 0 time: 44
bits: 1 time: 70
bits: 0 time: 43
bits: 1 time: 69
bits: 0 time: 44
bits: 1 time: 69
bits: 0 time: 44
bits: 1 time: 69
bits: 0 time: 44
bits: 1 time: 69
bits: 0 time: 44
bits: 1 time: 69
bits: 0 time: 157
bits: 1 time: 66
bits: 0 time: 160
bits: 1 time: 69
bits: 0 time: 157
bits: 1 time: 68
bits: 0 time: 157
bits: 1 time: 69
bits: 0 time: 158
bits: 1 time: 65
bits: 0 time: 160
bits: 1 time: 69
bits: 0 time: 157
bits: 1 time: 68
bits: 0 time: 158
bits: 1 time: 66
bits: 0 time: 46
bits: 1 time: 69
```

圖(十四) 通過 GPIO 讀取電位變化的時間差

發現到基本上大致符合 NEC 協定的電位變化，我們試著直接使用這個訊號進行發射，但電風扇並沒有任何的回應，初步推測是因為誤差導致構造出的封包不符合 NEC 協定的規範了。

NEC 協議開頭的部分，應該要有 9 us 的時間和一段 4.5 us 的時間，但是我們嘗試獲取我們發送的訊號，發現時間間隔誤差十分的巨大，我們推測正是因為這樣的時間差，造成我們發送的訊號無法成功被我們想控制的電風扇解析，因此我們需要處理延遲律問題。

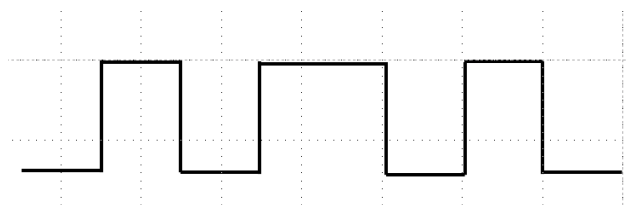
而會造成這樣的誤差原因（大概屬於 us 等級的誤差），是因為高電位信號的內容會比低電位的信號長約 160us，推測原因是因為在 IR Receiver 中接到的信號是反向的，也就是說高電位的信號會輸出低電位的信號，而當從低電位要恢復到高電位時，需要更多的時間，因此產生了這個延遲，這個問題在更換接收器之後成功解決，推測是硬體所造成。



```
UART #1
start debug
bits: 1 time: 899
bits: 0 time: 452
bits: 1 time: 56
bits: 0 time: 56
bits: 1 time: 55
bits: 0 time: 56
bits: 1 time: 57
bits: 0 time: 58
bits: 1 time: 57
bits: 0 time: 56
bits: 1 time: 55
bits: 0 time: 58
bits: 1 time: 54
bits: 0 time: 59
bits: 1 time: 54
bits: 0 time: 59
bits: 1 time: 54
bits: 0 time: 169
bits: 1 time: 57
bits: 0 time: 171
bits: 1 time: 55
bits: 0 time: 169
bits: 1 time: 57
bits: 0 time: 168
bits: 1 time: 57
bits: 0 time: 172
bits: 1 time: 56
bits: 0 time: 169
bits: 1 time: 55
bits: 0 time: 171
bits: 1 time: 55
bits: 0 time: 171
bits: 1 time: 54
bits: 0 time: 59
bits: 1 time: 54
bits: 0 time: 169
bits: 1 time: 57
bits: 0 time: 58
bits: 1 time: 55
bits: 0 time: 58
bits: 1 time: 55
bits: 0 time: 58
bits: 1 time: 55
bits: 0 time: 56
bits: 1 time: 57
bits: 0 time: 58
```

圖(十五) 更換接收器後，通過 GPIO 讀取電位變化的時間差

接著是發送訊號給電風扇的部分



上圖為我們發送的訊號，但是重新檢視 NEC 協議後，由上面 NEC 邏輯 0 與邏輯 1 的訊號表示（在大部分的紅外線通訊協議都有此現象），我們得知在發送高電位訊號時，不是在單位時間都是高電位輸出，而是不斷的在高電位與低電位之間切換，也就是在高電位和低電位之間應該會存在訊號的佔空比，這一點可以從我們接收訊號中看出。

```
bits: 1 time: 69
bits: 0 time: 43
bits: 1 time: 69
bits: 0 time: 44
bits: 1 time: 70
bits: 0 time: 43
```

我們整理實驗一到三目前的結論

1. 基本確認我們成功解讀 NEC 訊號，因為符合 NEC 的規範
2. 我們使用了 PWM_IrDA_NEC 的 sample code 作為發送訊號，發送到我們的輸入端，並且解析出來的結果也同樣符合 NEC 的協定，因此我們判斷，PWM_IrDA_NEC 的 sample code 中 SendNEC 基本上是正確的。

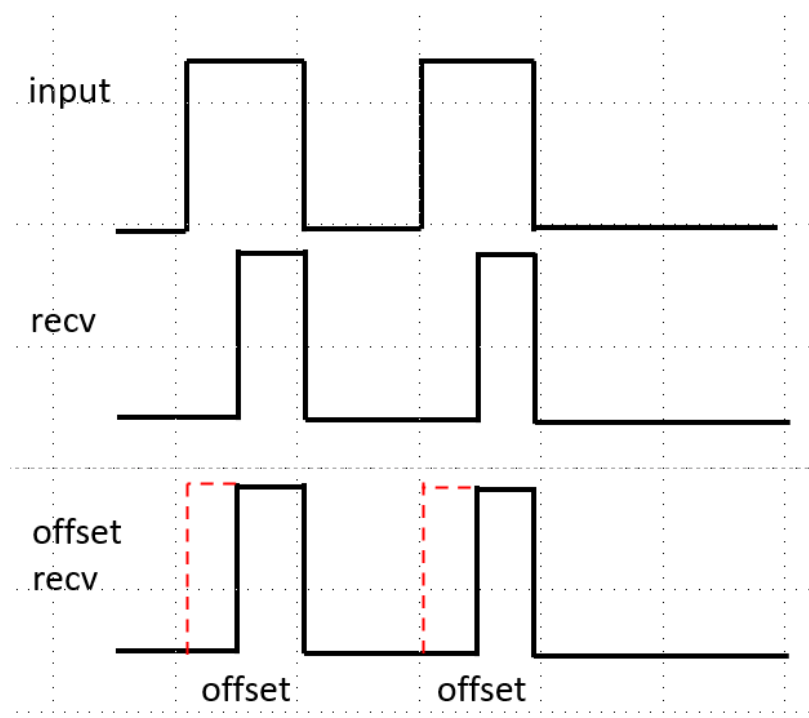
1. 我們要確保發送訊號的正確性，也就是讓我們能夠成功發送邏輯 0 以及邏輯 1。
2. 硬體所造成的讀取誤差

實驗四：實驗三改進，藉由修正訊號正確性，發送邏輯 0 和邏輯 1，並且修正誤差

預期結果：成功解析並且構造符合 NEC 規範的紅外線訊號，並成功被電風扇識別

首先我們先著手誤差問題，我們知道不同的紅外線協定會有不同的邏輯 0 和邏輯 1 的時間長度，而我們的做法為手動消除誤差，而為了成功完成，我們僅針對 NEC 協定進行誤差修正。

誤差部分我們使用了兩種方式進行修正，第一種方式為讓所有訊號進行偏移，如下圖所示



圖(十八) 針對接收訊號進行修正

修正的誤差長度為 160 us，但修正過後並發送並沒有成功。推論是經過修正之後，精確度依然不足，導致接收端無法正確解析。

實驗小節：我們需要將收到的訊號自行解析，將解析的訊號傳入

IDRA_NEC library 中 SendNEC 發送。

實驗五：通過 NEC 推导出訊號，並且直接通過 NEC sample code 中 SendNEC 發送訊號

預期結果：由於我們驗證了 NEC 的 SendNEC 發送訊號正確，而我們又成功通過 GPIO 解析訊號，因此我們發送的訊號會成功被電風扇識別

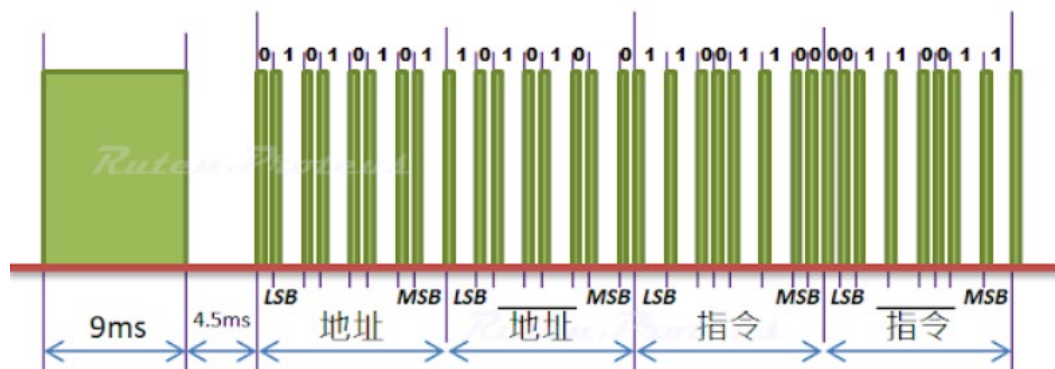
而上面我們通過 GPIO 成功讀取 NEC 訊號，得到了時間差，而我們打算通過時間差以及電位差，我們就能夠根據 NEC 協定推导出邏輯 0 和邏輯 1，從而得到正確的資料。

我們進行實驗，使用 SendNEC 發送訊息，並且成功被解析，但是我們隨後發現到，接收到的訊號與發送的訊號並不相同，而我們試著使用 16 進位印出。

發送訊號：807F01FE

接收訊號：01FE807F

我們發現到這個錯誤並不是接收錯誤，而是解析上的錯誤，原因是因為我們接收到的訊號是符合 NEC 規範的，也就是 01 FE 為互補關係，80 7F 為互補關係，為了修正這一個錯誤，我們重新檢視了 NEC 協定的規範，如下圖所示



圖(十九) NEC 紅外線通訊協定格式

我們可以發現每一個 bytes (8 bit) 的資料採用 LSB (Least Significant Bit) 先傳送，也就是最低位先傳送，而 MSB (Most Significant Bit) 最後傳送，這樣帶來的效果可以用下面表示，下面是常規的 MSB 到 LSB 的方式

10000000 = 128

而如果是 LSB 到 MSB，則會是下面解析方式

00000001 = 1

而回到我們的發送訊號與接收訊號，如果將我們接收到的訊號由 MSB to LSB 改變成 LSB to MSB，則為以下

```
01      FE
00000001 11111110

80      7F
10000000 01111111
```

可以看到我們將 MSB to LSB 改變成 LSB to MSB，就可以讓接收到的訊號和發送的訊號相同了，原理為上面的 NEC 協定所示。

如此，我們便初步實現了針對於 NEC 協定的萬用遙控器實作了。

專題總結

經過本專題，我們學習到了以下

- 如何使用 debugger 對程式碼進行追蹤，包含 stack trace 等等
- C 語言中巨集的展開以及巨集追蹤（在修正 sample code 時使用）
- 數位訊號的分析以及處理
- 訊號的延遲處理與發送
- NEC 與其他紅外線訊號協議的研讀，了解到使用補數對訊號進行校驗等等
- 如何閱讀外部連接裝置的規格書，並了解硬體方面的限制

本次專題研究過程幾經曲折，最終完成了針對於 NEC 協議的萬用遙控器，過程中也收穫了很多，對整個微處理機控制系統有了更加深入的了解。

參考資料

NUC140_Technical_Reference_Manual_EN_Rev2.05

Nu-LB-NUC140_User_Manual_EN_Rev2.0

紅外線遙控原理與 NEC IR

Protocol(<http://coopermaa2nd.blogspot.com/2010/01/nec-ir-protocol.html>)

NEC Infrared Transmission

Protocol(<https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol>)

Arduino-IRremote github(<https://github.com/Arduino-IRremote/Arduino-IRremote>)