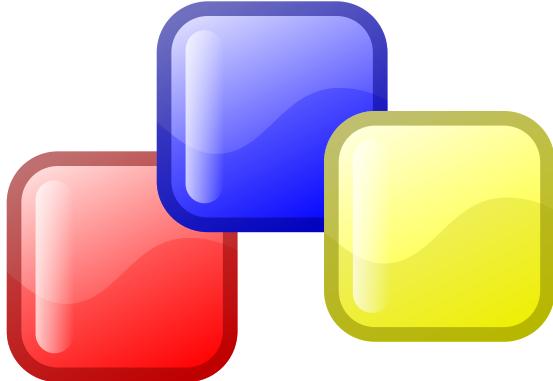


wxWidgets for Cross-platform GUI Programming in C++



စက်အမျိုးမျိုးအတွက် C++ ဖြင့် GUI ပရိဂရမ်များရေးသားခြင်း

ကိုအေး

Yan Naing Aye @ ကိုအေး

Cool Emerald: <http://coolemerald.blogspot.com>

Typeset with ~~X_ET_EX~~ for A4 paper.

၀.၁ အကြိမ်တည်းဖြတ်ခြင်း။

ဇန်နဝါရီ ၂၀၂၀။

ဤစာအုပ်၏ နောက်ဆုံးတည်းဖြတ်မှုကို <https://yan9a.github.io/wxwidgets/wxmm.pdf> တွင် ရယူ နိုင်ပြီး ဖော်ပြု ထားသော နမူနာ များကို <https://github.com/yan9a/wxwidgets> တွင် တွေ့နိုင်သည်။

ဤစာအုပ်ကို လွတ်လပ်စွာ ကူးယူ၊ ဖြန့်ဝေ၊ အသုံးပြု နိုင်သည်။ ခွင့်ပြုထား သော လိုင်စင်မှာ CC-BY-4.0 ဖြစ်သည်။ ထိုကြောင့် သင့်တင့် လျောက်ပတ်သော အသိအမှတ် ပြုမှု credit နှင့် link ဖော်ပြရန် လိုသည်။ အသေးစိတ်ကို <http://creativecommons.org/licenses/by/4.0/> တွင် တွေ့ နိုင်သည်။

မာတိကာ

မာတိကာ	i
၁ မိတ်ဆက်	၁
၁.၁ Windows တွင်တပ်ဆင်ခြင်း	၂
၁.၁.၁ wxDev-C++ ဖြန့်အသုံးပြုခြင်း	၂
၁.၁.၂ Visual Studio ဖြန့်အသုံးပြုခြင်း	၅
၁.၂ Linux တွင်တပ်ဆင်ခြင်း	၁၀
၁.၂.၁ Source မှ build လုပ်ခြင်း	၁၂
၁.၂.၂ Code::Blocks ဖြန့်အသုံးပြုခြင်း	၁၃
၁.၃ Mac တွင်တပ်ဆင်ခြင်း	၁၇
၂ OpenCV နှင့်တဲ့သုံးခြင်း	၂၀
၂.၁ Linux	၂၀
၂.၁.၁ Code::Blocks	၂၄
၂.၂ Windows	၂၀
၃ Network ချိတ်ဆက်အသုံးပြုခြင်း	၂၆
၃.၁ UDP	၂၆
၃.၂ TCP	၄၈
၃.၂.၁ TCP Server	၄၈
၃.၂.၂ TCP Client	၆၀

၄ Serial Port ကိုသုံးခြင်း	၇၄
၄.၁ UART	၇၄
၄.၁.၁ Start Bit	၇၄
၄.၁.၂ Baud Rate	၇၅
၄.၁.၃ Data Bits	၇၅
၄.၁.၄ Parity Bit	၇၅
၄.၁.၅ Stop Bit	၇၅
၄.၂ RS-232	၇၆
၄.၂.၁ Handshaking	၇၇
၄.၃ Programming Serial Port	၇၉
၄.၃.၁ Console	၈၀
၄.၃.၂ GUI	၈၂
၅ Multithreading	၉၇
၅.၁ ဘယ်လို အချိန်တွေမှာ သုံးသင့်၊ မသုံးသင့်	၉၇
၅.၂ wxThread သုံးခြင်း	၉၈
၅.၂.၁ Thread တစ်ခု ဖန်တီးခြင်း	၉၉
၅.၂.၂ Thread ကို ခေါ် ရပ်နားခြင်း	၁၀၀
၅.၂.၃ Thread ကို အဆုံးသတ်ခြင်း	၁၀၀
၅.၃ Synchronization Objects	၁၀၃
၅.၃.၁ Mutex	၁၀၄
၅.၃.၂ Deadlocks	၁၀၅
၅.၃.၃ Critical Section	၁၀၅
၅.၃.၄ Semaphore	၁၀၆
၅.၄ Simple Multithreading Example	၁၀၆
၅.၄.၁ Log Target ပြောင်းခြင်း	၁၀၆
၅.၄.၂ Thread များဖန်တီးခြင်း	၁၀၈
၅.၄.၃ Thread လုပ်ငန်းများသတ်မှတ်ခြင်း	၁၁၀
၅.၄.၄ Thread ကို ဖျက်ခြင်း	၁၁၁

၅.၄.၅ Pause နှင့် Resume	၁၁၄
၅.၅ Event များသုံးခြင်း	၁၂၆
၅.၆ Thread မှ GUI ကိုလှစ်သုံးခြင်း	၁၃၆
၅.၇ အခြားနည်းလမ်းများ	၁၄၈
A နောက်ဆက်တဲ့	၁၅၀
၁.၁ စကားလုံးဖွင့်ဆိုချက်များ	၁၅၀
B Code Listing	၁၅၁
J.၁ Serial.h	၁၅၁

အခန်း ၁

မိတ်ဆက်

wxWidgets က Windows । Linux နဲ့ Mac OSX အစ ရှိတဲ့ ပလက်ဖောင်း အမျိုးမျိုး ပေါ်မှာ GUI applications တွေ ရေးဖို့ အတွက် C++ library ပါ။ သူနဲ့ GUI code တွေ ရေးပြီး ရင် ပလက်ဖောင်း အမျိုးမျိုး ပေါ်မှာ ကုဒ် ကို သိပ်ပြင် စရာ မလိုပဲ တန်းပြီး compile လုပ်၊ run လုပ်လို ရပါတယ်။

wxWidgets က free လည်းပေး open source လည်း ဖြစ် တဲ့ software ပါ။ ကိုယ်ပိုင် ဆော့တဲ့ တွေ ထုတ်မယ် ဆိုရင် လည်း ဘာမှ ကန်သတ်ချက် တွေ မရှိ ပါဘူး [wxW98]။ အဲဒါက Qt နဲ့ အမိက ကွာခြားချက် ပါ။ Qt က LGPLv3 လိုင်စင် ကို free ပေးထားပြီး၊ ကိုယ်ပိုင် စီးပွားရေး အတွက် သုံးမယ် ဆိုရင် ကန်သတ်ချက် တရီး၊ ရှိတာကြောင့် လိုင်စင် ဝယ်ဖို့ လိုကောင်း လိုနိုင် ပါတယ် [Qt17]။

Native platform ကို တတ်နိုင် သလောက် သုံးထား တာမို့ wxWidgets သုံးပို့ ရလာတဲ့ GUI တွေဟာ သုံးတဲ့ platform နဲ့ လိုက်ဖက်ပြီး ပင်ကို အမြင် အတိုင်း တသားထဲ ကျ တာကို ခံစား ရမှာပါ [wxW12]။ Standard C++ ကိုပဲ သုံးထားပြီး Qt တို့လို အထူး extension တွေ မသုံးထား တဲ့ အတွက် ရှုပ်ထွေးမှု နည်းတာ ကလည်း ကောင်းတဲ့ အချက် တစ်ခုပါ။

wxWidgets နဲ့ ရလာတဲ့ binary application တွေဟာ သေးယော ပေါ့ပါး တာမို့ embedded system တွေအတွက် အထူး သင့်တော် ပါတယ်။ နောက်တစ်ခါ library အရွယ်အစား တွေ ယုဉ်ရင်လည်း ဥပမာ အနေနဲ့ Qt library ကို တပ်ဆင်ရင် \approx 200 MB လောက် အရွယ် ရှိပေမယ့် wxWidgets library က \approx 30 MB လောက်ပဲ ယူပါတယ်။

wxWidgets က C++ အတွက် သာ မကဲ့ python, perl, php, java, lua, lisp, erlang, eiffel, C# (.NET), BASIC, ruby နဲ့ javascript အတွက် တောင်မှ bindings [wxW15a] တွေ ရှိပါတယ်။ wxWidgets က တော်တော် ပြည့်စုံ ရင့်ကျက် တဲ့ GUI toolkits ဖြစ်ပြီး၊ utility classes လည်း အများကြီး ရှိတာမို့ ကောင်းမွန် သင့်တော် တဲ့ GUI toolkits အနေနဲ့ ညွှန်းဆို ချင်ပါတယ်။

wxWidgets ကို အသုံးပြုတဲ့ သူတွေ၊ အဖွဲ့အစည်းတွေ အများကြီး ရှိပြီး အဲဒီ အထဲမှာ လူသီများ တာတွေက NASA, AMD, Xerox, နဲ့ Open Source Applications Foundation (OSAF) တို့ ဖြစ်ပါတယ်။ ထင်ရှားတဲ့ wxWidgets applications တွေက AVG AntiVirus, Audacity, Filezilla, Code::Blocks, CodeLite တို့ ဖြစ်ပါတယ်။

၁.၁ Windows တွင်တပ်ဆင်ခြင်း

wxWidgets ကို Windows မှာ အသုံးပြု တဲ့ အခါ IDE အမျိုးမျိုး နဲ့ တွဲသုံးနိုင် ပါတယ်။ Microsoft Visual Studio IDE ကို သုံးလို ရသလို [wxDev-C++](#) | [CodeLite \[Cod17; Cod13\]](#) | [Code::Blocks](#) စတဲ့ IDE တွေ ကိုလည်း ရွေးချယ် အသုံး ပြုနိုင် ပါတယ်။

၁.၁.၁ wxDev-C++ ဖြင့်အသုံးပြုခြင်း

Windows ပေါ်မှာ wxWidgets applications တွေ ဖန်တီးဖို့ ရွေးချယ် စရာ တစ်ခု အနေနဲ့ ဆိုရင် wxDev-C++ ကို တော်တော် သဘောကျပါတယ်။ အဲဒီမှာ [Programming with wxDev-C++ \(http://wxdevcpp-book.sourceforge.net/\)](#) ဆိုတဲ့ စာအုပ်ကို လည်း တရာ့တဲ့ ရနိုင် တာကလည်း ပရိုဂရမ်တွေ လေ့လာ၊ ဖန်တီး တဲ့ အခါ လွယ်ကူ အဆင်ပြေ မြန်ဆန် ဖို့ အထောက် အကူ တစ်ခု ပါ။ သူကို [http://wxdsgn.sourceforge.net/](#) ကနေ download လုပ်ဖြီး install လုပ်နိုင် ပါတယ်။

အဲဒီနောက် wxDev-C++ ကို ဖွံ့ဖြိုး တဲ့ အခါ File menu → New... → Project ကို နိုပ်ပြီး project တစ်ခု ဖန်တီး လိုက် ပါမယ်။ New project ဝင်းဒီး ပေါ်လာ တဲ့ အခါ Basic tab ထဲက wxWidgets Frame အမျိုးအစား ကို ရွေး၊ wxTest စတဲ့ နာမည် တစ်ခုခု ပေး ပြီး OK ကို နိုပ်လိုက် ပါမယ် (ပုံ ၁.၁)။

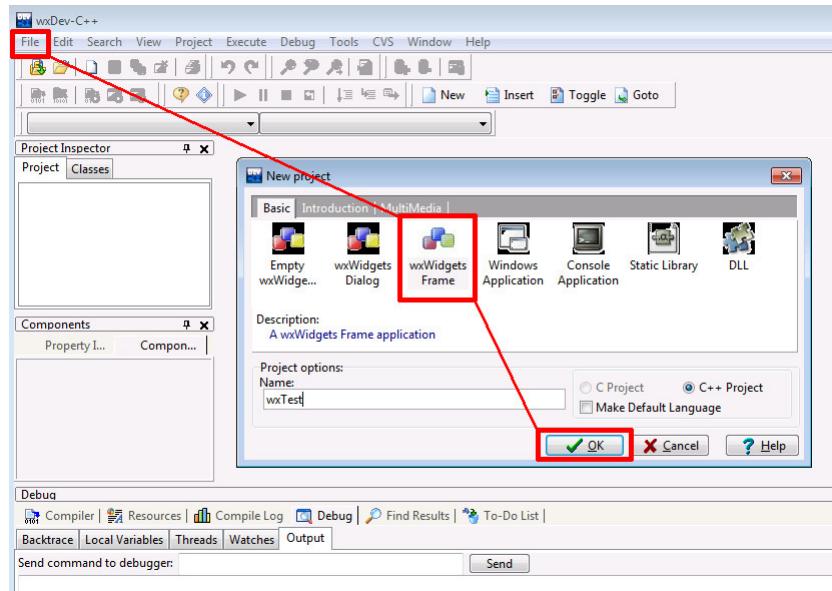
အဲဒီနောက် New wxFrame ဝင်းဒီး မှာ Window style အမျိုးအစား၊ MingW gcc စတဲ့ compiler တွေကို ရွေးပြီး Create ခလုတ် ကို နိုပ် နိုင် ပါတယ် (ပုံ ၁.၂)။

Project ဖန်တီး ရလာ တဲ့ အခါ GUI design ကို wxTestFrm.wxform ဆိုတဲ့ tab မှာ တွေ့နေ ရမှာ ဖြစ်ပါတယ်။ wxTestFrm.cpp စတဲ့ source ဖိုင် တွေမှာ လည်း အမျိုးမျိုး ပြုပြင် စမ်းသပ် နိုင် ပါတယ်။ Run ခလုတ် ကိုနိုပ် ဒါမှုမဟုတ် F8 ကို ကို နိုပ် လိုက်ရင် ပုံ ၁.၃ ကလို GUI application run လာတာ ကို တွေ့ရ ပါလိမ့်မယ်။

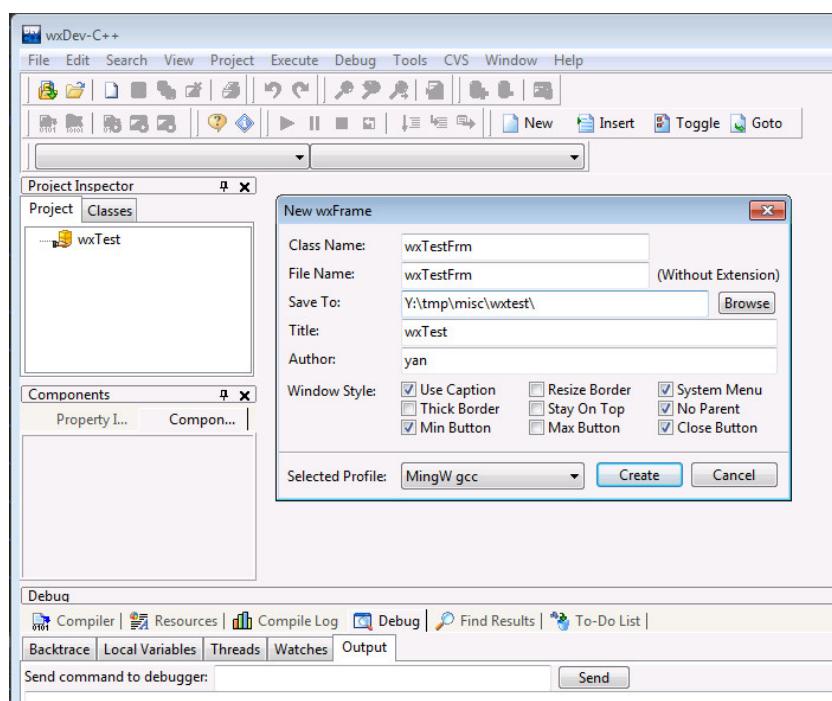
Windows application တွေမှာ form ရဲ့ icon ကို သတ်မှတ်လို ရပါတယ်။ wxWidgets ကုဒ်ထဲ မှာ icon ဖိုင်ကို xpm format နဲ့ တစ်ခါထည်း ထည့်လို ရပြီး ဥပမာ အနေနဲ့ favicon.xpm ကို ထည့်ချင်ရင် form ရဲ့ cpp ဖိုင်အစမှာ အောက်က အတိုင်း ထည့်နိုင်ပါတယ်။

၁.၁. WINDOWS တွင်တပ်ဆင်ခြင်း

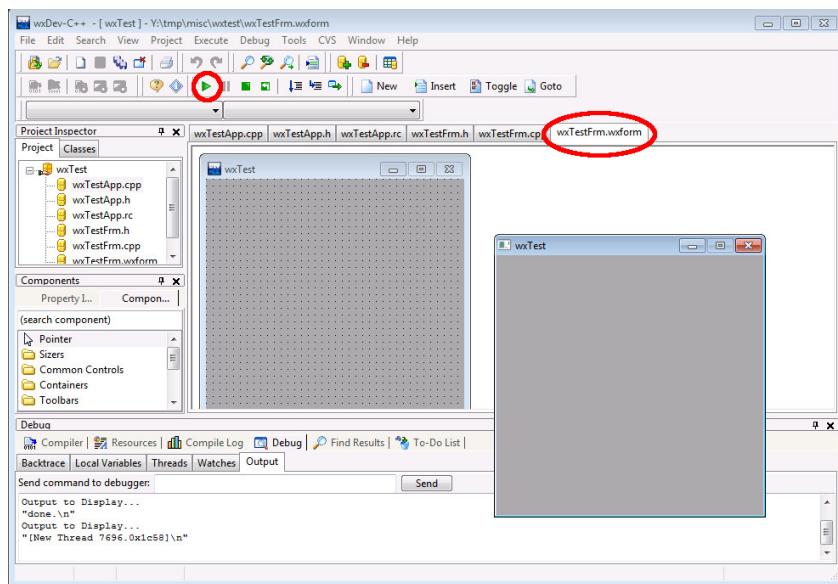
၃



ပုံ ၁.၁: wxWidgets frame based project တစ်ခုကို wxDev-C++ တွင် ဖန်တီးခြင်း။



ပုံ ၁.၂: Frame အမျိုးအစား ရွေးချယ်သတ်မှတ်ခြင်း။



ပုံ ၁.၃: GUI နမူနာ application တစ်ခု wxDev-C++ တွင် run ခြင်း။

```
#include "favicon.xpm"
```

နောက် ကုဒ်ထဲမှာ icon ကို အောက်က အတိုင်း ပြောင်းပေး ပြီး သတ်မှတ်နိုင်ပါတယ်။

```
SetIcon(wxIcon(favicon_xpm));
```

icon ဖိုင်ကို xpm format နဲ့ရချင်ရင် ဖိုတိ ရော့ လိမ့်း ပုံတွေကို ပြင်ပေးနိုင်တဲ့ GIMP ဆိုတဲ့ ဆော်ဖို့ ကို သုံးနိုင်ပါတယ်။ xpm ဖိုင်ဖန်တီးပြီးတဲ့ အခါမှာ အဲဒီဖိုင်ကို Notepad လိမ့်း text editor တစ်ခုခဲ့ ဖွင့်ပြီး ထိပ်ဆုံးနားမှာ ပါတဲ့ ဖိုင် path ကို ပြောင်းဖို့ လိုကောင်း လိုနိုင်ပါတယ်။ ရှော ဖိုင် path တွေကို အကုန်ဖျက်ပြီး ဖိုင်နာမည် တစ်ခုပဲ ချုန်တာ အဆင်ပြေ တာကို တွေ့ရ ပါတယ်။ Windows explorer မှာ ပေါ်မယ့် program ဖိုင်ရဲ့ icon ကို ပြင်ချင်ရင် တော့ ဖိုင် extension rc ဆိုပြီးရှိတဲ့ resource ဖိုင်ကို ပြင်ဖို့ လိုပါတယ်။ ဥပမာ အနေနဲ့ favicon48.ico ဆိုတဲ့ icon ကို သုံးချင်ရင် အောက်ကအတိုင်း ကုဒ်တစ်လိုင်း ထပ်ဖြည့်ပြီး ပြင်ဖို့ လိုပါတယ်။

```
aaaaAppIcon ICON "favicon48.ico"
#include <wx/msw/wx.rc>
```

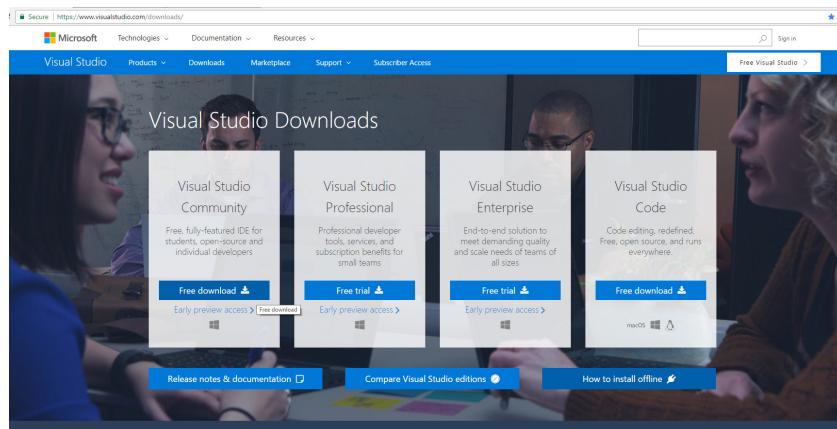
၁.၁. WINDOWS တပ်ဆင်ခြင်း

၅

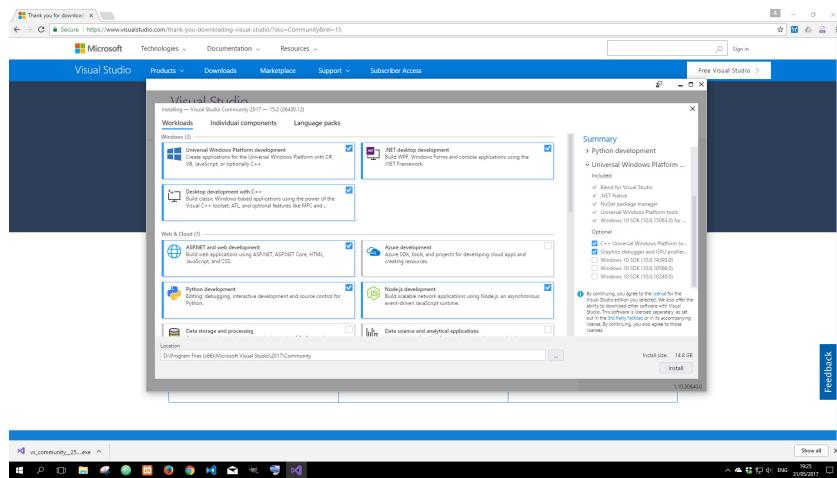
အခါမှာ icon ဖိုင်ရဲ့ နာမည်ကို aaaaAppIcon လို့ ပေးရတဲ့ အကြောင်းက icon ဖိုင်တွေကို alphabetically စဉ်လိုက်ရင် သူကို အရင်တွေအောင်လိုပါ။

၁.၁.၂ Visual Studio ဖြန့်အသုံးပြုခြင်း

IDE အတွက် Visual Studio ကို <https://www.visualstudio.com/downloads/> ကနေ download လုပ်ပြီး တဲ့အခါ ကိုယ် တပ်ဆင် ချင်တဲ့ အစိတ် အပိုင်း တွေကို သတ်မှတ်ပြီး တပ်ဆင် နှင့်ပါ တယ် (ပုံ ၁.၄ နှင့် ပုံ ၁.၅)။



ပုံ ၁.၄: Visual Studio Community 2017 ကို free download လုပ်၍ ရယူခြင်း။



ပုံ ၁.၅: Visual Studio Community 2017 ကို တပ်ဆင် ခြင်း။

နောက်တစ်ခါ wxWidgets ကို <https://www.wxwidgets.org/downloads/> ကင့် download လုပ်လိုက် ပါမယ် (ပုံ ၁.၆)။ ဒီ နမူနာ မှာတော့ latest stable release ဖြစ်တဲ့ 3.0.3 Windows installer ကို ရွေးလိုက် ပြီး C:/wxWidgets303/ ထဲမှာ install လုပ်လိုက် ပါမယ် (ပုံ ၁.၇)။



ပုံ ၁.၆: wxWidgets ကို free download လုပ်၍ ရယူ ခြင်း။

Latest Stable Release: 3.0.3

A screenshot of the wxWidgets 3.0.3 latest stable release download page. The page is divided into two main sections: 'Source Code' on the left and 'Documentation' on the right. The 'Source Code' section offers Windows ZIP (31 MB), Windows 7Z (16 MB), Windows Installer (47 MB), and Source for Linux, OS X, etc (20 MB). The 'Binaries' section provides wxMSW DLLs for Visual C++ 2008-2017 and TDM-GCC 4.9 and 5.1, along with Ubuntu/Debian and Fedora/openSUSE package links. The 'Documentation' section includes links to Readme, Changes, and various manual formats (HTML ZIP, BZIP, CHM). Below the sections, a note states 'Released: May 2nd, 2017' and 'API Stable Since: November 11th, 2013'.

ပုံ ၁.၇: wxWidgets latest stable release ကို Windows အတွက် installer ရယူ ခြင်း။

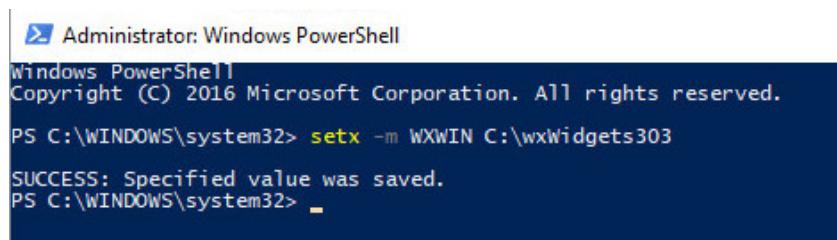
အဲဒီနောက် C:/wxWidgets303/docs/msw ထဲမှာ ဒါမေဟုတ် online မှာလည်း တွေ့နိုင် တဲ့ install.txt ဆိုတဲ့ guide ထဲက Microsoft Visual C++ compilation ဆိုတဲ့ အပိုင်း ကို ဖတ်နိုင် ပါတယ် [wxW15b]။ စစ်ခြင်း WXWIN ဆိုတဲ့ environment variable ကို သတ်မှတ်ဖို့ လိုပါတယ်။ အဲဒီ အတွက် command window ကို administrator အနေနဲ့ run ဖို့ start menu ကို right click နိပ်ပြီး command

၁.၁. WINDOWS တွင်တပ်ဆင်ခြင်း

၇

prompt (admin) ကို ရွေးလိုက် ပါမယ်။ အဲဒီမှာ အောက်က command သုံးပြီး WXWIN ကို ပုံ ၁.၈ ကလို သတ်မှတ် လိုက် ပါမယ်။

```
setx -m WXWIN C:\wxWidgets303
```



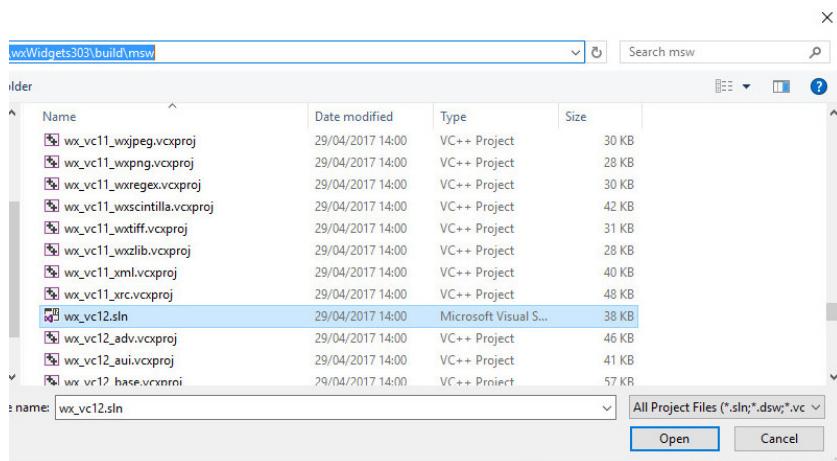
```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\WINDOWS\system32> setx -m WXWIN C:\wxWidgets303

SUCCESS: Specified value was saved.
PS C:\WINDOWS\system32>
```

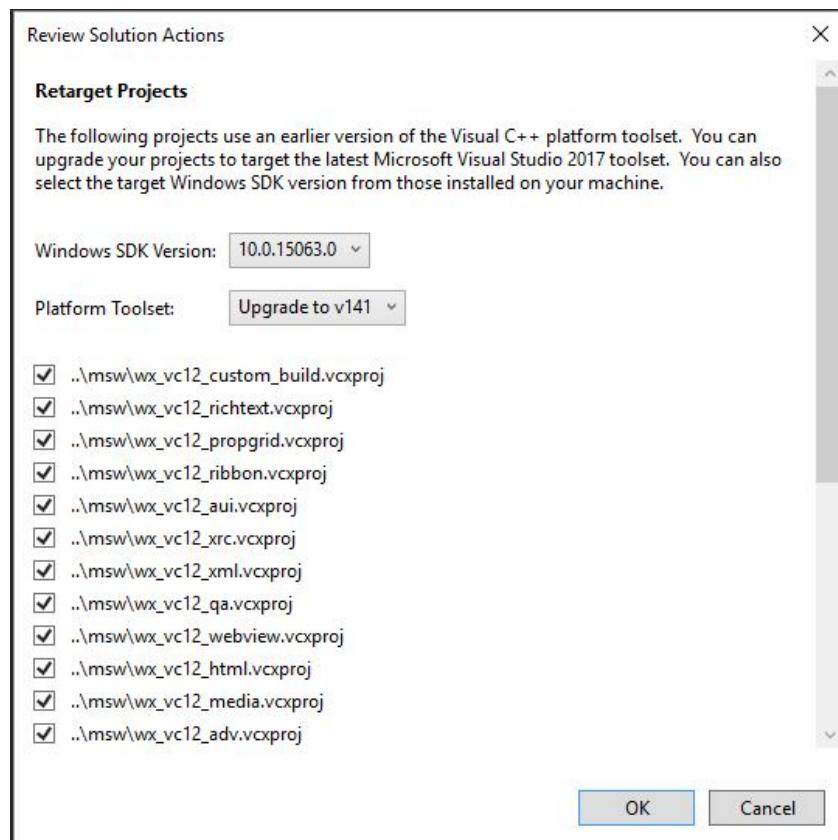
ပုံ ၁.၈: WXWIN environment variable သတ်မှတ်ခြင်း။

အဲဒီနောက် မှာ wxWidgets ကို build လုပ်ဖို့ အတွက် အသင့် ပါရှိတဲ့ C:/wxWidgets303/build/msw ထဲက wx_vc12.sln ဆိုတဲ့ solution (ပုံ ၁.၉) ကို Visual Studio 2017 နဲ့ ဖွင့်လိုက် ပါယယ်။ Retarget Projects ဆိုပြီး upgrade လုပ်ဖို့ မေးလာတဲ့ အခါ OK ကို နှိပ်လိုက် ပါမယ် (ပုံ ၁.၁၀)။ ပြီးတဲ့ အခါ ပုံ ၁.၁၁ မှာ ပြေထား သလို configuration အမျိုးမျိုးကို ရွေးပြီး build အကြိမ်ကြိမ် လုပ်နိုင် ပါတယ်။



ပုံ ၁.၉: wxWidgets ကို build လုပ်ခြင်း။

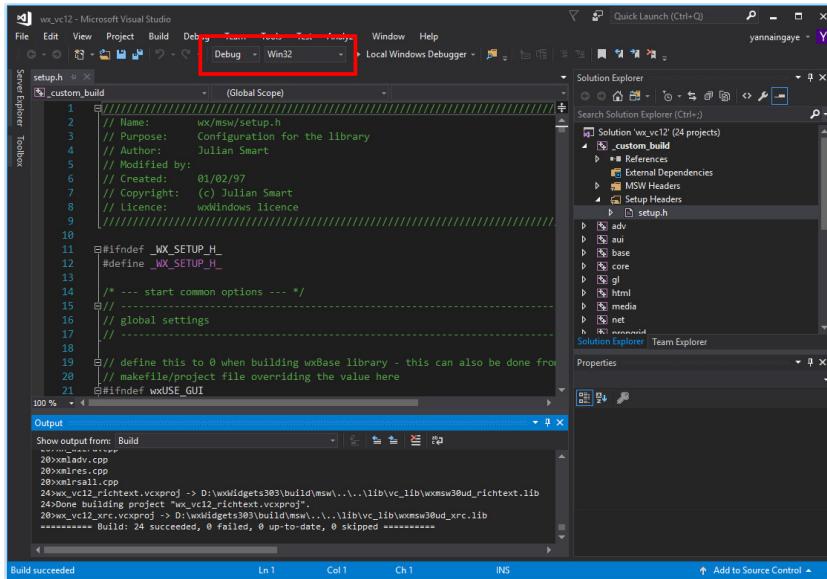
အဲဒီလို build လုပ်ပြီး တဲ့အခါ နမူနာ wxWidgets application တစ်ခု ကို compile လုပ်ပြီး run ကြည့်ပါမယ်။ ဒါကြောင့် C:/wxWidgets303/samples/ ထဲက minimal ဆိုတဲ့ folder ကို minimal2



ဂုဏ်သွင်းရန် Project မှာ upgrade လုပ်ခြင်း။

၁.၁. WINDOWS တွင်တပ်ဆင်ခြင်း

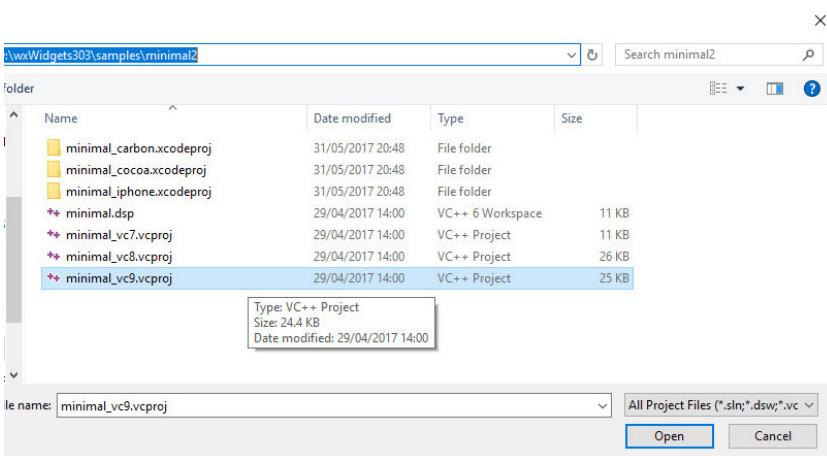
၃



ပုံ ၁.၁၁: Configuration အရှိုးမျိုးဖြင့် build လုပ်ခြင်း။

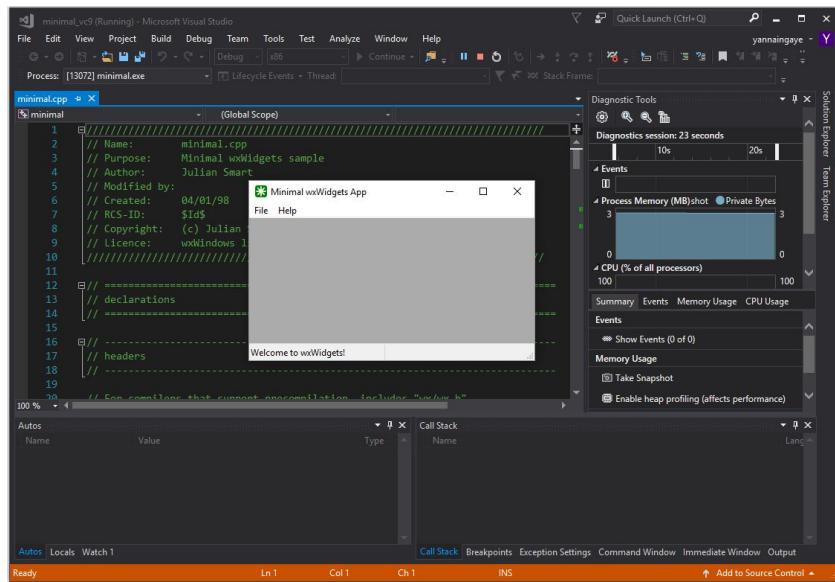
ဆိုတဲ့ နာမည်နဲ့ duplicate လုပ်လိုက် ပါမယ်။

ပြီးတဲ့ အခါ minimal_vc9.vcproj ကို ဖွင့်လိုက် ပါမယ် (ပုံ ၁.၁၂)။ အဲဒါ ကို build လုပ်ပြီး run လိုက်တဲ့ အခါ ပုံ ၁.၁၃ က အတိုင်း တွေ့ရမှာ ဖြစ် ပါတယ်။ အဲဒါမှာ minimal.cpp ဆိုတဲ့ source ကို အမျိုးမျိုး ပြုပြင် စမ်းသပ် ကြည့်နိုင် ပါတယ်။



ပုံ ၁.၁၂: နှမူနာ wxWidgets project တစ်ခုကို စမ်းသပ်ခြင်း။

ပရိုဂရမ် ရဲ့ icon ကို ပြင်ချင်ရင် C:/wxWidgets303/samples/sample.ico ကို ကိုယ်လို ချင်တဲ့



ပုံ ၁.၃၃: wxWidgets minimal project ကို run ခြင်း။

icon နဲ့ အစားထိုးလို့ ရပါတယ်။

၁.J Linux တပ်ဆင်ခြင်း

wxWidgets ကို Linux မှာ ထည့်ဖိုးအတွက် အောက်က စာရင်း ၁.၁ အတိုင်း terminal မှာ command တွေရှိက်ပြီး တပ်ဆင်ထည့်သွေးနိုင် ပါတယ် [wxW14]။

```
1 sudo apt-get install build-essential
2 sudo apt-get install libwxgtk3.0-dev
```

စာရင်း ၁.၁: Linux တွင် wxWidgets ကို တပ်ဆင်ခြင်း။

နဲ့မှန် အနေနဲ့ ရိုးရှင်း တဲ့ wxsimple.cpp (online) ဆိုတဲ့ ပရိုဂရမ် လေးကို စာရင်း ၁.J အတိုင်း ဖန်တီး လိုက် ပါမယ် [Zet13]။

```
1 #include <wx/wx.h>
2 class Simple : public wxFrame
3 {
4 public:
```

```

5     Simple(const wxString& title);
6
7 };
8 Simple::Simple(const wxString& title)
9     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(250, 150))
10 {
11     Centre();
12 }
13
14 class MyApp : public wxApp
15 {
16     public:
17     virtual bool OnInit();
18 };
19
20 IMPLEMENT_APP(MyApp)
21 bool MyApp::OnInit()
22 {
23     Simple *simple = new Simple(wxT("Simple"));
24     simple->Show(true);
25
26     return true;
27 }
```

စာရင်း ၁.၂: wxsimple.cpp

ပြီးတဲ့ အခါ အောက်က command ကို သုံးပြီး compile လုပ်နိုင် ပါတယ်။

```
g++ wxsimple.cpp `wx-config --cxxflags --libs` -o wxsimple
```

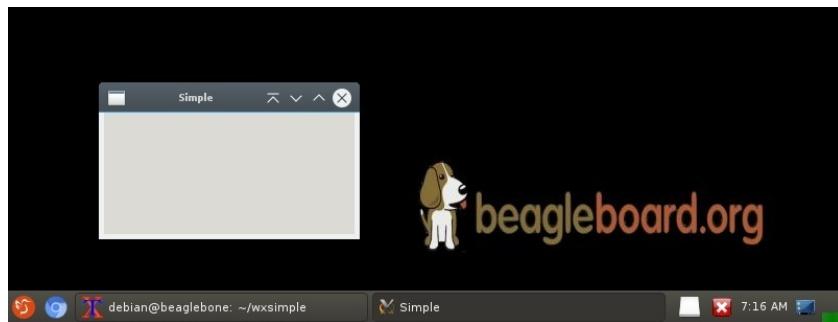
`wx-config --cxxflags` က compile လုပ်ဖို့ အတွက် လိုအပ်တဲ့ flags တွေကို ထူတ်ပေး ပြီး `wx-config --libs` က link လုပ်ဖို့ အတွက် လိုအပ်တဲ့ flags တွေကို ထူတ်ပေး ပါတယ်။

ရှုတဲ့ Terminal က အဆင် မပြု ရင် xterm ကို အောက်ပါ အတိုင်း တပ်ဆင် နိုင် ပါတယ်။

```
sudo apt-get install xterm
```

ပြီးရင် ခုနက ရလာတဲ့ binary ကို အောက်က အတိုင်း run လိုက်တဲ့ အခါ ပုံ ၁.၁၄ အတိုင်း ထွေ့နှင့် ပါတယ်။

```
./wxsimple
```



ပုံ ၁.၁၄: wxWidgets အတွက် wxsimple.cpp နှမူနာ ကို BeagleBone Black, Debian Wheezy တွင် run ခြင်း။

၁.၂.၁ Source မှ build လုပ်ခြင်း

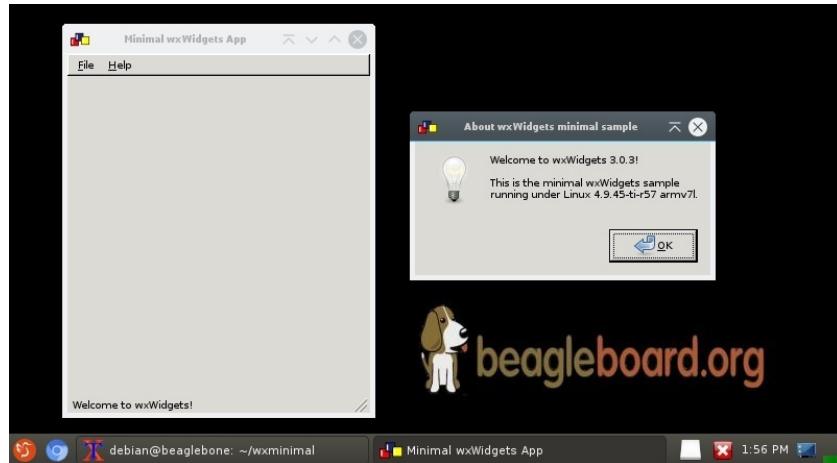
wxWidgets ကို source ကနေ စိတ်ကြိုက် build လုပ်ချင် ရင် အောက်က အတိုင်း build လုပ်ပြီး၊ install လုပ်နှင့် ပါတယ် [wxW14]။

```
sudo apt-get install build-essential
sudo apt-get install libgtk-3-dev
wget https://github.com/wxWidgets/wxWidgets/releases/download/v3.0.3/
    wxWidgets-3.0.3.tar.bz2
tar xjvf wxWidgets-3.0.3.tar.bz2
cd wxWidgets-3.0.3
mkdir gtk-build
cd gtk-build
../configure --enable-unicode --disable-shared --with-gtk=3
make
sudo make install
```

```
wx-config --version
```

Build လုပ်ဖြေတဲ့ wxWidgets ကို သုံးပြီး samples အခန်း ထဲက minimal ဆိုတဲ့ နမူနာကိုအောက်က command တွေနဲ့ run ကြည့် လိုက်တဲ့ အခါ ပုံ ၁.၁၅ အတိုင်း တွေ့ရ ပါတယ်။

```
cd gtk-build/samples/minimal  
make  
../minimal
```



ပုံ ၁.၁၅: wxWidgets ဖြင့် minimal နမူနာကို run ခြင်း။

၁.၂.၂ Code::Blocks ဖြင့်အသုံးပြုခြင်း

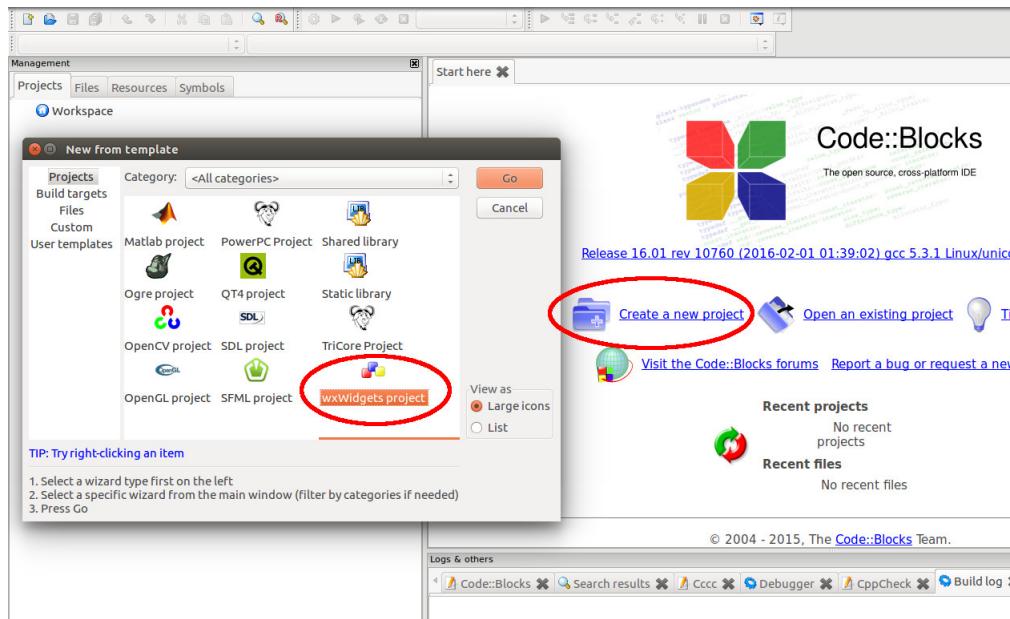
Cross platform GUI application တွေ အတွက် wxWidgets C/C++ ပရိုဂရမ် တွေ ရေးသားဖို့ အတွက် Code::Blocks ဆိုတဲ့ open source လည်းဖြစ်၊ free လည်းပေးတဲ့ IDE နဲ့ တွဲသုံး ကြည့်ပါမယ်။ သူက extension တွေလည်းများ၊ စိတ်ကြိုက် configure လည်း လုပ်နိုင်ပြီး တော်တော် လည်း ခေတ်စား တဲ့ IDE တစ်ခု ပါ။ Code::Blocks IDE ကို တပ်ဆင် ဖို့ အတွက် terminal မှာ စာရင်း ၁.၃ အတိုင်း ရိုက်ထည့်နိုင် ပါတယ်။

```
1 $ sudo apt-get install build-essential  
2 $ sudo apt-get install gdb  
3 $ sudo add-apt-repository ppa:damien-moore/codeblocks-stable
```

```
4 $ sudo apt-get update
5 $ sudo apt-get install codeblocks codeblocks-contrib
```

စာရင်း ၁.၃: Code Blocks တပ်ဆင်ခြင်း။

Code::Blocks ကို တပ်ဆင် ပြီးတဲ့ အခါ သူကို ဖွင့်လိုက်ပြီး ပုံ ၁.၁၆ မှာ ပြထား သလို File menu → New → Project... ဒါမှ မဟုတ် Start here ဆိုတဲ့ tab ထဲက Create a new project ကို နှိပ် လိုက် ပါမယ်။ ပြီးရင် ပေါ်လာတဲ့ ဝင်းချွဲး ထဲက project အမျိုးအစား အတွက် wxWidgets project ကို ရွေးပြီး Go ကို နှိပ်လိုက် ပါမယ်။

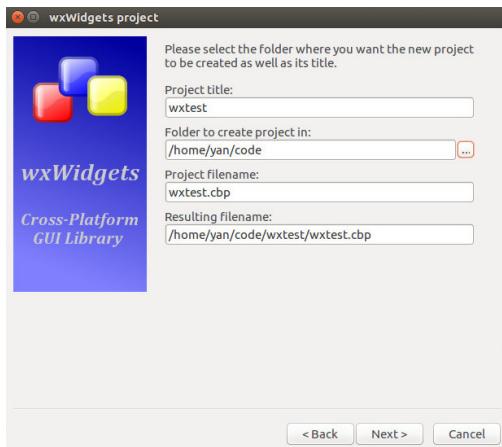


ပုံ ၁.၁၆: Code::Blocks IDE တွင် ပရောဂျက် အသစ် ဖန်တီးမှု။

နောက်ပေါ် လာတဲ့ ဝင်းချွဲးမှာ ပုံ ၁.၁၇ မှာ ပြထားတဲ့ အဆင့် တွေ အတိုင်း wxWidgets version ဥပမာ wxWidgets 3.0.x ကို ရွေးပြီး Next ကို နှိပ်နိုင် ပါတယ်။ ပြီးတဲ့ အခါ ပရောဂျက် နာမည် နဲ့ သိမ်းမည့် နေရာ ကို သတ်မှတ် နိုင် ပါတယ်။ နောက်တစ်ခါ GUI builder ကို None ရွေးပြီး Project အမျိုးအစား ကို Frame Based ရွေးလိုက် ပါမယ်။ ဆက်လက်ပြီး GNU GCC Compiler ကို ရွေးပြီး Default wxWidgets configuration သုံးဖို့ သတ်မှတ်ပြီး တဲ့ အခါ Finish ကို နှိပ်လိုက် ပါမယ်။

၁.၂. LINUX တွင်တပ်ဆင်ခြင်း

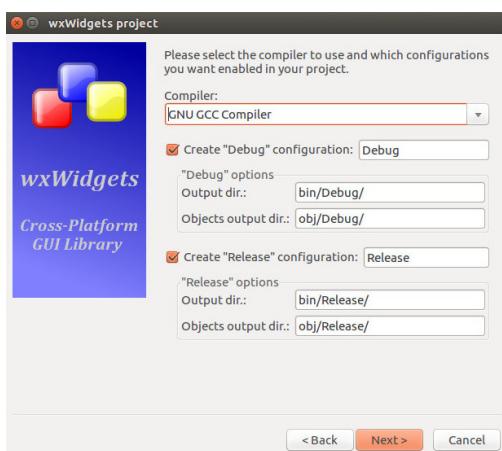
၁၅



(a) နာမည်နှင့်နေရာ သတ်မှတ်ခြင်း။



(b) Application အမျိုးအစား။



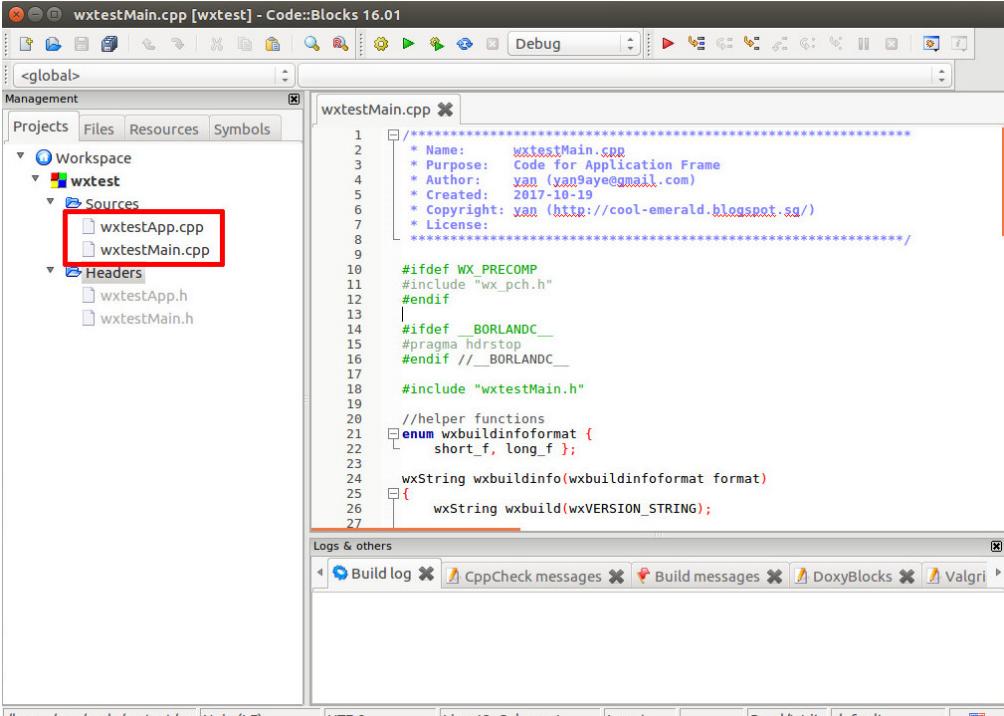
(c) Compiler သတ်မှတ်ခြင်း။



(d) Configuration သတ်မှတ်ခြင်း။

ပုံ ၁.၁၇: Code::Blocks တွင် wxWidgets အသုံးပြုခြင်း အဆင့်ဆင့်။

အဲဒီလို Project ကို ဖန်တီးပြီးတဲ့ အခါ ပုံ ၁.၁၈ မှာလို ညာဘက် project explorer ဝင်းခွဲး ထဲက wxtestMain.cpp | wxtestApp.cpp စတဲ့ source တွေကို အမျိုးမျိုး ပြုပြင် စမ်းသပ် ကြည့်နိုင် ပါတယ်။ ပြီးတဲ့ အခါ F9 ခလုပ် နှင့်ပြီး Build and run လုပ်လိုက်ရင် ပုံ ၁.၁၉ မှာ ပြထား တဲ့ အတိုင်း ပရိုဂရမ် ရဲ့ GUI output ကို တွေ့နိုင် ပါတယ်။



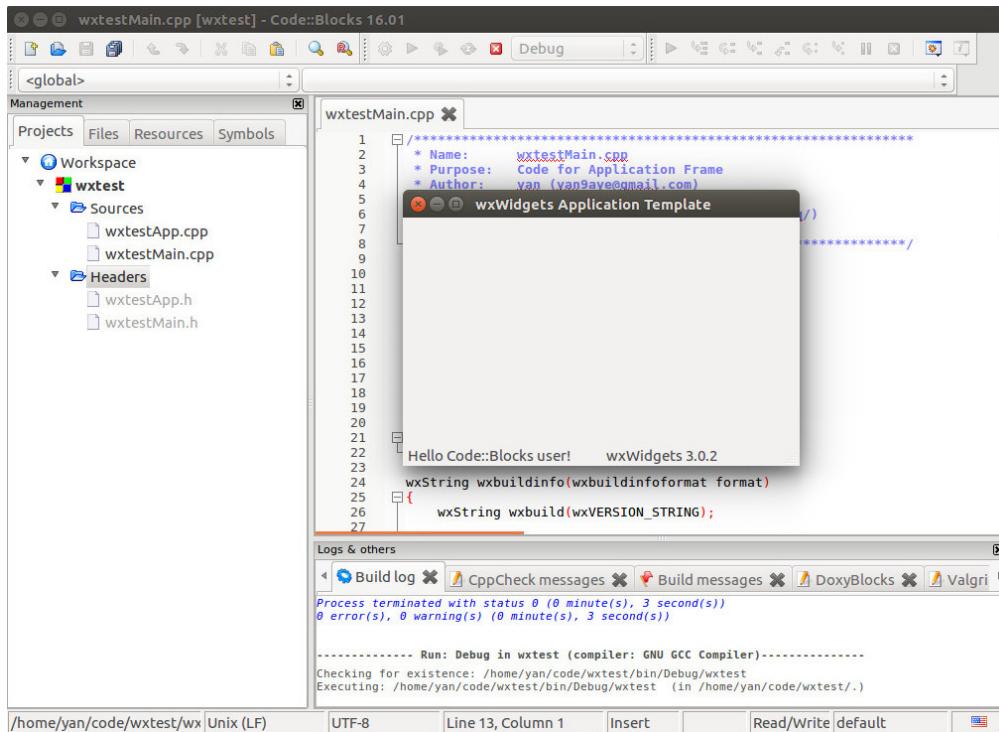
The screenshot shows the Code::Blocks IDE interface. The title bar says "wxtestMain.cpp [wxtest] - Code::Blocks 16.01". The left pane shows a project structure for "wxtest" with "Sources" containing "wxtestApp.cpp" and "wxtestMain.cpp", which is highlighted with a red rectangle. The right pane displays the content of "wxtestMain.cpp". The code includes comments at the top and defines helper functions like `wxString wxbuildinfo(wxbuildinfoformat format)` and `wxString wxbuild(wxVERSION_STRING);`. The bottom status bar shows the file path "/home/yan/code/wxtest/wx", encoding "Unix (LF)", and line/column information "Line 13, Column 1".

```

1  ****
2  * Name:      wxtestMain.cpp
3  * Purpose:   Code for Application Frame
4  * Author:    yan (yan9aye@gmail.com)
5  * Created:   2017-10-19
6  * Copyright: yan (http://cool-emerald.blogspot.sg/)
7  * License:
8
9
10 #ifndef WX_PRECOMP
11 #include "wx_pch.h"
12 #endif
13
14 #ifdef __BORLANDC__
15 #pragma hdrstop
16 #endif // __BORLANDC__
17
18 #include "wxtestMain.h"
19
20 //helper functions
21 enum wxbuildinfoformat {
22     short_f, long_f };
23
24 wxString wxbuildinfo(wxbuildinfoformat format)
25 {
26     wxString wxbuild(wxVERSION_STRING);
27

```

ပုံ ၁.၁၈: wxWidgets source နမူနာအား စမ်းသပ်ပြုပြင်ခြင်း။



ပုံ ၁.၁၉: wxWidgets နမူနာ ပရိုက်ရမ်း GUI output ။

၁.၄ Mac တွင်တပ်ဆင်ခြင်း

ပထာမဆုံး Mac Ports ကို အရင် install လုပ်ပါမယ်။ နောက် တစ်ခါ အောက်က ကွန်မန်း ကို သုံးပြီး wxWidgets ကို install လုပ်ပါတယ်။

```

port search wxWidgets
sudo port -v install wxWidgets-3.0
wx-config

```

အဲဒါတွေ ပြီးတော့ CodeLite ကို install လုပ်ပါတယ်။ wxWidgets code တွေကို ဖုန်မှန်ကန်ကန် compile လုပ်ဖို့ CodeLite ရဲ့ Compiler settings တွေကို ပြင်ဖို့ လိုပါတယ်။ အဲဒီလို ပြင်ဖို့ Settings menu → Build Settings command ကိုသွားပါ။ Bulid Settings ဆိုတဲ့ ဝင်းဒီးပေါ်လာရင် gnu g++ → Tools ဆိုတဲ့ အပိုင်းက PATH environment variable: ဆိုတဲ့ field မှာ အောက်က အတိုင်း ပြင်ပါတယ်။

```
/opt/local/bin:/usr/local/bin:/usr/bin:/bin
```

နောက်တခါ build လုပ်လို့ ထွက်လာတဲ့ output အမျိုးအစားကို မှန်ဖို့ CodeLite IDE ရဲ့ Workspace windows ထဲက project folder ကို right click နှိပ်ပြီး Make this project output a bundle command ကို နှိပ်ပါမယ်။ နောက် Debug နဲ့ Release ဆိုတဲ့ targets နှစ်ခု စလုံးကို မှတ်နိုင်ပါတယ်။

အဲဒီနေရာမှာပဲ ကိုပဲ ဆော်ဖဲ ရဲ့ icon file ကို သတ်မှတ်ပေး နိုင်ပါတယ်။ အဲဒီ icon ဖိုင် က icns format ဖြစ်ဖို့ လိုပါတယ်။ icns extension နဲ့ icon ဖန်တီးချင်ရင် App store မှာ free ရှိနိုင်တဲ့ Img2icns ဆိုတဲ့ ဆော်ဖဲ ကို သုံးနိုင်ပါတယ်။

အကိုးအကားများ

- [Cod13] CodeLite. Download CodeLite bundled with MinGW and wxWidgets. 2013. url: <https://sourceforge.net/projects/codelite/files/Releases/codelite-5.0/codelite-5.0.6213-mingw4.7.1-wx2.9.4.exe/download>.
- [Cod17] CodeLite Wiki. Compiling wxWidgets with MinGW. 2017. url: <http://codelite.org/Developers/BuildingWxWidgetsWin>.
- [Qt17] Qt. Licensing - Before you begin, make the right license choice. 2017. url: <https://www1.qt.io/licensing/>.
- [Zet13] ZetCode. wxWidgets tutorial. 2013. url: <http://zetcode.com/gui/wxwidgets/>.
- [wxW12] wxWiki. WxWidgets Compared To Other Toolkits. 2012. url: https://wiki.wxwidgets.org/WxWidgets_Compared_To_Other_Toolkits.
- [wxW14] wxWiki. Compiling and getting started. 2014. url: https://wiki.wxwidgets.org/Compiling_and_getting_started.
- [wxW15a] wxWiki. Bindings. 2015. url: <https://wiki.wxwidgets.org/Bindings>.
- [wxW15b] wxWiki. Microsoft Visual C++ Guide. 2015. url: https://wiki.wxwidgets.org/Microsoft_Visual_C%2B%2B_Guide.

[wxW98] wxWidgets. wxWindows Library Licence. 1998. url: <https://www.wxwidgets.org/about/licence/>.

အခန်း J

OpenCV နှင့်တဲ့သုံးခြင်း

Image processing နဲ့ computer vision တွေ လုပ်ဆောင်ဖို့ အတွက် ဆိုရင် OpenCV (<https://opencv.org/>) က လူသုံး များပါ တယ်။ ဒါ အခန်း မှာ OpenCV နဲ့ wxWidgets တဲ့ သုံးတဲ့ အကြောင်း ဆွေးနွေး ပါမယ်။

J.၁ Linux

OpenCV ကို Linux မှာ ရယူ တပ်ဆင်ဖို့ အတွက် လွယ်ကူ ရှိုးရှင်း တဲ့ နည်းလမ်း တစ်ခု အနေနဲ့ အောက်က အတိုင်း terminal မှာ ရိုက်ယူ နိုင်ပါတယ် [Eme17a]။

```
$ sudo apt-get install build-essential  
$ sudo apt-get install cmake git libgtk-3-dev pkg-config libavcodec-dev  
    libavformat-dev libswscale-dev  
$ sudo apt-get install libopencv-dev
```

Linux နဲ့ terminal ပေါ်မှာ command ရိုက်ထည့် ပြီး build လုပ်တာ လွယ်ကူ ရှိုးရှင်း ပါတယ်။ နှမူနာ အနေနဲ့ wxcvssimple.cpp (online) ဆိုတဲ့ ရှိုးရှင်း တဲ့ ပရိုဂရမ် လေး တစ်ခု ရေးကြည့် ပါမယ်။ ပရိုဂရမ် wxcvssimple.cpp ကို စာရင်း [J.၁](#) မှာ တွေ့နိုင် ပါတယ်။

```
1 //File: wxcvssimple.cpp  
2 //Description: A simple example to use OpenCV with wxWidgets  
3 //Author: Yan Naing Aye
```

```
4 //Date: 2017 November 01
5 //MIT License - Copyright (c) 2017 Yan Naing Aye
6
7 #include <wx/wx.h>
8 #include <opencv2/opencv.hpp>
9 #include "util.h"
10 using namespace cv;
11
12 class MyFrame : public wxFrame
13 {
14     wxStaticBitmap *lena;
15     wxStaticBitmap *thiri;
16 public:
17     MyFrame(const wxString& title);
18
19 };
20 MyFrame::MyFrame(const wxString& title)
21     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(512, 256))
22 {
23     Centre();
24     lena = new wxStaticBitmap(this, wxID_ANY, wxBitmap(wxT("lena.jpg"),
25                                         wxBITMAP_TYPE_JPEG), wxPoint(0,0), wxSize(256, 256));
26     thiri = new wxStaticBitmap(this, wxID_ANY, wxBitmap(wxT("lena.jpg"),
27                                         wxBITMAP_TYPE_JPEG), wxPoint(256,0), wxSize(256, 256));
28
29     //From opencv to wx
30     Mat imcv1=imread("thiri.jpg", CV_LOAD_IMAGE_COLOR);
31     wxBitmap imwx1=wx_from_mat(imcv1);
32     thiri->SetBitmap(imwx1);
33
34     //From wx to opencv
35     wxImage imwx2;
36     imwx2.LoadFile(wxT("lena.jpg"), wxBITMAP_TYPE_JPEG);
37     Mat imcv2=mat_from_wx(imwx2);
38     namedWindow("Img", CV_WINDOW_AUTOSIZE);
39     imshow("Img", imcv2);
```

```

38 }
39
40 class MyApp : public wxApp
41 {
42     public:
43         virtual bool OnInit();
44     };
45
46 IMPLEMENT_APP(MyApp)
47 bool MyApp::OnInit()
48 {
49     if ( !wxApp::OnInit() )
50         return false;
51     wxInitAllImageHandlers();
52     MyFrame *frame = new MyFrame(wxT("Simple wxWidgets + OpenCV"));
53     frame->Show(true);
54
55     return true;
56 }
```

စာရင်း J.C: wxcvssimple.cpp

ပရိုဂရမ် အစမှာ Application ရဲ့ OnInit() ဆိုတဲ့ method ထဲမှာ

```
wxInitAllImageHandlers();
```

ဆိုတာကို ထည့်ပါမယ်။ အဲဒီနောက် MyFrame ဆိုတဲ့ wxFrame ရဲ့ derived class ထဲမှာ ပုံရိပ် တွေကို ဖော်ပြန့် wxStaticBitmap variable တွေကို ကြော်လိုက် ပါမယ်။ MyFrame ရဲ့ constructor မှာ wxStaticBitmap တွေကို ဖန်တီးဖို့ အောက်က ကုဒ် ကို သုံးနိုင် ပါတယ်။

```
lena = new wxStaticBitmap(this,wxID_ANY,wxBitmap(wxT("lena.jpg"),
    wxBITMAP_TYPE_JPEG),wxPoint(0,0),wxSize(256, 256));
```

Image တွေ အတွက် OpenCV မှာသုံးတဲ့ Mat နဲ့ wxWidgets ရဲ့ wxImage ကို အပြန် အလှန် ပြောင်း

ပေးတဲ့ mat_from_wx | wx_from_mat အခါ ရှိတဲ့ ဖန်ရှင် တွက် စာရင်း J.J မှာ ဖော်ပြ ထားတဲ့ util.h ထဲမှာ တွေ့နိုင် ပါတယ [Dad10]!!

```

1 //File: util.h
2 //Description: Functions to convert wxImage and OpenCV Mat
3 //Author: Yan Naing Aye
4 //MIT License - Copyright (c) 2017 Yan Naing Aye
5
6 #include <wx/wx.h>
7 #include <opencv2/opencv.hpp>
8 using namespace cv;
9
10 wxImage wx_from_mat(Mat &img) {
11     Mat im2;
12     if(img.channels()==1){cvtColor(img,im2,COLOR_GRAY2RGB);}
13     else if (img.channels() == 4) { cvtColor(img, im2, COLOR_BGRA2RGB);}
14     else {cvtColor(img,im2,COLOR_BGR2RGB);}
15     long imsize = im2.rows*im2.cols*im2.channels();
16     wxImage wx(im2.cols, im2.rows,(unsigned char*)malloc(imsize), false);
17     unsigned char* s=im2.data;
18     unsigned char* d=wx.GetData();
19     for (long i = 0; i < imsize; i++) { d[i] = s[i];}
20     return wx;
21 }
22
23 Mat mat_from_wx(wxImage &wx) {
24     Mat im2(Size(wx.GetWidth(),wx.GetHeight()),CV_8UC3,wx.GetData());
25     cvtColor(im2,im2,COLOR_RGB2BGR);
26     return im2;
27 }
```

စာရင်း J.J: util.h

ဆက်ပြီး MyFrame ရဲ constructor ထဲမှာ သီရိ ရဲ ပုံကို OpenCV သံဃ္ပီး imread နဲ့ ဖတ်လိုက် ပါတယ်။ ဖတ်လို ရတဲ့ Mat အမျိုး အစား ပုံကို wx_from_mat ဖန်ရှင် ကို သံဃ္ပီး wxImage အမျိုး

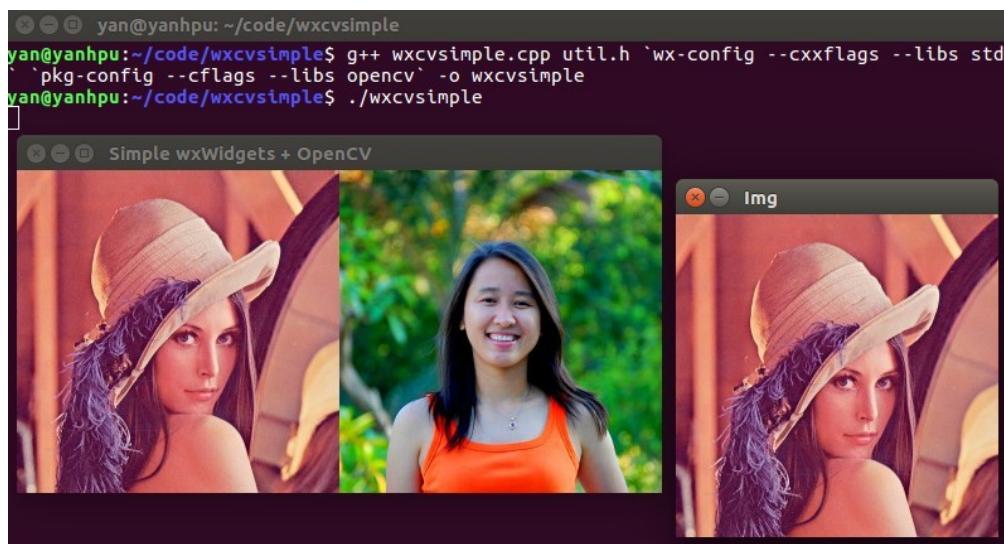
အစား ပြောင်းလိုက် ပါတယ်။ ပြီးတော့ wxStaticBitmap ရဲ့ SetBitmap method သုံးပြီး Frame ပေါ်မှာ ဖော်ပြု လိုက် ပါတယ်။

နောက်တစ်ခါ wxImage အမျိုး အစား variable တစ်ခု ကြော်ပြု ပြီး LoadFile method နဲ့ လိုနာ ရဲ့ ပုံကို wxWidget သုံးပြီး ဖတ်လိုက် ပါတယ်။ ရလာ တဲ့ wxImage အမျိုး အစား ပုံကို mat_from_wx ဆိုတဲ့ ဖန်ရှင် သုံးပြီး OpenCV အတွက် Mat အမျိုး အစား ပြောင်းလိုက် ပါတယ်။ ရလာတဲ့ ပုံရှင်ပို့ OpenCV ရဲ့ imshow နဲ့ ဖော်ပြု လိုက် ပါတယ်။

ပရီ ဂရမ် ကို build လုပ်ဖို့ အတွက် terminal မှာ အောက်က command ကို သုံးနိုင် ပါတယ်။

```
$ g++ wxcvssimple.cpp util.h `wx-config --cxxflags --libs std` `pkg-config --cflags --libs opencv` -o wxcvssimple
```

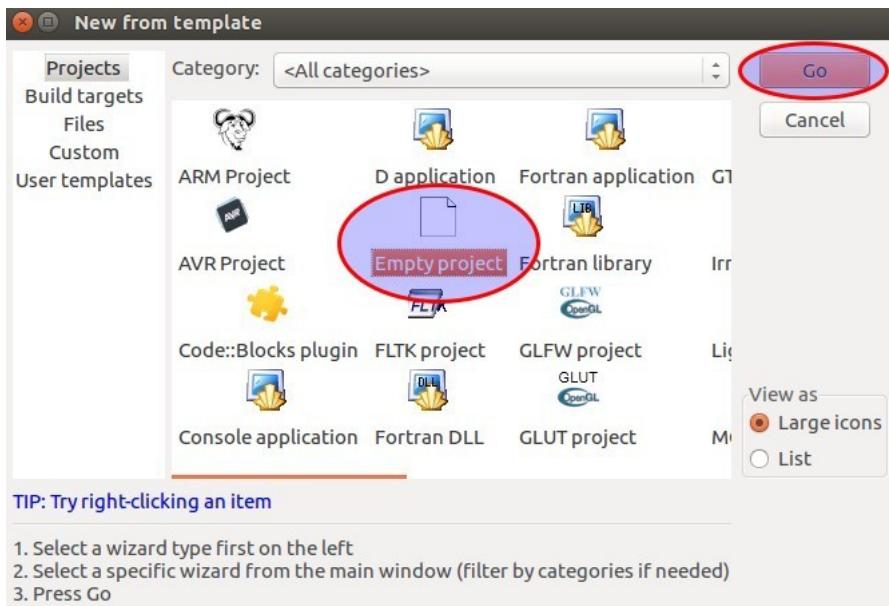
./wxcvssimple ကို ရှိကိုပြီး run လိုက်တဲ့ အခါ ရလာတဲ့ ရလဒ် ကို ပုံ J.၁ မှာ တွေ့နိုင် ပါတယ်။



ပုံ J.၁: wxWidgets နှင့် OpenCV ကို တွဲစပ် အသုံးပြု ထားသည့် ရှိုးရှင်းသော နမူနာ။

J.၁.၁ Code::Blocks

Code::Blocks မှာ wxWidgets ကို OpenCV နဲ့ တွဲသုံး ဖို့အတွက် wxWidgets project အသစ် တစ်ခု ကို ပုံ J.၂ မှာ ဖော်ပြု ထားတဲ့ အတိုင်း empty project တစ်ခု ဖန်တီးပြီး wxcv ဆိုတဲ့ project နာမည် ပေးလိုက် ပါမယ်။ ပြီးတဲ့ အခါ GNU GCC Compiler ကို ရွေး ပါမယ်။



ဗုဒ္ဓန: Code::Blocks စွဲငံး Empty project တစ်ခု ဖန်တီးခြင်း။

File Menu → New → Empty File ကို နိပ်ပြီး စာရင်း J.R မှာ ပြထားတဲ့ wxcv.cpp ဖိုင်ကို ဖန်တီးလိုက် ပါမယ်။

```

1
2 #include <wx/wx.h>
3 #include <opencv2/opencv.hpp>
4 using namespace cv;
5
6 class Simple : public wxFrame
7 {
8 public:
9     Simple(const wxString& title);
10
11 };
12 Simple::Simple(const wxString& title)
13     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(250, 150))
14 {
15     Centre();
16     Mat img=imread("./grove.jpg",CV_LOAD_IMAGE_COLOR);

```

J6

အခန်း J. OPENCV နှင့်တဲ့သုံးခြင်း

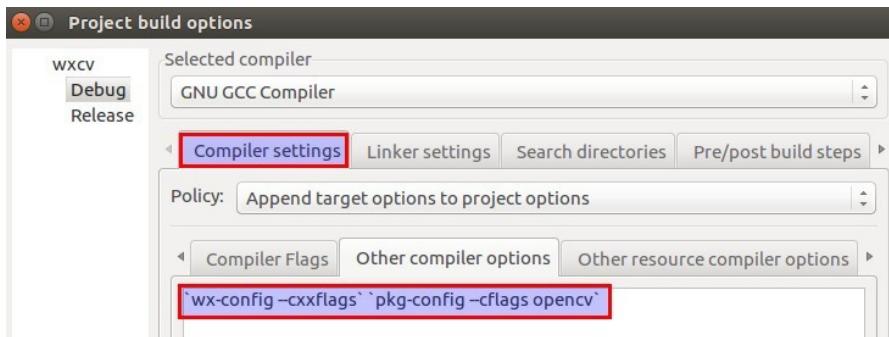
```
17     namedWindow("Img", CV_WINDOW_AUTOSIZE);
18     imshow("Img", img);
19 }
20
21 class MyApp : public wxApp
22 {
23     public:
24         virtual bool OnInit();
25 };
26
27 IMPLEMENT_APP(MyApp)
28 bool MyApp::OnInit()
29 {
30     Simple *simple = new Simple(wxT("Simple"));
31     simple->Show(true);
32
33     return true;
34 }
```

စာရင်း J.2: wxcv.cpp

ပြီးရင် Project Menu → Build Options... ကို နှိပ်ပြီး Compiler settings tab → Other compiler options မှာ ပုံ ပါ၏ အတိုင်း

```
`wx-config --cxxflags` `pkg-config --cflags opencv`
```

ကို Debug အတွက်ရော့၊ Release အတွက်ပါ သတ်မှတ် နှင့် ပါတယ်။

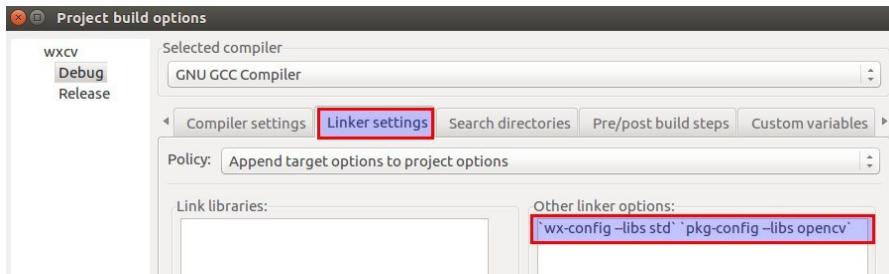


ጀ J.২: Compiler options মূৰাঃবাৰ্তাৰ্থে প্ৰিয়েন্দ্ৰিয়া।

ফেৱাৰ্কতাৰ্থে Linker settings tab → Other linker options মুৰা লভ্য়: ጀ J.৩ আটীড়ি়ে:

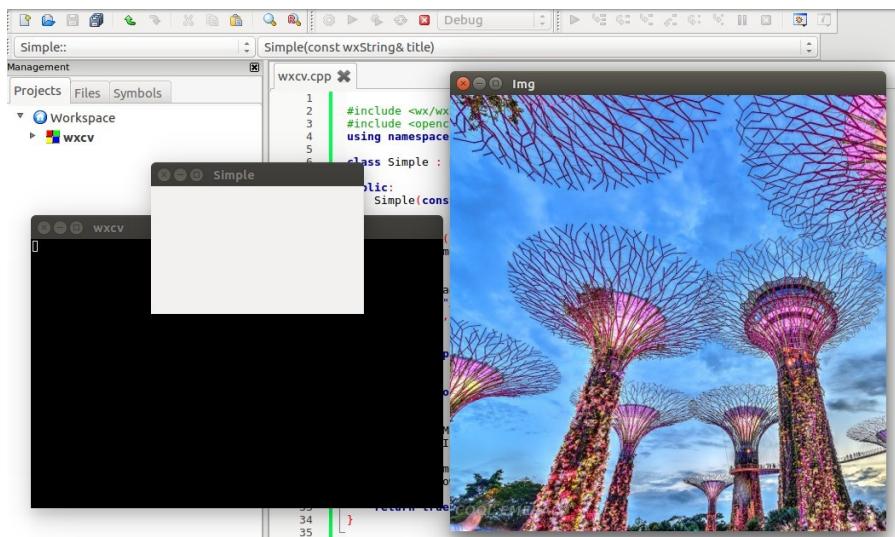
```
'wx-config --libs std` `pkg-config --libs opencv'
```

কো দৰ্শন বাৰ্তাৰ্থে প্ৰিয়েন্দ্ৰিয়া।



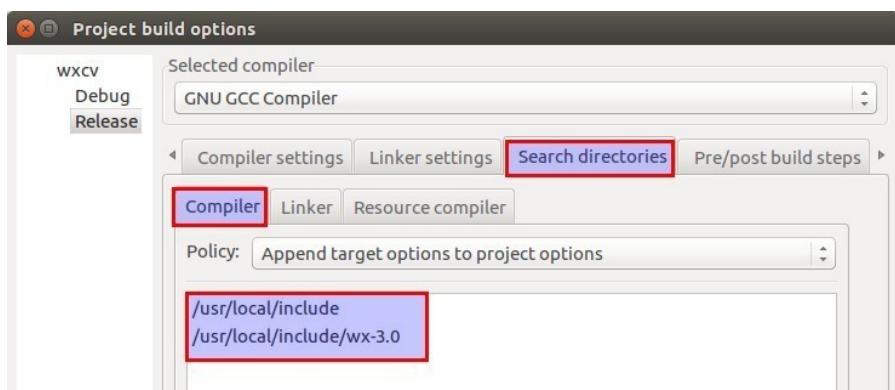
ጀ J.৪: Linker options মূৰাঃবাৰ্তাৰ্থে প্ৰিয়েন্দ্ৰিয়া।

ফেৱাৰ্ক এবং F9 এলডুনকী কুইিপ্ৰি: Build and run লুভলুভি তোআৱি ጀ J.৫ মুৰা প্ৰিয়েন্দ্ৰিয়া বলৈ পৰাইৰুৱা কৈছে।



ပုံ J.၅: OpenCV နမူနာ ပရီဂရမ်ကို Code::Blocks တွင် run ခြင်း။

ပရောဂျက် နဲ့ ပတ်သက် ထဲ အချက်အလက် တွေကို IDE ကို ပိုမြီး သိစေချင်ရင် optional အဆင့်တွေ အနေဖြင့် Project Menu → Build Options... ကို နိုင်ပြီး Search directories tab → Compiler tab မှာ ပုံ J.၆ အတိုင်း /usr/local/include နဲ့ /usr/local/include/wx-3.0 ကို Debug အတွက်ရော၊ Release အတွက်ပါ သတ်မှတ် နိုင် ပါတယ်။

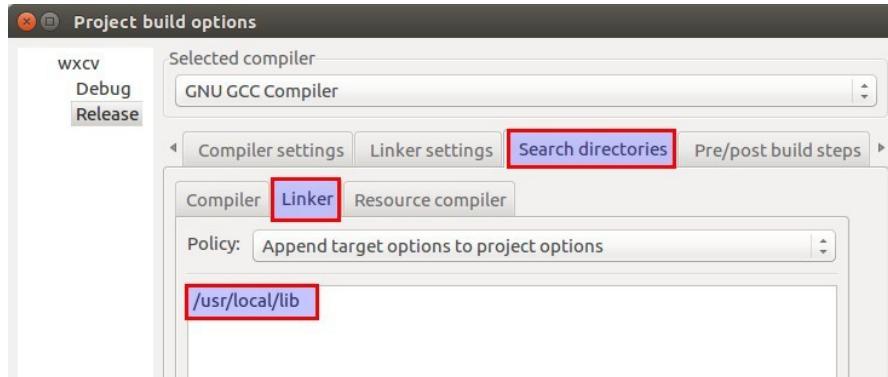


ပုံ J.၆: Compiler Search directories များသတ်မှတ်ခြင်း။

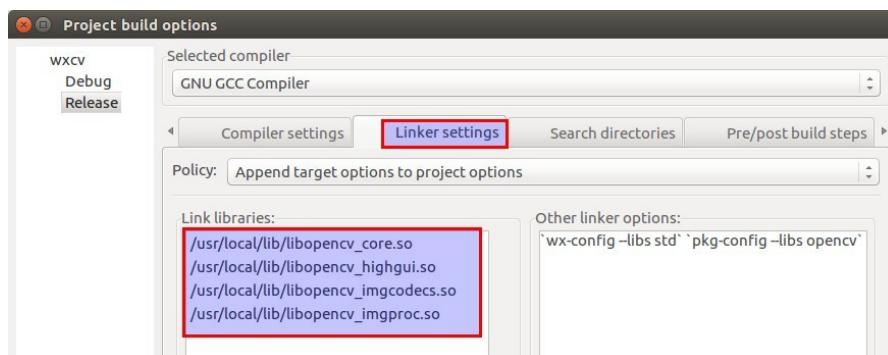
နောက်တစ်ခါ ဘေးဘက် က Linker tab မှာ ပုံ J.၇ အတိုင်း lib တွေရဲ့ လမ်းကြောင်း /usr/local/lib ကို သတ်မှတ် ပါမယ်။ Linker settings tab မှာလည်း ပုံ J.၈ အတိုင်း Link libraries တွေ ဖြစ်တဲ့

```
libopencv_core.so
libopencv_highgui.so
libopencv_imgcodecs.so
libopencv_imgproc.so
```

အေဒ ရှိတဲ့ library တွေကို လိုသလို သတ်မှတ် နိုင် ပါတယ်။



ံ J.ဂ: Library နှင့်များအတွက် Search directories များသတ်မှတ်ခြင်း။



ံ J.၈: Link libraries များသတ်မှတ်ခြင်း။

Search directories တွေနဲ့ပတ်သက် တဲ့ အချက်အလက် တွေကို အောက်က command တွေကို terminal မှာ ရှိက် ကြည့်ပြီး လည်း စစ်ဆေး သိရှိ နိုင် ပါတယ်။

```
$ wx-config --cxxflags
$ pkg-config --cflags opencv
$ wx-config --libs std
```

```
$ pkg-config --libs opencv
```

J.J Windows

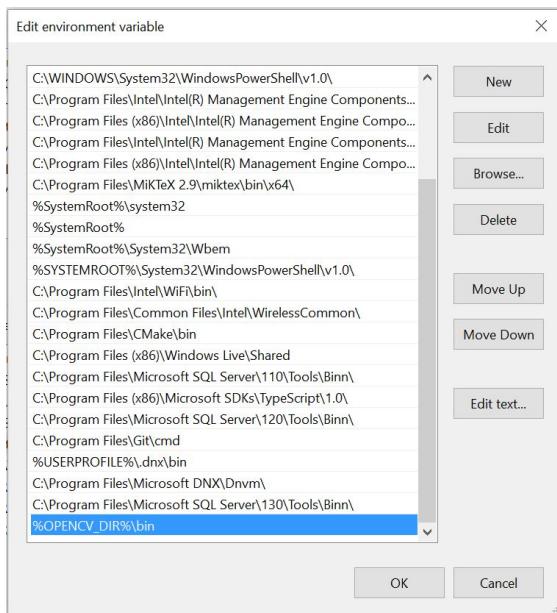
Windows အတွက် pre-built လုပ် ထားတဲ့ လက်ရှိ နောက်ဆုံးထွက် OpenCV အခြေကျ ဗာရုင်း ကို
တပ်ဆင် ပါမယ် [Eme17b]။ အဲဒါကို OpenCV for Windows <https://opencv.org/> မှာ ယူနိုင်
ပါတယ်။ ရလာတဲ့ opencv-3.3.0-vc14.exe ကို ဒီ နမူနာ မှာ C:\opencv အောက်မှာ opencv330
အနေနဲ့ extract လုပ်လိုက် ပါတယ်။

64 bit windows နဲ့ Visual Studio အတွက် environment variable တစ်ခု အနေနှင့် OPENCV_DIR
ကို သတ်မှတ် ပါမယ်။ အဲဒါ အတွက် Command window ကို administrator အနေနဲ့ run ဖို့ start
menu ကို right click နိမ့်ပြီး command prompt (admin) ကို ရွေးပြီး ရင် အောက်ပါ စာရင်း J.၄
အတိုင်း ရိုက်ထည့် နိုင် ပါတယ်။

```
1 > setx -m OPENCV_DIR C:\opencv\opencv330\build\x64\vc14
```

စာရင်း J.၄: OPENCV_DIR ကို သတ်မှတ် ခြင်း။

DLL တွေကို သုံးတဲ့ အခါမှာ bin folder ရှိတဲ့ နေရာကို system path မှာ ထည့်ဖို့ အတွက် Rapid
Environment Editor စတု utility တွေကို သုံးတာ ဖြစ်ဖြစ်၊ ဒါမှ မဟုတ် This PC(My Computer) →
Properties → Advanced System Settings → Advanced Tab → Environment variables → Edit
System Variables မှာ Path ကို ပဲ J.၅ မှာ ပြထား သလို edit လုပ်တာ ဖြစ်ဖြစ်၊ တနည်းနည်း သုံးပြီး
တော့ %OPENCV_DIR%\bin ကို system path အနေနဲ့ ထည့်နိုင် ပါတယ်။

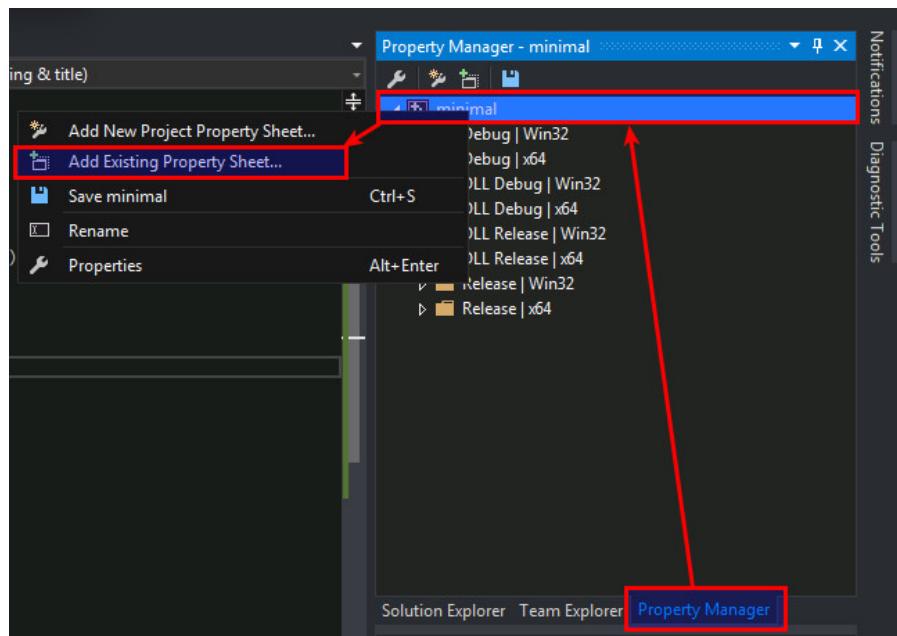


ဦး J.၉: Environment variable အသစ်ထည့်ခွင်း။

Windows ပေါ်မှာ wxWidgets နဲ့ OpenCV ကို တွေ့သုံးဖို့ အတွက် အပိုင်း ၁.၁.၂ မှာ ဖော်ပြ ခဲ့သလို C:/wxWidgets303/samples/ ထဲက minimal ဆိုတဲ့ folder ကို minimalcv ဆိုတဲ့ နာမည်နဲ့ duplicate လုပ်လိုက် ပါမယ်။ ပြီးတဲ့ အခါ minimal_vc9.vcproj ကို ဖွင့်လိုက် ပါမယ်။ အောက်ပါ link မှာရှိတဲ့ OpenCV320.props ကို ယူပြီး ပရောဂျက် အခန်း ထဲမှာ ကူးထည့် လိုက် ပါမယ်။

https://github.com/yan9a/OpenCV_Projects/blob/master/img_wins_vs2015_x64/img_wins/OpenCV320.props

အပေါ် menu bar မှာရှိတဲ့ View → Other Windows → Property Manager ကိုသွားဖွင့် ပြီးတဲ့ အခါ ပုံ J.၁၀ မှာ ပြထား သလို ညာဘက် Property window ထဲက minimal ပေါ်မှာ ညာဘက် ကလစ် နှုပ်ပြီး Add existing property sheet ကို ရွေး၊ OpenCV320.props ကို ရွေးထည့် လိုက် ပါမယ်။



ပုံ J.၁၀: OpenCV Property Sheet ကို ထည့်ခြင်း။

အပေါ်က Solution Platforms ဆိုတဲ့ drop down list မှာ x64 ကို ရွေးထား လိုက် ပါမယ်။ Solution Explorer ထဲက Source Files ပေါ်မှာ ညာဘက် ကလစ် နှင့်ပြီး Add Existing Item ... ကို ရွေးပြီး စာရင်း J.J မှာ ဖော်ပြ ထားတဲ့ util.h ကိုလည်း ထည့်ထား လိုက် ပါမယ်။ ပြီးတဲ့ အခါ minimal.cpp ကို စာရင်း J.၅ မှာ ပြထား သလို ပြင်လိုက် ပါမယ်။ အဲဒီနောက် ပရိုဂရမ် ကို run လိုက်ရင် ပုံ J.၁၁ မှာ ပြထား သလို OpenCV နဲ့ process လုပ်ထား တဲ့ ပုံရှိပိုက် wxWidgets ရဲ့ frame ပေါ်မှာ ဖော်ပြ ထားတဲ့ output ကို ထွေ့နိုင် ပါတယ်။

```

1 //File: minimalcv.cpp
2 //Description: A simple example to use OpenCV with wxWidgets
3 //Author: Yan Naing Aye
4 //Date: 2017 November 07
5 //MIT License - Copyright (c) 2017 Yan Naing Aye
6
7 #include <wx/wx.h>
8 #include <opencv2/opencv.hpp>
9 #include "util.h"
10 #include <string>
11 using namespace std;

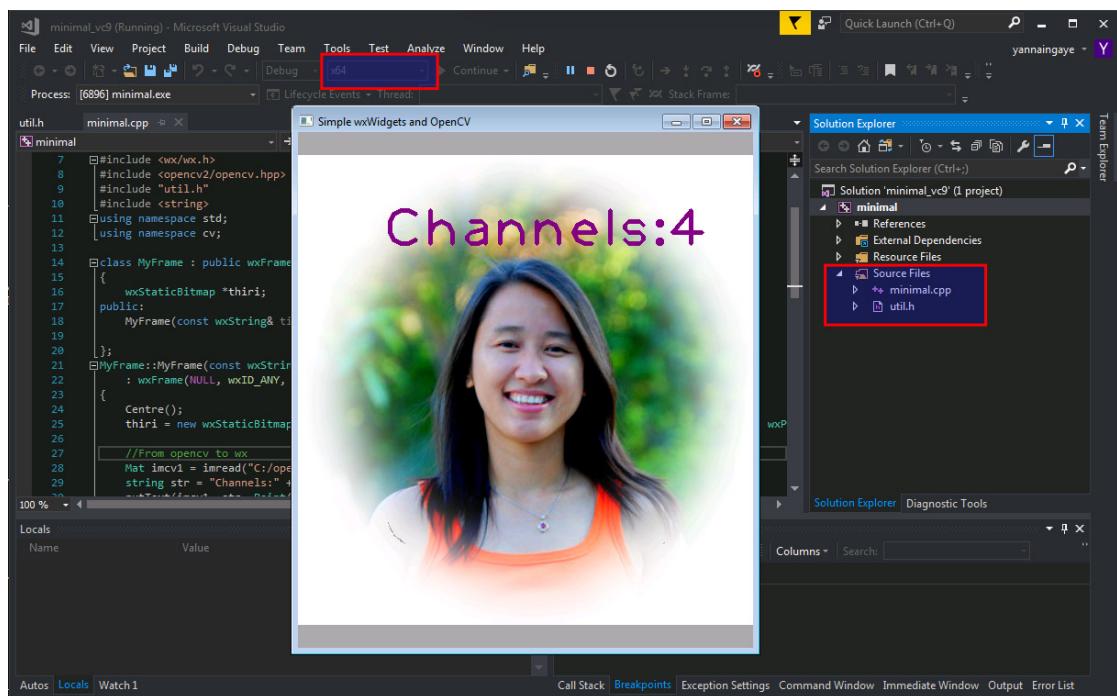
```

```
12 using namespace cv;
13
14 class MyFrame : public wxFrame
15 {
16     wxStaticBitmap *thiri;
17 public:
18     MyFrame(const wxString& title);
19
20 };
21 MyFrame::MyFrame(const wxString& title)
22     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(512, 600))
23 {
24     Centre();
25     thiri = new wxStaticBitmap(this, wxID_ANY, wxBitmap(wxT("./thiri.png"),
26                                         wxBITMAP_TYPE_PNG), wxPoint(256, 0), wxSize(512,512));
27
28     //From opencv to wx
29     Mat imcv1 = imread("./thiri.png",IMREAD_UNCHANGED);
30     string str = "Channels:" + to_string(imcv1.channels());
31     putText(imcv1, str, Point(100, 100), FONT_HERSHEY_PLAIN, 4.0, CV_RGB(128,
32     0, 128), 4.0);
33     wxBitmap imwx1 = wx_from_mat(imcv1);
34     thiri->SetBitmap(imwx1);
35 }
36
37 class MyApp : public wxApp
38 {
39 public:
40     virtual bool OnInit();
41 };
42 IMPLEMENT_APP(MyApp)
43 bool MyApp::OnInit()
44 {
45     if (!wxApp::OnInit())
46         return false;
```

```

46     wxInitAllImageHandlers();
47
48     MyFrame *frame = new MyFrame(wxT("Simple wxWidgets and OpenCV"));
49     frame->Show(true);
50
51     return true;
52 }
```

စာရင်း J.၂: minimal.cpp



ဦး J.၁၁: wxWidgets နှင့် OpenCV ကို Windows ပေါ်တွင် Visual Studio ဖြင့် အသုံးပြုခြင်း။

အကိုးအကားများ

[Dad10] Jive Dadson. OpenCV Image and wxImage Conversion. 2010. url: <https://stackoverflow.com/a/2241517>.

- [Eme17a] Cool Emerald. OpenCV on Linux using g++, CMake, Qt, Code::Blocks. 2017. url: <http://coolemerald.blogspot.sg/2017/11/opencv-on-linux-using-g-cmake-qt.html>.
- [Eme17b] Cool Emerald. Opencv on Windows Using Visual Studio. 2017. url: <http://coolemerald.blogspot.sg/2017/02/opencv-320-on-windows-10-64-bit-using.html>.

အခန်း ၃

Network ချိတ်ဆက်အသုံးပြုခြင်း

အင်တာနက် စတဲ့ network တွေ ပေါ်မှာ TCP ဒါမှမဟုတ် UDP တွေသုံးပြီး စက်တစ်ခု နဲ့ တစ်ခု ဒေတာ တွေ အပြန်အလှန် ဖို့ဖို့ အတွက် socket တွေကို အသုံးပြု နိုင် ပါတယ်။ ခေတ်ပေါ် operating system တွေ အားလုံးက socket layer ကို အထောက် အပံ့ ပေးကြ ပေမယ့် platform ပေါ်မူ တည်ပြီး socket ကို အသုံး ပြုရတဲ့ ပုံစံ တွေက အမျိုးမျိုး ကွဲပြား နိုင်ပါတယ်။ wxWidgets မှာ အောက်ခံ platform အတွက် ပူစရာ မလိုပဲ အလွယ် တကူ အသုံးပြုနိုင်တဲ့ socket class ပါ ပါတယ်။ အဲဒီ class ကို မတူညီတဲ့ ပုံစံ နည်းလမ်း အမျိုးမျိုး နဲ့ အသုံးပြုနိုင်ပြီး၊ အသုံးပြုပဲ နမူနာ တရာ့ကို အောက်မှာ ဆက်ပြီး ဖော်ပြထား ပါတယ်။

၃.၁ UDP

IP (Internet Protocol) network တွေ ပေါ်မှာ စက်တစ်ခု ကနေ တစ်ခု ကို UDP (User Datagram Protocol) သုံးပြီး (Datagram လိုလည်း ခေါ်ကြတဲ့) message တွေကို ပို့လို ရပါတယ်။ UDP က ရိုးရှင်း တဲ့ connectionless communication ပုံစံ ကို သုံးတဲ့ အတွက် ဒေတာ မပို့ခင် ချိတ်ဆက် တာတွေ၊ handshake လုပ်တာတွေ၊ အမှား ပြင်တာ တွေ မပါပဲ ပေါ်ဝါး မူ ရှိပြီး ပို့လိုက်တဲ့ ဒေတာ မှန်မမှန် ကိုပဲ checksum သုံးပြီး စစ်ပါတယ်။ ပို့လိုက်တဲ့ ဒေတာ က မှားသွား၊ ထပ်သွား လည်း ပြန်ပို့ စရာ မလို၊ ပြင်စရာ မလိုပဲ မြန်ဆန် သွက်လက် ဖို့ပဲ အရေးကြီး တဲ့ နေရာ (ဥပမာ Voice over IP) တွေက UDP နဲ့ သင့်တော် ပါတယ်။

wxWidgets ကို ထည့်သွင်း တပ်ဆင်ပြီး တဲ့ အခါ samples ဆိုတဲ့ အခန်းထဲက sockets ထဲမှာ network ချိတ်ဆက် အသုံးပြု တဲ့ နမူနာ [GZ09] ထဲမှာ UDP သုံးတာ ပြထား ပါတယ်။ အဲဒီ နမူနာ က တခြား

TCP တွေကို ရော ပြည့်စုံ အောင် ပေါင်းပြ ထားတဲ့ အတွက် ရှုပ်ထွေး ခက်ခ မှ အနည်းငယ် ရှိတာ ရယ်၊ event ကို မသုံးပဲ data ပြန်မရ မခြင်း ရပ် နေတာ တွေ ရှိတာ ကြောင့် သူကို အခြေခံ ပြင်ဆင် ထားတဲ့၊ ပိုပြီး ရှိုးရှင်း လွယ်ကူတဲ့ UDP သီးသန် ce_wx_udp.cpp ဆိုတဲ့ နမူနာ တစ်ခု ကို စာရင်း ၃.၁ မှာ ဖော်ပြ ထားပါတယ်။

```

1 // File: ce_wx_udp.cpp
2 // Description: A simpler version of wxWidgets UDP sample
3 // Author: Yan Naing Aye
4 // Web: http://cool-emerald.blogspot.com
5 // MIT License (https://opensource.org/licenses/MIT)
6 // Copyright (c) 2018 Yan Naing Aye
7
8 // References
9 // [1] Guillermo Rodriguez Garcia, Vadim Zeitlin,
10 //      "Server for wxSocket demo",
11 //      https://github.com/wxWidgets/wxWidgets/blob/master/samples/sockets/
12 //          server.cpp, 2009.
13
14 #include "wx/wxprec.h"
15
16 #ifdef __BORLANDC__
17 #pragma hdrstop
18 #endif
19
20 #ifndef WX_PRECOMP
21 #include "wx/wx.h"
22 #endif
23
24 #include "wx/socket.h"
25
26 // this example is currently written to use only IP or only IPv6 sockets, it
27 // should be extended to allow using either in the future
28 #if wxUSE_IPV6
29 typedef wxIPV6address IPaddress;
30 #else

```

```

31 typedef wxIPV4address IPaddress;
32 #endif
33
34 #ifndef wxHAS_IMAGES_IN_RESOURCES
35 #include "./sample.xpm"
36 #endif
37
38 // Define a new application type, each program should derive a class from
39 // wxApp
40 class MyApp : public wxApp
41 {
42     public:
43     // override base class virtuals
44     // -----
45     // this one is called on application startup and is a good place for the app
46     // initialization (doing it here and not in the ctor allows to have an error
47     // return: if OnInit() returns false, the application terminates)
48     virtual bool OnInit();
49 }
50
51 // Define a new frame type: this is going to be our main frame
52 class MyFrame : public wxFrame
53 {
54     public:
55     // ctor(s)
56     MyFrame(const wxString& title);
57
58     // event handlers (these functions should _not_ be virtual)
59     void OnQuit(wxCommandEvent& event);
60     void OnAbout(wxCommandEvent& event);
61     void OnSend(wxCommandEvent& event);
62     void OnSocketEvent(wxSocketEvent& event);
63     private:
64
65     wxDatagramSocket *sock;

```

```
66 wxButton *btnSend;
67 wxTextCtrl *txtSend;
68 wxTextCtrl *txtRx;
69 // any class wishing to process wxWidgets events must use this macro
70 wxDECLARE_EVENT_TABLE();
71 };
72
73 // constants
74 const int ID_BTNSEND = 101;
75 const int ID_TXTSEND = 102;
76 const int ID_TXTRX = 103;
77
78 // IDs for the controls and the menu commands
79 enum
80 {
81 Button_Send = ID_BTNSEND,
82 Txt_Send = ID_TXTSEND,
83 Txt_Rx = ID_TXTRX,
84 SOCKET_ID,
85 // menu items
86 Minimal_Quit = wxID_EXIT,
87
88 // it is important for the id corresponding to the "About" command to have
89 // this standard value as otherwise it won't be handled properly under Mac
90 // (where it is special and put into the "Apple" menu)
91 Minimal_About = wxID_ABOUT
92 };
93
94
95 // the event tables connect the wxWidgets events with the functions (event
96 // handlers) which process them. It can be also done at run-time, but for the
97 // simple menu events like this the static method is much simpler.
98 wxBEGIN_EVENT_TABLE(MyFrame, wxFrame)
99 EVT_MENU(Minimal_Quit, MyFrame::OnQuit)
100 EVT_MENU(Minimal_About, MyFrame::OnAbout)
101 EVT_SOCKET(SOCKET_ID, MyFrame::OnSocketEvent)
```

```

102 wxEND_EVENT_TABLE()
103
104 // Create a new application object: this macro will allow wxWidgets to create
105 // the application object during program execution (it's better than using a
106 // static object for many reasons) and also implements the accessor function
107 // wxGetApp() which will return the reference of the right type (i.e. MyApp
108 // and
109 // not wxApp)
110 IMPLEMENT_APP(MyApp)
111
112 // 'Main program' equivalent: the program execution "starts" here
113 bool MyApp::OnInit()
114 {
115 // call the base class initialization method, currently it only parses a
116 // few common command-line options but it could be do more in the future
117 if ( !wxApp::OnInit() )
118 return false;
119
120 // create the main application window
121 MyFrame *frame = new MyFrame("wxWidgets UDP App");
122
123 // and show it (the frames, unlike simple controls, are not shown when
124 // created initially)
125 frame->Show(true);
126
127 // success: wxApp::OnRun() will be called which will enter the main message
128 // loop and the application will run. If we returned false here, the
129 // application would exit immediately.
130 return true;
131
132 // frame constructor
133 MyFrame::MyFrame(const wxString& title)
134 : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(390, 280),
135     wxDEFAULT_FRAME_STYLE ^ wxRESIZE_BORDER)
136 {

```

```
137 // set the frame icon
138 SetIcon(wxICON(sample));
139
140 #if wxUSE_MENUS
141 // create a menu bar
142 wxMenu *fileMenu = new wxMenu;
143
144 // the "About" item should be in the help menu
145 wxMenu *helpMenu = new wxMenu;
146 helpMenu->Append(Minimal_About, "&About\tF1", "Show about dialog");
147
148 fileMenu->Append(Minimal_Quit, "E&xit\tAlt-X", "Quit this program");
149
150 // now append the freshly created menu to the menu bar...
151 wxMenuBar *menuBar = new wxMenuBar();
152 menuBar->Append(fileMenu, "&File");
153 menuBar->Append(helpMenu, "&Help");
154
155 // ... and attach this menu bar to the frame
156 SetMenuBar(menuBar);
157 #endif // wxUSE_MENUS
158
159 #if wxUSE_STATUSBAR
160 // create a status bar just for fun (by default with 1 pane only)
161 CreateStatusBar(2);
162 SetStatusText("UDP using wxWidgets");
163 #endif // wxUSE_STATUSBAR
164 btnSend = new wxButton(this, Button_Send, wxT("Send"),
165                     wxPoint(5, 5), wxSize(100, 25));
166 txtSend = new wxTextCtrl(this, Txt_Send, wxT("Hello!"),
167                     wxPoint(120, 5), wxSize(250, 25));
168 txtRx = new wxTextCtrl(this, Txt_Rx, wxT(""), wxPoint(5, 35),
169                     wxSize(365, 125), wxTE_MULTILINE);
170
171 Connect(Button_Send, wxEVT_COMMAND_BUTTON_CLICKED,
172         wxCommandEventHandler(MyFrame::OnSend));
```

```
173
174 // Create the address - defaults to localhost:0 initially
175 IPaddress addr;
176 addr.AnyAddress();
177 addr.Service(3000);
178 txtRx->AppendText(wxString::Format(wxT("Creating UDP socket at %s:%u \n")),
179     addr.IPAddress(), addr.Service()));
180
181 // Create the socket
182 sock = new wxDatagramSocket(addr);
183
184 // We use IsOk() here to see if the server is really listening
185 if (!sock->IsOk()){
186     txtRx->AppendText(wxString::Format(wxT("Could not listen at the specified
187         port !\n")));
188     return;
189 }
190 IPaddress addrReal;
191 if (!sock->GetLocal(addrReal)){
192     txtRx->AppendText(wxString::Format(wxT("ERROR: couldn't get the address we
193         bound to. \n")));
194 }
195 else{
196     txtRx->AppendText(wxString::Format(wxT("Server listening at %s:%u \n"),
197         addrReal.IPAddress(), addrReal.Service())));
198 }
199 // Setup the event handler
200 sock->SetEventHandler(*this, SOCKET_ID);
201 sock->SetNotify(wxSOCKET_INPUT_FLAG);
202 sock->Notify(true);
203
204 // event handlers
205 void MyFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
```

```

207 {
208 // true is to force the frame to close
209 Close(true);
210 }
211
212 void MyFrame::OnAbout(wxCommandEvent& WXUNUSED(event))
213 {
214 wxMessageBox(wxString::Format
215 (
216 "wxWidgets UDP sample\n"
217 "\n"
218 "Author: Yan Naing Aye \n"
219 "Web: http://cool-emerald.blogspot.com"
220 ),
221 "About wxWidgets UDP sample",
222 wxOK | wxICON_INFORMATION,
223 this);
224 }
225
226 void MyFrame::OnSend(wxCommandEvent& WXUNUSED(event))
227 {
228 wxString str = txtSend->GetValue();
229 wxCharBuffer buffer = str.ToUTF8();
230 size_t txn = str.length();
231
232 IPaddress addrLocal;
233 addrLocal.AnyAddress();
234 wxDatagramSocket sock2tx(addrLocal);
235 if (!sock2tx.IsOk())
236 {
237 txtRx->AppendText(wxT("Failed to create UDP socket.\n"));
238 return;
239 }
240 txtRx->AppendText(wxString::Format(wxT("Created socket at %s:%u \n"),
241 addrLocal.IPAddress(), addrLocal.Service()));
242

```

```

243     IPaddress raddr;
244     raddr.Hostname("localhost");
245     //raddr.Hostname("192.168.7.2");
246     raddr.Service(3001);
247     if (sock2tx.SendTo(raddr, buffer.data(), txn).LastCount() != txn)
248     {
249         txtRx->AppendText(wxT("Write error.\n"));
250         return;
251     }
252     else {
253         txtRx->AppendText("Tx: "+str+"\n");
254     }
255 }

256

257 void MyFrame::OnSocketEvent(wxSocketEvent& event)
258 {
259     IPaddress addr;
260     addr.Service(3000);
261     char buf[1024];
262     size_t n;
263     switch(event.GetSocketEvent())
264     {
265     case wxSOCKET_INPUT:
266         //txtRx->AppendText("OnSocketEvent: wxSOCKET_INPUT\n");
267         sock->Notify(false);
268         n = sock->RecvFrom(addr, buf, sizeof(buf)).LastCount();
269         if (!n) {
270             txtRx->AppendText("ERROR: failed to receive data \n");
271             return;
272         }
273         //txtRx->AppendText(wxString::Format(wxT("Received \"%s\" from %s:%u.\n"))
274         ,
275         //  wxString::From8BitData(buf, n), addr.IPAddress(), addr.Service()));
276         txtRx->AppendText("Rx: "+wxString::From8BitData(buf, n) + "\n");
277         sock->Notify(true);
278     break;

```

```

278     default: txtRx->AppendText("OnSocketEvent: Unexpected event !\n"); break;
279 }
280 }
```

စာရင်း ၃.၁: ce_wx_udp.cpp

ပရိုဂရမ် အစမှာ wxWidgets နဲ့ socket အတွက် header ဖိုင်တွေ နဲ့ IP address အမျိုးအစား တွေကို
အောက်ပါ အတိုင်း သတ်မှတ် နိုင်ပါတယ်။

```

#include "wx/wx.h"
#include "wx/socket.h"
#if wxUSE_IPV6
    typedef wxIPV6address IPaddress;
#else
    typedef wxIPV4address IPaddress;
#endif
```

ပြီးတဲ့ အခါ လိုချင်တဲ့ IP address + port number တွေနဲ့ UDP socket တစ်ခု ကို ဖန်တီးပြီး
အသုံးပြုချင်တဲ့ event တွေကို သတ်မှတ် နိုင် ပါတယ်။

```

// Create the address - defaults to localhost:0 initially
IPaddress addr;
addr.AnyAddress();
addr.Service(3000);

// Create the socket
sock = new wxDatagramSocket(addr);

// Setup the event handler
sock->SetEventHandler(*this, SOCKET_ID);
sock->SetNotify(wxSOCKET_INPUT_FLAG);
sock->Notify(true);
```

ဒေတာ တွေလက်ခံ ရရှိတဲ့ အခါ OnSocketEvent ရဲ့ wxSOCKET_INPUT အမျိုးအစား event မှာ

RecvFrom method ကို သုံးပြီး ဖတ်နှင့် ပါတယ်။

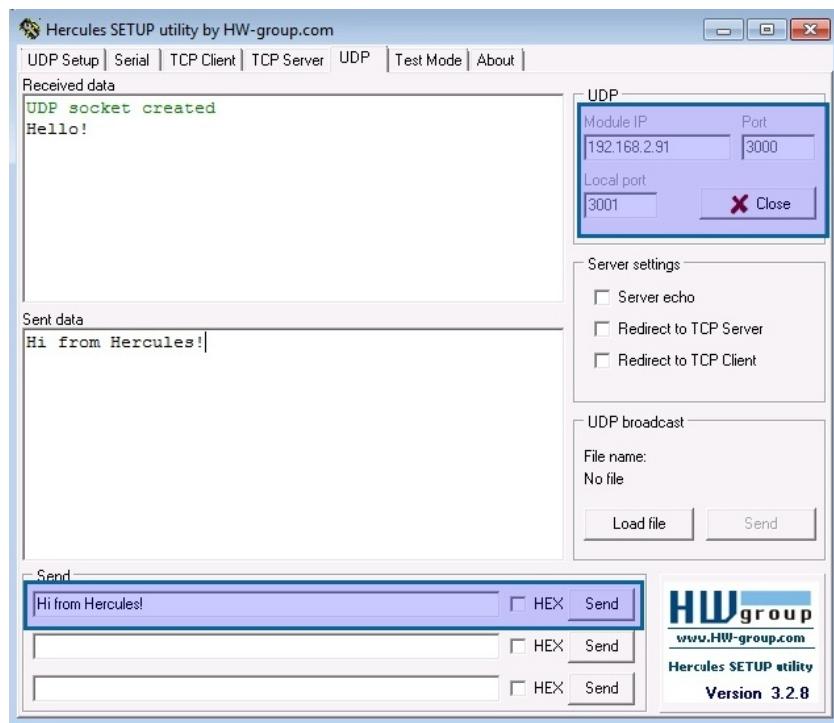
```
n = sock->RecvFrom(addr, buf, sizeof(buf)).LastCount();
```

ဒေတာ ပိုမို အတွက် ပိုမယ့် remote host ရဲ့ address နဲ့ port number ကို သတ်မှတ်ပြီး၊ SendTo method နဲ့ ပို့နိုင် ပါတယ်။

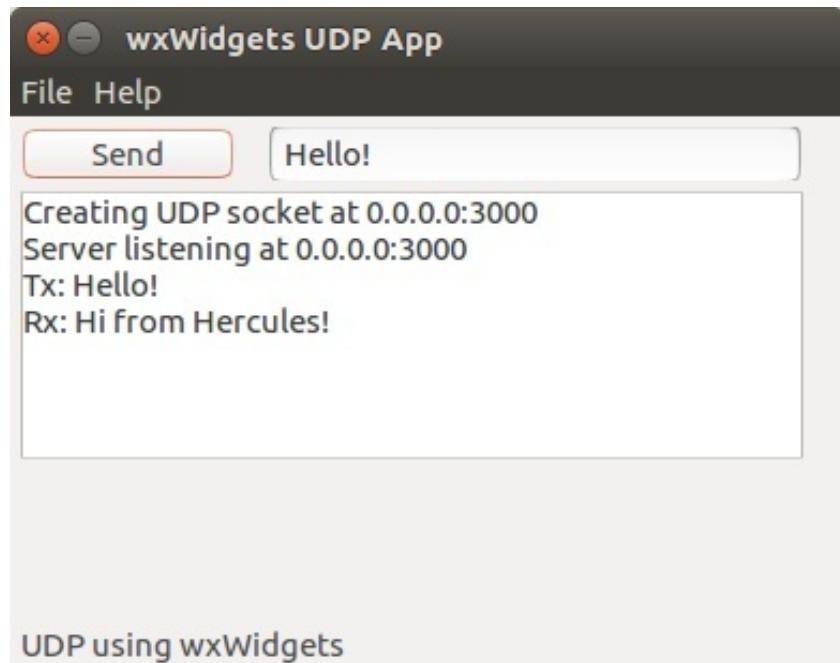
```
IPaddress raddr;
raddr.Hostname("localhost");
raddr.Service(3001);
sock->SendTo(raddr, buf, n)
```

ဒါ ပရိုဂရမ် ကို စမ်းကြည့်ဖို့ အတွက် https://www.hw-group.com/products/hercules/index_en.html မှာ free ရနိုင်တဲ့ Hercules Utility ကို သုံးနိုင် ပါတယ်။ Hercules ကို သူနဲ့ ဆက်သွယ် စမ်းသပ်မယ့် Windows ကွန်ပျူးတာ မှာ တပ်ဆင် လိုက်ပြီး၊ ပရိုဂရမ် ကို ဖွင့်ပြီး တဲ့ အခါ IP address နဲ့ UDP port ကို သတ်မှတ်၊ နားထောင် ပါမယ်။ စာရင်း ၃.၁ က ပရိုဂရမ်ကို အောက်က အတိုင်း build လုပ်ပြီး run လိုက်တဲ့ အခါ သူတို့ အချင်းချင်း ဒေတာ တွေ အပြန်အလှန် ပို့နိုင် တာကို ပုံ ၃.၁ နဲ့ ပုံ ၃.၂ မှာ ပြထား သလို တွေ့နိုင် ပါတယ်။

```
g++ ce_wx_udp.cpp `wx-config --cxxflags --libs` -o ce_wx_udp
gksudo ./ce_wx_udp
```



ပုံ ၃၁: Hercules utility ကို အသိပါမြင်း။



ပုံ ၃.၂: UDP ဖြင့် ဒေတာများ ပို့ခြင်း နှင့် လက်ခံခြင်း။

၃.၂ TCP

TCP (Transmission Control Protocol) က အဓိက ကျတဲ့ network protocol တွေထဲမှာ တရု အပါအဝင် ဖြစ်ပါတယ်။ သူက ဒေတာ တွေပို့ တဲ့ အခါ စိတ်ချ ယံကြည် ရပြီး၊ အစီအစဉ် တကျ ရောက်အောင်၊ အမှား မပါ အောင် ပို့နိုင် ပါတယ်။ TCP နဲ့ ဒေတာ တွေ ပို့နိုင် ဖို့ အရင်ဆုံး connection ကို ချိတ်ဆက်ဖို့ လိုပါတယ်။ အဲဒီ အတွက် သတ်မှတ်ထား တဲ့ port number ကို နားထောင် နေမယ့် server နဲ့ အဲဒီ ကို လုမ်း ဆက်သွယ် ပြီး ချိတ်ဆက်မှု ကို စတင် မယ့် client ဆိုပြီး နှစ်မျိုး ရှုပါ တယ်။

၃.၂.၁ TCP Server

TCP server က သတ်မှတ် ထားတဲ့ port number တစ်ခု မှာ passively နားထောင် နေပြီး၊ သူကို လာဆက် သွယ်တဲ့ client နဲ့ ဒေတာ တွေ အပြန်အလှန် ပို့နိုင် ပါတယ်။ Client တွေ ဆီက ဒေတာ တွေကို လက်ခံ ဖော်ပြပြီး၊ ပြန်ပို့ပေး၊ ပြီးတော့ လက်ရှိ ဆက်သွယ် နေတဲ့ client အရေအတွက် တွေကို ပါဖော်ပြ ပေးတဲ့ [ce_wx_tcp_server.cpp](#) ဆိုတဲ့ TCP server နမူနာ [GZ09] တစ်ခု ကို စာရင်း ၃.၂ မှာ ဖော်ပြထား ပါတယ်။

```
1 // File: ce_wx_tcp_server.cpp
2 // Description: A simple wxWidgets TCP server sample
3 // Author: Yan Naing Aye
4 // Web: http://cool-emerald.blogspot.com
5 // MIT License (https://opensource.org/licenses/MIT)
6 // Copyright (c) 2018 Yan Naing Aye
7
8 // References
9 // [1] Guillermo Rodriguez Garcia, Vadim Zeitlin, "Server for wxSocket demo",
10 //      https://github.com/wxWidgets/wxWidgets/blob/master/samples/sockets/
11 //          server.cpp, 2009.
12 // [2] Julian Smart and Kevin Hock, "Cross-Platform GUI Programming with
13 //      wxWidgets,"
14 //      Pearson Education, Inc. 2006. ISBN: 0-13-147381-6.
15
16 #include "wx/wxprec.h"
17
18 #ifdef __BORLANDC__
19 #pragma hdrstop
20 #endif
21
22 #ifndef WX_PRECOMP
23 #include "wx/wx.h"
24 #endif
25
26 // this example is currently written to use only IP or only IPv6 sockets, it
27 // should be extended to allow using either in the future
28 #if wxUSE_IPV6
29 typedef wxIPV6address IPEndPoint;
30 #else
31 typedef wxIPV4address IPEndPoint;
32 #endif
```

```

34 #ifndef wxHAS_IMAGES_IN_RESOURCES
35 #include "./sample.xpm"
36 #endif
37
38 class MyApp : public wxApp
39 {
40 public:
41
42 virtual bool OnInit();
43 };
44
45 // Define a new frame type: this is going to be our main frame
46 class MyFrame : public wxFrame
47 {
48 public:
49 // ctor(s)
50 MyFrame(const wxString& title);
51 ~MyFrame();
52 // event handlers (these functions should _not_ be virtual)
53 void OnQuit(wxCommandEvent& event);
54 void OnAbout(wxCommandEvent& event);
55 void OnServerEvent(wxSocketEvent& event);
56 void OnSocketEvent(wxSocketEvent& event);
57 private:
58
59 wxSocketServer *sock;
60 wxTextCtrl *txtRx;
61 int numClients;
62 // any class wishing to process wxWidgets events must use this macro
63 wxDECLARE_EVENT_TABLE();
64 };
65
66 // IDs for the controls and the menu commands
67 enum
68 {
69 ID_TXTRX=101,

```

```
70 SOCKET_ID,
71 SERVER_ID,
72 // menu items
73 Minimal_Quit = wxID_EXIT,
74
75 Minimal_About = wxID_ABOUT
76 };
77
78 // event tables and other macros for wxWidgets
79 wxBEGIN_EVENT_TABLE(MyFrame, wxFrame)
80 EVT_MENU(Minimal_Quit, MyFrame::OnQuit)
81 EVT_MENU(Minimal_About, MyFrame::OnAbout)
82 EVT_SOCKET(SOCKET_ID, MyFrame::OnSocketEvent)
83 EVT_SOCKET(SERVER_ID, MyFrame::OnServerEvent)
84 wxEND_EVENT_TABLE()
85
86 IMPLEMENT_APP(MyApp)
87
88 // 'Main program' equivalent: the program execution "starts" here
89 bool MyApp::OnInit()
90 {
91 if ( !wxApp::OnInit() )
92 return false;
93 MyFrame *frame = new MyFrame("wxWidgets TCP Server");
94 frame->Show(true);
95 return true;
96 }
97
98 // frame constructor
99 MyFrame::MyFrame(const wxString& title)
100 : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(390, 280),
101 wxDEFAULT_FRAME_STYLE ^ wxRESIZE_BORDER)
102 {
103 // set the frame icon
104 SetIcon(wxICON(sample));
105 }
```

```

106 #if wxUSE_MENUS
107 // create a menu bar
108 wxMenu *fileMenu = new wxMenu;
109
110 // the "About" item should be in the help menu
111 wxMenu *helpMenu = new wxMenu;
112 helpMenu->Append(Minimal_About, "&About\tF1", "Show about dialog");
113
114 fileMenu->Append(Minimal_Quit, "E&xit\tAlt-X", "Quit this program");
115
116 // now append the freshly created menu to the menu bar...
117 wxMenuBar *menuBar = new wxMenuBar();
118 menuBar->Append(fileMenu, "&File");
119 menuBar->Append(helpMenu, "&Help");
120
121 // ... and attach this menu bar to the frame
122 SetMenuBar(menuBar);
123 #endif // wxUSE_MENUS
124
125 #if wxUSE_STATUSBAR
126 // create a status bar just for fun (by default with 1 pane only)
127 CreateStatusBar(2);
128 SetStatusText("TCP server using wxWidgets");
129 #endif // wxUSE_STATUSBAR
130 txtRx = new wxTextCtrl(this, ID_TXTRX, wxT(""), wxPoint(5, 5),
131 wxSize(365, 125), wxTE_MULTILINE);
132
133 // Create the address - defaults to localhost:0 initially
134 IPaddress addr;
135 addr.AnyAddress();
136 addr.Service(3000);
137 txtRx->AppendText(wxString::Format(wxT("Creating server at %s:%u \n"))
138 ,addr.IPAddress(), addr.Service()));
139
140 // Create the socket
141 sock = new wxSocketServer(addr);

```

```
142
143 // We use IsOk() here to see if the server is really listening
144 if (!sock->IsOk()){
145     txtRx->AppendText(wxString::Format(wxT("Could not listen at the specified
146         port !\n")));
147     return;
148 }
149 IPaddress addrReal;
150 if (!sock->GetLocal(addrReal)){
151     txtRx->AppendText(wxString::Format(wxT("ERROR: couldn't get the address we
152         bound to. \n")));
153 }
154 else{
155     txtRx->AppendText(wxString::Format(wxT("Server listening at %s:%u \n"),
156         addrReal.IPAddress(), addrReal.Service())));
157 }
158 // Setup the event handler and subscribe to connection events
159 sock->SetEventHandler( *this, SERVER_ID );
160 sock->SetNotify(wxSOCKET_CONNECTION_FLAG);
161 sock->Notify(true);
162 numClients = 0;
163 }
164
165 MyFrame::~MyFrame()
166 {
167     // No delayed deletion here, as the frame is dying anyway
168     delete sock;
169 }
170
171 // event handlers
172 void MyFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
173 {
174     // true is to force the frame to close
175     Close(true);
```

```

176 }
177
178 void MyFrame::OnAbout(wxCommandEvent& WXUNUSED(event))
179 {
180 wxMessageBox(wxString::Format
181 (
182 "wxWidgets TCP server sample\n"
183 "\n"
184 "Author: Yan Naing Aye \n"
185 "Web: http://cool-emerald.blogspot.com"
186 ),
187 "About wxWidgets TCP server sample",
188 wxOK | wxICON_INFORMATION,
189 this);
190 }
191
192 void MyFrame::OnServerEvent(wxSocketEvent& event)
193 {
194 txtRx->AppendText(wxT("OnServerEvent: "));
195 wxSocketBase *sockBase;
196
197 switch (event.GetSocketEvent())
198 {
199 case wxSOCKET_CONNECTION: txtRx->AppendText(wxT("wxSOCKET_CONNECTION\n"));
200 break;
201 default: txtRx->AppendText(wxT("Unexpected event !\n")); break;
202 }
203
204 // Accept new connection if there is one in the pending
205 // connections queue, else exit. We use Accept(false) for
206 // non-blocking accept (although if we got here, there
207 // should ALWAYS be a pending connection).
208
209 sockBase = sock->Accept(false);
210 if (sockBase)

```

```
211 {
212     IPEndPoint addr;
213     if (!sockBase->GetPeer(addr))
214     {
215         txtRx->AppendText(wxT("New connection from unknown client accepted.\n"))
216     }
217     else
218     {
219         txtRx->AppendText(wxString::Format(wxT("New client connection from %s:%
220 u accepted \n")),
221                         addr.IPEndPoint(), addr.Service()));
222     }
223 }
224 else
225 {
226     txtRx->AppendText(wxT("Error: couldn't accept a new connection \n"));
227     return;
228 }
229
230 sockBase->SetEventHandler(*this, SOCKET_ID);
231 sockBase->SetNotify(wxSOCKET_INPUT_FLAG | wxSOCKET_LOST_FLAG);
232 sockBase->Notify(true);
233
234 numClients++;
235 SetStatusText(wxString::Format(wxT("%d clients connected"), numClients), 1)
236 ;
237 }
238 void MyFrame::OnSocketEvent(wxSocketEvent& event)
239 {
240     txtRx->AppendText(wxT("OnSocketEvent: "));
241     wxSocketBase *sockBase = event.GetSocket();
242
243     // First, print a message
244     switch (event.GetSocketEvent())
```

```

244 {
245     case wxSOCKET_INPUT: txtRx->AppendText(wxT("wxSOCKET_INPUT\n")); break;
246     case wxSOCKET_LOST: txtRx->AppendText(wxT("wxSOCKET_LOST\n")); break;
247     default: txtRx->AppendText(wxT("Unexpected event !\n")); break;
248 }

249
250 // Now we process the event
251 switch (event.GetSocketEvent())
252 {
253     case wxSOCKET_INPUT:
254     {
255         // We disable input events, so that the test doesn't trigger
256         // wxSocketEvent again.
257         sockBase->SetNotify(wxSOCKET_LOST_FLAG);
258
259         // Receive data from socket and send it back. We will first
260         // get a byte with the buffer size, so we can specify the
261         // exact size and use the wxSOCKET_WAITALL flag. Also, we
262         // disabled input events so we won't have unwanted reentrance.
263         // This way we can avoid the infamous wxSOCKET_BLOCK flag.
264
265         sockBase->SetFlags(wxSOCKET_WAITALL);
266
267         // Read the size @ first byte
268         unsigned char len;
269         sockBase->Read(&len, 1);
270         char buf[256];
271
272         // Read the message
273         wxUint32 lenRd = sockBase->Read(buf, len).LastCount();
274         if (!lenRd) {
275             txtRx->AppendText(wxT("Failed to read message.\n"));
276             return;
277         }
278         else {
279             txtRx->AppendText(wxString::Format(wxT("Read %d bytes.\n"), lenRd));
280         }
281     }
282 }

```

```
280     txtRx->AppendText(wxString::Format(wxT("Rx: %s \n"), wxString::FromUTF8(
281         buf, len)));
282
283     // Write it back
284     len = 2;
285     buf[0] = '0';
286     buf[1] = 'K';
287     sockBase->Write(&len,1);
288     sockBase->Write(buf, len);
289     txtRx->AppendText("Tx: " + wxString::From8BitData(buf, len) + "\n");
290
291     // Enable input events again.
292     sockBase->SetNotify(wxSOCKET_LOST_FLAG | wxSOCKET_INPUT_FLAG);
293     break;
294 }
295 case wxSOCKET_LOST:
296 {
297     numClients--;
298
299     // Destroy() should be used instead of delete wherever possible,
300     // due to the fact that wxSocket uses 'delayed events' (see the
301     // documentation for wxPostEvent) and we don't want an event to
302     // arrive to the event handler (the frame, here) after the socket
303     // has been deleted. Also, we might be doing some other thing with
304     // the socket at the same time; for example, we might be in the
305     // middle of a test or something. Destroy() takes care of all
306     // this for us.
307
308     txtRx->AppendText(wxT("Deleting socket.\n"));
309     sockBase->Destroy();
310     break;
311 }
312 default:;
313
314 SetStatusText(wxString::Format(wxT("%d clients connected"), numClients),
```

```
    1);
315 }
```

စာရင်း ၃.J: ce_wx_tcp_server.cpp

အစ frame constructor မှာ socket server တစ်ခု ကို အောက်က အတိုင်း သတ်မှတ် လိုတဲ့ port number နဲ့ ဖန်တီးပြီး၊ ဆက်သွယ်မှု တစ်ခု ရောက်လာရင် လုပ်ဆောင်ဖို့ event handler ကို သတ်မှတ် နိုင်ပါတယ်။

```
// Create the address - defaults to localhost:0 initially
IPaddress addr;
addr.AnyAddress();
addr.Service(3000);

// Create the socket
sock = new wxSocketServer(addr);

// Setup the event handler and subscribe to connection events
sock->SetEventHandler( *this, SERVER_ID );
sock->SetNotify(wxSOCKET_CONNECTION_FLAG);
sock->Notify(true);
```

OnServerEvent မှာ ရောက်လာတဲ့ ဆက်သွယ်မှု ကို လက်ခံပြီး ရင် ရလာတဲ့ socket အတွက် ဒေတာတွေ ဝင်လာတဲ့ အခါး ဒါမှုမဟုတ် ဆက်သွယ်မှု ပြတ်တောက် သွားတဲ့ အခါ လုပ်ဆောင် ဖို့ event handler တွေကို သတ်မှတ် ပါမယ်။ နောက်ပြီး စုစုပေါင်း client အရေအတွက် ကို ဖော်ပြနိုင် ပါတယ်။

```
wxSocketBase *sockBase;
sockBase = sock->Accept(false);

sockBase->SetEventHandler( *this, SOCKET_ID );
sockBase->SetNotify(wxSOCKET_INPUT_FLAG | wxSOCKET_LOST_FLAG);
sockBase->Notify(true);

numClients++;
```

```
SetStatusText(wxString::Format(wxT("%d clients connected"), numClients), 1);
```

ဒေတာ တွေ ဝင်လာရင် OnSocketEvent နဲ့ လက်ခံ ရယူ တဲ့ ပုံစံ က လက်ရှိ socket ရဲ့ setting ထွေပေါ် မူတည်ပြီး အမျိုးမျိုး ဖြစ်နိုင် ပါတယ်။ Setting တွေ သတ်မှတ် တာ မမှန်ရင် ပရိုဂရမ် ရပ်သွား နိုင်တာကြောင့် မှန်မှန် ကန်ကန် သတ်မှတ်ဖို့ အရေးကြီး ပြီး၊ အဲဒီ အကြောင်း အသေးစိတ်ကို Julian Smart ရဲ့ Cross-Platform GUI Programming with wxWidgets [SH06] ဆိုတဲ့ စာအုပ် အဆုံး ၁၈ မှာ ဖော်ပြု ထားတာကို ဖတ်ကြည့် သင့်ပါတယ်။

ဒီ နမူနာ မှာတော့ ပထမ ဆုံး byte မှာ message ရဲ့ အရွယ် အစား ကို ပို့ပေးဖို့ လိုပြီး၊ အဲဒီ အရေ အတွက် မရ မချင်း စောင့်ပြီး ဖတ်တဲ့ wxSOCKET_WAITALL ကို အသုံးပြု ထား ပါတယ်။

```
sockBase->SetFlags(wxSOCKET_WAITALL);

// Read the size @ first byte
unsigned char len;
sockBase->Read(&len, 1);

char buf[256];
// Read the message
wxUint32 lenRd = sockBase->Read(buf, len).LastCount();
```

ဒေတာ တွေ လက်ခံ ရရှိ ပြီးတဲ့ အခါ OK ဆိုတဲ့ message ကို သူရဲ့ အရွယ် အစား 2 ကို ရှုံးဆုံး byte မှာ ထည့်ပြီး client ဆီကို အကြောင်း ပြန်ပါ မယ်။

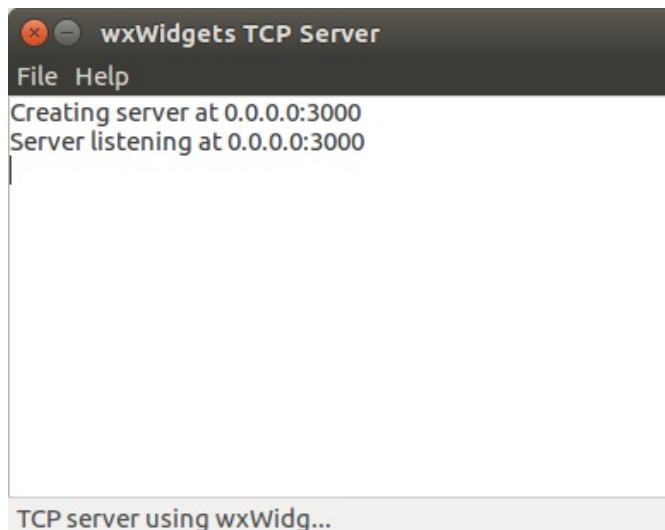
```
len = 2;
buf[0] = '0';
buf[1] = 'K';
sockBase->Write(&len, 1);
sockBase->Write(buf, len);
```

Connection ကို ဖြတ်တောက် လိုက်တဲ့ အခါ မှာ wxSOCKET_LOST ဆိုတဲ့ OnSocketEvent ဖြစ်တဲ့ အခါမှာ ပြတ်တောက် သွားတဲ့ socket ကို ဖျက်လိုက် ပါမယ်။

```
sockBase->Destroy();
```

ပရီဂရမ် ကို အောက်က command တွေနဲ့ build လုပ်ပြီး၊ run လိုက်တဲ့ အခါ ပုံ ၃.၃ မှာ ပြထားသလို client ရဲ့ ဆက်သွယ်မူ ကို နားထောင် နေတာ တွေ့နိုင် ပါတယ်။

```
g++ ce_wx_tcp_server.cpp `wx-config --cxxflags --libs` -o ce_wx_tcp_server
gksudo ./ce_wx_tcp_server
```



ပုံ ၃.၃: TCP server

၃.၂.၂ TCP Client

TCP client က လိပ်စာ တစ်ခု က port number တစ်ခု မှာ နားထောင် နေတဲ့ server ကို သွားရောက်ဆက်သွယ်ပြီး၊ ဒေတာ တွေ အပြန်အလှန် ပို့နိုင် ပါတယ်။ Connection တစ်ခု ကို ပြုလုပ်ပြီး ဒေတာတွေကို ပိုပေး၊ server က ပြန်ပို တာကို လက်ခံ ဖော်ပြ ပေးတဲ့ `ce_wx_tcp_client.cpp` ဆိုတဲ့ TCP client နမူနာ [Gar99] တစ်ခု ကို စာရင်း ၃.၃ မှာ ဖော်ပြထား ပါတယ်။

```
1 // File: ce_wx_tcp_client.cpp
2 // Description: A simple wxWidgets TCP client sample
3 // Author: Yan Naing Aye
4 // Web: http://cool-emerald.blogspot.com
5 // MIT License (https://opensource.org/licenses/MIT)
6 // Copyright (c) 2018 Yan Naing Aye
7 // References
```

```
8 // [1] Guillermo Rodriguez Garcia, "Client for wxSocket demo,"  
9 //     https://github.com/wxWidgets/wxWidgets/blob/master/samples/sockets/  
10 //      client.cpp, 1999.  
11 // [2] Julian Smart and Kevin Hock, "Cross-Platform GUI Programming with  
12 //      wxWidgets,"  
13 //      Pearson Education, Inc. 2006. ISBN: 0-13-147381-6.  
14  
15  
16 #ifdef __BORLANDC__  
17 #pragma hdrstop  
18 #endif  
19  
20 #ifndef WX_PRECOMP  
21 #include "wx/wx.h"  
22 #endif  
23  
24 #include "wx/socket.h"  
25  
26 #if wxUSE_IPV6  
27 typedef wxIPV6address IPaddress;  
28 #else  
29 typedef wxIPV4address IPaddress;  
30 #endif  
31  
32 #ifndef wxHAS_IMAGES_IN_RESOURCES  
33 #include "./sample.xpm"  
34 #endif  
35  
36 class MyApp : public wxApp  
37 {  
38 public:  
39     virtual bool OnInit();  
40 };
```

```
42 // Define a new frame type: this is going to be our main frame
43 class MyFrame : public wxFrame
44 {
45 public:
46     // ctor(s)
47     MyFrame(const wxString& title);
48     ~MyFrame();
49     // event handlers (these functions should _not_ be virtual)
50     void OnQuit(wxCommandEvent& event);
51     void OnAbout(wxCommandEvent& event);
52
53     // event handlers for Socket menu
54     void OnOpenConnection(wxCommandEvent& event);
55     void OnCloseConnection(wxCommandEvent& event);
56     void OnSend(wxCommandEvent& event);
57     void OnSocketEvent(wxSocketEvent& event);
58
59     // convenience functions
60     void UpdateStatusBar();
61
62 private:
63
64     wxSocketClient *sock;
65     wxButton *btnSend;
66     wxTextCtrl *txtSend;
67     wxTextCtrl *txtRx;
68     wxMenu *fileMenu;
69     wxMenu *helpMenu;
70     // any class wishing to process wxWidgets events must use this macro
71     wxDECLARE_EVENT_TABLE();
72 };
73
74 // IDs for the controls and the menu commands
75 enum
76 {
77     ID_BTNSEND=101,
```

```
78 ID_TXTSEND ,
79 ID_TXTRX ,
80 SOCKET_ID ,
81 CLIENT_OPEN=wxID_OPEN ,
82 CLIENT_CLOSE=wxID_CLOSE ,
83 // menu items
84 Minimal_Quit = wxID_EXIT ,
85 Minimal_About = wxID_ABOUT
86 };
87
88 wxBEGIN_EVENT_TABLE(MyFrame, wxFrame)
89 EVT_MENU(Minimal_Quit, MyFrame::OnQuit)
90 EVT_MENU(Minimal_About, MyFrame::OnAbout)
91 EVT_SOCKET(SOCKET_ID, MyFrame::OnSocketEvent)
92 EVT_MENU(CLIENT_OPEN, MyFrame::OnOpenConnection)
93 EVT_MENU(CLIENT_CLOSE, MyFrame::OnCloseConnection)
94 wxEND_EVENT_TABLE()
95
96 IMPLEMENT_APP(MyApp)
97
98 // 'Main program'
99 bool MyApp::OnInit()
100 {
101     if (!wxApp::OnInit())
102         return false;
103
104     // create the main application window
105     MyFrame *frame = new MyFrame("wxWidgets TCP Client");
106
107     frame->Show(true);
108     return true;
109 }
110
111 // frame constructor
112 MyFrame::MyFrame(const wxString& title)
113     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(390, 280)
```

```
114         , wxDEFAULT_FRAME_STYLE ^ wxRESIZE_BORDER)  
115     {  
116         // set the frame icon  
117         SetIcon(wxICON(sample));  
118  
119 #if wxUSE_MENUS  
120     // create a menu bar  
121     fileMenu = new wxMenu;  
122  
123     // the "About" item should be in the help menu  
124     helpMenu = new wxMenu;  
125     helpMenu->Append(Minimal_About, "&About\tF1", "Show about dialog");  
126  
127     fileMenu->Append(CLIENT_OPEN, "&Open session\tAlt-O", "Connect to server");  
128     fileMenu->Append(CLIENT_CLOSE,"&Close session\tAlt-C","Close connection");  
129     fileMenu->Append(Minimal_Quit, "E&xit\tAlt-X", "Quit this program");  
130  
131     // now append the freshly created menu to the menu bar...  
132     wxMenuBar *menuBar = new wxMenuBar();  
133     menuBar->Append(fileMenu, "&File");  
134     menuBar->Append(helpMenu, "&Help");  
135  
136     // ... and attach this menu bar to the frame  
137     SetMenuBar(menuBar);  
138 #endif // wxUSE_MENUS  
139  
140 #if wxUSE_STATUSBAR  
141     // create a status bar just for fun (by default with 1 pane only)  
142     CreateStatusBar(2);  
143     SetStatusText("TCP client using wxWidgets");  
144 #endif // wxUSE_STATUSBAR  
145     btnSend = new wxButton(this, ID_BTNSEND, wxT("Send"),  
146                         wxPoint(5, 5), wxSize(100, 25));  
147     txtSend = new wxTextCtrl(this, ID_TXTSEND, wxT("Hello!"),  
148                           wxPoint(120, 5), wxSize(250, 25));  
149     txtRx = new wxTextCtrl(this, ID_TXTRX, wxT(""),
```

```
150     wxPoint(5, 35), wxSize(365, 125), wxTE_MULTILINE);
151
152 Connect(ID_BTNSEND, wxEVT_COMMAND_BUTTON_CLICKED,
153         wxCommandEvent::OnSend));
154
155 // Create the socket
156 sock = new wxSocketClient();
157
158 // Setup the event handler and subscribe to most events
159 sock->SetEventHandler(*this, SOCKET_ID);
160 sock->SetNotify(wxSOCKET_CONNECTION_FLAG |
161                  wxSOCKET_INPUT_FLAG |
162                  wxSOCKET_LOST_FLAG);
163 sock->Notify(true);
164 }
165
166 MyFrame::~MyFrame()
167 {
168     // No delayed deletion here, as the frame is dying anyway
169     delete sock;
170 }
171
172 // event handlers
173 void MyFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
174 {
175     // true is to force the frame to close
176     Close(true);
177 }
178
179 void MyFrame::OnAbout(wxCommandEvent& WXUNUSED(event))
180 {
181     wxMessageBox(wxString::Format
182 (
183     "wxWidgets TCP client sample\n"
184     "\n"
185     "Author: Yan Naing Aye \n"
```

```

186     "Web: http://cool-emerald.blogspot.com"
187 ),
188 "About wxWidgets TCP client sample",
189 wxOK | wxICON_INFORMATION,
190 this);
191 }
192
193 void MyFrame::OnSend(wxCommandEvent& WXUNUSED(event))
194 {
195     wxString str = txtSend->GetValue();
196     wxCharBuffer buffer = str.ToUTF8();
197     size_t txn = str.length();
198
199     unsigned char len;
200     len = txn;
201     sock->Write(&len, 1); //send the length of the message first
202     if (sock->Write(buffer.data(), txn).LastCount() != txn)
203     {
204         txtRx->AppendText(wxT("Write error.\n"));
205         return;
206     }
207     else {
208         txtRx->AppendText("Tx: " + str + "\n");
209     }
210 }
211
212 void MyFrame::OnOpenConnection(wxCommandEvent& WXUNUSED(event))
213 {
214     // Create the address - defaults to localhost:0 initially
215     IPaddress addr;
216     //addr.AnyAddress();
217     addr.Hostname("localhost");
218     addr.Service(3000);
219     txtRx->AppendText(wxString::Format(wxT("Trying to connect to %s:%u \n"),
220                                         addr.IPAddress(), addr.Service())));
221 }
```

```
222 fileMenu->Enable(CLIENT_OPEN, false);
223 fileMenu->Enable(CLIENT_CLOSE, false);
224 // we connect asynchronously and will get a wxSOCKET_CONNECTION event when
225 // the connection is really established
226 //
227 // if you want to make sure that connection is established right here you
228 // could call WaitOnConnect(timeout) instead
229
230 sock->Connect(addr, false);
231
232 //update status
233 UpdateStatusBar();
234 }
235
236 void MyFrame::OnCloseConnection(wxCommandEvent& WXUNUSED(event))
237 {
238     sock->Close();
239
240 //update status
241 UpdateStatusBar();
242 }
243
244 void MyFrame::OnSocketEvent(wxSocketEvent& event)
245 {
246     txtRx->AppendText(wxT("OnSocketEvent: "));
247     wxSocketBase *sockBase = event.GetSocket();
248
249 // First, print a message
250     switch (event.GetSocketEvent())
251     {
252         case wxSOCKET_INPUT: txtRx->AppendText(wxT("wxSOCKET_INPUT\n")); break;
253         case wxSOCKET_LOST: txtRx->AppendText(wxT("wxSOCKET_LOST\n")); break;
254         case wxSOCKET_CONNECTION: txtRx->AppendText(wxT("wxSOCKET_CONNECTION\n"));
255             break;
256         default: txtRx->AppendText(wxT("Unexpected event !\n")); break;
257     }
258 }
```

```
257  
258 // Now we process the event  
259 switch (event.GetSocketEvent())  
260 {  
261 case wxSOCKET_INPUT:  
262 {  
263 // We disable input events, so that the test doesn't trigger  
// wxSocketEvent again.  
264 sockBase->SetNotify(wxSOCKET_LOST_FLAG);  
265  
266 // Receive data from socket and send it back. We will first  
// get a byte with the buffer size, so we can specify the  
// exact size and use the wxSOCKET_WAITALL flag. Also, we  
// disabled input events so we won't have unwanted reentrance.  
// This way we can avoid the infamous wxSOCKET_BLOCK flag.  
267  
268  
269  
270  
271  
272  
273 sockBase->SetFlags(wxSOCKET_WAITALL);  
274  
275 // Read the size @ first byte  
276 unsigned char len;  
277 sockBase->Read(&len, 1);  
278 char buf[256];  
279 // Read the message  
280 wxUint32 lenRd = sockBase->Read(buf, len).LastCount();  
281 if (!lenRd) {  
282 txtRx->AppendText(wxT("Failed to read message.\n"));  
283 return;  
284 }  
285 else {  
286 txtRx->AppendText(wxString::Format(wxT("Read %d bytes.\n"), lenRd));  
287 }  
288  
289 txtRx->AppendText(wxString::Format(wxT("Rx: %s \n"),  
290 wxString::FromUTF8(buf, len)));  
291 // Enable input events again.  
292 sockBase->SetNotify(wxSOCKET_LOST_FLAG | wxSOCKET_INPUT_FLAG);
```

```

293     break;
294 }
295 default:;
296 }

297
298 //update status
299 UpdateStatusBar();
300 }

301
302 void MyFrame::UpdateStatusBar()
303 {
304     fileMenu->Enable(CLIENT_OPEN, !sock->IsConnected());
305     fileMenu->Enable(CLIENT_CLOSE, sock->IsConnected());
306     if (sock->IsConnected()) {
307         //SetStatusText(wxString::Format(wxT("%s:%u"),
308         //    addr.IPAddress(), addr.Service()), 1);
309         SetStatusText(wxString::Format(wxT("Connected")), 1);
310     }
311     else {
312         SetStatusText(wxString::Format(wxT("Not connected")), 1);
313     }
314 }
```

စာရင်း ၃.၃: ce_wx_tcp_client.cpp

ပရိုဂရမ် အစမှာ wxSocketClient တစ်ခု ကို ဖန်တီးပြီး၊ သူအတွက် event handler တွေကို သတ်မှတ်ပါတယ်။

```

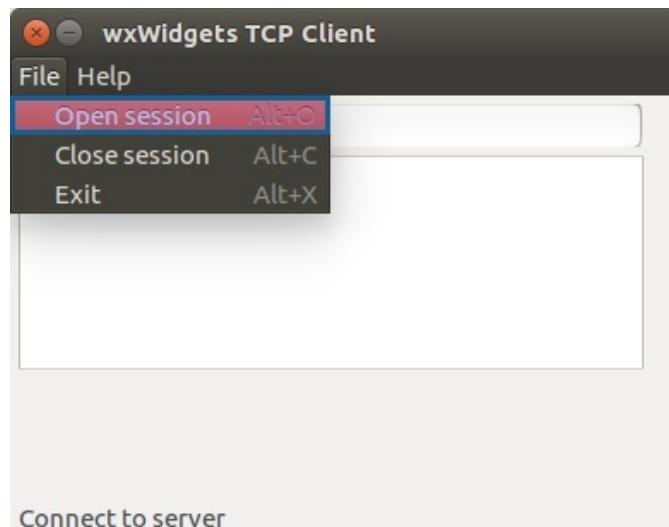
// Create the socket
sock = new wxSocketClient();

// Setup the event handler and subscribe to most events
sock->SetEventHandler( *this, SOCKET_ID );
sock->SetNotify(wxSOCKET_CONNECTION_FLAG |
    wxSOCKET_INPUT_FLAG |
```

```
wxSOCKET_LOST_FLAG);
sock->Notify(true);
```

ပရိုဂရမ် ကို အောက်က command ထွေ သုံးပြီး build နဲ့ run လုပ်ပြီးတဲ့ အခါ ပုံ ၃.၄ မှာ ပြထား သလို File→Open session ကို နှိပ်ပြီး အရင် အပိုင်း က run ထားတဲ့ TCP server ကို ဆက်သွယ်မှု စတင် ပြုလုပ် နိုင် ပါတယ်။ ဆက်သွယ်မှု ကို ပြန်ပို့ ချင်ရင် တော့ Close session ကို နှိပ်နိုင် ပါတယ်။

```
g++ ce_wx_tcp_client.cpp `wx-config --cxxflags --libs` -o ce_wx_tcp_client
gksudo ./ce_wx_tcp_client
```



ပုံ ၃.၄: TCP client မှ open session ပြုလုပ်ပုံ။

ဆက်သွယ်မှု ပြုလုပ်ဖို့ အတွက် လိုင်စာ နဲ့ port number တွေကို သတ်မှတ်ပြီး Connect ဆိုတဲ့ method ကို သုံးပြီး ဆက်သွယ် နိုင် ပါတယ်။

```
IPaddress addr;
addr.Hostname("localhost");
addr.Service(3000);
sock->Connect(addr, false);
```

ဆက်သွယ် မှု ပြုလုပ်ပြီး တဲ့ အခါ ဒေတာ တွေကို Send ခလုတ် နှိပ်ပြီး ပုံ နှင့် ပါတယ်။ အောက်မှာ

ဖော်ပြထားတဲ့ အတိုင်း ပိုမယ့် ဒေတာ တွေရဲ့ ပထမ byte မှာ အရေ အတွက် ကို အရင်ထားပြီး buffer ထဲက ဒေတာ တွေကို နောက်က နေ Write ကိုသုံးပြီး ပိုနိုင်ပါတယ်။

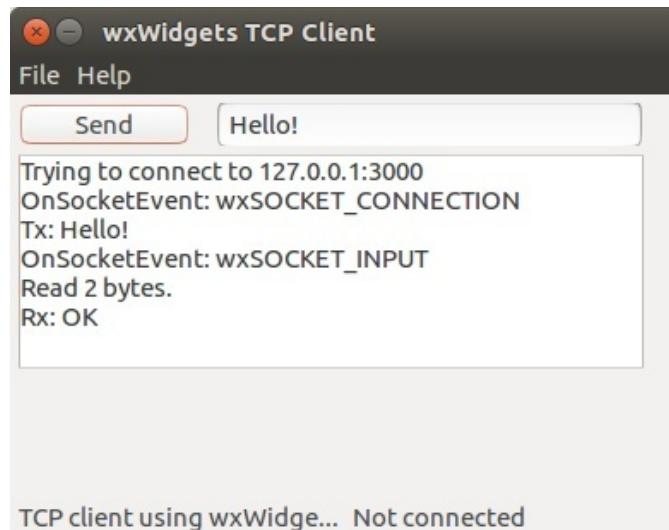
```
void MyFrame::OnSend(wxCommandEvent& WXUNUSED(event))
{
    wxString str = txtSend->GetValue();
    wxCharBuffer buffer = str.ToUTF8();
    size_t txn = str.length();
    unsigned char len;
    len = txn;
    sock->Write(&len, 1); //send the length of the message first
    if (sock->Write(buffer.data(), txn).LastCount() != txn)
    {
        txtRx->AppendText(wxT("Write error.\n"));
        return;
    }
    else {
        txtRx->AppendText("Tx: " + str + "\n");
    }
}
```

OnSocketEvent မှာ wxSOCKET_INPUT ဆိုပြီး ဒေတာ တွေ လက်ခံ ရရှိတဲ့ အခါ အရင် အပိုင်းက TCP server မှာလိုပဲ Read method ကိုသုံးပြီး ဖတ်နိုင်ပါတယ်။ ဆက်သွယ်မှု ကို အဆုံးသတ်ဖို့ အတွက် File→Close session ကို နိုပ်နိုင်ပါတယ်။ ဒေတာ တွေကို ပိုပြီး တဲ့ အခါ server က ပြန်ပို့ တာကို ဖော်ပြထားတာကို Client ပုံ ၃.၅ နဲ့ server ပုံ ၃.၆ မှာ တွေ့နိုင်ပါတယ်။

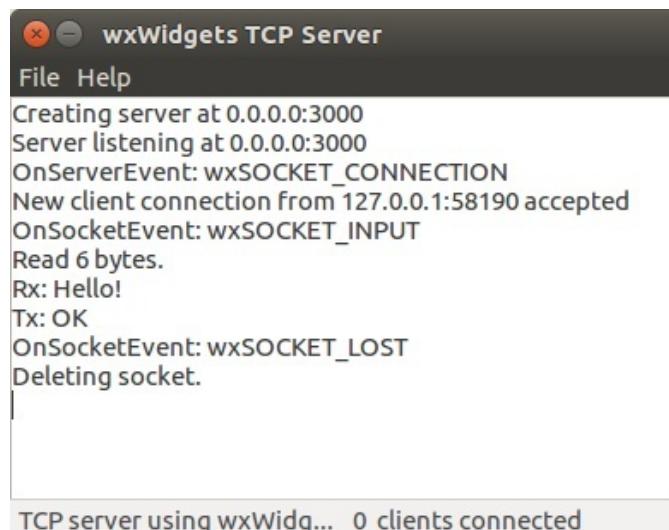
```
wxUint32 lenRd = sockBase->Read(buf, len).LastCount();
```

၇၂

အခန်း ၃. NETWORK ချိတ်ဆက်အသုံးပြုခြင်း



ပုံ ၃.၅: TCP client ဖြင့် ဒေတာ ပို့ခြင်း နှင့် လက်ခံခြင်း။



ပုံ ၃.၆: TCP server တွင် ဒေတာ လက်ခံရရှိပုံ။

အကိုးအကားများ

- [Gar99] Guillermo Rodriguez Garcia. Client for wxSocket demo. 1999. url: <https://github.com/wxWidgets/wxWidgets/blob/master/samples/sockets/client.cpp>.
- [GZ09] Guillermo Rodriguez Garcia and Vadim Zeitlin. Server for wxSocket demo. 2009. url: <https://github.com/wxWidgets/wxWidgets/blob/master/samples/sockets/server.cpp>.
- [SH06] Julian Smart and Kevin Hock. Cross-Platform GUI Programming with wxWidgets. 1st. Pearson Education, Inc., 2006. isbn: 0-13-147381-6.

အခန်း ၄

Serial Port ကိုသုံးခြင်း

Serial communication မှာ data byte တွေရဲ့ bit တွေကို အစီအစဉ်လိုက် တစ်ခုပြီး တစ်ခု ပိုတဲ့ အတွက် ဝါယာ တွေ အများကြီး သုံးစရာ မလို တော့ပဲ ဝါယာ တစ်ခု နဲ့တင် ပိုလို ရတဲ့ အားသာချက် ရှိပါ တယ်။ ပိုတဲ့ အခါ byte ကနေ bit တစ်ခု ချင်းစီ အနေ နဲ့ serial ပြောင်းပြီး ထွက်လာ ဖို့ လက်ခံတဲ့ အခါ တစ်ခု ပြီး တစ်ခု ဝင်လာ တဲ့ serial bit တွေကို byte အဖြစ် ပြန် တည်ဆောက် ဖို့ အတွက် UART (universal asynchronous receiver-transmitter) ကို သုံးကြ ပါတယ်။ UART တစ်ခု မှာ transmit (TX) လို ခေါ်တဲ့ အထွက် တစ်ခု နဲ့ receive (Rx) လို ခေါ်တဲ့ အဝင် တစ်ခု ပါရှိ ပါတယ်။

၄.၁ UART

UART တစ်ခု ဟာ serial bit stream ပြောင်းပေး နိုင်ရုံ တင် မက ပဲ start bit, parity bit, stop bit တွေ ထည့် ပေးခြင်း စတဲ့ လုပ်ငန်း တွေကို လည်း Asynchronous Serial Communication Protocol နဲ့ ကိုက်ညီ အောင် ဖြည့်စွက် လုပ်ဆောင် ပေးပါ တယ်။

၄.၁.၁ Start Bit

UART တစ်ခု က ပိုစရာ data မရှိပဲ idle ဖြစ်နေ ရင် output ကို 1 မှာထား ပါတယ်။ အဲဒီ အခါ လက်ခံ မယ့် UART ဟာ idle ဖြစ်လို ထုတ်ထား တဲ့ 1 လား၊ ပိုပေး နေတဲ့ ဒေတာ က 1 လား ခွဲခြား သိဖို့ လိုလာ ပါတယ်။ အဲဒါကို ဖြောင်းဖို့ အတွက် 0 bit တစ်ခု ကို start bit အနေနဲ့ သုံး ပါတယ်။ လက်ခံ နေတဲ့ UART က 0 ရောက် မလာ မချင်းရှိနေ တဲ့ 1 ကို idle လို ယူဆ ပါတယ်။ ပထမ ဆုံး 0 ရောက်လာ တာနဲ့ အဲဒါ ကို start bit လို ယူဆပြီး နောက်ဝင် လာမယ့် bit တစ်ခု က စပြီး data အနေနဲ့ လက်ခံ ပါတယ်။

၄.၁.၂ Baud Rate

Data bit တစ်ခု နဲ့ တစ်ခု ကြားက အခိုန် အကွာ အထေး ကို baud rate က သတ်မှတ် ပါတယ်။ ဥပမာ baud rate က 1 kHz ဆိုရင်၊ bit တစ်ခု အတွက် time period က $1/(1 \text{ kHz}) = 1 \text{ ms}$ ကြာမှာ ဖြစ် ပါတယ်။ အသုံး များတဲ့ baud rate တွေကတော့ 9600 တို့၊ 115200 တို့ ဖြစ် ပါတယ်။

၄.၁.၃ Data Bits

Data ပိုတဲ့ အခါ ပိုတဲ့ UART နဲ့ လက်ခံ တဲ့ UART တို့ရဲ့ data bit အရေ အတွက် ခြင်း တူဖို့ လို ပါတယ်။ ပိုတဲ့ အခါ 5 bit data ကနေ 8 bit data ထိ ပိုတတ် ပါတယ်။ ဥပမာ ASCII data ဆိုရင် ပိုတာ 7 bit ပဲ ရှိတဲ့ အတွက် လက်ခံ ရင်လဲ 7 bit လက်ခံ ဖို့ လို ပါတယ်။ Data bits တွေကို ပိုတဲ့ အခါ LSB ကနေ အရင် စပို ပါတယ်။ အသုံး များတာ ကတော့ 8 bit ပါ။

၄.၁.၄ Parity Bit

လက်ခံ လိုက်တဲ့ data က မှန်ကန် ကိုက်ညီ မှု ရှိရဲ့ လား ပြန်စစ်ဖို့ အတွက် parity bit ကိုလဲ data bits တွေရဲ့ နောက်မှာ အပို ထည့်ပေး တတ် ပါတယ်။ 1 အရေ အတွက် စုံ ကဏ္ဍး ဖြစ်အောင် ထည့်ပေး ရင် Even parity । မ ကဏ္ဍး ဖြစ်အောင် ပေါင်းထည့် ပေးရင် Odd parity ပါ။ ပိုတဲ့ UART နဲ့ လက်ခံ တဲ့ UART parity ခြင်းလည်း ကိုက်ညီ ဖို့ လို ပါတယ်။ များသော အားဖြင့် Parity ထည့် သုံးလေ့ မရှိ ပါဘူး။

၄.၁.၅ Stop Bit

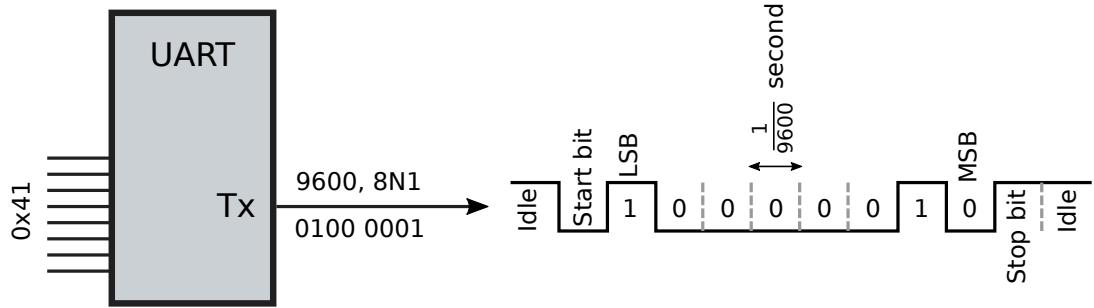
8 bits ရှိတဲ့ data byte တစ်ခု နဲ့ တစ်ခု ကြားမှာ ခြား ပေးဖို့ အတွက် stop bits တွေကို သုံး ပါတယ်။ 1 ကို stop bit အနေ နဲ့ သုံး ပါတယ်။ Stop bit ကို တစ်ခု သို့မဟုတ် နှစ်ခု သုံး လို ရ ပါတယ်။ data တွေ parity bit တွေ ပို့ပြီး တဲ့ အခါ နောက်မှာ 1 တစ်ခု ပေါင်း ပေး၊ ဒါမှ မဟုတ် နှစ်ခု ပေါင်းထည့် ပေး တာပါ။ အသုံး များတာ တော့ stop bit တစ်ခု ထဲ ပါပဲ။

အောက်က ပုံ ၄.၁ မှာ နှမူနာ အနေ နဲ့ data byte 0x41 ကို UART တစ်ခု ကနေ 9600, 8N1 နဲ့ ပိုတာ ကို ပြထား ပါတယ်။ ဆိုလို တာက

- Baud rate = 9600
- Data bit = 8 bits
- Parity = No parity (N= No parity, E= Even parity, O= Odd parity)

- Stop bit= 1 stop bit

လို့ ဆိုလို တာပါ။ No parity ဖြစ်တဲ့ အတွက် data ကို ပိုမြဲး တဲ့ အခါ parity bit ကို မထည့် ပဲ stop bit တန်းလာ ပါတယ်။



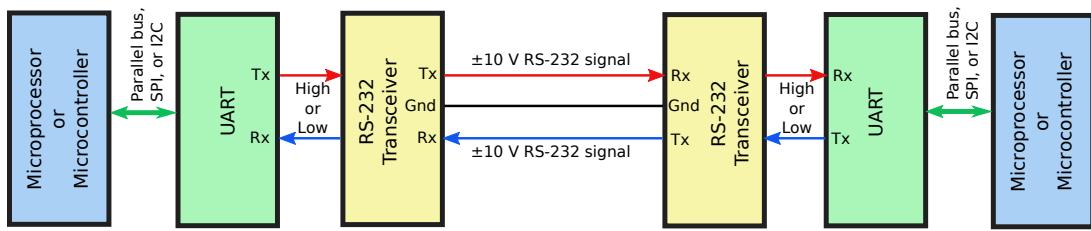
ပုံ ၄.၁: UART တစ်ခု မှ data byte (0x41) ကို serial bit stream (9600, 8N1) အဖြစ် ပြောင်းပေးခြင်း။

၄.J RS-232

UART တစ်ခု က ပုံမှန် အားဖြင့် 1 အတွက် high level voltage (5 V, 3.3 V စတာ တွေ) ထုတ်ပေးပြီး 0 အတွက် ဆိုရင် low level votage (0 V) ထုတ်ပေး ပါတယ်။ များသော အားဖြင့် UART တစ်ခု က ထွက်လာတဲ့ voltage signal ကို external device တွေနဲ့ ဆက်ဖို့ သင့်တော် တဲ့ signaling levels တွေ အဖြစ် ပြောင်းထိုး transceiver တစ်ခုခု ခံပြီး ထုတ်ပေး လေ့ ရှိ ပါတယ်။ အသုံး များတဲ့ standard တစ်ခု ကတော့ RS-232 (Recommended Standard 232) ပါ။ RS-232 transceiver တစ်ခုရဲ့ အလုပ်က 1 အတွက် -10 V နဲ့ 0 အတွက် +10 V အဖြစ် အပြန် အလုန် ပြောင်းပေး တာပါ။ ပုံမှန်က ± 10 V ဆိုပေမယ့် အပေါင်းအနှစ် 3 V ကနေ 25 V အထိက valid ဖြစ်ပါတယ်။ အသုံး များတဲ့ transceiver တွေ ကတော့ MAX232 တို့၊ MAX202 တို့ ဖြစ် ပါတယ်။

ပုံ မှာ RS-232 ဆက်သွယ် အသုံး ပြုပဲ နမေနာ တစ်ခု ကို ပြထား ပါတယ်။ ပုံမှန် အားဖြင့် transmit နဲ့ receive data လိုင်း တွေကို ပဲ သုံးလေ့ ရှိတဲ့ အတွက် Tx တစ်လိုင်း၊ Rx တစ်လိုင်း နဲ့ Ground ဝါယာ တစ်လိုင်း စုစုပေါင်း သုံးလိုင်း သုံးဖို့လိုပါတယ်။

RS-232 ရဲ့ Tx နဲ့ Rx ကို တစ်ခါ တစ်လေ ခဲ့ခြား ဖို့လို လာပြီ ဆိုရင် ကြိုး တွေကို open လုပ်ပြီး မိတာ ကို သုံးပြီး အလွယ် တကူ တိုင်း ကြည့်နိုင် ပါတယ်။ Tx နဲ့ Ground ကြားမှာ ပုံမှန် အားဖြင့် -10V လောက် တွေ့ ရမှာ ဖြစ်ပြီး၊ Rx နဲ့ Ground ကြားမှာ မို့ မရှိ ပါဘူး။ RS232 ကို unbalanced lines လို



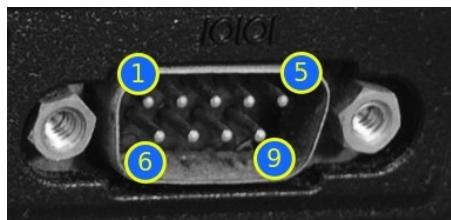
ံ ၄.J: RS-232 ဆက်သွယ် အသုံးပြုခြင်း။

ဆို ပါတယ်။ Tx ဝါယာနဲ့ Ground ဝါယာ ကြေားက ဖိုက transmit voltage ဖြစ်ပြီး Rx ဝါယာ နဲ့ Ground ဝါယာ ကြေားက ဖိုက receive voltage ဖြစ်ပါတယ်။

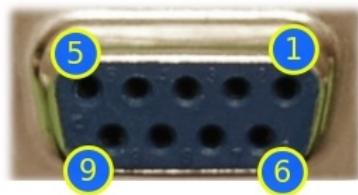
၄.J.၁ Handshaking

RS232 communication မှာ ပုံမှန် အားဖြင့် data transmit line (Tx) နဲ့ data receive line (Rx) ကိုပဲ သုံး ပေမယ့် တစ်ခု နဲ့ တစ်ခု handshaking လုပ်ဖို့ အတွက် control နဲ့ status lines တွေ ကိုလဲ သုံးတတ် ကြ ပါတယ်။ RS232 device တွေ အကြောင်း ပြောတဲ့ အခါ မှာ ကွန်ပျူးတာ ကို DTE (Data Terminal Equipment) လို ခေါ်ပြီး၊ ကွန်ပျူးတာ နဲ့ ဆက်သုံး တဲ့ Modem သို့ device ကို DCE (Data Circuit-Terminating Equipment) လို ခေါ် ပါတယ်။

ံ ၄.၃ မှာ ပြထား တဲ့ အတိုင်း ကွန်ပျူးတာ (DTE) မှာ RS232 အတွက် Male DB9 ပါပြီး သူနဲ့ ဆက်သုံး ရမယ့် (DCE) device မှာ Female DB9 connector ပါလေ့ရှိ ပါတယ်။



(a) Male DB9 connector on DTE



(b) Female DB9 connector on DCE

ံ ၄.၃: RS-232 အတွက် DB9 connector များ။

DTE (ကွန်ပျူးတာ) ဘက်က male connector ရဲ့ pin connection တွေက အောက်ပါ အတိုင်း ဖြစ် ပါတယ်။

1. CD- Carrier Detect (input)

2. Rx- Receive Data (input)
3. Tx- Transmit Data (output)
4. DTR- Data Terminal Ready (output)
5. GND- System Ground
6. DSR- Data Set Ready (input)
7. RTS- Request To Send (output)
8. CTS- Clear To Send (input)
9. RI - Ring Indicator (input)

DCE (Device) ဘက်က female connector ၏ pin connection တွက် တော့ အောက်က စာရင်းမှာ တွေ့နိုင် ပါတယ်။

1. CD- Carrier Detect (output)
2. Tx- Transmit Data (output)
3. Rx- Receive Data (input)
4. DTR- Data Terminal Ready (input)
5. GND- System Ground
6. DSR- Data Set Ready (output)
7. CTS- Clear To Send (input)
8. RTS- Request To Send (output)
9. RI - Ring Indicator (output)

Primary Communication lines တွေ ထဲမှာ ဆို Tx နဲ့ RTS က အထွက် ဖြစ်ပြီး၊ Rx နဲ့ CTS က အဝင် ဖြစ် ပါတယ်။ Status and Control lines တွေမှာ ဆိုရင် တော့ DTR က အထွက် လိုင်း ဖြစ်ပြီး၊ DSR နဲ့ CD က အဝင် လိုင်း ဖြစ် ပါတယ်။

RTS Request To Send signal ကတေသာ DTE ကနေ DCE ကို ပိုချင်တဲ့ အကြောင်း လုမ်း request လုပ်တာပါ။ ပုံမှန် အားဖြင့် logic 1 အနှစ် ဖို့ မှာပဲ ရှိပြီး request လုပ်တာ နဲ့ logic 0 အပေါင်း ဖို့ ထုတ်ပေးပါတယ်။

CTS Clear To Send signal ကတေသာ DCE က ပိုစိုး ready ဖြစ်တဲ့ အတွက် စ ပိုတေသာ ဆိုတဲ့ အကြောင်း DTE ကို အကြောင်း ပြန် တာပါ။ ပုံမှန် အားဖြင့် logic 1 အနှစ် ဖို့ မှာပဲ ရှိပြီး DCE ကနေ အကြောင်း ပြန်ဖို့ logic 0 အပေါင်း ဖို့ ထုတ်ပေးပါတယ်။

DSR DCE Ready (Data Set Ready) ကတေသာ DCE (device) က turn on ဖြစ်ပြီး ready ဖြစ် နေပြီ ဆိုတဲ့ အကြောင်း DTE ကွန်ပျူးတာ ကို လုမ်းပြော တာပါ။ Ready ဖြစ်ပြီ ဆို DCE ကနေ logic 0 အပေါင်း ဖို့ ထုတ်ပေးပါတယ်။

DTR DTE Ready ကတေသာ DTE ကွန်ပျူးတာ ကနေ communication ကို လုပ်ချင် လို့ အသုံး ပြင်တေသာ လို့ DCE (device) ကို လုမ်းပြော တာပါ။ ပုံမှန် အားဖြင့် logic 1 အနှစ် ဖို့ မှာပဲ ရှိပြီး DTE ကနေ DTR လိုင်းကို true လုပ်တဲ့ အခါ logic 0 အပေါင်း ဖို့ ထုတ်ပေးပါတယ်။

CD Carrier Detect (Received Line Signal Detector) ကတေသာ DCE က modem ဖြစ်တဲ့ အခါ မှာ သုံးပါတယ်။ တယ်လီဖုန်း လိုင်း ကို တဗြား ဘက် အဝေးမှာ ရှိတဲ့ modem က ဖြေ လိုက်တဲ့ answer tone ကို ရတဲ့ အခါမှာ ဒီဖက် modem က ကွန်ပျူးတာ ကို logic 0 အပေါင်း ဖို့ ထုတ်ပေးပါတယ်။

RI Ring Indicator ကတေသာ ring signal ကို ရတဲ့ အခါ modem က ကွန်ပျူးတာ ကို logic 0 အပေါင်း ဖို့ ထုတ်ပေးပါတယ်။ Ring signal ရတဲ့ အချိန်ပဲ အပေါင်း ထုတ် ပေးပြီး ring signal အချင်းချင်း ကြားနဲ့ ring signal မရှိတဲ့ အချိန် တွေမှာ အနှစ် ဖြစ်နေ ပါတယ်။

၄.၃ Programming Serial Port

serial port ကို အသုံးပြု တဲ့ အကြောင်း ကျေနော် ဖန်တီး ထားတဲ့ C++ class library လေး တစ်ခု နဲ့ နမူနာ သုံးပြီး ပြောချင် ပါတယ် [Wik16; Lab17; Den95]။ Cross-platform ဖြစ်ပြီး Windows ရော့ Linux မှာပါ သုံးနိုင် ပါတယ်။ ပထမ နမူနာ အနေနဲ့ ရှိုးရွှေ့တဲ့ C++ console program လေး တစ်ခု ကို ဖော်ပြ ထားပြီး၊ နောက်ထပ် GUI application နမူနာ အနေနဲ့ wxWidgets ကို သုံးထား တဲ့ program ကိုပါ ဖော်ပြထားပါတယ်။ Serial class ရဲ့ source code ([ce_serial.h](#)) ကို နောက်ဆက်တဲ့ B - စာရင်း [J.C](#) မှာ တွေ့နှိုင် ပါတယ်။

၄.၃.၁ Console

ကွန်ပျိုတာ ရဲ့ serial port ကို ဖွံ့ဖြိုး ဒေတာ ပို့ ပြီးတော့ character တစ်လုံး ကို ပြန် ဖတ်ပြ တဲ့ ရှိုးရှင်းတဲ့ serialcon.cpp ဆိုတဲ့ နူမူနာ လေးကို စာရင်း ၄.၁ မှာ ပြထား ပါတယ်။

```

1 //File: serialcon.cpp
2 //Description: Serial communication console program for Windows and Linux
3 //WebSite: http://cool-emerald.blogspot.sg/2017/05/serial-port-programming-in
4 //           -c-with.html
5 //MIT License (https://opensource.org/licenses/MIT)
6 //Copyright (c) 2017 Yan Naing Aye
7
8
9 #include<stdio.h>
10 #include "ce_serial.h"
11 using namespace std;
12
13 int main()
14 {
15
16 #if defined (__WIN32__) || defined(_WIN32) || defined(WIN32) || defined(
17     __WINDOWS__) || defined(__TOS_WIN__)
18     Serial com("\\\\.\COM1",9600,8,'N',1); //Windows
19 #else
20     Serial com("/dev/ttyS0",9600,8,'N',1); //Linux
21 #endif
22
23     printf("Opening port %s.\n",com.GetPort().c_str());
24     if (com.Open() == 0) {
25         printf("OK.\n");
26     }
27     else {
28         printf("Error.\n");
29         return 1;
30     }
31
32     bool successFlag;
33     printf("Writing.\n");
34     char s[]="Hello";

```

```

31 successFlag=com.Write(s); //write string
32 successFlag=com.WriteChar('!'); //write a character
33
34 printf("Waiting 3 seconds.\n");
35 delay(3000); //delay 5 sec to wait for a character
36
37 printf("Reading.\n");
38 char c=com.ReadChar(successFlag); //read a char
39 if(successFlag) printf("Rx: %c\n",c);
40
41 printf("Closing port %s.\n",com.GetPort().c_str());
42 com.Close();
43 return 0;
44 }
```

စာရင်း ၄.၁: serialcon.cpp

အစ မှာ #include "Serial.h" လို့ ကြော်ပြီး Serial class ကို ထည့်လိုက် ပါတယ်။ ပြီးတော့ 'com' လို့ ခေါ်တဲ့ object တစ်ခု ကို port number, baud rate စတဲ့ တန်ဖိုး တွေနဲ့ ကြော်ပါတယ်။ ဒီနမူနာ အတွက် ဆိုရင် Windows မှာ COM1 ကို သုံးမှာ ဖြစ်ပြီး Linux မှာ ttyS0 ကို သုံးမှာ ဖြစ် ပါတယ်။ Linux မှာ USB device ကို သုံးမယ် ဆိုရင်တော့ ttyUSB0 ဖြစ်ကောင်း ဖြစ်နိုင် ပါတယ်။ Parity အတွက် က 'N','E','O','M', သို့ 'S' တို့ကို none, even, odd, mark, နဲ့ space တို့ အတွက် သုံးလို့ ရ ပါတယ်။

Comm port ကို ဖွင့်ဖို့ အတွက် com.Open() ကို သုံးနိုင် ပါတယ်။ Write နဲ့ WriteChar methods တွေကို သုံးပြီး null terminated string ဒါမှမဟုတ် character တစ်လုံး ကို ပို့နိုင် ပါတယ်။ ReadChar method နဲ့ character ကို ဖတ်တဲ့ အခါ စေနေ ပဲ non-blocking ပုံစံ နဲ့ ဖတ်မှာ ဖြစ်တဲ့ အတွက် character ရှိ မနေရင် successFlag က false လို့ ပြ ပါမယ်။ Close method နဲ့ com ကို ပို့ပို့ ပါတယ်။

Control နဲ့ status လိုင်းတွေ ကိုလည်း အသုံးပြု နိုင် ပါတယ်။ RTS နဲ့ DTR lines တွေကို control လုပ်နိုင် ပြီး CTS တို့၊ DSR တို့လို့ Status lines တွေကိုလည်း ဖတ်နိုင်ပါတယ်။ ဒီ ပရိုဂရမ် "serialcon.cpp" ကို Ubuntu Linux ပေါ်မှာ အောက်က အတိုင်း compiled လုပ်၊ run လုပ်နိုင် ပါတယ်။

```

g++ serialcon.cpp Serial.h -o serialcon
sudo ./serialcon
```

Windows ပေါ်မှာတော့ Visual Studio ဒါမှမဟုတ် tdm-gcc ထို့ပါ mingw တို့လို compiler တစ်ခုခဲ့ကို သုံးပြီးအောက်က လိုပြီး compile လုပ်ပြီး run နိုင် ပါတယ်။

```
g++ serialcon.cpp Serial.h -o serialcon.exe -std=c++11
.\serialcon.exe
```

၄.၃.၂ GUI

နောက်ထပ် နမူနာ တစ်ခု အနေနဲ့ wxWidgets နဲ့ GUI application တစ်ခု ဖန်တီးပြီး serial port ကို အသုံးပြု တဲ့ အကြောင်း ဆွေးနွေး ပါမယ်။ File menu ကနေ သုံးမယ့် serial port ၊ baud rate စတာ တွေကို ရွေးချယ် သတ်မှတ် လို ရပြီး၊ လက်ခံ ရရှိ တဲ့ ဒေတာ တွေကို text box မှာ ဖော်ပြ ပေးနိုင် ပါတယ်။ အပေါ်ဘက် က text box ထဲမှာ ပို့ချင် တဲ့ text ကို ထည့်ပြီး send ခလုတ် ကို နှိပ်ပြီး ပို့နိုင် ပါတယ်။ Control နဲ့ status လိုင်းတွေ ကို အသုံးပြု ပုံကိုပါ ပြထား ပါတယ်။ ဒါ `wxserial.cpp` ဆိုတဲ့ နမူနာ ပရိုဂရမ် ကို စာရင်း ၄.၂ မှာ ဖော်ပြ ထား ပါတယ်။

```
1 //File: wxserial.cpp
2 //Description: Serial communication for wxWidgets
3 //WebSite: http://cool-emerald.blogspot.sg/2017/05/serial-port-programming-in
4 //           -c-with.html
5 //MIT License (https://opensource.org/licenses/MIT)
6 //Copyright (c) 2017 Yan Naing Aye
7
8 // For compilers that support precompilation, includes "wx/wx.h".
9
10 #include "wx/wxprec.h"
11
12 #ifdef __BORLANDC__
13     #pragma hdrstop
14 #endif
15
16 // for all others, include the necessary headers (this file is usually all
17 // you
18
19 // need because it includes almost all "standard" wxWidgets headers)
20 #ifndef WX_PRECOMP
21     #include "wx/wx.h"
```

```
18 #endif
19
20 // the application icon (under Windows and OS/2 it is in resources and even
21 // though we could still include the XPM here it would be unused)
22 #ifndef wxHAS_IMAGES_IN_RESOURCES
23     #include "sample.xpm"
24 #endif
25
26 #include <wx/numdlg.h>
27
28 #include "ce_serial.h"
29
30 // Define a new application type, each program should derive a class from
31 // wxApp
32 class MyApp : public wxApp
33 {
34 public:
35     virtual bool OnInit();
36 };
37
38 class MyFrame : public wxFrame
39 {
40 public:
41     // ctor(s)
42     MyFrame(const wxString& title);
43     wxButton *btnSend;
44     wxTextCtrl *txtSend;
45     Serial com;
46     wxTimer m_timer;
47     wxTextCtrl *txtRx;
48     wxCheckBox *chkRTS;
49     wxCheckBox *chkDTR;
50     wxCheckBox *chkCTS;
51     wxCheckBox *chkDSR;
52     wxCheckBox *chkRI;
53     wxCheckBox *chkCD;
```

```
53 // event handlers (these functions should _not_ be virtual)
54 void OnQuit(wxCommandEvent& event);
55 void OnAbout(wxCommandEvent& event);
56 void OnOpen(wxCommandEvent& event);
57 void OnClose(wxCommandEvent& event);
58 void SelPort(wxCommandEvent& event);
59 void SetDataSize(wxCommandEvent& event);
60 void SetParity(wxCommandEvent& event);
61 void SetStopBits(wxCommandEvent& event);
62 void SetBaud(wxCommandEvent& event);
63 void OnSend(wxCommandEvent& event);
64 void OnTimer(wxTimerEvent& event);
65 void ProcessChar(char ch);
66 void ClearText(wxCommandEvent& event);
67 void OnChkRTS(wxCommandEvent& event);
68 void OnChkDTR(wxCommandEvent& event);
69 void UpdateCommStatus();
70 private:
71
72 };
73
74 // IDs for the controls and the menu commands
75 const int ID_BTNSEND = 101;
76 const int ID_TXTSEND = 102;
77 const int ID_CHKRTS = 103;
78 const int ID_BAUDRATE = 103;
79 const int ID_TIMER = 104;
80 const int ID_TXTRX = 105;
81 const int ID_CHKDTR = 106;
82 const int ID_SELPORT = 107;
83 const int ID_CHKCTS = 108;
84 const int ID_CHKDSR = 109;
85 const int ID_CHKRI = 110;
86 const int ID_CHKCD = 111;
87 const int ID_DATASIZE = 112;
88 const int ID_PARITY = 113;
```

```
89 const int ID_STOPBITS = 114;  
90  
91 enum  
92 {  
93     Button_Send = ID_BTNSEND,  
94     Txt_Send = ID_TXTSEND,  
95     Chk_RTS = ID_CHKRTS,  
96     Serial_Baud = ID_BAUDRATE,  
97     Timer1 = ID_TIMER,  
98     Txt_Rx = ID_TXTRX,  
99     Chk_DTR = ID_CHKDTR,  
100    Serial_Port = ID_SELPORT,  
101    Serial_DataSize = ID_DATASIZE,  
102    Serial_Parity = ID_PARITY,  
103    Serial_StopBits = ID_STOPBITS,  
104    Txt_Clear = wxID_CLEAR,  
105    Serial_Open = wxID_OPEN,  
106    Serial_Close = wxID_CLOSE,  
107    Minimal_Quit = wxID_EXIT,  
108  
109    Minimal_About = wxID_ABOUT  
110  
111};  
112  
113 IMPLEMENT_APP(MyApp)  
114 // 'Main program' equivalent: the program execution "starts" here  
115 bool MyApp::OnInit()  
116 {  
117     // call the base class initialization method, currently it only parses a  
118     // few common command-line options but it could be do more in the future  
119     if (!wxApp::OnInit())  
120         return false;  
121  
122     // create the main application window  
123     MyFrame *frame = new MyFrame(wxT("Serial Com"));  
124
```

```

125 // and show it (the frames, unlike simple controls, are not shown when
126 // created initially)
127 frame->Show(true);
128
129 // success: wxApp::OnRun() will be called which will enter the main
130 // message
131 // loop and the application will run. If we returned false here, the
132 // application would exit immediately.
133 return true;
134
135 // frame constructor
136 MyFrame::MyFrame(const wxString& title)
137     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(390, 280),
138             wxDEFAULT_FRAME_STYLE ^ wxRESIZE_BORDER), m_timer(this, ID_TIMER)
139 {
140     // set the frame icon
141     SetIcon(wxICON(sample));
142
143 #if wxUSE_MENUS
144     // create a menu bar
145     wxMenu *fileMenu = new wxMenu();
146
147     //Edit menu
148     wxMenu *editMenu = new wxMenu();
149
150     // the "About" item should be in the help menu
151     wxMenu *helpMenu = new wxMenu();
152
153     helpMenu->Append(Minimal_About, wxT("&About\tF1"), wxT("Show about dialog
154     "));
155     fileMenu->Append(Serial_Open, wxT("&Open\tAlt-0"), wxT("Open serial port
156     "));
157     fileMenu->Append(Serial_Close, wxT("&Close\tAlt-C"), wxT("Close serial port
158     "));
159     editMenu->Append(Txt_Clear, wxT("Clea&r\tAlt-R"), wxT("Clear text"));

```

```

157     fileMenu->Append(Serial_Port, wxT("&Serial Port\tAlt-S"), wxT("Select
158         serial port"));
159     fileMenu->Append(Serial_Baud, wxT("&Baud Rate\tAlt-B"), wxT("Set baud rate"
160         ));
161     fileMenu->Append(Serial_DataSize, wxT("&Data Size\tAlt-D"), wxT("Set data
162         size"));
163     fileMenu->Append(Serial_Parity, wxT("&Parity\tAlt-P"), wxT("Set parity"));
164     fileMenu->Append(Serial_StopBits, wxT("S&top Bits\tAlt-t"), wxT("Set stop
165         bits"));
166     fileMenu->Append(Minimal_Quit, wxT("E&xit\tAlt-X"), wxT("Quit this program"
167         ));
168
169
170
171     // now append the freshly created menu to the menu bar...
172
173     wxMenuBar *menuBar = new wxMenuBar();
174
175     menuBar->Append(fileMenu, wxT("&File"));
176     menuBar->Append(editMenu, wxT("&Edit"));
177     menuBar->Append(helpMenu, wxT("&Help"));
178
179     // ... and attach this menu bar to the frame
180     SetMenuBar(menuBar);
181
182 #endif // wxUSE_MENUS
183
184
185 #if wxUSE_STATUSBAR
186
187     // create a status bar just for fun (by default with 1 pane only)
188     CreateStatusBar(2);
189     SetStatusText(wxT("Serial Communication"));
190
191 #endif // wxUSE_STATUSBAR
192
193     btnSend = new wxButton(this, Button_Send, wxT("Send"), wxPoint(5, 5), wxSize
194         (100, 25));
195
196     txtSend = new wxTextCtrl(this, Txt_Send, wxT("Hello!"), wxPoint(120,5), wxSize
197         (250,25));
198
199     //lblRx = new wxStaticText(this, ID_LBLRX, wxT("Rx:"), wxPoint(5, 75),
200     wxSize(35, 25));
201
202     txtRx = new wxTextCtrl(this, Txt_Rx, wxT(""), wxPoint(5, 35), wxSize(365,
203         125), wxTE_MULTILINE);

```

```

184 chkRTS = new wxCheckBox(this, Chk_RTS, wxT("RTS"), wxPoint(5, 170),
185     wxDefaultSize);
186 chkDTR = new wxCheckBox(this, Chk_DTR, wxT("DTR"), wxPoint(55, 170),
187     wxDefaultSize);
188 chkCTS = new wxCheckBox(this, ID_CHKCTS, wxT("CTS"), wxPoint(155, 170),
189     wxDefaultSize);
190 chkDSR = new wxCheckBox(this, ID_CHKDSR, wxT("DSR"), wxPoint(205, 170),
191     wxDefaultSize);
192 chkRI = new wxCheckBox(this, ID_CHKRI, wxT("RI"), wxPoint(255, 170),
193     wxDefaultSize);
194 chkCD = new wxCheckBox(this, ID_CHKCD, wxT("CD"), wxPoint(305, 170),
195     wxDefaultSize);

196 Connect(Button_Send, wxEVT_COMMAND_BUTTON_CLICKED,wxCommandEventHandler(
197     MyFrame::OnSend));
198 Connect(Minimal_About,wxEVT_COMMAND_MENU_SELECTED,wxCommandEventHandler(
199     MyFrame::OnAbout));
200 Connect(Minimal_Quit,wxEVT_COMMAND_MENU_SELECTED,wxCommandEventHandler(
201     MyFrame::OnQuit));
202 Connect(Serial_Open,wxEVT_COMMAND_MENU_SELECTED,wxCommandEventHandler(
203     MyFrame::OnOpen));
204 Connect(Serial_Close,wxEVT_COMMAND_MENU_SELECTED,wxCommandEventHandler(
205     MyFrame::OnClose));
206 Connect(Serial_Port, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
207     MyFrame::SelPort));
208 Connect(Serial_Baud, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
209     MyFrame::SetBaud));
210 Connect(Serial_DataSize, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
211     MyFrame::SetDataSize));
212 Connect(Serial_Parity, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
213     MyFrame::SetParity));
214 Connect(Serial_StopBits, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
215     MyFrame::SetStopBits));
216 Connect(Timer1,wxEVT_TIMER, wxTimerEventHandler(MyFrame::OnTimer));
217 Connect.Txt_Clear, wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
218     MyFrame::ClearText));

```

```

203 Connect(Chk_RTS, wxEVT_COMMAND_CHECKBOX_CLICKED, wxCommandEventHandler(
204     MyFrame::OnChkRTS));
205 Connect(Chk_DTR, wxEVT_COMMAND_CHECKBOX_CLICKED, wxCommandEventHandler(
206     MyFrame::OnChkDTR));
207 //Bind(wxEVT_MENU, &MyFrame::OnClose, this, Serial_Close);
208 m_timer.Start(250);
209 chkCTS->Disable();
210 chkDSR->Disable();
211 chkRI->Disable();
212 chkCD->Disable();
213 }
214
215 void MyFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
216 {
217     // true is to force the frame to close
218     Close(true);
219 }
220
221 void MyFrame::OnAbout(wxCommandEvent& WXUNUSED(event))
222 {
223     wxMessageBox(wxString::Format(
224         wxT("Serial Communication! \n "))
225         wxT("Author: Yan Naing Aye \n ")
226         wxT("Web: https://github.com/yan9a/serial")
227         ),
228     wxT("About Serial Comm"),
229     wxOK | wxICON_INFORMATION,
230     this);
231 }
232
233 void MyFrame::OnOpen(wxCommandEvent& WXUNUSED(event))
234 {
235     if(com.Open()) txtRx->AppendText(wxString::Format(wxT("Error opening port %
236         s.\n"), com.GetPort())));

```

```

236     else txtRx->AppendText(wxString::Format(wxT("Port %s is opened.\n"), com.
237         GetPort())));
238
239 void MyFrame::OnClose(wxCommandEvent& WXUNUSED(event))
240 {
241     com.Close();
242     txtRx->AppendText(wxString::Format(wxT("Port %s is closed.\n"), com.GetPort()
243         ()));
244
245 void MyFrame::SelPort(wxCommandEvent& WXUNUSED(event))
246 {
247     if (com.IsOpened()) {
248         txtRx->AppendText(wxString::Format(wxT("Close Port %s first.\n"), com.
249             GetPort()));
250     }
251     else {
252         wxString cdev=wxString::Format(wxT("%s"), com.GetPort());
253         wxString device = wxGetTextFromUser(wxT("Enter the port"), wxT("Set Port"
254             ), cdev);
255         string str = device.ToStdString();
256         if (str.length() > 0) {
257             com.SetPort(str);
258         }
259         txtRx->AppendText(wxString::Format(wxT("Port: %s\n"), com.GetPort()))
260         ;
261     }
262 }
263
264 void MyFrame::SetParity(wxCommandEvent& WXUNUSED(event))
265 {
266     if (com.IsOpened()) {
267         txtRx->AppendText(wxString::Format(wxT("Close Port %s first.\n"), com.
268             GetPort()));

```

```
266 }
267 else {
268     wxString cdev = wxString::Format(wxT("%c"), com.GetParity());
269 #if defined(__WINDOWS__)
270     wxString parity = wxGetTextFromUser(wxT("Enter the parity ( N, E, O, M,
271 or S )"), wxT("Set Parity"), cdev);
272 #else
273     wxString parity = wxGetTextFromUser(wxT("Enter the parity ( N, E, or O )"
274 ), wxT("Set Parity"), cdev);
275 #endif
276
277     string pstr = parity.ToStdString();
278     if (pstr.length() > 0) {
279         com.SetParity(pstr.at(0));
280     }
281     txtRx->AppendText(wxString::Format(wxT("Parity: %c\n"), com.GetParity()));
282 }
283
284 void MyFrame::SetBaud(wxCommandEvent& WXUNUSED(event))
285 {
286     if (com.IsOpened()) {
287         txtRx->AppendText(wxString::Format(wxT("Close port %s first.\n"), com.
288             GetPort()));
289     }
290     else {
291         long n = wxGetNumberFromUser(wxT("Enter the baud rate"), wxT("Baud rate")
292             , wxT("Set Baud Rate"), com.GetBaudRate(), 0, 1000000);
293         if (n >= 0) {
294             com.SetBaudRate(n);
295         }
296         txtRx->AppendText(wxString::Format(wxT("Baud rate: %ld\n"), com.
297             GetBaudRate()));
298     }
299 }
```

```
296
297 void MyFrame::SetDataSize(wxCommandEvent& WXUNUSED(event))
298 {
299     if (com.IsOpened()) {
300         txtRx->AppendText(wxString::Format(wxT("Close port %s first.\n"), com.
301             GetPort()));
302     }
303     else {
304         long n = wxGetNumberFromUser(wxT("Enter the data size"), wxT("Data Size")
305             , wxT("Set Data Size"), com.GetDataSize(), 5, 8);
306         if (n >= 0) {
307             com.SetDataSize(n);
308         }
309         txtRx->AppendText(wxString::Format(wxT("Data size: %ld\n"), com.
310             GetDataSize()));
311     }
312 }
313
314 void MyFrame::SetStopBits(wxCommandEvent& WXUNUSED(event))
315 {
316     if (com.IsOpened()) {
317         txtRx->AppendText(wxString::Format(wxT("Close port %s first.\n"), com.
318             GetPort()));
319     }
320     else {
321         long n = wxGetNumberFromUser(wxT("Enter the number of stop bits"), wxT("Data
322             Size"), wxT("Set stop bits"), long(com.GetStopBits()), 1, 2);
323         if (n > 0) {
324             com.SetStopBits(float(n));
325         }
326         txtRx->AppendText(wxString::Format(wxT("Stop bits: %ld\n"), long(com.
327             GetStopBits())));
328     }
329 }
330
331 void MyFrame::OnSend(wxCommandEvent& WXUNUSED(event))
```

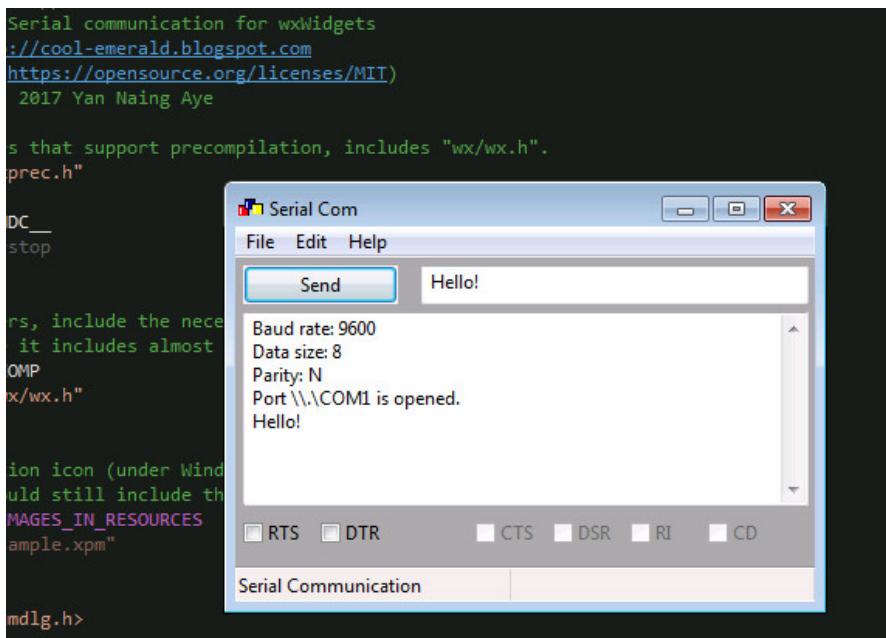
```
326 {
327     wxString str = txtSend->GetValue();
328     wxCharBuffer buffer = str.ToUTF8();
329     if (com.Write(buffer.data())) {
330         txtRx->AppendText(str);
331     }
332     else {
333         txtRx->AppendText(wxT("Write error.\n"));
334     }
335 }
336
337 void MyFrame::OnTimer(wxTimerEvent& WXUNUSED(event))
338 {
339     char ch; bool r;
340     do {ch = com.ReadChar(r);if (r) ProcessChar(ch);} while (r);
341     UpdateCommStatus();
342 }
343
344 void MyFrame::ProcessChar(char ch)
345 {
346     txtRx->AppendText(wxString::Format(wxT("%c"), ch));
347 }
348
349 void MyFrame::ClearText(wxCommandEvent& WXUNUSED(event))
350 {
351     txtRx->Clear();
352 }
353
354 void MyFrame::OnChkRTS(wxCommandEvent& WXUNUSED(event))
355 {
356     if (!com.SetRTS(chkRTS->IsChecked())) {
357         txtRx->AppendText(wxT("RTS error.\n"));
358     }
359 }
```

```

362 {
363     if (!com.SetDTR(chkDTR->IsChecked())) {
364         txtRx->AppendText(wxT("DTR error.\n"));
365     }
366 }
367
368 void MyFrame::UpdateCommStatus()
369 {
370     bool s;
371     bool v;
372     v = com.GetCTS(s);
373     if (s) chkCTS->SetValue(v);
374     v = com.GetDSR(s);
375     if (s) chkDSR->SetValue(v);
376     v = com.GetRI(s);
377     if (s) chkRI->SetValue(v);
378     v = com.GetCD(s);
379     if (s) chkCD->SetValue(v);
380 }
```

စာရင်း ၄.J: wxserial.cpp

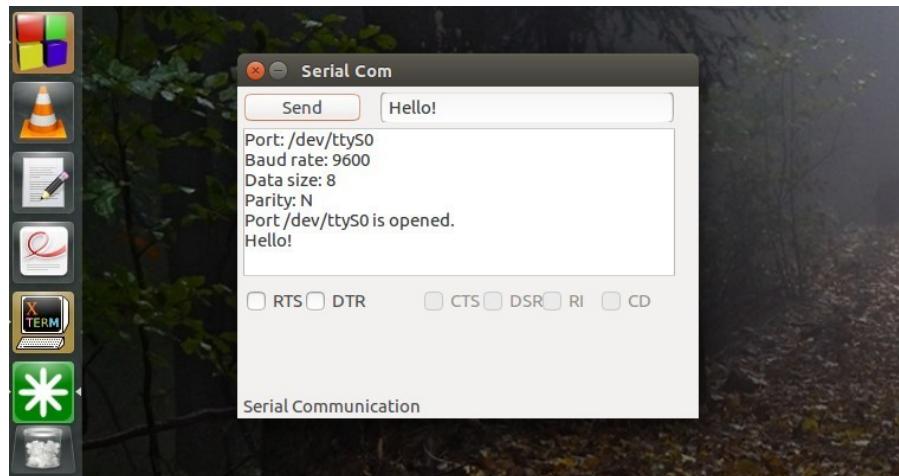
Windows ပေါ်မှာ Visual Studio 2017 နဲ့ wxserial.cpp ကို run လိုက်တဲ့ အခါ ပုံ ၄.၄ မှာ ပြထားသလို တွေ့နိုင် ပါတယ်။



ံ ၄.၄: Serial class သုံးထားသော wxWidgets GUI application ကို Visual Studio 2017 ဖွင့် run ခြင်း။

အဲဒီ wxserial.cpp program ကိုပဲ ဘာမှ မပြင်ပဲ Debian အခြေဖြူ Linux တွေပေါ်မှာ အောက်ကအတိုင်း builtလုပ်၊ run လုပ်လိုဂျပ်တယ်။ ပရိုကရမ် ရဲ့ GUI ကို ပုံ မှာ တွေ့နိုင် ပါတယ်။

```
g++ wxserial.cpp Serial.h wx-config --cxxflags --libs -o wxserial -DNDEBUG
sudo ./wxserial
```



ဦး ၄.၂: Serial class သုံးထားသော wxWidgets GUI application ကို Ubuntu ပေါ်တွင် run ခြင်း။

အကိုးအကားများ

- [Den95] Allen Denver. Serial Communications. 1995. url: <https://msdn.microsoft.com/en-us/library/ff802693.aspx>.
- [Lab17] Silicon Labs. AN197: Serial Communications Guide for the CP210x. 2017. url: <https://www.silabs.com/documents/public/application-notes/an197.pdf>.
- [Wik16] Wikibooks. Serial Programming/termios. 2016. url: https://en.wikibooks.org/wiki/Serial_Programming/termios.

အခန်း ၅

Multithreading

ဒီ အခန်း မှာ thread တွက် wxWidgets နဲ့ ဘယ်လို ထိန်းချုပ် မလဲ ဆိုတာ ရယ်၊ multithreading အစား ရွေးချယ် လို့ရတဲ့ အခြား နည်းလမ်း တွက် အေးနေး ပါမယ်။

၅.၁ ဘယ်လို အချိန်တွေမှာ သုံးသင့်၊ မသုံးသင့်

Thread ဆိုတာ program ကို လုပ်ဆောင် သွားတဲ့ လမ်းကြောင်း လို အခြေခံ အားဖြင့် ပြောနိုင် ပါတယ်။ Thread တွက် process အပေါ်စား တွေလို လည်း ပြောကြ ပါတယ်။ သူတို့ ရဲ့ ကွာခြား ချက်က process တွေမှာ မတူညီ တဲ့ memory space တွေရှိကြပြီး၊ process တစ်ခု ရဲ့ thread တွေက တော့ memory address space တစ်ခု ထဲကို မျှသုံး ကြပါတယ်။ Multithreading သုံးတဲ့ အခါ အဖြစ် များတဲ့ data structure တွက် ပြုပြင်တဲ့ သုံးမိ တတ်တဲ့ ပြဿနာ ရှိတာ မို့ mutex တွေ၊ critical section တွက် သုံးဖို့ လိုတတ် ပါတယ်။

Multithreading ကို သင့်တော် သလို သုံးမယ် ဆိုရင် user interface နဲ့ program ရဲ့ နောက်ခံ အလုပ် တွေကို သီးသန် စီ ခွဲပြီး လုပ်ဆောင် နိုင်တာ မို့ ပိုပြီး responsive ဖြစ်တဲ့ user interface ကို ရနိုင် ပါတယ်။

wxWidgets မှာ thread class တစ်ခု ရော့ လိုအပ် တဲ့ synchronization object တွေဖြစ်တဲ့ mutexes နဲ့ critical sections with conditions တွေပါ ပါရှိ ပါတယ်။ wxWidgets ရဲ့ threading API က pthreads လို လူသီများတဲ့ POSIX threading API နဲ့ ဆင်တူ ပါတယ်။ အဲဒီ class တွေ သုံးတဲ့ အခါ native thread API သုံးတာ နဲ့ ယုံးရင် multithreading ကို ရေးရ တာ ပိုလွယ် သွားပြီး၊ error checking အတွက် လည်း အထောက် အပံ့ တွေ ပိုရ တာကို တွေ့ရ မှာပါ။

Application တစ်ခု မှာ multithreading သုံးလိုက် ပြီ ဆိုတာနဲ့ မှားနှင့် တာတွေ၊ သတိထား စရာ တွေ၊ ပိုမို ရှုပ်တွေ၊ မူတွေ တဲ့ ပါလာ တာမို့ မသုံးခင် အခြား နည်းလမ်း နဲ့ ဖြေရှင်း လို့ ရမရ သေချာ စဉ်းစား သင့် ပါတယ်။ ဥပမာ progress dialog ကို ရှုပ်တွေ၊ ကြာမြင့် တဲ့ တွက်ချက်မှု အတွက် ပြရုံ အတွက် ဆိုရင်၊ idle handler တစ်ခု မှာ တွက်ပြီး wxWindow::Update ကို ပုံမှန် ခေါ်ပေး နေရှုံး နဲ့ ရရှင် ပါတယ်။

နောက်တစ်ခါ GUI function တွေကို ခေါ်သုံး တဲ့ အခါ thread တစ်ခု ထဲက ပဲ ခေါ်သုံး သင့် ပါတယ်။ ကျွန်ုတဲ့ worker thread တွေက main thread ကို event နဲ့ ပဲ ဆက်သွယ် တာက အများကြီး ပို စိတ်ချ ရပါတယ်။ wxEvtHandler::AddPendingEvent သို့မဟုတ် သူ့ရဲ့ အတိုကောက် wxPostEvent ဆိုတဲ့ function တွေက thread safe ဖြစ်တာ မို့ thread အချင်းချင်း ဆက်သွယ် ဖို့ သုံးနှင့် ပါတယ်။

၅.၂ wxThread သုံးခြင်း

Thread တစ်ခု ကို အသုံးပြု ဖို့ အတွက် wxThread ကနေ derive လုပ်ထား တဲ့ class တစ်ခု ကို အသုံးချဖို့ လိုပြီး၊ အဲဒီ thread ထဲမှာ လုပ်ဆောင် ချင်တဲ့ လုပ်ငန်း တွေကို Entry ဆိုတဲ့ method ထဲမှာ လုပ်ဆောင် ရမှာ ဖြစ်ပါတယ်။ ဥပမာ နံပါတ် တစ်ခု ကို thread တစ်ခု နဲ့ သပ်သပ် ရေတွက် မယ့် MyThread ဆိုတဲ့ class တစ်ခု ကို အောက်ပါ အတိုင်း ကြော်နိုင် ပါတယ်။

```
class MyThread : public wxThread
{
public:
    MyThread();
    virtual ~MyThread();

    // thread execution starts here
    virtual void *Entry();

private:
    int m_count;
};
```

၅.၂.၁ Thread တစ်ခု ဖန်တီးခြင်း

Thread တစ်ခု ကို ဖန်တီးဖို့ အဆင့် နှစ်ဆင့် လိုပါတယ်။ စစ်ဆေး object ကို instance တစ်ခု ဖန်တီး ဖို့ပြီး၊ နောက် တစ်ဆင့် မှာ Create method ကို ခေါ်ဖို့ လိုပါတယ်။ အဲဒါ လို ဖန်တီး ရရှိ လာတဲ့ thread က အလို အလျောက် run မလုပ် ပါဘူး။ သူကို စတင် အလုပ်လုပ် ဖို့ အတွက် Run method ကို ခေါ်လိုက် မှ thread ရဲ့ Entry function ကို လုပ်ဆောင် မှာ ဖြစ်ပါတယ်။ Thread တစ်ခု ကို Create နဲ့ ဖန်တီးပြီး၊ စတင်လုပ် ဆောင် ဖို့ Run ကို အသုံးပြု တဲ့ နမူနာ ကို အောက်မှာ တွေ့နှင့် ပါတယ်။

```
MyThread *th = new MyThread;
if ( th->Create() != wxTHREAD_NO_ERROR )
{
    printf("Can't create thread! \n");
}
else {
    printf("New thread created.\n");
    if ( th->Run() != wxTHREAD_NO_ERROR )
    {
        printf("Can't start thread! \n");
    }
    else printf("New thread started.\n");
}
```

Thread အမျိုးအစား တွေက အောက်ပါ အတိုင်း နှစ်မျိုး ရှိပါတယ်။

- Detached thread
- Joinable thread

Detached thread တွေက စတင် ပြီးရင် လွှတ်ထား လိုက်ရုံ ပါပဲ။ သူတို့ က လုပ်ဆောင် စရာ ရှိတာ လုပ်ပြီး ရင် သူဟာသူ အဆုံးသတ်ပြီး၊ သူဟာသူ ဖျက်သွား ပါတယ်။ Thread တစ်ခု လုပ်ဆောင် ပြီးတဲ့ အခါ ရတာတဲ့ အဖြေ ကို စောင့်ချင် ရင်တော့ joinable thread ကို သုံးနိုင် ပြီး၊ wxThread::Wait နဲ့ ပြန်စောင့် ဖို့ လိုပါတယ်။ မဟုတ်ရင် သူသုံးထားတဲ့ system resources တွေကို အလိုလို ပြန် free လုပ်မှာ မဟုတ်ပဲ၊ Delete နဲ့ ကိုပုံးပာ့ ကိုယ် ဖျက်ဖို့ လိုပါတယ်။

Thread ကို ဖန်တီးတဲ့ အခါ၊ သူရဲ့ အမျိုးအစား ကို wxThread ရဲ့ constructor မှာ wxTHREAD_DETACHED (the default) ဒါမှ မဟုတ် wxTHREAD_JOINABLE လို့ ထည့်ပေး နိုင်

ပါတယ်။

Thread တစ်ခု ရဲ့ priority ကို သတ်မှတ် ချင်ရင် တော့ Create နဲ့ဖန်တီး ပြီးတဲ့ နောက် Run နဲ့ စစ်ဆေးမှု wxThread::SetPriority ကို ၀ နဲ့ ၁၀၀ ကြေား တန်ဖိုး တစ်ခု ထည့်ပြီး ခေါ်နိုင် ပါတယ်။ ၀ ဆိုရင် အနိမ့်ဆုံး ဖြစ်ပြီး ၁၀၀ ဆိုရင် အမြင့်ဆုံး ဖြစ်ပါတယ်။ ဒါမူ မဟုတ် WXTHREAD_MIN_PRIORITY ၁ wxTHREAD_DEFAULT_PRIORITY နဲ့ wxTHREAD_MAX_PRIORITY တို့ကို ၀, ၵ၀, နဲ့ ၁၀၀ အတွက် သုံးနိုင် ပါတယ်။

၅.၂၂ Thread ကို အော့အားလုံးခြင်း

Thread ကို စကြန် အနည်းငယ် ရပ်နား ချင်ရင် polling တို့ idling တို့ လုပ်ပြီး processor ကို အလုပ်များမော်လုံး အစား: wxThread::Sleep ကို နားချင်တဲ့ milliseconds အရေအတွက် သတ်မှတ် ပြီး ခေါ်လို့ ရပါတယ်။ Pause နဲ့ Resume functions တွေကို သုံးလို့ ရပေါ်မယ့် Deadlock ဖြစ်နိုင်တာ တွေ၊ POSIX စနစ်တွေမှာ TestDestroy ဆိုတဲ့ function နဲ့ thread ထဲမှာ စစ်နေဖို့လိုတာ တွေ ကြောင့် သိပ် မသုံးသင့်ပါဘူး။ အဲဒီ အစား နောက်အပိုင်း မှာ ဆွေးနွေးမယ့် synchronization objects တွေကို သုံးတာ ပို့ကောင်း ပါတယ်။ ဒါမူ မဟုတ် blocking call တွေ သုံးတာ၊ joinable thread သုံးပြီး wait နဲ့ block လုပ်တာ တွေ သုံးနိုင် ပါတယ်။

၅.၂၃ Thread ကို အဆုံးသတ်ခြင်း

Detached thread တွေက လုပ်ဆောင်မှု ပြီးတဲ့ အခါ အလိုအလျောက် သူ့ဟာသူ ဖျက်သွား မှာဖြစ် ပါတယ်။ Joinable thread တွေ အတွက်တော့ wxThread::Wait ကို ချက်ချင်း ခေါ်ပြီး စောင့်ရင် စောင့်၊ ဒါမူမဟုတ် GUI application တွေမှာ ဆိုရင် wxThread::IsAlive နဲ့ poll လုပ်ကြည့်ပြီး false ရလာမှ Wait ကို ခေါ်လို့ ရပါတယ်။ ပို့ကောင်း တဲ့ နည်းကတော့ detached thread ကို သုံးလိုက်ပြီး ပြီးသွားတဲ့ အခါ event တစ်ခု post လုပ်လိုက် တဲ့ နည်းပါ။ Thread တစ်ခု ကို ဖျက်ဖို့ အတွက် wxThread::Delete ကို သုံးနိုင် ပြီး၊ အဲဒီ အတွက် thread ထဲမှာ TestDestroy ကို ပုံမှန် ခေါ်နေဖို့လိုပါတယ်။

ရိုးရှင်းတဲ့ thread နမူနာ တစ်ခု ကို စာရင်း ၅.၁ မှာ ဖော်ပြ ထားပါတယ်။

```

1 #include "wx/wx.h"
2 #include <stdio.h>
3 using namespace std;
4
```

```
5 // -----
6 // a simple thread
7
8 class MyThread : public wxThread
9 {
10 public:
11     MyThread();
12     virtual ~MyThread();
13
14     // thread execution starts here
15     virtual void *Entry();
16
17 private:
18     int m_count;
19 };
20 // -----
21 // MyThread
22 MyThread::MyThread() : wxThread()
23 {
24     m_count = 0;
25 }
26
27 MyThread::~MyThread()
28 {
29 }
30
31 wxThread::ExitCode MyThread::Entry()
32 {
33     printf("Thread started (Priority = %d).\n", GetPriority());
34     for (m_count = 0; m_count < 10; m_count++)
35     {
36         // check if just this thread was asked to exit
37         if (TestDestroy()) break;
38         printf("Thread progress: %d \n", m_count);
39         // wxSleep() can't be called from non-GUI thread!
40         wxThread::Sleep(1000);
41 }
```

```
41 }
42     printf("Thread finished.\n");
43     return NULL;
44 }
45 // -----
46 class MyFrame : public wxFrame
47 {
48 public:
49     MyFrame(const wxString& title);
50
51 };
52 MyFrame::MyFrame(const wxString& title)
53     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(250, 150))
54 {
55     Centre();
56
57     MyThread *th = new MyThread;
58     if ( th->Create() != wxTHREAD_NO_ERROR )
59     {
60         printf("Can't create thread! \n");
61     }
62     else {
63         printf("New thread created.\n");
64         if ( th->Run() != wxTHREAD_NO_ERROR )
65         {
66             printf("Can't start thread! \n");
67         }
68         else printf("New thread started.\n");
69     }
70 }
71
72 class MyApp : public wxApp
73 {
74 public:
75     virtual bool OnInit();
76 }
```

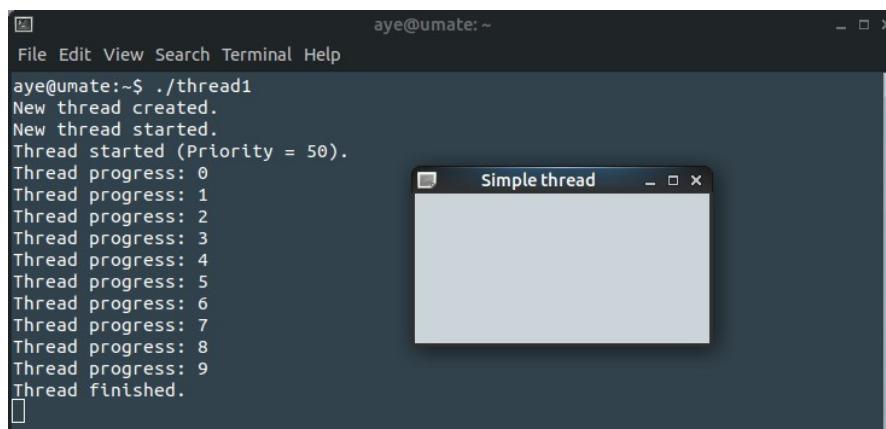
```

77
78 IMPLEMENT_APP(MyApp)
79 bool MyApp::OnInit()
80 {
81     MyFrame *frame = new MyFrame(wxT("Simple thread"));
82     frame->Show(true);
83
84     return true;
85 }

```

စာရင်း ၅.၃: thread1.cpp

အဲဒီ နမူနာ မှာ MyFrame ရဲ constructor မှာ MyThread တစ်ခု ကို ဖန်တီး ပြီး၊ သူရဲ Entry method မှာ သတ်မှတ် ထားတဲ့ အတိုင်း တစ်စက္းနံ တစ်ခါ progress ကို print လုပ်ပြ ပြီးသွား တဲ့ အခါ သူဟာသူ ရပ်ပြီး၊ destroy လုပ်သွား ပါတယ်။ ပရိုဂရမ် ရဲ ရလဒ် ကို ပုံ ၅.၃ မှာ ပြထား ပါတယ်။



ပုံ ၅.၃: thread1.cpp ၏ ရလဒ်။

၅.၃ Synchronization Objects

ပရိုဂရမ် တစ်ခု မှာ thread တွေကို သုံးကြ တဲ့ အခါ ဒေတာ တွေကို thread တွေကြား မျှသုံး ဖို့ လိုတတ် ပါတယ်။ အဲဒီ အခါ ဒေတာ တစ်ခု ကို ပြုပြင် နေတဲ့ အခိုက် အတန် မှာ တခြား thread တွေက အဲဒီ ဒေတာ ကို အသုံး မပြု မိမိ အရေးကြီး ပါတယ်။ အဲဒီ အတွက် အကာ အကွယ် လုပ်ဖို့ အစီ အမံ တွေ

အနေနဲ့ အောက်မှာ ဖော်ပြ ထား တဲ့ class တွေကို သုံးနိုင် ပါတယ်။

၅.၃.၁ Mutex

Mutex ဆိုတဲ့ နာမည် က mutual exclusion ဆိုတဲ့ စကားလုံး ကနေ ဆင်းသက် လာတာ ပါ။ သူက synchronization objects တွေထဲ မှာ အသုံးပြုရ အလွယ်ဆုံး ဖြစ်ပါတယ်။ ကာကွယ် ချင်တဲ့ ဒေတာ အတွက် wxMutex အမျိုးအစား variable တစ်ခု ဥပမာ m_mutex ဆိုပြီး သတ်မှတ် ထားပါမယ်။ ဒေတာ ကို သုံးခါနီးရင် wxMutex::Lock ကို အရင် ခေါ်ကြည့် ပါမယ်။ အကယ်၍ တွေား thread တစ်ခု ခု က သူကို lock လုပ်ထား ရင် အဲဒီ ဒေတာ မအား မချင်း ပရိုကရမ က Lock မှာ ရပ်စောင့် (halt) နေမှာ ဖြစ်ပါတယ်။ ဒေတာ ကို သုံးပြီး တဲ့ အခါမှာ လည်း wxMutex::Unlock နဲ့ ပြန်ပြီး free လုပ်ပေးဖို့ လိုပါတယ်။ wxMutex ရဲ့ Lock နဲ့ Unlock တို့ကို တိုက်ရှိက် သုံးလို့ ရပေမယ့်၊ wxMutexLocker class ကို သုံးတာ က exception ဖြစ်ရင် တောင်မှ mutex ကို အလို အလျောက် release ပေးစေ ပါတယ်။ ဥပမာ အနေနဲ့ MyApp ဆိုတဲ့ class ထဲက wxMutex အမျိုးအစား ဖြစ်တဲ့ m_mutex ကို wxMutexLocker သုံးပြီး lock လုပ်တဲ့ နမူနာ ကို ဖော်ပြ ထား ပါတယ်။

```
void MyApp ::DoSomething()
{
    wxMutexLocker lock(m_mutex);
    if (lock.IsOk())
    {
        ... do something
    }
    else
    {
        ... we have not been able to
        ... acquire the mutex, fatal error
    }
}
```

Mutex တွေ သုံးတဲ့ အခါမှာ အရေးကြီး တဲ့ စည်းမျဉ်း သုံးခု က အောက်ပါ အတိုင်း ဖြစ်ပါတယ်။

- Lock ချထား တဲ့ Mutex တစ်ခု ကို ထပ်ပြီး lock လုပ်လို့ မရ ပါဘူး။
- Thread တစ်ခု က lock လုပ်ထား တဲ့ mutex ကို နောက် တစ်ခြား thread တစ်ခု ကနေ unlock

လုပ်လို မရ ပါဘူး။ အဲဒီ လို လုပ်ဖို့ လိုရင် semaphores တွေကို သုံးရ မှာ ဖြစ်ပါတယ်။

- တွေး အလုပ် တွေ လုပ်စရာ ရှိတဲ့ thread တစ်ခု (ဥပမာ GUI thread) လိုမျိုးမှာ Lock ကို ခေါ်ပြီး မရ မချင်း blocked ဖြစ်နေမယ့် အစား return ချက်ချင်း ပြန်ပေး တဲ့ wxMutex::TryLock ကို အရင် ခေါ်ကြည့် နိုင်ပါတယ်။ Lock လုပ်လို ရရင် wxMUTEX_NO_ERROR ကို ရမှာ ဖြစ်ပြီး၊ မဟုတ်ရင် wxMUTEX_DEAD_LOCK or wxMUTEX_BUSY တို့ကို ရလာမှာ ဖြစ်ပါတယ်။

၅.၃.၂ Deadlocks

Thread နှစ်ခု က တစ်ခု lock ချထားတဲ့ resources တွေကို တစ်ခု က အပြန် အလုန် စောင့်နေ ကြမယ် ဆိုရင် deadlock ဖြစ်သွား ပါတယ်။ တချို့ စနစ် တွေမှာ အဲဒီ အခြေ အနေကို အထူး error code အနေနဲ့ wxMUTEX_DEAD_LOCK ကို return လုပ် ပေးနိုင် ပေမယ့် အဲဒီ လို လုပ်မပေး နိုင် တဲ့ စနစ် တွေမှာ တော့ ပရိုကရမ က ဘာမှ ရွှေ မဆက် နိုင်ပဲ kill မလုပ် မချင်း ရပ် (hang) သွား မှာ ဖြစ်ပါတယ်။ အဲဒီ ပြဿနာ ကို ဖြေရှင်းဖို့ အတွက် အောက်မှာ ဖော်ပြ ထားတဲ့ နည်း နှစ်နည်း ကို သုံးလေ့ ရှိပါတယ်။

Fixed order : Thread အားလုံးမှာ mutex ကို ယူတဲ့ အစီ အစဉ် ကို mutex 1, mutex 2 စသဖြင့် အစီ အစဉ် အသေ သတ်မှတ် ထားရင် deadlock မဖြစ် နိုင် ပါဘူး။

Try lock : Fixed order သုံးဖို့ သိပ် မဆင် မပြောတဲ့ အခြေ အနေ မျိုး ဆိုရင် တော့ ပထမ ဆုံး mutex ကို lock လုပ်ပြီး၊ ကျနိုင်တဲ့ mutex တွေကို ယူတဲ့ အခါ try lock နဲ့ ယူပါမယ်။ မရရင် အကုန် ပြန် release လုပ်ပြီး၊ အစ ကနေ ပြန်စ တဲ့ နည်းပါ။

၅.၃.၃ Critical Section

အကာ အကွယ် ပေးချင် တာက ဒေတာ မဟုတ်ပဲ ကုဒ် ရဲ့ အချို့အဝိုင်း တွေကို ကာကွယ် ချင်တာ ဆိုရင် တော့ critical section တွေကို သုံးနိုင် ပါတယ်။ လက်တွေ့ အသုံး မှာ mutex နဲ့ တော်တော် ဆင်ပါတယ်။ Mutex ရဲ့ lock နဲ့ unlock လိုပဲ၊ critical section ရဲ့ enter နဲ့ left ကို သုံးနိုင် ပါတယ်။ နောက်တစ်ခါ wxMutex ရဲ့ Lock ကို တိုက်ရှိက် သုံးမယ့် အစား wxMutexLocker ကို သုံးသလို၊ wxCriticalSection ကို တိုက်ရှိက် သုံးမယ့် အစား wxCriticalSectionLocker ကို တတ်နိုင် သလောက် သုံးသင့် ပါတယ်။

၅.၃.၄ Semaphore

Semaphores တွေက mutex နဲ့ counter ပေါင်းထား တယ်လို့ ပြောလို့ ရပါတယ်။ Mutex နဲ့ အမိက ကွာတာ ကတေသ့ သူတို့ ကို ပိုင်ထားတဲ့ thread တင်မက ပဲ ဘယ် thread က မလို လှစ်း signal လုပ်လို ရတာ ပါ။ Semaphores တွေကို ပိုင်ရှင်မဲ့ counter တွေလို့ မြင်ကြည့် နိုင် ပါတယ်။ Wait ကို ခေါ်ပြီး semaphore တစ်ခု ကို စောင့်နေ တဲ့ thread တစ်ခု က counter က အပေါင်း မဖြစ် မချင်း စောင့်နေမှာပါ။ ပြီးရင် counter ကို လျှော့ပြီး၊ return ပြန်ပါတယ်။ တွေား thread တစ်ခု က semaphore ရဲ့ Post ကို ခေါ်ရင် counter ကို တိုးပေးပြီး၊ return ပြန်ပါ တယ်။ wxSemaphore ရဲ့ constructor မှာ counter အတွက် maximum တန်ဖိုး ကို သတ်မှတ် ပေးနိုင် ပါတယ်။ ဘာမှ မသတ်မှတ်ရင် ၀ ဖြစ်ပြီး၊ counter အတွက် limit မရှိပါဘူး။ သူည့် မဟုတ်တဲ့ maximum တန်ဖိုး သတ်မှတ်ပြီး၊ မလို အင်ပဲ Post ကို ခေါ်တဲ့ အတွက် အမြင့်ဆုံး တန်ဖိုး ကို ကျော်သွားရင် wxSEMA_OVERFLOW error ကို ရမှာ ဖြစ်ပါတယ်။

၅.၄ Simple Multithreading Example

ရိုးရှင်း တဲ့ multithreading နမူနာ အနေနဲ့ စာရင်း ၅.၂ မှာ ဖော်ပြ ထားတဲ့ ./thread/th-simple/thread-simple.cpp ဆိုတဲ့ ပရိုဂရမဲ့ လေးကို ခွေးနေး ပါမယ်။

ပရိုဂရမဲ့ မှာ start | stop | pause နဲ့ resume ဆိုတဲ့ button လေးခု ကို အရင် ထည့် ပါမယ်။ Start button က thread အသစ် တစ်ခု ကို ဖန်တီး ပြီး၊ စတင် ဖို့ပါ။ Stop က နောက်ဆုံး ဖန်တီး လိုက်တဲ့ thread ကို ရှာပြီး ဖျက်ပေး ပါလိမ့်မယ်။ Pause က နောက်ဆုံး စတင် တဲ့ thread ကို ရှာပြီး ခေါ်ဗျာရပ် (pause) လုပ်ပေး ပါတယ်။ Resume က အရင်ဆုံး paused လုပ်ခဲ့ တဲ့ thread ကို ရှာပြီး ပြန် run ပေးဖို့ပါ။

၅.၄.၁ Log Target ပြောင်းခြင်း

နောက် တစ်ခါ message တွေကို user interface မှာ ဖော်ပြ ဖို့ အတွက် logList လို့ နာမည် ပေးထားတဲ့ List Box တစ်ခုကို ဖန်တီး ပါမယ်။ ဖန်တီး လိုက်တဲ့ List box မှာ message တွေကို timestamp နဲ့ တွေ့ပြီး log လုပ်ဖို့ လိုင်း အရေအတွက် ကန်သတ်ပြီး အလို အလျောက် ရှင်းလင်း ဖို့ တွေ အတွက် WriteList ဆိုတဲ့ function တစ်ခု ကို အောက်ပါ အတိုင်း ရေးလိုက် ပါတယ်။

```
void MyFrame::WriteList(string mes)
```

```
{
    int nl=logList->GetCount();
    if(nl>=20){ logList->Delete(0); }

    wxString wstr = mes;
    wxDateTime wdt;
    wdt.SetToCurrent();
    wstr = wdt.Format(wxT("%Y-%m-%d %H:%M:%S"), wxDateTime::Local)+"
    " : "+wstr; // add timestamp
    wxArrayString astr;
    astr.Add(wstr);
    nl=logList->GetCount();
    logList->InsertItems(astr,nl);
}
```

Log target အဟောင်းကို မှတ်ထားပြီး၊ ခုနက list box ကို log target အသစ် အနေနဲ့ အစားထိုးဖို့ အတွက် ပြင်ဆင် ပါမယ်။ အဟောင်း ကို သိမ်းဖို့ အတွက် m_oldLogger ဆိုတဲ့ variable တစ်ခု ကို အောက်ပါ အတိုင်း ကြော်ပြု ပါမယ်။

```
wxLog *m_oldLogger;
```

ပြီးရင် GUI အတွက် ဖြစ်တဲ့ MyFrame ၏ wxLog ကင် derived လုပ်ယူ ဖို့ လိုပါတယ်။

```
class MyFrame : public wxFrame, private wxLog
```

အဲဒီ နောက် MyFrame ရဲ့ constructor နဲ့ destructor တွေမှာ log target သတ်မှတ် မှု တွေကို အောက်ပါ အတိုင်း လုပ်နိုင် ပါတယ်။

```
MyFrame::MyFrame(...) : ...
{
    m_oldLogger = wxLog::GetActiveTarget();

    ... do somethings

    wxLog::SetActiveTarget(this);
```

```

}

MyFrame::~MyFrame()
{
    wxLog::SetActiveTarget(m_oldLogger);
    ... do somethings
}

```

Derived class ဖြစ်တဲ့ MyFrame အတွက် log လုပ်ရင် လုပ်ဆောင်တဲ့ virtual DoLogRecord ဖော်ရှင် ကို အောက်ပါ အတိုင်း သတ်မှတ် ထားဖို့ လည်း လိုပါတယ်။

```

void MyFrame::DoLogRecord(wxLogLevel level, const wxString& msg, const
                           wxLogRecordInfo& info)
{
    this->WriteList(msg.ToStdString());
}

```

အဲဒါ ဆိုရင် wxLogStatus ၊ wx.LogError စတဲ့ ဖော်ရှင် တွေ ခေါ်လိုက် တိုင်း သူတို့ ရဲ့ message တွေက list box မှာ လာပေါ်မှာ ဖြစ်ပါတယ်။

၅.၄.၂ Thread များဖန်တီးခြင်း

Start ခလုတ် ကို တစ်ချက် နှိပ်တိုင်း thread အသစ် တစ်ခု ဖန်တီးပြီး၊ wxThread array ထဲမှာ element တစ်ခု အနေနဲ့ ထပ်ထည့် သိမ်းပါမယ်။ အဲဒါ အတွက် wxThread array အပျိုးအစား အင် နဲ့ wxArrayThread ဆိုပြီး type define လုပ်ပါမယ်။

```
WX_DEFINE_ARRAY_PTR(wxThread *, wxArrayThread);
```

အဲဒါ နောက် MyApp class ထဲမှာ wxThread array တစ်ခု ကို အောက်ပါ အတိုင်း ကြော်ဖို့ ဖို့ပြုပါမယ်။

```
wxArrayThread m_threads;
```

MyApp class ထဲက အဲဒါ module variable ကို access လုပ်ချင်ရင် တော့ အောက်ပါ အတိုင်း ပြုလုပ် နိုင် ပါတယ်။

```
wxArrayThread& threads = wxGetApp().m_threads;
```

Application ကို ပိတ်လိုက် တဲ့ အခါ လက်ရှိ လုပ်လက်စ thread တွေ အားလုံး ကို ရပ်ဖို့ bool variable တစ်ခု နဲ့ လျမ်း အကြောင်းကြား ပြီး semaphore တစ်ခု နဲ့ Wait လုပ်နိုင် ပါတယ်။ အဲဒီ အတွက် အောက်ပါ variable နှစ်ခု ကို MyApp ထဲမှာ module variable အနေနဲ့ ထပ်ကြော် လိုက်ပါမယ်။

```
wxSemaphore m_semAllDone;
bool m_shuttingDown;
```

Thread အားလုံး က မျှသုံး ကြော်မယ့် အဲဒီ module variables တွေကို ကာကွယ် နဲ့ အတွက် Mutex ဒါမှ မဟုတ် critical section ကို သုံးနိုင် ပါတယ်။ Critical section က application အပြင် ကနေ မမြင် ရပဲ၊ နဲ့ ပိုမြဲး efficient ဖြစ်တဲ့ အတွက် သူကို သုံးဖို့ အောက်ပါ အတိုင်း ကြော် လိုက်ပါမယ်။

```
wxCriticalSection m_critsect;
```

Start ခလုတ် ကို နှိပ်လိုက် တိုင်း MyThread ရဲ့ instance တစ်ခု ကို ဖန်တီး ပြီး၊ Create ကို ခေါ်ပြီး thread တစ်ခု ဖန်တီး ပါမယ်။ ဖန်တီး လို့ ရတဲ့ thread ကို module variable ဖြစ်တဲ့ thread array ထဲမှာ ထပ်ထည့် မသိမ်းခင် wxCriticalSectionLocker နဲ့ အရင် lock လုပ်ပါမယ်။ အဲဒီ ဖန်ရှင် က return ဘာနဲ့ အလိုလို unlock ပြန်ဖြစ် သွားမှာ ပါ။ ပြီးတဲ့ အခါ Run နဲ့ Entry ထဲက လုပ်ငန်း တွေကို စတင် ပါတယ်။

```
void MyFrame::OnStart(wxCommandEvent& WXUNUSED(event))
{
    MyThread *thread = CreateThread(); // call the following function
    if ( thread->Run() != wxTHREAD_NO_ERROR )
    {
        wxLogStatus(wxT("Can't start thread!"));
    }
}

MyThread *MyFrame::CreateThread()
{
```

```

MyThread *thread = new MyThread;

if ( thread->Create() != wxTHREAD_NO_ERROR )
{
    wxLogStatus(wxT("Can't create thread!"));
}

wxCriticalSectionLocker enter(wxGetApp().m_critsect);
wxGetApp().m_threads.Add(thread);

return thread;
}

```

၅.၄.၃ Thread လုပ်ငန်းများသတ်မှတ်ခြင်း

Thread တစ်ခု ထဲမှာ လုပ်ဆောင် မယ့် လုပ်ငန်း တွေကို virtual Entry ထဲမှာ သတ်မှတ် နိုင် ပါတယ်။ သူကို Delete လုပ်ရင် ထွက်နိုင် အောင် TestDestroy() ဆိုတဲ့ ဖန်ရှင်ကို thread ထဲမှာ ပုံမှန် ခေါ်ပေး နေဖို့ လည်း လိုအပ် ပါတယ်။ Application က ပိတ်လိုက် တဲ့ အကြောင်း signal လုပ်တဲ့ flag ကို လည်း ပုံမှန် စစ်ပြီး လိုအပ် ရင် ချက်ချင်း exit လုပ်နိုင် ပါတယ်။ နမူနာ Entry ဖန်ရှင် တစ်ခု ကို အောက်မှာ တွေ့နိုင် ပါတယ်။

```

wxThread::ExitCode MyThread::Entry()
{
    wxLogMessage("Thread started (priority = %u).", GetPriority());

    for ( m_count = 0; m_count < 10; m_count++ )
    {
        // check if the application is shutting down: in this case all threads
        // should stop a.s.a.p.
        {

            wxCriticalSectionLocker locker(wxGetApp().m_critsect);
            if ( wxGetApp().m_shuttingDown )
                return NULL;
        }
    }
}

```

```

// check if just this thread was asked to exit
if ( TestDestroy() )
break;

wxLogMessage("Thread progress: %u", m_count);

// wxSleep() can't be called from non-GUI thread!
wxThread::Sleep(1000);

}

wxLogMessage("Thread finished.");

return NULL;
}

```

၅.၄.၄ Thread ကို ဖျက်ခြင်း:

ပုံမှန် detached thread တစ်ခု က သူရဲ့ လုပ်ငန်း ပြီးပြီး သွား တာနဲ့ အလို အလေ့ အလျောက် destroy ဖြစ်သွား ပါတယ်။ အကယ်၍ မပြီး ခင် ဖျက်ချင် ရင်တော့ Delete နဲ့ ဖျက်နိုင် ပြီး၊ thread ထဲမှာ လည်း TestDestroy() ကို ခေါ်ပို့ လိုပါတယ်။ Stop ခလုတ် ကို နိုပ်ရင် thread array ထဲက နောက်ဆုံး thread ကို delete လုပ်ပေး တဲ့ လုပ်ပေး တဲ့ ဖန်ရှင် ကို အောက်ပါ အတိုင်း တွေ့နှင့် ပါတယ်။

```

void MyFrame::OnStop(wxCommandEvent& WXUNUSED(event))
{
    wxThread* toDelete = NULL;

    {
        wxCriticalSectionLocker enter(wxGetApp().m_critsect);

        // stop the last thread
        if ( wxGetApp().m_threads.IsEmpty() )
        {
            wxLogStatus(wxT("No thread to stop!"));
        }
    }
}

```

```

    else
    {
        toDelete = wxGetApp().m_threads.Last();
    }

}

if ( toDelete )
{
    // This can still crash if the thread gets to delete itself
    // in the mean time.
    toDelete->Delete();
    wxLogStatus(wxT("Last thread stopped."), 1);
}
}

```

ဖျက်ပြီးတဲ့ အခါ thread ရဲ့ destructor ထဲမှာ module variable ဖြစ်တဲ့ thread array ထဲက ငော်လုပ်မှုတွေမှာ ဖြန့်ဖယ် ဖို့ အောက်ပါ အတိုင်း ပြုလုပ် နိုင် ပါတယ်။ Application ၏ ဝိတ်နေ တဲ့ အခါ thread array ထဲမှာ လည်း အကုန် ဖယ်ပြီးတဲ့ အခါ semaphore ကို Post လုပ် ပါမယ်။

```

MyThread::~MyThread()
{
    wxCriticalSectionLocker locker(wxGetApp().m_critsect);

    wxArrayThread& threads = wxGetApp().m_threads;
    threads.Remove(this);

    if ( threads.IsEmpty() )
    {
        // signal the main thread that there are no more threads left if it is
        // waiting for us
        if ( wxGetApp().m_shuttingDown )
        {
            wxGetApp().m_shuttingDown = false;

            wxGetApp().m_semAllDone.Post();
        }
    }
}

```

```

    }
}
}

```

MyFrame ရဲ့ destructor ထဲမှာ လည်း သူကို ပိတ်လိုက် တဲ့ အခါ thread array ထဲမှာ လည်း
thread တွေ ရှိနေ သေးရင် flag ပြေား၊ semaphore တို့ အောက်ပါ အတိုင်း စောင့် ပါမယ်။

```

// NB: although the OS will terminate all the threads anyhow when the main
//      one exits, it's good practice to do it ourselves -- even if it's not
//      completely trivial in this example

// tell all the threads to terminate: note that they can't terminate while
// we're deleting them because they will block in their OnExit() -- this is
// important as otherwise we might access invalid array elements

{
    wxCriticalSectionLocker locker(wxGetApp().m_critsect);

    // check if we have any threads running first
    const wxArrayThread& threads = wxGetApp().m_threads;
    size_t count = threads.GetCount();

    if ( !count )
        return;

    // set the flag indicating that all threads should exit
    wxGetApp().m_shuttingDown = true;
}

// now wait for them to really terminate
wxGetApp().m_semAllDone.Wait();

```

၅.၄.၂ Pause နဲ့ Resume

Thread ကို ရော် ရပ်စီ အတွက် pause လုပ်တာ နဲ့ ပြန်စီ resume လုပ်တဲ့ နမူနာ တွေကို အောက်ပါ အတိုင်း တွေ့နှင့် ပါတယ်။

```
void MyFrame::OnPause(wxCommandEvent& WXUNUSED(event))
{
    wxCriticalSectionLocker enter(wxGetApp().m_critsect);

    // pause last running thread
    int n = wxGetApp().m_threads.Count() - 1;
    while (n >= 0 && !wxGetApp().m_threads[n]->IsRunning()) n--;

    if (n < 0) {
        wxLogStatus(wxT("No thread to pause!"));
    }
    else {
        wxGetApp().m_threads[n]->Pause();
        wxLogStatus(wxT("Thread paused."), 1);
    }
}

void MyFrame::OnResume(wxCommandEvent& WXUNUSED(event))
{
    wxCriticalSectionLocker enter(wxGetApp().m_critsect);

    // resume first suspended thread
    size_t n = 0, count = wxGetApp().m_threads.Count();
    while (n < count && !wxGetApp().m_threads[n]->IsPaused())
        n++;

    if (n == count) {
        wxLogStatus(wxT("No thread to resume!"));
    }
    else {
        wxGetApp().m_threads[n]->Resume();
    }
}
```

၅.၄. SIMPLE MULTITHREADING EXAMPLE

၁၁၁

```
    wxLogStatus(wxT("Thread resumed."), 1);  
}  
}
```

နူမူနာ တစ်ပုဒ် လို့ အပြည့် အစုံ ကိုတော့ အောက်က စာရင်း ၅.၂ မှာ ဖော်ပြ ထားပါတယ်။

```
1 // File: th-simple.cpp  
2 // Description: Using wxThread  
3 // WebSite: http://cool-emerald.blogspot.com  
4 // MIT License (https://opensource.org/licenses/MIT)  
5 // Copyright (c) 2018 Yan Naing Aye  
6  
7 // References (Original example)  
8 // https://github.com/wxWidgets/wxWidgets/blob/master/samples/thread/thread.  
     cpp  
9  
10 #include "wx/wx.h"  
11 #include "./sample.xpm"  
12 #include <string>  
13 #include <stdio.h>  
14 using namespace std;  
15  
16 WX_DEFINE_ARRAY_PTR(wxThread *, wxArrayThread);  
17 // -----  
18 // a simple thread  
19  
20 class MyThread : public wxThread  
21 {  
22 public:  
23     MyThread();  
24     virtual ~MyThread();  
25  
26     // thread execution starts here  
27     virtual void *Entry();  
28  
29 public:
```

```

30     int m_count;
31 };
32
33 // -----
34 class MyApp : public wxApp
35 {
36 public:
37     MyApp();
38     virtual ~MyApp() {};
39     virtual bool OnInit();
40     wxCriticalSection m_critsect;
41     wxArrayThread m_threads;
42
43     // semaphore used to wait for the threads to exit, see MyFrame::OnQuit()
44     wxSemaphore m_semAllDone;
45
46     // indicates that we're shutting down and all threads should exit
47     bool m_shuttingDown;
48 };
49
50 // -----
51 class MyFrame : public wxFrame, private wxLog
52 {
53 public:
54     MyFrame(const wxString& title);
55     ~MyFrame();
56     void OnQuit(wxCommandEvent& event);
57     void OnAbout(wxCommandEvent& event);
58     void OnStart(wxCommandEvent& event);
59     void OnStop(wxCommandEvent& event);
60     void OnPause(wxCommandEvent& event);
61     void OnResume(wxCommandEvent& event);
62     void WriteList(string mes);
63 protected:
64     virtual void DoLogRecord(wxLogLevel level, const wxString& msg, const
wxLogRecordInfo& info);

```

9.5. SIMPLE MULTITHREADING EXAMPLE

CC

```
65 private:
66     wxListBox *logList;
67     wxButton *btnStart;
68     wxButton *btnStop;
69     wxButton *btnPause;
70     wxButton *btnResume;
71     // old log target, we replace it with one using m_textctrl during this
72     // frame life time
73     wxLog *m_oldLogger;
74
75     // helper function - creates a new thread (but doesn't run it)
76     MyThread *CreateThread();
77
78     wxDECLARE_EVENT_TABLE();
79 };
80 // -----
81 // IDs for the controls and the menu commands
82 enum {
83     ID_MNU_QUIT = wxID_EXIT,
84     ID_MNU_ABOUT = wxID_ABOUT,
85     ID_LST_LOG,
86     ID_BTN_START,
87     ID_BTN_STOP,
88     ID_BTN_PAUSE,
89     ID_BTN_RESUME
90 };
91
92 wxBEGIN_EVENT_TABLE(MyFrame, wxFrame)
93     EVT_MENU(ID_MNU_QUIT, MyFrame::OnQuit)
94     EVT_MENU(ID_MNU_ABOUT, MyFrame::OnAbout)
95 wxEND_EVENT_TABLE()
96
97 IMPLEMENT_APP(MyApp)
98
99 MyApp::MyApp()
100 {
```

```

101     m_shuttingDown = false;
102 }
103
104 // 'Main program' equivalent: the program execution "starts" here
105 bool MyApp::OnInit()
106 {
107     if ( !wxApp::OnInit() )
108         return false;
109     MyFrame *frame = new MyFrame("wxWidgets App");
110     frame->Show(true);
111     return true;
112 }
113 // -----
114 // MyFrame
115
116 MyFrame::MyFrame(const wxString& title)
117     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(600, 450),
118       wxDEFAULT_FRAME_STYLE ^ wxRESIZE_BORDER)
119 {
120     m_oldLogger = wxLog::GetActiveTarget();
121     SetIcon(wxICON(sample));
122     wxMenu *fileMenu = new wxMenu();
123     wxMenu *helpMenu = new wxMenu();
124     helpMenu->Append(ID_MNU_ABOUT, "&About\tF1", "Show about dialog");
125     fileMenu->Append(ID_MNU_QUIT, "E&xit\tAlt-X", "Quit this program");
126     wxMenuBar *menuBar = new wxMenuBar();
127     menuBar->Append(fileMenu, "&File");
128     menuBar->Append(helpMenu, "&Help");
129     SetMenuBar(menuBar);
130     CreateStatusBar(1);
131     SetStatusText("wxWidgets App");
132
133     btnStart = new wxButton(this, ID_BTN_START, wxT("Start"),
134       wxPoint(20,20), wxSize(100, 25));
135     btnStop = new wxButton(this, ID_BTN_STOP, wxT("Stop"),
136       wxPoint(140,20), wxSize(100, 25));

```

9.5. SIMPLE MULTITHREADING EXAMPLE

CC3

```
137     btnPause = new wxButton(this, ID_BTN_PAUSE, wxT("Pause"),
138         wxPoint(260,20), wxSize(100, 25));
139     btnResume = new wxButton(this, ID_BTN_RESUME, wxT("Resume"),
140         wxPoint(380,20), wxSize(100, 25));
141     logList=new wxListBox(this, ID_LST_LOG,wxPoint(20,70),wxSize(560, 250));
142
143     Connect(ID_BTN_START, wxEVT_COMMAND_BUTTON_CLICKED,
144             wxCommandEventHandler(MyFrame::OnStart));
145     Connect(ID_BTN_STOP, wxEVT_COMMAND_BUTTON_CLICKED,
146             wxCommandEventHandler(MyFrame::OnStop));
147     Connect(ID_BTN_PAUSE, wxEVT_COMMAND_BUTTON_CLICKED,
148             wxCommandEventHandler(MyFrame::OnPause));
149     Connect(ID_BTN_RESUME, wxEVT_COMMAND_BUTTON_CLICKED,
150             wxCommandEventHandler(MyFrame::OnResume));
151     wxLog::SetActiveTarget(this);
152 }
153
154 MyFrame::~MyFrame()
155 {
156     wxLog::SetActiveTarget(m_oldLogger);
157
158     // NB: although the OS will terminate all the threads anyhow when the
159     // main
160     // one exits, it's good practice to do it ourselves -- even if it's
161     // not
162     // completely trivial in this example
163
164     // tell all the threads to terminate: note that they can't terminate
165     // while
166     // we're deleting them because they will block in their OnExit() -- this
167     // is
168     // important as otherwise we might access invalid array elements
169
170     {
171         wxCriticalSectionLocker locker(wxGetApp().m_critsect);
```

```
169
170     // check if we have any threads running first
171     const wxArrayThread& threads = wxGetApp().m_threads;
172     size_t count = threads.GetCount();
173
174     if ( !count )
175         return;
176
177     // set the flag indicating that all threads should exit
178     wxGetApp().m_shuttingDown = true;
179 }
180
181 // now wait for them to really terminate
182 wxGetApp().m_semAllDone.Wait();
183 }
184
185 void MyFrame::WriteList(string mes)
186 {
187     int nl=logList->GetCount();
188     if(nl>=20){ logList->Delete(0); }
189     wxString wstr = mes;
190     wxDateTime wdt;
191     wdt.SetToCurrent();
192     wstr = wdt.Format(wxT("%Y-%m-%d %H:%M:%S"), wxDateTime::Local) +
193             " : "+wstr; // add timestamp
194     wxArrayString astr;
195     astr.Add(wstr);
196     nl=logList->GetCount();
197     logList->InsertItems(astr,nl);
198 }
199
200 void MyFrame::DoLogRecord(wxLogLevel level,
201                           const wxString& msg, const wxLogRecordInfo& info)
202 {
203     /*
204     if ( level <= wxLOG.Warning || level == wxLOG_Trace )
```

9.5. SIMPLE MULTITHREADING EXAMPLE

©JO

```
205     {
206         m_oldLogger->LogRecord(level, msg, info);
207         return;
208     }
209 */
210     this->WriteList(msg.ToString());
211 }
212
213 void MyFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
214 {
215     Close(true);
216 }
217
218 void MyFrame::OnAbout(wxCommandEvent& WXUNUSED(event))
219 {
220     wxMessageBox(wxString::Format
221                 (
222                     "wxWidgets thread\n"
223                     "\n"
224                     "Author: Yan Naing Aye \n"
225                     "Web: http://cool-emerald.blogspot.com"
226                 ),
227                 "About wxWidgets thread",
228                 wxOK | wxICON_INFORMATION,
229                 this);
230 }
231
232 void MyFrame::OnStart(wxCommandEvent& WXUNUSED(event))
233 {
234     MyThread *thread = CreateThread(); // call the following function
235     if ( thread->Run() != wxTHREAD_NO_ERROR )
236     {
237         wxLogStatus(wxT("Can't start thread!"));
238     }
239 }
```

```
241 MyThread *MyFrame::CreateThread()
242 {
243     MyThread *thread = new MyThread;
244
245     if ( thread->Create() != wxTHREAD_NO_ERROR )
246     {
247         wxLogStatus(wxT("Can't create thread!"));
248     }
249
250     wxCriticalSectionLocker enter(wxGetApp().m_critsect);
251     wxGetApp().m_threads.Add(thread);
252
253     return thread;
254 }
255
256 void MyFrame::OnStop(wxCommandEvent& WXUNUSED(event))
257 {
258     wxThread* toDelete = NULL;
259     {
260         wxCriticalSectionLocker enter(wxGetApp().m_critsect);
261
262         // stop the last thread
263         if ( wxGetApp().m_threads.IsEmpty() )
264         {
265             wxLogStatus(wxT("No thread to stop!"));
266         }
267         else
268         {
269             toDelete = wxGetApp().m_threads.Last();
270         }
271     }
272
273     if ( toDelete )
274     {
275         // This can still crash if the thread gets to delete itself
276         // in the mean time.
```

9.5. SIMPLE MULTITHREADING EXAMPLE

©JR

```
277     toDelete->Delete();
278     wxLogStatus(wxT("Last thread stopped."), 1);
279 }
280 }
281
282 void MyFrame::OnPause(wxCommandEvent& WXUNUSED(event))
283 {
284     wxCriticalSectionLocker enter(wxGetApp().m_critsect);
285
286     // pause last running thread
287     int n = wxGetApp().m_threads.Count() - 1;
288     while (n >= 0 && !wxGetApp().m_threads[n]->IsRunning()) n--;
289
290     if (n < 0) {
291         wxLogStatus(wxT("No thread to pause!"));
292     }
293     else {
294         wxGetApp().m_threads[n]->Pause();
295         wxLogStatus(wxT("Thread paused."), 1);
296     }
297 }
298
299 void MyFrame::OnResume(wxCommandEvent& WXUNUSED(event))
300 {
301     wxCriticalSectionLocker enter(wxGetApp().m_critsect);
302
303     // resume first suspended thread
304     size_t n = 0, count = wxGetApp().m_threads.Count();
305     while (n < count && !wxGetApp().m_threads[n]->IsPaused())
306         n++;
307
308     if (n == count) {
309         wxLogStatus(wxT("No thread to resume!"));
310     }
311     else {
312         wxGetApp().m_threads[n]->Resume();
```

```
313         wxLogStatus(wxT("Thread resumed."), 1);
314     }
315 }
316
317 // -----
318 // MyThread
319 MyThread::MyThread()
320     : wxThread()
321 {
322     m_count = 0;
323 }
324
325 MyThread::~MyThread()
326 {
327     wxCriticalSectionLocker locker(wxGetApp().m_critsect);
328
329     wxArrayThread& threads = wxGetApp().m_threads;
330     threads.Remove(this);
331
332     if ( threads.IsEmpty() )
333     {
334         // signal the main thread that there are no more threads left if it
335         // is
336         if ( wxGetApp().m_shuttingDown )
337         {
338             wxGetApp().m_shuttingDown = false;
339
340             wxGetApp().m_semAllDone.Post();
341         }
342     }
343 }
344
345 wxThread::ExitCode MyThread::Entry()
346 {
347     wxLogMessage("Thread started (priority = %u).", GetPriority());
```

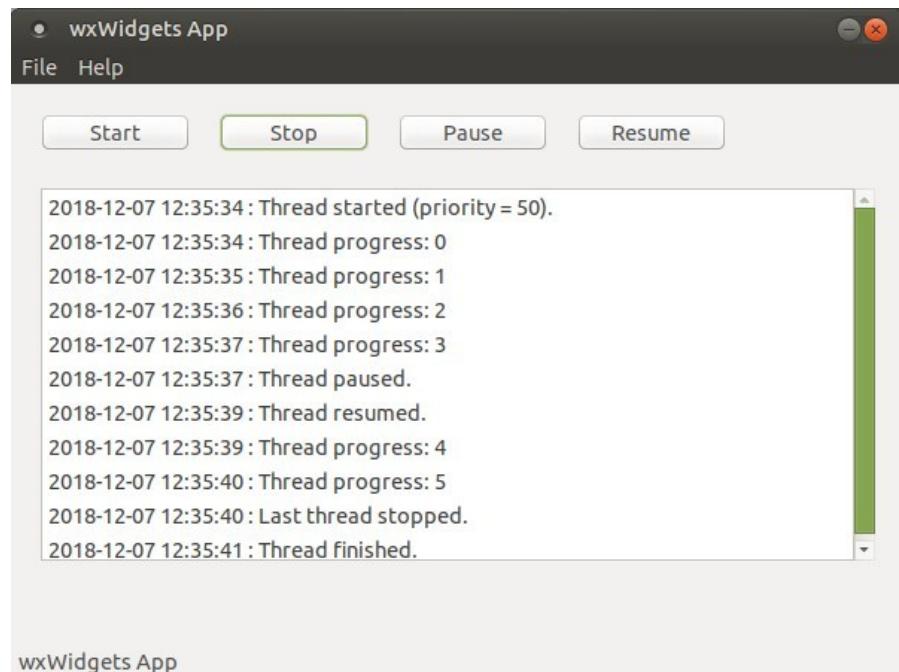
၂.၅. SIMPLE MULTITHREADING EXAMPLE

၁၂၂

```
348
349     for ( m_count = 0; m_count < 10; m_count++ )
350     {
351         // check if the application is shutting down: in this case all
352         // threads
353         // should stop a.s.a.p.
354         {
355             wxCriticalSectionLocker locker(wxGetApp().m_critsect);
356             if ( wxGetApp().m_shuttingDown )
357                 return NULL;
358         }
359         // check if just this thread was asked to exit
360         if ( TestDestroy() )
361             break;
362
363         wxLogMessage("Thread progress: %u", m_count);
364
365         // wxSleep() can't be called from non-GUI thread!
366         wxThread::Sleep(1000);
367     }
368
369     wxLogMessage("Thread finished.");
370
371     return NULL;
372 }
373 // -----
```

စာရင်း ၂.၂: th-simple.cpp

ပရိုဂရမ် ရဲ့ GUI နဲ့ thread ကို run ပြီး၊ ပြန်ရပ် လိုက်တဲ့ အခါ ရလာတဲ့ ရလဒ် ကို ပုံ ၂.၂ မှာ ထွေးနိုင် ပါတယ်။



ပုံ ၅.၂: th-simple.cpp ၏ ရလဒ်။

၅.၅ Event များသုံးခြင်း

နောက်ထပ် နှမူနာ တစ်ခု အနေနဲ့ worker thread တစ်ခု ကနေ application ကို event တွေ လှမ်းပို အကြောင်း ကြား တဲ့ [./thread/th-worker/th-worker.cpp](#) ဆိုတဲ့ ပရီဂရမ် ကို ဆွေးနွေး ပါမယ်။ MyFrame ကို event တွေလှမ်းပို ဖို့ thread ၏ constructor မှာ MyFrame ကို ထည့်ပေး ရပါမယ်။ ပြီးတဲ့ အခါ wxThreadEvent တစ်ခု ဖန်တီး၊ SetInt နဲ့ တန်ဖိုး တစ်ခု ကို ထည့်ပေး ပြီး wxQueueEvent နဲ့ event ကို ပို့နိုင် ပါတယ်။ အဲဒါလို thread ၏ constructor နဲ့ Entry နှမူနာ ကို အောက်မှာ တွေ့နိုင် ပါတယ်။

```
MyWorkerThread::MyWorkerThread(MyFrame *frame)
: wxThread()
{
    m_frame = frame;
    m_count = 0;
}
```

```

wxThread::ExitCode MyWorkerThread::Entry()
{
    for ( m_count = 0; !m_frame->Cancelled() && (m_count < 100); m_count++ )
    {
        // check if we were asked to exit
        if ( TestDestroy() )
            break;

        // create any type of command event here
        wxThreadEvent event( wxEVT_THREAD, ID_WORKER_THREAD );
        event.SetInt( m_count );

        // send in a thread-safe way
        wxQueueEvent( m_frame, event.Clone() );
        wxMilliSleep(200);
    }

    wxThreadEvent event( wxEVT_THREAD, ID_WORKER_THREAD );
    event.SetInt(-1); // that's all
    wxQueueEvent( m_frame, event.Clone() );
    return NULL;
}

```

Start ဆိုတဲ့ ခလုတ် ကို နိုပ်လိုက် တာနဲ့ အဲဒီ thread ကို ဖန်တီး၊ စတင်ပြီး သူက ပိုပေး တဲ့ တန်ဖိုး တွေကို ဖော်ပြန့် dialog တစ်ခု ကို ဖန်တီး လိုက် ပါမယ်။ Dialog မှာ ၁ ကနေ ၁၀၀ ထိ တန်ဖိုး တွေကို ဖော်ပြ ပေးမှာ ဖြစ်ပြီး ပြီးသွားရင် ဒါမှုမဟုတ် dialog ကို ပိတ်လိုက် ရင် thread ကို အဆုံး သတ်ပြီး၊ အနုတ် ၁ တန်ဖိုး ကို event နဲ့ ပိုပေး ပါမယ်။ Application ဘက်မှာ လည်း အနုတ် ၁ ရလာ ရင် dialog ကို destroy လုပ်လိုက် ပါမယ်။ Thread ရဲ့ event ကို handle လုပ်မယ့် OnWorkerEvent လို့ နာမည် ပေးလိုက် တဲ့ ဖန်ရှင် ကို event table မှာ ထည့်သွင်း တာရယ်၊ သူရဲ့ လုပ်ဆောင်မှု တွေကို အောက်မှာ ပြထား ပါတယ်။

```

wxBEGIN_EVENT_TABLE(MyFrame, wxFrame)
...
EVT_THREAD(ID_WORKER_THREAD, MyFrame::OnWorkerEvent)

```

```

...
wxEND_EVENT_TABLE()

...

void MyFrame::OnWorkerEvent(wxThreadEvent& event)
{
    int n = event.GetInt();
    if (n == -1)
    {
        m_dlgProgress->Destroy();
        m_dlgProgress = (wxProgressDialog *)NULL;

        // the dialog is aborted because the event came from another thread, so
        // we may need to wake up the main event loop for the dialog to be
        // really closed
        wxWakeUpIdle();
    }
    else
    {
        if (!m_dlgProgress->Update(n))
        {
            wxCriticalSectionLocker lock(m_csCancelled);

            m_cancelled = true;
        }
    }
}

```

ပရိုကရမဲ့ တစ်ပုဒ် လုံး ကို အောက်က စာရင်း ၅.၃ မှာ ဖော်ပြ ထားပါတယ်။

```

1 // File: th-worker.cpp
2 // Description: Using wxThread
3 // WebSite: http://cool-emerald.blogspot.com
4 // MIT License (https://opensource.org/licenses/MIT)

```

၁၂. EVENT များသုံးခွင့်

၁၂၂

```
5 // Copyright (c) 2018 Yan Naing Aye
6
7 // References (Original example)
8 // https://github.com/wxWidgets/wxWidgets/blob/master/samples/thread/thread.
9
10 #include "wx/wx.h"
11 #include "wx/progdlg.h"
12 #include "./sample.xpm"
13
14 // -----
15 class MyApp : public wxApp
16 {
17 public:
18     virtual bool OnInit();
19
20 };
21 // -----
22 class MyFrame : public wxFrame
23 {
24 public:
25     MyFrame(const wxString& title);
26     ~MyFrame();
27     void OnQuit(wxCommandEvent& event);
28     void OnAbout(wxCommandEvent& event);
29     void OnStart(wxCommandEvent& event);
30     void OnWorkerEvent(wxThreadEvent& event);
31     // accessors for MyWorkerThread (called in its context!)
32     bool Cancelled();
33 private:
34     wxButton *btnStart;
35     wxStaticText *lblText;
36
37     // the progress dialog which we show while worker thread is running
38     wxProgressDialog *m_dlgProgress;
39
```

```

40 // was the worker thread cancelled by user?
41 bool m_cancelled;
42 wxCriticalSection m_csCancelled;           // protects m_cancelled
43
44 wxDECLARE_EVENT_TABLE();
45 };
46 // -----
47 // a worker thread
48
49 class MyWorkerThread : public wxThread
50 {
51 public:
52     MyWorkerThread(MyFrame *frame);
53
54     // thread execution starts here
55     virtual void *Entry();
56
57     // called when the thread exits - whether it terminates normally or is
58     // stopped with Delete() (but not when it is Kill()ed!)
59     virtual void OnExit();
60
61 public:
62     MyFrame *m_frame;
63     unsigned m_count;
64 };
65
66 // -----
67 // IDs for the controls and the menu commands
68 enum
69 {
70     ID_MNU_QUIT = wxID_EXIT,
71     ID_MNU_ABOUT = wxID_ABOUT,
72     ID_BTN_START,
73     ID_WORKER_THREAD,
74     ID_LBL_TEXT
75 };

```

၁၂. EVENT များသုံးခွင့်

၁၃၃

```
76 wxBEGIN_EVENT_TABLE(MyFrame, wxFrame)
77     EVT_MENU(ID_MNU_QUIT, MyFrame::OnQuit)
78     EVT_MENU(ID_MNU_ABOUT, MyFrame::OnAbout)
79     EVT_THREAD(ID_WORKER_THREAD, MyFrame::OnWorkerEvent)
80 wxEND_EVENT_TABLE()
81
82 IMPLEMENT_APP(MyApp)
83
84 // 'Main program' equivalent: the program execution "starts" here
85 bool MyApp::OnInit()
86 {
87     if (!wxApp::OnInit())
88         return false;
89     MyFrame *frame = new MyFrame("wxWidgets App");
90     frame->Show(true);
91     return true;
92 }
93 // -----
94 // MyFrame
95
96 MyFrame::MyFrame(const wxString& title)
97     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(600, 450),
98               wxDEFAULT_FRAME_STYLE ^ wxRESIZE_BORDER)
99 {
100     SetIcon(wxICON(sample));
101     wxMenu *fileMenu = new wxMenu();
102     wxMenu *helpMenu = new wxMenu();
103     helpMenu->Append(ID_MNU_ABOUT, "&About\tF1", "Show about dialog");
104     fileMenu->Append(ID_MNU_QUIT, "E&xit\tAlt-X", "Quit this program");
105     wxMenuBar *menuBar = new wxMenuBar();
106     menuBar->Append(fileMenu, "&File");
107     menuBar->Append(helpMenu, "&Help");
108     SetMenuBar(menuBar);
109     CreateStatusBar(1);
110     SetStatusText("wxWidgets App");
111 }
```

```
112     btnStart = new wxButton(this, ID_BTN_START, wxT("Start"),
113         wxPoint(20,20), wxSize(100, 25));
114     Connect(ID_BTN_START, wxEVT_COMMAND_BUTTON_CLICKED,
115             wxCommandEventHandler(MyFrame::OnStart));
116     lblText = new wxStaticText(this, ID_LBL_TEXT,
117         wxT("Click the button to start the worker thread"),
118         wxPoint(20,50));
119 }
120
121 MyFrame::~MyFrame()
122 {
123
124 }
125
126 void MyFrame::OnQuit(wxCommandEvent& WXUNUSED(event))
127 {
128     Close(true);
129 }
130
131 void MyFrame::OnAbout(wxCommandEvent& WXUNUSED(event))
132 {
133     wxMessageBox(wxString::Format
134     (
135         "wxWidgets thread\n"
136         "\n"
137         "Author: Yan Naing Aye \n"
138         "Web: http://cool-emerald.blogspot.com"
139     ),
140     "About wxWidgets thread",
141     wxOK | wxICON_INFORMATION,
142     this);
143 }
144
145 void MyFrame::OnStart(wxCommandEvent& WXUNUSED(event))
146 {
147     MyWorkerThread *thread = new MyWorkerThread(this);
```

```

148
149     if ( thread->Create() != wxTHREAD_NO_ERROR )
150     {
151         // wxLogError(wxT("Can't create thread!"));
152         return;
153     }
154
155     m_dlgProgress = new wxProgressDialog
156     (
157         wxT("Progress dialog"),
158         wxT("Wait until the thread terminates or press [Cancel]"),
159         100,
160         this,
161         wxPD_CAN_ABORT |
162         wxPD_APP_MODAL |
163         wxPD_ELAPSED_TIME |
164         wxPD_ESTIMATED_TIME |
165         wxPD_REMAINING_TIME
166     );
167
168     // thread is not running yet, no need for crit sect
169     m_cancelled = false;
170
171     thread->Run();
172 }
173
174 bool MyFrame::Cancelled()
175 {
176     wxCriticalSectionLocker lock(m_csCancelled);
177
178     return m_cancelled;
179 }
180
181 void MyFrame::OnWorkerEvent(wxThreadEvent& event)
182 {
183     int n = event.GetInt();

```

```

184     if ( n == -1 )
185     {
186         m_dlgProgress->Destroy();
187         m_dlgProgress = (wxProgressDialog *)NULL;
188
189         // the dialog is aborted because the event came from another thread,
190         // so
191         // we may need to wake up the main event loop for the dialog to be
192         // really closed
193         wxWakeUpIdle();
194     }
195     else
196     {
197         if ( !m_dlgProgress->Update(n) )
198         {
199             wxCriticalSectionLocker lock(m_csCancelled);
200
201             m_cancelled = true;
202         }
203     }
204
205 // -----
206 // MyWorkerThread
207
208 MyWorkerThread::MyWorkerThread(MyFrame *frame)
209     : wxThread()
210 {
211     m_frame = frame;
212     m_count = 0;
213 }
214
215 void MyWorkerThread::OnExit()
216 {
217 }
```

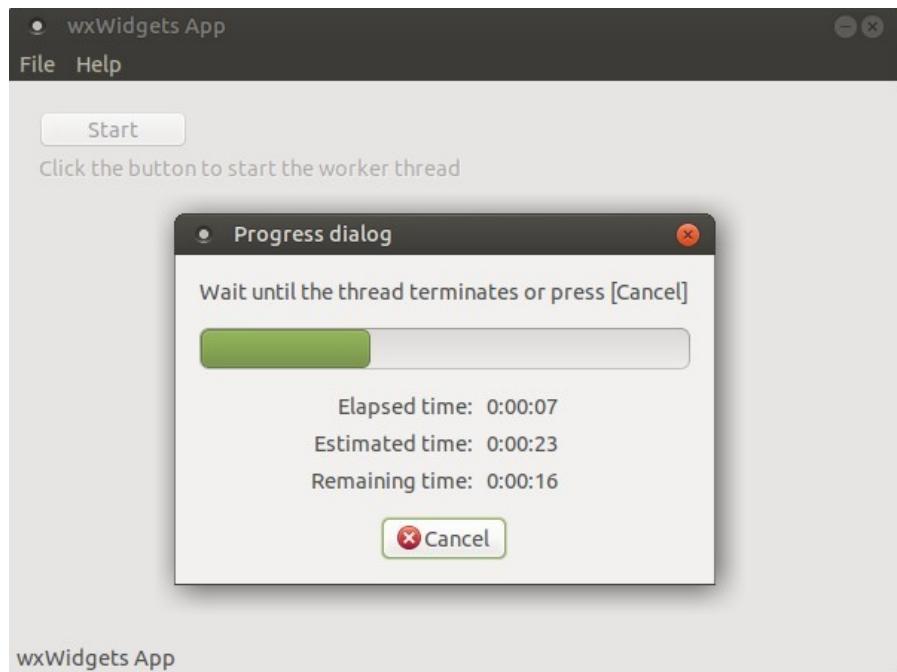
```

219 wxThread::ExitCode MyWorkerThread::Entry()
220 {
221     for ( m_count = 0; !m_frame->Cancelled() && (m_count < 100); m_count++ )
222     {
223         // check if we were asked to exit
224         if ( TestDestroy() )
225             break;
226
227         // create any type of command event here
228         wxThreadEvent event( wxEVT_THREAD, ID_WORKER_THREAD );
229         event.SetInt( m_count );
230
231         // send in a thread-safe way
232         wxQueueEvent( m_frame, event.Clone() );
233
234         wxMilliSleep(200);
235     }
236
237     wxThreadEvent event( wxEVT_THREAD, ID_WORKER_THREAD );
238     event.SetInt(-1); // that's all
239     wxQueueEvent( m_frame, event.Clone() );
240
241     return NULL;
242 }
243
244 // -----

```

စာရင်း ၅.၃: th-worker.cpp

ပရိုဂရမ် ရဲ့ GUI နဲ့ thread ကို run လိုက်တဲ့ အခါ ပေါ်လာမယ့် dialog ကို ဖုံး၍ မှာ ထွေးစိုင်ပါတယ်။



ଫିଲ୍ ନାମ: th-worker.cpp ଏଇ ରହିଛି॥

၅.၆ Thread မှ GUI ကိုလျမ်းသုံးခြင်း

Application တဲ့မှာ thread တွေကို သုံးမယ် ဆိုရင် main thread ကပဲ GUI လုပ်ဆောင်မှု တွေကို လုပ်ဆောင်ပြီး ကျန်တဲ့ thread တွေနဲ့ event တွေက ပဲ ဆက်သွယ်တာ ကောင်းပါတယ်။ ဒါ နမူနာ ./thread/th-gui/th-gui.cpp မှာ တော့ တစ်ခြား thread တွေက နေလည်း GUI ကို လုမ်းသုံးလို ရတယ် ဆိုတဲ့ အကြောင်း လုပ်ပြရုံ သက်သက် ဖြစ်ပြီး ဒီမိမိ အရ မကောင်း တာကြောင့် အဲဒီ လို သုံးဖို့ ကို အား မပေး ပါဘူး။

GUI element တစ်ခုဖြစ်တဲ့ image dialog ကို လှမ်းပြီး အသုံးပြု မယ့် gui thread တစ်ခု ကို အောက်ပါ အတိုင်း သတ်မှတ်နိုင် ပါတယ်။ သူလှမ်းပြီး ထိန်းချုပ်မယ့် image dialog အတွက် လည်း pointer တစ်ခု ပါဝင် ပါမယ်။

```
class MyGUITHread : public wxThread
{
public:
    MyGUITHread(MyImageDialog *dlg) : wxThread(wxTHREAD_JOINABLE) {
```

```
m_dlg = dlg;
}

virtual ExitCode Entry();

private:
MyImageDialog *m_dlg;
};
```

အဲဒီ နောက် သူရဲ့ လုပ်ဆောင် မှု တွေ ဖြစ်တဲ့ Entry မှာ image dialog ပေါ်က bitmap image ပေါ်မှာ rectangle တွေကို ကြံ့ရော ကျပန်း နေရာ တွေမှာ အပြာ၊ အစိမ်း တစ်လျည့်စီ ဆွဲမှာ ဖြစ်ပါတယ်။ GUI ကို secondary ဖြစ်တဲ့ thread ထဲက နေ လုမ်း မသုံးခင် မှာ wxMutexGuiEnter(); နဲ့ lock လုပ်ပြီး၊ ပြီးသွားတဲ့ အခါ wxMutexGuiLeave(); နဲ့ release လုပ် ပါမယ်။ Image dialog ကို သုံးဖို့ အတွက် လည်း သူကို ကာကွယ် ထားတဲ့ critical section ကို ဝင်ဖို့ wxCriticalSectionLocker သုံး ပါတယ်။ တစ်ဆင့် ပြီးတဲ့ အခါတိုင်း image dialog ကို event ပို့ပေး ပါတယ်။ Entry ဖန်ရှင် ကို အောက်မှာ ဖော်ပြ ထားပါတယ်။

```
wxThread::ExitCode MyGUIThread::Entry()
{
    for (int i=0; i<GUITHREAD_NUM_UPDATES && !TestDestroy(); i++)
    {
        // inform the GUI toolkit that we're going to use GUI functions
        // from a secondary thread:
        wxMutexGuiEnter();

        {
            wxCriticalSectionLocker lock(m_dlg->m_csBmp);

            // draw some more stuff on the bitmap
            wxMemoryDC dc(m_dlg->m_bmp);
            dc.SetBrush((i%2)==0 ? *wxBLUE_BRUSH : *wxGREEN_BRUSH);
            dc.DrawRectangle(rand()%GUITHREAD_BMP_SIZE, rand()%GUITHREAD_BMP_SIZE,
                30, 30);

            // simulate long drawing time:
```

```

wxMilliSleep(200);

}

// if we don't release the GUI mutex the MyImageDialog won't be able to
// refresh
wxMutexGuiLeave();

// notify the dialog that another piece of our masterpiece is complete:
wxThreadEvent event( wxEVT_THREAD, GUILTHREAD_EVENT );
event.SetInt(i+1);
wxQueueEvent( m_dlg, event.Clone() );

// give the main thread the time to refresh before we lock the GUI mutex
// again
// FIXME: find a better way to do this!
wxMilliSleep(100);
}

return (ExitCode)0;
}

```

Image dialog မှတေတာ့ ခုန် က ပြောခဲ့ တဲ့ bitmap image တစ်ခုရယ်၊ သူကို ကာကွယ်ဖို့ critical section တစ်ခု ရယ်၊ gui thread ရယ် အမိက ပါဝင် ပါတယ်။ ပြီးရင် သူ အတွက် event table တစ်ခု သပ်သပ် ထပ် ကြော်ဖို့ လည်း လိုပါတယ်။

```

class MyImageDialog: public wxDialog
{
public:
    // ctor
    MyImageDialog(wxFrame *frame);
    ~MyImageDialog();

    // stuff used by MyGUIThread:
    wxBitmap m_bmp;      // the bitmap drawn by MyGUIThread
    wxCriticalSection m_csBmp;        // protects m_bmp

```

```

private:
void OnGUIThreadEvent(wxThreadEvent& event);
void OnPaint(wxPaintEvent&);

MyGUIThread m_thread;
int m_nCurrentProgress;

wxDECLARE_EVENT_TABLE();
};

...
}

wxBEGIN_EVENT_TABLE(MyImageDialog, wxDialog)
EVT_THREAD(GUTHREAD_EVENT, MyImageDialog::OnGUIThreadEvent)
EVT_PAINT(MyImageDialog::OnPaint)
wxEND_EVENT_TABLE()

```

Image dialog ရဲ့ thread event မှာ လက်ရှိ progress ကို ပဲ တွက်ပြီး၊ Refresh ကို ခေါ်ပေးပါတယ်။ အဲဒီ နောက် OnPaint event handler မှာ bitmap ကို paint လုပ်ပေးပြီး၊ progress bar ဆွဲပြုပါတယ်။ ပရိုဂရမ် အပြည့် အစုံကို စာရင်း ၅.၄ မှာ ဖော်ပြုထားပါတယ်။

```

1 // File: th-gui.cpp
2 // Description: Using wxThread
3 // WebSite: http://cool-emerald.blogspot.com
4 // MIT License (https://opensource.org/licenses/MIT)
5 // Copyright (c) 2018 Yan Naing Aye
6
7 // References (Original example)
8 // https://github.com/wxWidgets/wxWidgets/blob/master/samples/thread/thread.
9 // cpp
10 #include "wx/wx.h"
11 #include "./sample.xpm"
12
13 // -----

```

```
14 // a thread which executes GUI calls using wxMutexGuiEnter/Leave
15
16 #define GUITHREAD_BMP_SIZE          300
17 #define GUITHREAD_NUM_UPDATES       50
18 class MyImageDialog;
19
20 class MyGUIThread : public wxThread
21 {
22 public:
23     MyGUIThread(MyImageDialog *dlg) : wxThread(wxTHREAD_JOINABLE) {
24         m_dlg = dlg;
25     }
26     virtual ExitCode Entry();
27
28 private:
29     MyImageDialog *m_dlg;
30 };
31
32 // -----
33 // an helper dialog used by MyFrame::OnStartGUIThread
34
35 class MyImageDialog: public wxDialog
36 {
37 public:
38     // ctor
39     MyImageDialog(wxFrame *frame);
40     ~MyImageDialog();
41
42     // stuff used by MyGUIThread:
43     wxBitmap m_bmp;      // the bitmap drawn by MyGUIThread
44     wxCriticalSection m_csBmp;           // protects m_bmp
45
46 private:
47     void OnGUIThreadEvent(wxThreadEvent& event);
48     void OnPaint(wxPaintEvent&);
```

၅.၆. THREAD နဲ့ GUI ကိုလုမ်းသံးခြင်း

```
50     MyGUIThread m_thread;
51     int m_nCurrentProgress;
52
53     wxDECLARE_EVENT_TABLE();
54 };
55
56 // -----
57 class MyApp : public wxApp
58 {
59 public:
60     virtual bool OnInit();
61
62 };
63 // -----
64 class MyFrame : public wxFrame
65 {
66 public:
67     MyFrame(const wxString& title);
68     ~MyFrame();
69     void OnQuit(wxCommandEvent& event);
70     void OnAbout(wxCommandEvent& event);
71     void OnStart(wxCommandEvent& event);
72 private:
73     wxButton *btnStart;
74     wxStaticText *lblText;
75
76     wxDECLARE_EVENT_TABLE();
77 };
78
79 // -----
80 // IDs for the controls and the menu commands
81 enum
82 {
83     ID_MNU_QUIT = wxID_EXIT,
84     ID_MNU_ABOUT = wxID_ABOUT,
85     ID_BTN_START,
```

```
86     ID_LBL_TEXT,
87     GUITHREAD_EVENT
88 };
89 wxBEGIN_EVENT_TABLE(MyFrame, wxFrame)
90     EVT_MENU(ID_MNU_QUIT, MyFrame::OnQuit)
91     EVT_MENU(ID_MNU_ABOUT, MyFrame::OnAbout)
92 wxEND_EVENT_TABLE()
93
94 IMPLEMENT_APP(MyApp)
95
96 // 'Main program' equivalent: the program execution "starts" here
97 bool MyApp::OnInit()
98 {
99     if ( !wxApp::OnInit() )
100         return false;
101     MyFrame *frame = new MyFrame("wxWidgets App");
102     frame->Show(true);
103     return true;
104 }
105 // -----
106 // MyFrame
107
108 MyFrame::MyFrame(const wxString& title)
109     : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition, wxSize(600, 450),
110               wxDEFAULT_FRAME_STYLE ^ wxRESIZE_BORDER)
111 {
112     SetIcon(wxICON(sample));
113     wxMenu *fileMenu = new wxMenu;
114     wxMenu *helpMenu = new wxMenu;
115     helpMenu->Append(ID_MNU_ABOUT, "&About\tF1", "Show about dialog");
116     fileMenu->Append(ID_MNU_QUIT, "E&xit\tAlt-X", "Quit this program");
117     wxMenuBar *menuBar = new wxMenuBar();
118     menuBar->Append(fileMenu, "&File");
119     menuBar->Append(helpMenu, "&Help");
120     SetMenuBar(menuBar);
121     CreateStatusBar(1);
```

၅.၆. THREAD နဲ့ GUI ကိုလုမ်းသံးခြင်း

၁၄၃

```
122     SetStatusText("wxWidgets App");  
123  
124     btnStart = new wxButton(this, ID_BTN_START, wxT("Start"),  
125         wxPoint(20,20), wxSize(100, 25));  
126     Connect(ID_BTN_START, wxEVT_COMMAND_BUTTON_CLICKED,  
127             wxCommandEventHandler(MyFrame::OnStart));  
128     lblText = new wxStaticText(this, ID_LBL_TEXT, wxT("Click the button to  
129         start the gui thread"),  
130         wxPoint(20,50));  
131  
132 MyFrame::~MyFrame()  
133 {  
134 }  
135  
136  
137 void MyFrame::OnQuit(wxCommandEvent& WXUNUSED(event))  
138 {  
139     Close(true);  
140 }  
141  
142 void MyFrame::OnAbout(wxCommandEvent& WXUNUSED(event))  
143 {  
144     wxMessageBox(wxString::Format  
145         (  
146             "wxWidgets thread\n"  
147             "\n"  
148             "Author: Yan Naing Aye \n"  
149             "Web: http://cool-emerald.blogspot.com"  
150         ),  
151         "About wxWidgets thread",  
152         wxOK | wxICON_INFORMATION,  
153         this);  
154 }  
155  
156 void MyFrame::OnStart(wxCommandEvent& WXUNUSED(event))
```

```
157 {
158     MyImageDialog dlg(this);
159     dlg.ShowModal();
160 }
161
162 // -----
163 // MyImageDialog
164
165 wxBEGIN_EVENT_TABLE(MyImageDialog, wxDialog)
166     EVT_THREAD(GUITHREAD_EVENT, MyImageDialog::OnGUIThreadEvent)
167     EVT_PAINT(MyImageDialog::OnPaint)
168 wxEND_EVENT_TABLE()
169
170 MyImageDialog::MyImageDialog(wxFrame *parent)
171     : wxDialog(parent, wxID_ANY, "Image created by a secondary thread",
172                 wxDefaultPosition, wxSize(GUITHREAD_BMP_SIZE,
173                                         GUITHREAD_BMP_SIZE)*1.5, wxDEFAULT_DIALOG_STYLE),
174     m_thread(this)
175 {
176     m_nCurrentProgress = 0;
177
178     CentreOnScreen();
179
180     // NOTE: no need to lock m_csBmp until the thread isn't started:
181
182     // create the bitmap
183     if (!m_bmp.Create(GUITHREAD_BMP_SIZE, GUITHREAD_BMP_SIZE) || !m_bmp.IsOk())
184     {
185         wxLogError("Couldn't create the bitmap!");
186         return;
187     }
188
189     // clean it
190     wxMemoryDC dc(m_bmp);
191     dc.SetBackground(*wxBLACK_BRUSH);
```

၅.၆. THREAD နဲ့ GUI ကိုလုမ်းသံးခြင်း

၁၄၅

```
191     dc.Clear();  
192  
193     // draw the bitmap from a secondary thread  
194     if ( m_thread.Create() != wxTHREAD_NO_ERROR ||  
195         m_thread.Run() != wxTHREAD_NO_ERROR )  
196     {  
197         wxLogError(wxT("Can't create/run thread!"));  
198         return;  
199     }  
200 }  
201  
202 MyImageDialog::~MyImageDialog()  
203 {  
204     // in case our thread is still running and for some reason we are  
205     // destroyed,  
206     // do wait for the thread to complete as it assumes that its  
207     MyImageDialog  
208     // pointer is always valid  
209     m_thread.Delete();  
210 }  
211  
212 void MyImageDialog::OnGUIThreadEvent(wxThreadEvent& event)  
213 {  
214     m_nCurrentProgress = int(((float)event.GetInt()*100)/  
215     GUILTHREAD_NUM_UPDATES);  
216  
217     Refresh();  
218 }  
219  
220 void MyImageDialog::OnPaint(wxPaintEvent& WXUNUSED(evt))  
221 {  
222     wxPaintDC dc(this);  
223     const wxSize& sz = dc.GetSize();  
224     {
```

```

224     // paint the bitmap
225     wxCriticalSectionLocker locker(m_csBmp);
226     dc.DrawBitmap(m_bmp, (sz.GetWidth()-GUILTHREAD_BMP_SIZE)/2,
227                   (sz.GetHeight()-GUILTHREAD_BMP_SIZE)/2);
228 }
229
230 // paint a sort of progress bar with a 10px border:
231 dc.SetBrush(*wxRED_BRUSH);
232 dc.DrawRectangle(10,10, m_nCurrentProgress*(sz.GetWidth()-20)/100,30);
233 dc.SetTextForeground(*wxBLUE);
234 dc.DrawText(wxString::Format("%d%%", m_nCurrentProgress),
235             (sz.GetWidth()-dc.GetCharWidth()*2)/2,
236             25-dc.GetCharHeight()/2);
237 }
238 // -----
239 // MyGUIThread
240
241 wxThread::ExitCode MyGUIThread::Entry()
242 {
243
244     for (int i=0; i<GUILTHREAD_NUM_UPDATES && !TestDestroy(); i++)
245     {
246         // inform the GUI toolkit that we're going to use GUI functions
247         // from a secondary thread:
248         wxMutexGuiEnter();
249
250     {
251         wxCriticalSectionLocker lock(m_dlg->m_csBmp);
252
253         // draw some more stuff on the bitmap
254         wxMemoryDC dc(m_dlg->m_bmp);
255         dc.SetBrush((i%2)==0 ? *wxBLUE_BRUSH : *wxGREEN_BRUSH);
256         dc.DrawRectangle(rand()%GUILTHREAD_BMP_SIZE, rand()%GUILTHREAD_BMP_SIZE, 30, 30);
257
258         // simulate long drawing time:

```

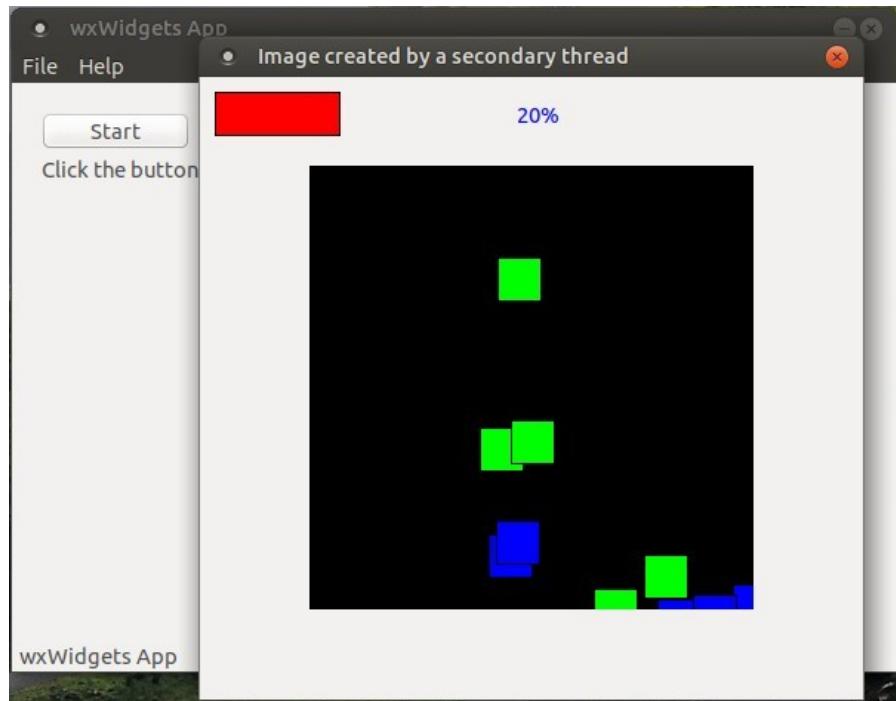
```

259         wxMilliSleep(200);
260     }
261
262     // if we don't release the GUI mutex the MyImageDialog won't be able
263     // to refresh
264     wxMutexGuiLeave();
265
266     // notify the dialog that another piece of our masterpiece is
267     // complete:
268     wxThreadEvent event( wxEVT_THREAD, GUILTHREAD_EVENT );
269     event.SetInt(i+1);
270     wxQueueEvent( m_dlg, event.Clone() );
271
272     // give the main thread the time to refresh before we lock the GUI
273     // mutex again
274     // FIXME: find a better way to do this!
275     wxMilliSleep(100);
276
277 }
278 // -----

```

စာရင်း ၅.၆: th-gui.cpp

ပရီဂရမ် ရဲ့ GUI နဲ့ thread ကို runလိုက်တဲ့ အခါ ရလာတဲ့ ရလဒ် ကို ပုံ ၅.၄ မှာ တွေ့နိုင် ပါတယ်။



ပုံ ၅.၄: th-gui.cpp ၏ ရလဒ်။

၅.၇ အခြားနည်းလမ်းများ

Thread တွေ သုံးရ တာ ရှုပ်ထွေး ဖြီး ဂရိစိက်ဖို့ လိုတာကြောင့် များသော အားဖြင့် ပိုမြီး ရှင်းလင်း လွယ်ကူ တဲ့ timer တွေ၊ idle time processing တွေ လှည့်သုံး ဖြီး ရှောင်လွှဲ နိုင် ပါတယ်။ တခြား event processing တွေ အားလုံး ဖြီးသွား တဲ့ အခါတိုင်း application object နဲ့ windows အားလုံး ကို idle event လုပ်ပါတယ်။ အကယ်၍ event handler မှာ RequestMore ကို မခေါ်ရင် နောက် တစ်ကြိမ် user interface event တွေ ပြန်ဖြစ်ပြီး၊ ပြန် process လုပ်ပြီး မှ ပြန် ခေါ်မှာ ဖြစ်ပါတယ်။ Idle event သုံးတဲ့ နမူနာ တစ်ခု ကို အောက်မှာ တွေ့နိုင် ပါတယ်။

```
class MyFrame : public wxFrame
{
public:
    ...
    void OnIdle(wxIdleEvent& event);
    // Do a little bit of work, return true if
```

```

// task finished
bool FinishedIdleTask();
};

BEGIN_EVENT_TABLE(MyFrame, wxFrame)
EVT_IDLE(MyFrame::OnIdle)
END_EVENT_TABLE()

void MyFrame::OnIdle(wxIdleEvent& event)
{
    // Do idle processing, ask for more idle
    // processing if we 'havent finished the task
    if (!FinishedIdleTask())
        event.RequestMore();
    event.Skip();
}

```

အကယ်၍ idle event ကို ချက်ချင်း စလုပ် စေချင် ရင် wxWakeUpIdle ကို ခေါ်နိုင် ပါတယ်။ Idle event processing တွေကို ချက်ချင်း လုပ်ဖို့ အတွက် wxApp::ProcessIdle ကိုလည်း ခေါ်နိုင် ပါတယ်။

အကယ်၍ processing တွေ အကြောကြီး လုပ်ရတဲ့ အလုပ် ကို လုပ်နေတဲ့န်း user interface က update မလုပ်နိုင် တော့ ပဲ သေနေ ရင် wxApp::Yield ကို ပုံမှန် ကြားညွှန် ခေါ်ပေးပြီး pending events တွေကို လုပ်ဆောင် နိုင် ပါတယ်။ အကယ်၍ display ကို ပဲ update လုပ်ချင်တာ ဆိုရင် တော့ yield အစား wxWindow::Update ကို ခေါ်နိုင် ပါတယ်။ Yield ကို ခေါ်တာ က reentrancy ပြဿနာ ဖြစ်နိုင် တာမို့ လက်ရှိ yield လုပ်နေတဲ့န်း ဆိုရင် ပြန်မ ခေါ်မ အောင် သူကို ခေါ်တဲ့ အချိန်မှာ true ကို pass လုပ်ပေးပြီး ကာကွယ် နိုင် ပါတယ်။

Appendix A

နောက်ဆက်တွဲ

၁.၁ စကားလုံးဖွင့်ဆိုချက်များ

GUI Graphical User Interface

IP Internet Protocol

TCP

Transmission Control Protocol

UART Universal asynchronous receiver-transmitter

UDP User Datagram Protocol

ဒေသ Data

Appendix B

Code Listing

J. Serial.h

```
1 //File: ce_serial.h
2 //Description: Serial communication class for Windows and Linux
3 //WebSite: http://cool-emerald.blogspot.sg/2017/05/serial-port-programming-in
4 //c-with.html
5 //MIT License (https://opensource.org/licenses/MIT)
6 //Copyright (c) 2017 Yan Naing Aye
7
8 // References
9 // https://en.wikibooks.org/wiki/Serial\_Programming/termios
10 // http://www.silabs.com/documents/public/application-notes/an197.pdf
11 // https://msdn.microsoft.com/en-us/library/ff802693.aspx
12
13 #ifndef SERIAL_H
14 #define SERIAL_H
15
16 #include <stdlib.h>
17 #include <stdio.h>
18 #include <string.h>
19 #include <string>
```

```
20 using namespace std;
21
22 #if defined (_WIN32_) || defined(_WIN32) || defined(WIN32) || defined(
23     __WINDOWS__) || defined(__TOS_WIN__)
24     #define CEWIN
25
26 #endif
27
28 #ifdef CEWIN
29
30 #include <windows.h>
31 #define READ_TIMEOUT 10      // milliseconds
32 inline void delay(unsigned long ms)
33 {
34     Sleep(ms);
35 }
36
37 #else
38
39 #include <unistd.h>
40 #include <fcntl.h>
41 #include <termios.h>
42 #include <sys/ioctl.h>
43 inline void delay(unsigned long ms)
44 {
45     usleep(ms*1000);
46 }
47
48 //Class definition
49 class Serial {
50     char rxchar;
51     string port;
52     long baud;
53     long dsizes;
54     char parity;
```

```
55 float stopbits;
56 #ifdef CEWIN
57     HANDLE hComm; //handle
58     OVERLAPPED osReader;
59     OVERLAPPED osWrite;
60     BOOL fWaitingOnRead;
61     COMMTIMEOUTS timeouts_ori;
62 #else
63     long fd;//serial_fd
64 #endif
65 public:
66     Serial();
67     Serial(string Device, long BaudRate, long DataSize, char ParityType, float
68             NStopBits);
69     ~Serial();
70     long Open(void); //return 0 if success
71     void Close();
72     char ReadChar(bool& success); //return read char if success
73     bool WriteChar(char ch); //return success flag
74     bool Write(char *data); //write null terminated string and return success
75             flag
76     bool SetRTS(bool value); //return success flag
77     bool SetDTR(bool value); //return success flag
78     bool GetCTS(bool& success);
79     bool GetDSR(bool& success);
80     bool GetRI(bool& success);
81     bool GetCD(bool& success);
82     bool IsOpened();
83     void SetPort(string Port);
84     string GetPort();
85     void SetBaudRate(long baudrate);
86     long GetBaudRate();
87     void SetDataSize(long nbits);
88     long GetDataSize();
89     void SetParity(char p);
90     char GetParity();
```

```
89     void SetStopBits(float nbits);
90     float GetStopBits();
91 }
92
93
94 Serial::Serial()
95 {
96 #ifdef CEWIN
97     hComm = INVALID_HANDLE_VALUE;
98     port = "\\\\.\\COM1";
99 #else
100    fd = -1;
101    port = "/dev/ttyS0";
102 #endif // defined
103    SetBaudRate(9600);
104    SetDataSize(8);
105    SetParity('N');
106    SetStopBits(1);
107 }
108
109 Serial::Serial(string Device, long BaudRate, long DataSize, char ParityType,
110                 float NStopBits)
111 {
112 #ifdef CEWIN
113     hComm = INVALID_HANDLE_VALUE;
114 #else
115     fd = -1;
116 #endif // defined
117     port = Device;
118     SetBaudRate(BaudRate);
119     SetDataSize(DataSize);
120     SetParity(ParityType);
121     SetStopBits(NStopBits);
122 }
123 Serial::~Serial()
```

J.O. SERIAL.H

©GO

```
124 {
125     Close();
126 }
127
128 void Serial::SetPort(string Device) {
129     port = Device;
130 }
131
132 string Serial::GetPort() {
133     return port;
134 }
135
136 void Serial::SetDataSize(long nbits) {
137     if ((nbits < 5) || (nbits > 8)) nbits = 8;
138     dsize=nbits;
139 }
140
141 long Serial::GetDataSize() {
142     return dsize;
143 }
144
145 void Serial::SetParity(char p) {
146     if ((p != 'N') && (p != 'E') && (p != 'O')) {
147 #ifdef CEWIN
148         if ((p != 'M') && (p != 'S')) p = 'N';
149 #else
150         p = 'N';
151 #endif
152     }
153     parity = p;
154 }
155
156 char Serial::GetParity() {
157     return parity;
158 }
```

```
160 void Serial::SetStopBits(float nbits) {
161     if (nbits >= 2) stopbits = 2;
162 #ifdef CEWIN
163     else if(nbits >= 1.5) stopbits = 1.5;
164 #endif
165     else stopbits = 1;
166 }
167
168 float Serial::GetStopBits() {
169     return stopbits;
170 }
171
172
173 #ifdef CEWIN
174
175 void Serial::SetBaudRate(long baudrate) {
176     if (baudrate < 300) baud = CBR_110;
177     else if (baudrate < 600) baud = CBR_300;
178     else if (baudrate < 1200) baud = CBR_600;
179     else if (baudrate < 2400) baud = CBR_1200;
180     else if (baudrate < 4800) baud = CBR_2400;
181     else if (baudrate < 9600) baud = CBR_4800;
182     else if (baudrate < 14400) baud = CBR_9600;
183     else if (baudrate < 19200) baud = CBR_14400;
184     else if (baudrate < 38400) baud = CBR_19200;
185     else if (baudrate < 57600) baud = CBR_38400;
186     else if (baudrate < 115200) baud = CBR_57600;
187     else if (baudrate < 128000) baud = CBR_115200;
188     else if (baudrate < 256000) baud = CBR_128000;
189     else baud = CBR_256000;
190 }
191
192 long Serial::GetBaudRate() {
193     return baud;
194 }
```

```
196 long Serial::Open()
197 {
198     if (IsOpened()) return 0;
199 #ifdef UNICODE
200     wstring wtext(port.begin(),port.end());
201 #else
202     string wtext = port;
203 #endif
204     hComm = CreateFile(wtext.c_str(),
205                         GENERIC_READ | GENERIC_WRITE,
206                         0,
207                         0,
208                         OPEN_EXISTING,
209                         FILE_ATTRIBUTE_NORMAL | FILE_FLAG_OVERLAPPED,
210                         0);
211     if (hComm == INVALID_HANDLE_VALUE) {return 1;}
212
213     if (PurgeComm(hComm, PURGE_TXABORT | PURGE_RXABORT | PURGE_TXCLEAR |
214 PURGE_RXCLEAR) == 0) {return 2;} //purge
215
216     //get initial state
217     DCB dcbOri;
218     bool fSuccess;
219     fSuccess = GetCommState(hComm, &dcbOri);
220     if (!fSuccess) {return 3;}
221
222     DCB dcb1 = dcbOri;
223
224     dcb1.BaudRate = baud;
225
226     if (parity == 'E') dcb1.Parity = EVENPARITY;
227     else if (parity == 'O') dcb1.Parity = ODDPARITY;
228     else if (parity == 'M') dcb1.Parity = MARKPARITY;
229     else if (parity == 'S') dcb1.Parity = SPACEPARITY;
230     else dcb1.Parity = NOPARITY;
```

```
231     dcb1.ByteSize = (BYTE)dsize;
232
233     if(stopbits==2) dcb1.StopBits = TWOSTOPBITS;
234     else if (stopbits == 1.5) dcb1.StopBits = ONE5STOPBITS;
235     else dcb1.StopBits = ONESTOPBIT;
236
237     dcb1.fOutxCtsFlow = false;
238     dcb1.fOutxDsrFlow = false;
239     dcb1.fOutX = false;
240     dcb1.fDtrControl = DTR_CONTROL_DISABLE;
241     dcb1.fRtsControl = RTS_CONTROL_DISABLE;
242     fSuccess = SetCommState(hComm, &dcb1);
243     delay(60);
244     if (!fSuccess) {return 4;}
245
246     fSuccess = GetCommState(hComm, &dcb1);
247     if (!fSuccess) {return 5;}
248
249     osReader = { 0 };// Create the overlapped event.
250     // Must be closed before exiting to avoid a handle leak.
251     osReader.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
252
253     if (osReader.hEvent == NULL) {return 6;}// Error creating overlapped
254     event; abort.
255     fWaitingOnRead = FALSE;
256
257     osWrite = { 0 };
258     osWrite.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
259     if (osWrite.hEvent == NULL) {return 7;}
260
261     if (!GetCommTimeouts(hComm, &timeouts_ori)) { return 8; } // Error
262     getting time-outs.
263     COMMTIMEOUTS timeouts;
264     timeouts.ReadIntervalTimeout = 20;
265     timeouts.ReadTotalTimeoutMultiplier = 15;
266     timeouts.ReadTotalTimeoutConstant = 100;
```

```
265     timeouts.WriteTotalTimeoutMultiplier = 15;
266     timeouts.WriteTotalTimeoutConstant = 100;
267     if (!SetCommTimeouts(hComm, &timeouts)) { return 9;} // Error setting
268     // time-outs.
269
270
271 void Serial::Close()
272 {
273     if (IsOpened())
274     {
275         SetCommTimeouts(hComm, &timeouts_ori);
276         CloseHandle(osReader.hEvent);
277         CloseHandle(osWrite.hEvent);
278         CloseHandle(hComm); //close comm port
279         hComm = INVALID_HANDLE_VALUE;
280     }
281 }
282
283 bool Serial::IsOpened()
284 {
285     if (hComm == INVALID_HANDLE_VALUE) return false;
286     else return true;
287 }
288
289
290
291 bool Serial::Write(char *data)
292 {
293     if (!IsOpened()) {
294         return false;
295     }
296     BOOL fRes;
297     DWORD dwWritten;
298     long n = strlen(data);
299     if (n < 0) n = 0;
```

```
300    else if(n > 1024) n = 1024;
301
302    // Issue write.
303
304    if (!WriteFile(hComm, data, n, &dwWritten, &osWrite)) {
305        // WriteFile failed, but it isn't delayed. Report error and abort.
306        if (GetLastError() != ERROR_IO_PENDING) {fRes = FALSE;}
307        else { // Write is pending.
308            if (!GetOverlappedResult(hComm, &osWrite, &dwWritten, TRUE)) fRes =
309                FALSE;
310            else fRes = TRUE; // Write operation completed successfully.
311        }
312    }
313
314
315    bool Serial::WriteChar(char ch)
316    {
317        char s[2];
318        s[0]=ch;
319        s[1]=0; //null terminated
320        return Write(s);
321    }
322
323    char Serial::ReadChar(bool& success)
324    {
325        success = false;
326        if (!IsOpened()) {return 0;}
327
328        DWORD dwRead;
329        DWORD length=1;
330        BYTE* data = (BYTE*)(&rxchar);
331        //the creation of the overlapped read operation
332        if (!fWaitingOnRead) {
333            // Issue read operation.
334            if (!ReadFile(hComm, data, length, &dwRead, &osReader)) {
```

```
335     if (GetLastError() != ERROR_IO_PENDING) { /*Error*/
336     else { fWaitingOnRead = TRUE; /*Waiting*/
337     }
338     else {if(dwRead==length) success = true;}//success
339   }
340
341
342 //detection of the completion of an overlapped read operation
343 DWORD dwRes;
344 if (fWaitingOnRead) {
345     dwRes = WaitForSingleObject(osReader.hEvent, READ_TIMEOUT);
346     switch (dwRes)
347     {
348     // Read completed.
349     case WAIT_OBJECT_0:
350         if (!GetOverlappedResult(hComm, &osReader, &dwRead, FALSE)) {/*Error*/
351     }
352     else {
353         if (dwRead == length) success = true;
354         fWaitingOnRead = FALSE;
355         // Reset flag so that another operation can be issued.
356     } // Read completed successfully.
357     break;
358
359     case WAIT_TIMEOUT:
360         // Operation isn't complete yet.
361     break;
362
363     default:
364         // Error in the WaitForSingleObject;
365     break;
366   }
367   return rxchar;
368 }
369 }
```

```
370 bool Serial::SetRTS(bool value)
371 {
372     bool r = false;
373     if (IsOpened()) {
374         if (value) {
375             if (EscapeCommFunction(hComm, SETRTS)) r = true;
376         }
377         else {
378             if (EscapeCommFunction(hComm, CLRRTS)) r = true;
379         }
380     }
381     return r;
382 }
383
384 bool Serial::SetDTR(bool value)
385 {
386     bool r = false;
387     if (IsOpened()) {
388         if (value) {
389             if (EscapeCommFunction(hComm, SETDTR)) r = true;
390         }
391         else {
392             if (EscapeCommFunction(hComm, CLRDTR)) r = true;
393         }
394     }
395     return r;
396 }
397
398 bool Serial::GetCTS(bool& success)
399 {
400     success = false;
401     bool r = false;
402     if (IsOpened()) {
403         DWORD dwModemStatus;
404         if (GetCommModemStatus(hComm, &dwModemStatus)) {
405             r = MS_CTS_ON & dwModemStatus;
```

```
406     success = true;
407 }
408 }
409 return r;
410 }

411

412 bool Serial::GetDSR(bool& success)
413 {
414     success = false;
415     bool r = false;
416     if (IsOpened()) {
417         DWORD dwModemStatus;
418         if (GetCommModemStatus(hComm, &dwModemStatus)) {
419             r = MS_DSR_ON & dwModemStatus;
420             success = true;
421         }
422     }
423     return r;
424 }

425

426 bool Serial::GetRI(bool& success)
427 {
428     success = false;
429     bool r = false;
430     if (IsOpened()) {
431         DWORD dwModemStatus;
432         if (GetCommModemStatus(hComm, &dwModemStatus)) {
433             r = MS_RING_ON & dwModemStatus;
434             success = true;
435         }
436     }
437     return r;
438 }

439

440 bool Serial::GetCD(bool& success)
441 {
```

```
442     success = false;
443     bool r = false;
444     if (IsOpened()) {
445         DWORD dwModemStatus;
446         if (GetCommModemStatus(hComm, &dwModemStatus)) {
447             r = MS_RLSD_ON & dwModemStatus;
448             success = true;
449         }
450     }
451     return r;
452 }
453
454 #else //for POSIX
455
456 long Serial::Open(void) {
457
458     struct termios settings;
459     memset(&settings, 0, sizeof(settings));
460     settings.c_iflag = 0;
461     settings.c_oflag = 0;
462
463     settings.c_cflag = CREAD | CLOCAL; //see termios.h for more information
464     if (dsize==5) settings.c_cflag |= CS5; //no change
465     else if (dsize == 6) settings.c_cflag |= CS6;
466     else if (dsize == 7) settings.c_cflag |= CS7;
467     else settings.c_cflag |= CS8;
468
469     if (stopbits==2) settings.c_cflag |= CSTOPB;
470
471     if (parity != 'N') settings.c_cflag |= PARENB;
472
473     if (parity == '0') settings.c_cflag |= PARODD;
474
475     settings.c_lflag = 0;
476     settings.c_cc[VMIN] = 1;
477     settings.c_cc[VTIME] = 0;
```

```
478
479 fd = open(port.c_str(), O_RDWR | O_NONBLOCK);
480 if (fd == -1) {
481     return -1;
482 }
483 cfsetospeed(&settings, baud);
484 cfsetispeed(&settings, baud);
485
486 tcsetattr(fd, TCSANOW, &settings);
487
488 int flags = fcntl(fd, F_GETFL, 0);
489 fcntl(fd, F_SETFL, flags | O_NONBLOCK);
490
491 return 0;
492 }
493
494 void Serial::Close() {
495     if(IsOpened()) close(fd);
496     fd=-1;
497 }
498
499 bool Serial::IsOpened()
500 {
501     if(fd==(-1)) return false;
502     else return true;
503 }
504
505 void Serial::SetBaudRate(long baudrate) {
506     if (baudrate < 50) baud = B0;
507     else if (baudrate < 75) baud = B50;
508     else if (baudrate < 110) baud = B75;
509     else if (baudrate < 134) baud = B110;
510     else if (baudrate < 150) baud = B134;
511     else if (baudrate < 200) baud = B150;
512     else if (baudrate < 300) baud = B200;
513     else if (baudrate < 600) baud = B300;
```

```
514     else if (baudrate < 1200) baud = B600;
515     else if (baudrate < 2400) baud = B1200;
516     else if (baudrate < 4800) baud = B2400;
517     else if (baudrate < 9600) baud = B4800;
518     else if (baudrate < 19200) baud = B9600;
519     else if (baudrate < 38400) baud = B19200;
520     else if (baudrate < 57600) baud = B38400;
521     else if (baudrate < 115200) baud = B57600;
522     else if (baudrate < 230400) baud = B115200;
523     else baud = B230400;
524 }
525
526 long Serial::GetBaudRate() {
527     long baudrate=9600;
528     if (baud < B50) baudrate = 0;
529     else if (baud < B75) baudrate = 50;
530     else if (baud < B110) baudrate = 75;
531     else if (baud < B134) baudrate = 110;
532     else if (baud < B150) baudrate = 134;
533     else if (baud < B200) baudrate = 150;
534     else if (baud < B300) baudrate = 200;
535     else if (baud < B600) baudrate = 300;
536     else if (baud < B1200) baudrate = 600;
537     else if (baud < B2400) baudrate = 1200;
538     else if (baud < B4800) baudrate = 2400;
539     else if (baud < B9600) baudrate = 4800;
540     else if (baud < B19200) baudrate = 9600;
541     else if (baud < B38400) baudrate = 19200;
542     else if (baud < B57600) baudrate = 38400;
543     else if (baud < B115200) baudrate = 57600;
544     else if (baud < B230400) baudrate = 115200;
545     else baudrate = 230400;
546     return baudrate;
547 }
548 char Serial::ReadChar(bool& success)
549 {
```

```
550     success=false;
551     if (!IsOpened()) {return 0; }
552     success=read(fd, &rxchar, 1)==1;
553     return rxchar;
554 }
555
556 bool Serial::Write(char *data)
557 {
558     if (!IsOpened()) {return false; }
559     long n = strlen(data);
560     if (n < 0) n = 0;
561     else if(n > 1024) n = 1024;
562     return (write(fd, data, n)==n);
563 }
564
565 bool Serial::WriteChar(char ch)
566 {
567     char s[2];
568     s[0]=ch;
569     s[1]=0;//null terminated
570     return Write(s);
571 }
572
573 bool Serial::SetRTS(bool value) {
574     long RTS_flag = TIOCM_RTS;
575     bool success=true;
576     if (value) {//Set RTS pin
577         if (ioctl(fd, TIOCMBIS, &RTS_flag) == -1) success=false;
578     }
579     else {//Clear RTS pin
580         if (ioctl(fd, TIOCMBIC, &RTS_flag) == -1) success=false;
581     }
582     return success;
583 }
584
585 bool Serial::SetDTR(bool value) {
```

```
586 long DTR_flag = TIOCM_DTR;
587 bool success=true;
588 if (value) { //Set DTR pin
589     if (ioctl(fd, TIOCMBIS, &DTR_flag) == -1) success=false;
590 }
591 else { //Clear DTR pin
592     if (ioctl(fd, TIOCMBIC, &DTR_flag) == -1) success=false;
593 }
594 return success;
595 }
596
597 bool Serial::GetCTS(bool& success) {
598     success=true;
599     long status;
600     if (ioctl(fd, TIOCMGET, &status) == -1) success=false;
601     return ((status & TIOCM_CTS) != 0);
602 }
603
604 bool Serial::GetDSR(bool& success) {
605     success=true;
606     long status;
607     if (ioctl(fd, TIOCMGET, &status) == -1) success=false;
608     return ((status & TIOCM_DSR) != 0);
609 }
610
611 bool Serial::GetRI(bool& success) {
612     success=true;
613     long status;
614     if (ioctl(fd, TIOCMGET, &status) == -1) success=false;
615     return ((status & TIOCM_RI) != 0);
616 }
617
618 bool Serial::GetCD(bool& success) {
619     success=true;
620     long status;
621     if (ioctl(fd, TIOCMGET, &status) == -1) success=false;
```

J.C. SERIAL.H

၁၆၃

```
622     return ((status & TIOCM_CD) != 0);  
623 }  
624 #endif  
625  
626 #endif // SERIAL_H
```

စာရင်း J.C: ce_serial.h