1 User manual for Scilab2C

This section describes steps to be followed for using Scilab2C. Pre-requisites are mentioned followed by procedure to install Scilab2C.

1.1 Installation

1.1.1 Prerequisites

There are few prerequisites or some packages must be pre installed before we can use Scilab2C. These are:

- scilab >=5.5.1.
- scilab-arduino toolbox (If using Scilab2C to generate code for Arduino)
- Arduino makefile (https://github.com/sudar/Arduino-Makefile). Install using 'sudo apt-get install arduino-mk.
- BCM2835 C library for RasberryPi (http://wiringpi.com/)
- RasberryPi tools (For cross compiling code for RasberryPi)

1.1.2 Installing Scilab2C

Before we can use 'Scilab2C' extension, we need to install latest version of Scilab2C. Follow following procedure to get latest source code from github repo.

- Open terminal window. (Ctrl+Shift+T is shortcut).
- Change current directory to '/path/to/scilab/share/scilab/contrib'
- Clone the git repo using following command:
  `git clone https://github.com/siddhu8990/Scilab2C.git`
- Make sure a directory named 'Scilab2C' is present in 'contrib' folder.
- Open Scilab.
- Run 'builder.sce file present in Scilab2C/2.3-1 using 'exec'. This generates binary files from source files.
  `exec(/path/to/Scilab2C/2.3-1/builder.sce)`

- In 'Home/.scilab/scilabx.x.x' make a new file '.scilab' if it does not exist already. Open .scilab using suitable editor. Add following line in this file:
  `exec(/path/to/Scilab2C/2.3-1/loader.sce)`
  This will load the 'scilab2c' everytime scilab is started.

1.1.3 Installing supporting packages

Most of the supporting packages or libraries which are required are provided with the toolbox. But they were compiled using latest source code available at release of toolbox. If you want to use latest libraries, steps to compile the same are listed below. You can follow these steps and replace old files with newly generated ones.

- **scilab-arduino toolbox**
- **RasberryPi tools**
  - Make a folder named 'RasberryPi tools' somewhere on the harddisk.
  - Open terminal and change directory to 'RasberryPi tools'. Clone 'Tools' repo using 'git clone https://github.com/raspberrypi/tools.git'.
  - Add location of toolchain to your 'PATH' variable.
    `'export PATH=$PATH:/location/of/tools/folder/arm-bcm2708/gcc-linaro-arm-li`
- **BCM2835 C library for RasberryPi**
  - Before going further, make sure that you have installed 'RasberryPi tool' following the instructions given above.
  - Download latest source code from '`http://www.airspayce.com/mikem/bcm2835`'.
  - Extract source code at some suitable location on harddrive.
  - Open the terminal window and change current directory to the location where source is extracted.
  - Execute following command
    `./configure -host=arm CC=arm-linux-gnueabihf-gcc ar=arm-linux-gnueabihf-ar`
  - Then execute 'make' to cross compile the library. Don't do 'make install' as it is noramlly next step.
- **Cross compiling Lapack and Blas for RasberryPi**
  - Download latest source code for Lapack from '`http://www.netlib.org/lapack/`'. Extract source files at some suitable location.
  - Open file 'make.inc.example' given in Lapack folder using some editor.

- Edit following items as shown:
  - FORTRAN = arm-linux-gnueabihf-gfortran
  - LOADER = arm-linux-gnueabihf-gfortran
  - CC = arm-linux-gnueabihf-gcc
  - ARCH = arm-linux-gnueabihf-ar
  - RANLIB = arm-linux-gnueabihf-ranlib
- Since we are cross compiling for some other platform, normal way compiling will not work.
- Open terminal window and change current directory to laplack directory.
- We will need to compile BLAS, CBLAS and Lapack separately and in same order.
- Change current directory to /path/to/lapack/BLAS/SRC and run 'make'. This will generate 'librefblas.a' in Lapack folder.
- Now change current directory to /path/to/laplack/CBLAS and run 'make'. This will generate 'libcblas.a' in Lapack folder.
- Now change current directory to /path/to/laplack/SRC and run 'make'. This will generate 'liblapack.a' in Lapack folder. Now replace the generated lib files in 'src/c/hardware/rasberrypi/libraries' in 'scilab2c' source folder.

- **GNU Scientific Library (GSL) for RasberryPi**
  - Before going further, make sure that you have installed 'RasberryPi tool' following the instructions given here.
  - Get latest source code for GSL from `ftp://ftp.gnu.org/gnu/gsl/`.
  - Extract source code at some suitable location on harddrive.
  - Open the terminal window and change current directory to the location where source is extracted.
  - Execute following command
    `./configure -host=arm CC=arm-linux-gnueabihf-gcc ar=arm-linux-gnueabihf-ar --enable-static`
  - Then execute 'make libgsl.la' to cross compile the library. Don't do 'make install' as it is noramlly next step.
  - Library 'libgsl.a' is created in folder '.libs'. By default this folder is hidden.
  - Now replace the generated lib file in 'src/c/hardware/rasberrypi/libraries' in 'scilab2c' source folder.

1.2 Using Scilab2C for C code generation

Scilab2C extension in scilab can be used for generating C code from a scilab script. Currently it supports three types of target platforms:

- **Standalone C code**: General C code which can be compiled using any compiler.
- **Arduino** : Generated code with few additions can be run on an Arduino board. (A scilab-arduino extension is required)
- **AVR** : Code can be generated for using hardware peripherals of AVR microcontroller

You can follow following steps for generating C code using scilab2c extension for required target platforms.

- **Generating standalone C code**

  1. Write the scilab script first which is to be converted to C. Scilab code can contain single file or many files, but each file must be a scilab function. There must be one main scilab file in case project contains many files, from which execution of code starts. All scilab files must be in a single folder.
  2. Before a scilab can be translated to c file, some function annotations should be added manually. Function annotations gives information about no. of inputs/outputs, their types etc. Refer 'Function annotations' for more details.
  3. Type 'sci2c_gui' or 'scilab2c' in scilab console. This will prompt the GUI of Scilab2C extension as shown in figure 1
  4. Click 'Browse' next to 'Main file name' textbox, browse to location of main scilab file and select it. (Refer figure 2)
  5. If scilab code contains many files, select folder containing these file by clicking 'Browse' next to 'Sub-functions' textbox.
  6. Create a new folder somewhere on the disk, preferably in same folder containing scilab files. Select this newly created folder by clicking 'Browse' next to 'Directory name' textbox. (Refer figure 3)
  7. Choose appropriate options from 'Options' box. Different options are explained below:

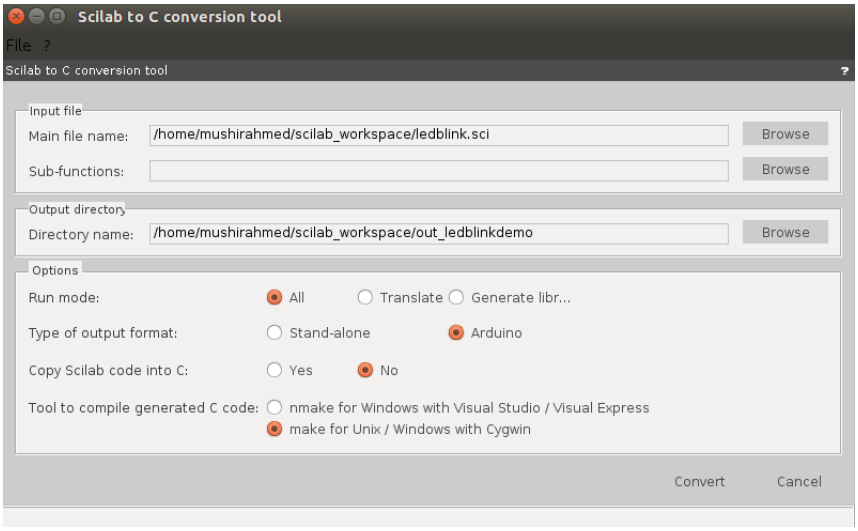

Figure 1: GUI for 'Scilab2C'

(a) Run mode : If only directory structure is to generated in output directory, select 'Generate library'. If only conversion of scilab files is to be done, select 'Translate'. In case both are to be done, select 'All'.

(b) Target platform : To generate standalone C code, select 'Standalone C' from dropdown. (Refer figure 4)

(c) Copy scilab code into C: Select 'Yes' or 'No' accordingly.

(d) Tool to complie generated C code: Select appropriate option depending upon platform on which generated code will be complied.

8. Confirm everything again and then press 'Convert' button. (Refer figure 5)
9. After clicking 'Convert', scilab code will be run in scilab, to check for any errors. If code runs successfully, a prompt will occur asking if you want to continue to code conversion or not. Select 'Yes'. If scilab code doesnt run correctly then code conversion is stopped there itself. Correct the scilab code and follow the steps again.
10. After selecting 'Yes' for code conversion, code conversion starts. If code conversion is done successfully, you will see the message in command window.
11. Generated code can be seen in output folder. By default a makefile is generated which uses 'GCC' compiler to compile the C code. You can compile this code using 'make'. Open output folder in terminal and type 'make' and press Enter. Once code is compiled successfully, it is

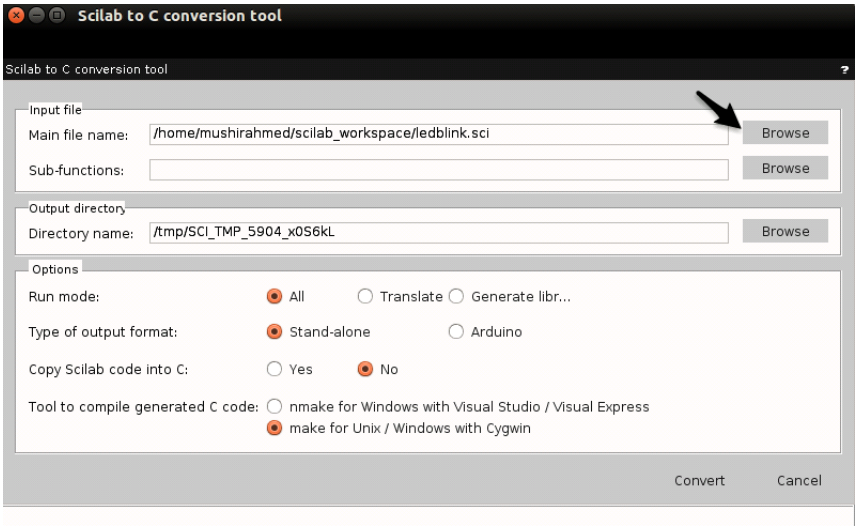

Figure 2: Select 'main' scilab file for conversion

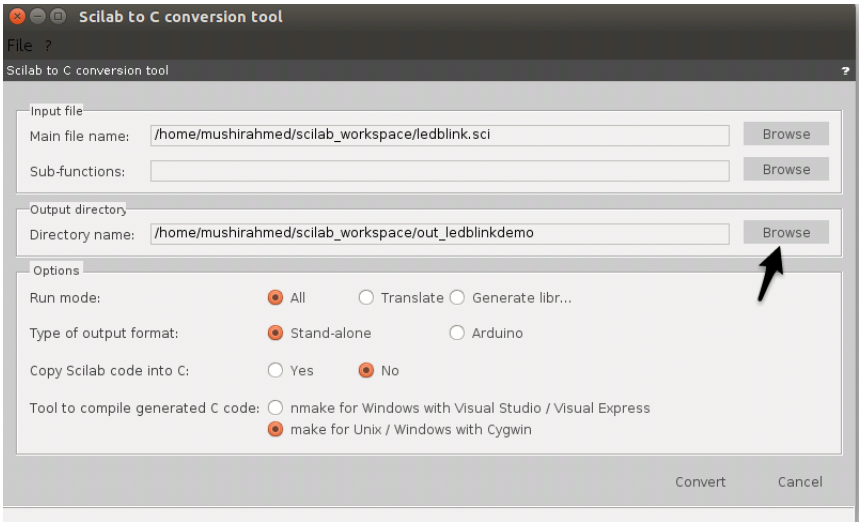

Figure 3: Select output folder

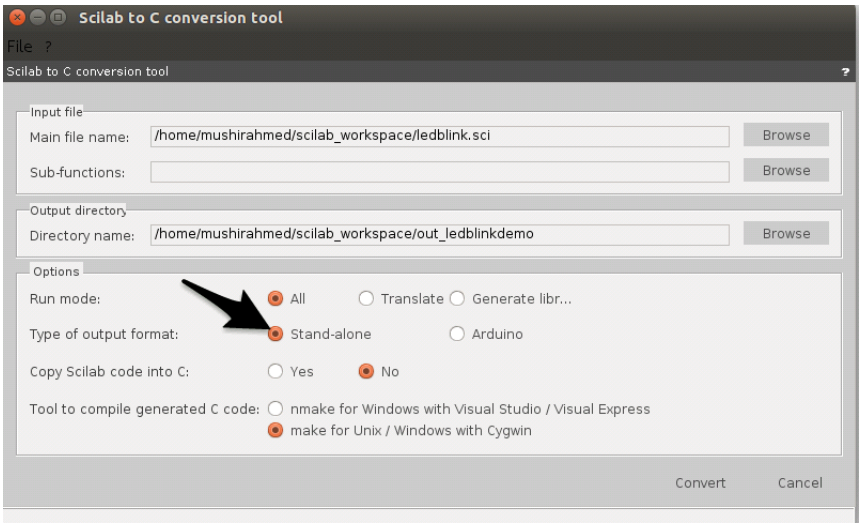

Figure 4: Select 'Standalone C' from dropdown

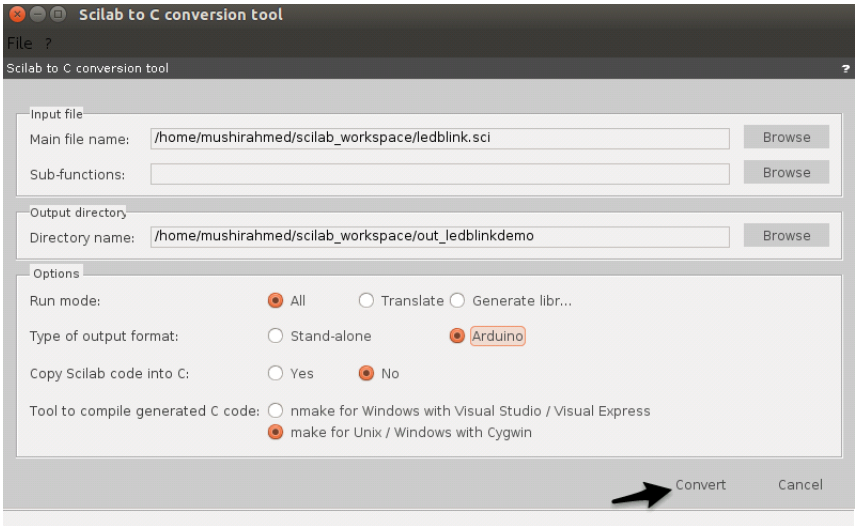

Figure 5: Select output folder

run in terminal and output can be seen in terminal window. Check the output for correctness. If code did not behave as expected, correct the scilab code and follow the process again.

- **Generating code for Arduino**

  1. Write the scilab script first which is to be converted to C. Scilab code can contain single file or many files, but each file must be a scilab function. There must be one main scilab file in case project contains many files, from which execution of code starts. All scilab files must be in a single folder. You can verify working of scilab script by runnig it on an Arduino board. Modify the script untill code behaves as expected. Once script is finalised, remove the commands 'open_serial' and 'close_serial'.
  2. Type 'sci2c_gui' or 'scilab2c' in scilab console. This will prompt the GUI of Scilab2C extension as shown in figure 1
  3. Click 'Browse' next to 'Main file name' textbox, browse to location of main scilab file and select it. (Refer figure 2)
  4. If scilab code contains many files, select folder containing these file by clicking 'Browse' next to 'Sub-functions' textbox.
  5. Create a new folder somewhere on the disk, preferably in same folder containing scilab files. Select this newly created folder by clicking 'Browse' next to 'Directory name' textbox. (Refer figure 3)
  6. Choose appropriate options from 'Options' box. Different options are explained below:
     (a) Run mode : If only directory structure is to generated in output directory, select 'Generate library'. If only conversion of scilab files is to be done, select 'Translate'. In case both are to be done, select 'All'.
     (b) Target platform : To generate C code for arduino, select 'Arduino' from dropdown. (Refer figure 6)
     (c) Copy scilab code into C: Select 'Yes' or 'No' accordingly.
     (d) Tool to complie generated C code: Select appropriate option depending upon platform on which generated code will be complied.
  7. Confirm everything again and then press 'Convert button. (Refer figure 5)
  8. Code conversion will start, promting different messages in command window. If conversion completes successfully, prompt will occur in command window indicationg the same.

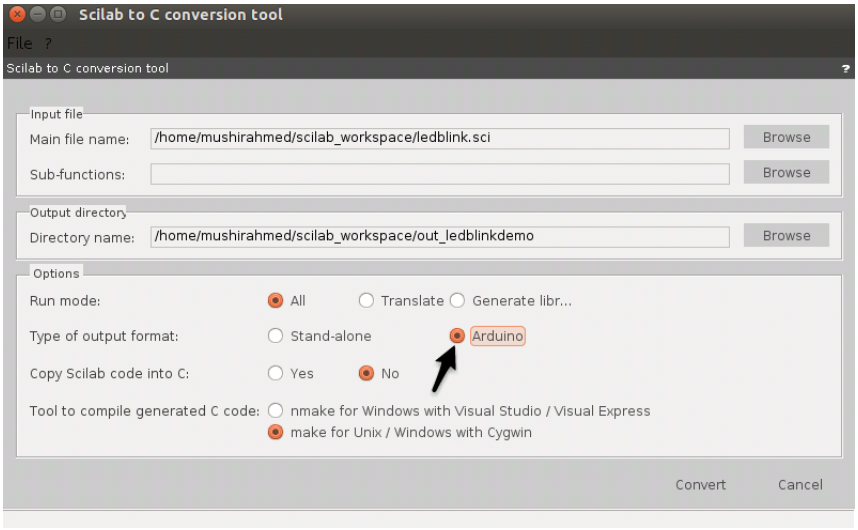

Figure 6: Select 'Standalone C' from dropdown

9. Generated code can be seen in output folder. A separate folder named 'Arduino' is created, which contains a makefile and an arduino sketch file – sci2c_arduino.ino.
10. Open 'Makefile' using suitable text editor. Change following parameters according to board and connection:
    (a) BOARD_TAG
    (b) ARDUINO_PORT
11. Open the terminal and change current directory to the directory containing modified Arduino sketch and then compile by typing 'make' in terminal.
12. If code is compiled successfully, you can upload it to arduino using 'make upload' command.
13. If code doesnot behave as expected, modify sclab code and follow the steps again.

- **Generating code for AVR**

1.3 Function Annotations

Each scilab function/file should start with the function annotation section having following structure: