AVR Development with avr-gcc

Installation

First, you will need to install the toolchain with a package manager. The toolchain consists of the compiler `avr-gcc` and the uploader `avrdude`. The following commands installs these packages with `brew` on MacOS, with the first line making `brew` aware of the packages and the second installing them. If you're on a Linux system and/or have a different package manager, simply replace `brew` with the name of your package manager and omit the first line

```
brew tap osx-cross/avr
brew install avr-gcc avrdude
```

Once you've run the installation commands, you can check that your installation was successfull by typing `avr-gcc` and `avrdude` into a terminal. You should see error messages like those shown below, which indicate the toolchain is working as expected.

```
avr-gcc: fatal error: no input files
compilation terminated.
```

```
Usage: avrdude [options]
Options:
  -p <partno>                Required. Specify AVR device.
  -b <baudrate>              Override RS-232 baud rate.
...
```

Now's also a good time to check you have `make` installed for later. If you type `make` into a terminal and see an error like

```
make: *** No targets specified and no makefile found. Stop.
```

Then `make` is installed and working correctly. If you see a `command not found: make` error, you can either install the Command Line Developer Tools if you're on MacOS (you may be prompted to do this if `make` is not already installed) or you can install it through a package manager.

Automated Build with make

Instead of manually typing a series of commands each time you want to compile and upload your project, you can use a build tool like `make` to streamline the process. The first stage of using `make` is to supply a configuration, or Makefile, so that it knows how to build your project. For each project you make, you will need to create a separate folder and copy the Makefile off Blackboard into a file named `Makefile` (note there is no file extension).

Configuration

To upload projects to the AVR microcontroller, the Makefile must be configured with the correct upload port for the programmer. Before connecting the programmer to your computer, run the following command to list the connected USB devices.

```
ls /dev/cu.usbmodem*
```

On Linux, USB port names are somewhat different so you should instead run

```
ls /dev/tty*
```

Now connect the programmer to the AVR and your computer, and run the command again. Two new USB connections should appear, in the general format of `cu.usbmodemNNNNNNNNN` where the `N` represents a digit. The port with the lower value is the programmer itself, while the port with the higher value is the serial port provided by the programmer. Copy the full path of the port and paste it after the = sign on line 2 of the Makefile (the line starting with `UPLOAD_PORT =` . To test if the programmer is working correctly, run `make test` (ensuring you are in the project directory). If an AVR board is connected to the programmer, you should see some information about the programmer and AVR board with no error messages. If the programmer is connected but the AVR is not, you will see information about the programmer and an error message at the end as it attempts to connect. The output is quite long, but the last few lines should look something like this

```
         Programmer Type : STK500V2
         Description     : Atmel STK500 Version 2.x firmware
         Programmer Model: STK500
         Hardware Version: 15
         Firmware Version Master : 2.10
         Topcard         : Unknown
         Vtarget         : 4.7 V
         SCK period      : 0.5 us
         Varef           : 0.0 V
         Oscillator      : Off

avrdude: stk500v2_command(): command failed
avrdude: initialization failed, rc=-1
         Double check connections and try again, or use -F to override
         this check.


avrdude done.  Thank you.

make: *** [test] Error 1
```

If you have specified the serial port instead of the programming port, or another port not provided by the programmer, you will see a slow stream of `avrdude: stk500v2_ReceiveMessage(): timeout` messages. Terminate the program by pressing Control-C and change the port in the Makefile.

If the programmer is not connected, or you specify another unconnected port, you will receive an error that ends with the message shown below.

```
avrdude: ser_open(): can't open device "/dev/cu.usbmodemNNNNNNNN": No such file or directory
avrdude: opening programmer "stk500v2" on port "/dev/cu.usbmodemNNNNNNNN" failed

avrdude done.  Thank you.

make: *** [test] Error 1
```

The port specification process only needs to be performed once, the programmer will have the same port each time you connect it. It may, however, be useful to run `make test` when you connect the programmer to ensure it is working as expected.

Use

The Makefile provides a variety of helpful commands, one of which you have already seen. A complete list of available commands is below `make` - compiles any changes that have been made since the last compile, but does not upload

`make test` - tests programmer connectivity

`make upload` - uploads to the AVR, compiling first if necessary

`make clean` - deletes intermediate files, good for forcing a full recompile

`make disasm` - produces a full assembly listing for your program

Extracting the Hex File

As part of the second assignment, you will be required to submit a `.hex` file of your code. Before extracting the hex file, run `make clean` and then `make` to ensure the hex file contains the latest version of your code. The hex file will then be located in the same directory as your project with the name `main.hex`.