
專案一 產線機台介面

2018/09/21

專案說明

- 設計一個簡易的使用者介面，可供顯示目前機台運作狀態。
- 使用LED作為實際機台的顯示介面。
- 使用者端則使用QT作為操作介面，用以機台互動。
- 每一個小項目功能5分。

專案要求

- 在本專案中，機台分為下列幾種工作狀態。
 1. 啟動中(程式開啟)
 - ① 三個LED同時閃爍(亮暗間距為一秒)。
 - ② Slider及Slider Editor數值改變時，LED不受影響。
 - ③ Start按鈕為可操作狀態(enable)。
 - ④ Stop按鈕及Reset按鈕為無法操作狀態(disable)。
 - ⑤ Slider初始狀況為最小值(0)，Slider Editor初始值為0。

專案要求

2. 開始(Start按下去之後)

- ① 第一個LED維持亮，不再閃爍。
- ② 另外兩個LED為當前Slider的數值(0為暗暗，1為暗亮，2為亮暗，3為亮亮)。
- ③ Slider最大值為3，最小值為0，Slider Editor需跟著Slider拉動而改變，反之Slider Editor改變Slider也會跟著改變。
- ④ 若Slider Editor輸入大於3則需維持3，小於0則維持0。
- ⑤ Slider及Slider Editor改變時，LED顯示也會跟著一起改變。
- ⑥ Start按鈕為不可操作狀態(disable)。
- ⑦ Stop按鈕及Reset按鈕為可操作狀態(enable)。

專案要求

3. 停止(Stop按下去之後)

- ① 所有LED維持暗。
- ② Slider及Slider Editor數值改變時，LED不受影響(保持暗)。
- ③ Start按鈕及Reset按鈕為可操作狀態(enable)。
- ④ Stop按鈕為不可操作狀態(disable)。

專案要求

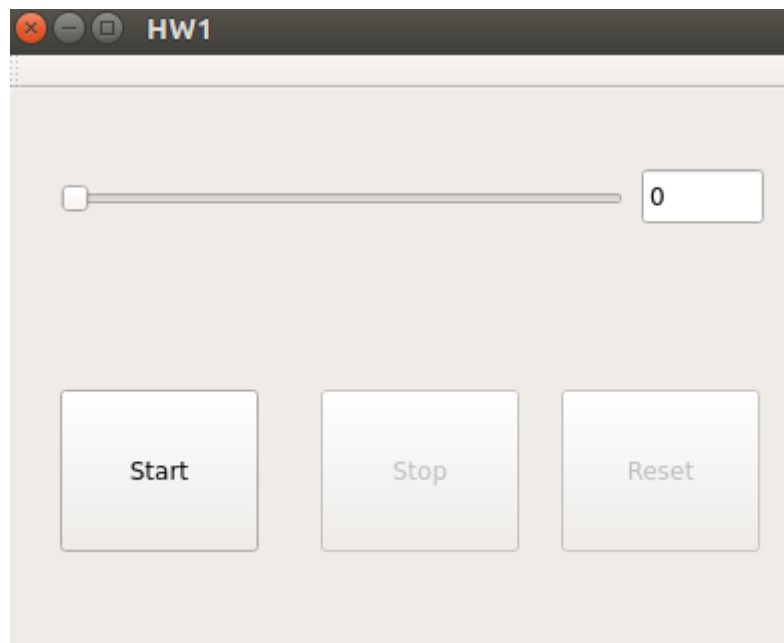
4. 重置(Reset按下去之後)

- ① 系統回到啟動中(程式開啟狀態)。

5. 關閉(程式結束)

- ① 關閉GPIO(unexport)。

使用者端介面



執行說明

1. 請在虛擬機安裝filezilla (Lecture1 有說明)
2. 在Windows10 打開CMD輸入 `arp -a`可以看到如下圖，然後根據板子查看IP，有可能很多組，ping看看哪組能連。

```
介面: 192.168.137.1 --- 0x2
IP 地址          實體位址      類型
192.168.137.28    00-04-4b-3a-d8-be 靜態
192.168.137.47    00-04-4b-3a-d8-be 靜態
192.168.137.117   00-04-4b-3a-d8-be 靜態
192.168.137.255   11-11-11-11-11-11 靜態
224.0.0.22        01-00-5e-00-00-16 靜態
224.0.0.251       01-00-5e-00-00-fb 靜態
224.0.0.252       01-00-5e-00-00-fc 靜態
230.0.0.1         01-00-5e-00-00-01 靜態
239.255.255.250   01-00-5e-7f-ff-fa 靜態
255.255.255.255   ff-ff-ff-ff-ff-ff 靜態
```

3. 打開QT做完專案，在虛擬機端編譯完成後，在資料夾中會產生makefile，將 `x86_64-linux-gnu` 全部換成 `arm-linux-gnueabi`

執行說明

3. 打開QT做完專案，在虛擬機端編譯完成後，在資料夾中會產生makefile，將x86_64-linux-gnu 全部換成 arm-linux-gnueabi

```
Makefile x
##### Compiler, tools and options
CC      = gcc
CXX      = g++
DEFINES  = -DQT_QML_DEBUG -DQT_DECLARATIVE_DEBUG -DQT_WIDGETS_LIB -DQT_GUI_LIB -DQT_CORE_LIB
CFLAGS   = -m64 -pipe -g -Wall -W -D_REENTRANT -fPIE $(DEFINES)
CXXFLAGS = -m64 -pipe -g -Wall -W -D_REENTRANT -fPIE $(DEFINES)
INCPATH  = -I/usr/lib/arm-linux-gnueabi/qt5/mkspecs/linux-g++-64 -I. -I/usr/include/qt5 -I/usr/include/qt5/QtWidgets -I/usr/
include/qt5/QtGui -I/usr/include/qt5/QtCore -I. -I.
LINK      = g++
LFLAGS   = -m64
LIBS      = $(SUBLIBS) -L/usr/X11R6/lib64 -lQt5Widgets -L/usr/lib/arm-linux-gnueabi/qt5 -lQt5Gui -lQt5Core -lGL -lpthread
AR        = ar cqs
RANLIB    =
QMAKE     = /usr/lib/arm-linux-gnueabi/qt5/bin/qmake
TAR        = tar -cf
COMPRESS  = gzip -9f
COPY      = cp -f
SED        = sed
COPY_FILE = cp -f
COPY_DIR  = cp -f -R
STRIP     = strip
INSTALL_FILE = install -m 644 -p
INSTALL_DIR = $(COPY_DIR)
INSTALL_PROGRAM = install -m 755 -p
DEL_FILE  = rm -f
SYMLINK   = ln -f -s
DEL_DIR   = rmdir
MOVE      = mv -f
CHK_DIR_EXISTS = test -d
MKDIR     = mkdir -p

##### Output directory
OBJECTS_DIR = ./
```

執行說明

5. 用虛擬機的filezilla把專案傳到板子。
6. 使用ssh -Y ubuntu@板子IP，連上板子，打開剛剛傳過來的資料夾，先make clean在qmake 最後 make就會產生執行檔。
7. 由於使用GPIO所以執行時要加 sudo。
8. 板子時間有問題可能會產生編譯問題，因此需先修改時間
sudo date MMDDhhmmYYYY

```
ubuntu@tegra-ubuntu: ~/untitled
ubuntu@tegra-ubuntu:~/untitled$ ls
Makefile  mainwindow.cpp  mainwindow.ui  ui_mainwindow.h  untitled.pro.user
main.cpp  mainwindow.h    moc_mainwindow.cpp  untitled
main.o    mainwindow.o    moc_mainwindow.o    untitled.pro
ubuntu@tegra-ubuntu:~/untitled$
```

GPIO說明文件及範例

1. <https://github.com/derekmolloy/boneDeviceTree/>
2. https://elinux.org/Jetson/Tutorials/Vision-controlled_GPIO