Searduino Manual

C/C++ environment
Stubs
Simulator
... for Arduino

Version: 20120110-235112
Date: 10 Jan 2012

Henrik Sandklef

Table of Contents

1 Background

2 Abbreviations

- Arduino program - a program that is written for the Arduino board. Uses only the Arduio and avr APIs.
- Arduino stub - library implementing the Arduino and avr APIs
- Streamed input/output - instead of a fullblown simulator GUI Searduino provides you with a stdin/stdout interface. This can be used to script (bash, Python..) your test cases. Using programs such as netcat you can also run the Arduino program on one PC and the test on another PC.

3 Supported boards and platforms

Supported Arduino boards

- Uno - http://arduino.cc/en/Main/arduinoBoardUno
- Mega - http://arduino.cc/en/Main/ArduinoBoardMega
- Due - http://arduino.cc/en/Main/arduinoBoardDuemilanove

Supported Operating Systems

- GNU/Linux (source code supports both 32 and 64 bits. Binaries available for 32 bits only)

4 Installing Searduino

4.1 Source code releases

Download from
http://download.savannah.gnu.org/releases/searduino/

4.2 Latest Source code

Download from
http://git.savannah.gnu.org/gitweb/?p=searduino.git;a=snapshot;h=HEAD;sf=tgz

4.3 From git repository

Download from
`git clone git://git.savannah.nongnu.org/searduino.git`

4.4 Binary releases

4.4.1 GNU/Linux

Create a installation directiory (e.g /opt/searduino)
`mkdir -p /opt/searduino`

Go to the installation directory
`cd /opt/searduino`

Download a release from
http://download.savannah.gnu.org/releases/searduino/bin/
E.g http://download.savannah.gnu.org/releases/searduino/bin/searduino-bin-0.4-x86.tar.gz

Unpack
`tar zxvf searduino-bin-0.4-x86.tar.gz`

Verify installation - with the digpins example
`cd example/digpins/`
Make sure that the SEARDUINO_PATH in the Makefile points to your Searduino installation dir.

Build blinker program for PC
`make`

Set up environment to find the Searduino shared libs
`export LD_LIBRARY_PATH=/opt/searduino/libs`

Execute blinker
`make check-sw`

The blinker program should run and print out (the printouts comes from the stub libraries). Interrupt the program by sending a signal, e g by pressing Ctrl-C.

5 Using Searduino

In the previous chapter we looked a bit at the digpins example, so we now have some feeling for what a Searduino Makefile contains. We will now proceed by writing our first Arduino program using Searduino.

5.1 Writing the first program

5.1.1 The first C file

To use the Arduino functionality you need to inlude Arduino.h, so we need to add this:

```
#include <Arduino.h>
```

When using Arduino IDE you're used to having the `loop` function as the starting point for the program. With Searduino we're back to the normal C way with a `main` function, so we need to define a main function.

```
int main(void) { }
```

As with the loop function you write when you're using the Arduino IDE, the main function needs to never exit or return. It's a simple control loop (see http://en.wikipedia.org/wiki/Embedded_system#Simple_control_loop).

So a very simple main function looks like this

```
int main(void)
{
 for(;;)
  {
 digitalWrite(13, 1);
 delay(100);
 digitalWrite(13, 0);
 delay(100);
  }
}
```

Note: this program sets pin 13 high and low with 0.1 secs interval. You don't need to connect a led to output pin 13, since pin 13 already has a built in led on the board.

5.1.2 Write the first Makefile

Inporant settings in the Makefile

- SEARDUINO_PATH - should be set to the directory of your Searduino installation
- PROG - name of the program to build
- SRC_C - a list (separated with space) of C files to compile
- SRC_CXX - a list (separated with space) of C++ files to compile
- MAIN_SRC - the C (not C++) file containing the main fuction. Should not be included in the SRC_C variable

- ARDUINO - should be set to the type of software you want to build (see **Build types** below)

Include the searduino makefile
You need to include somce settings, targets and rules from Searduino. This is done by adding the following line to your Makefile.

```
include $(SEARDUINO_PATH)/mk/searduino.mk
```

Note: You don't have to use the makefiles provided by Searduino. The makefiles do however provide a lot of help (board settings etc).

5.1.3 Building the program for PC

To build your software to be executed on your PC:

make sure the the variable **ARDUINO** in the Makefile is set to **stub**.

and type:

```
make clean
make
```

To run the program

```
./blinker
```

5.1.4 Building the program for UNO

To build your software to be executed on your PC:

make sure the the variable **ARDUINO** in the Makefile is set to **uno** and type:

By setting ARDUINO to uno the Searduino makefiles will use the settings for building and uploading for the Arduino UNO board.
To build the program, all we have to do now is to type:

```
make clean
make
```

To upload and run the program on the Arduino UNO board:

```
make upload
```

You should now be able to see the built in led (pin 13) flash. If not, the author of this document need to his homework.

6 Build for different targets

With Searduino it's (relatively) easy to compile your program for various boards. You decide what targets to build for with the ARDUINO variable in the Makefile. The following values of that variable are implemented.

Build types

- uno - builds software for the Arduino UNO board
- mega - builds software for the Arduino Mega board
- due - builds software for the Arduino Mega board
- stub - builds software for the PC

7 Various Searduino functions/macros

7.1 print functions/macros

8 Simulators

With Searduino you can easily build your code with the following stubs/simulators:

- stub - the Arduino fuctions print whne they are being called (stdout by default)
- stream - same as with stub, but now also with a listening (stdin) thread to which you can send commands (such as settig digital input pin 2 to 1).
- pardon - a Simulator, written in Python, using the Python simulator interface
- xxxx - a simulator, written in C++/Qt/Qml. simulator interface

8.1 Stub

8.2 Streamed simulator

8.2.1 Scripting with bash

8.2.2 Scripting over the network

8.3 Pardon simulator

8.4 xxx simulator

9 Using the Python scripting interface

10 Debugging Arduino code

11 Examples

11.1 Example Makefile

```
SEARDUINO_PATH=/opt/searduino
PROG=blinker
SRC_C=blink.c
SRC_CXX= stuff.cpp morestuff.cpp
MAIN_SRC=main.c
ARDUINO=stub
$(PROG):
upload:
include $(SEARDUINO_PATH)/mk/searduino.mk
```

11.2 Example C code

```
#include <Arduino.h>

int main(void)
{
 for(;;)
  {
 digitalWrite(13, 1);
 delay(100);
 digitalWrite(13, 0);
 delay(100);
  }
}
```

12 Write your own simulator

12.0.1 Writing a simulator in C/C++

12.0.2 Writing a simulator in Python