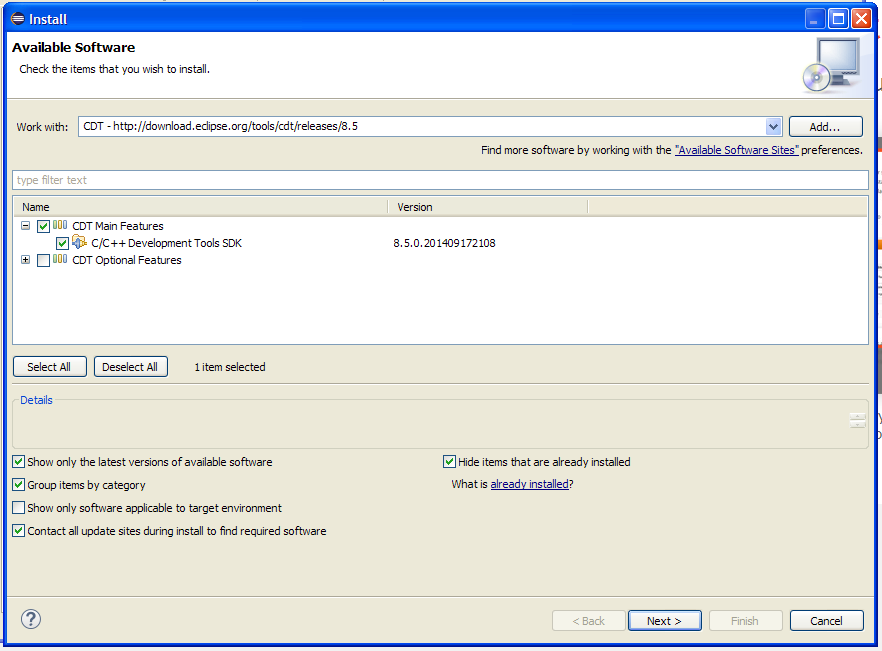
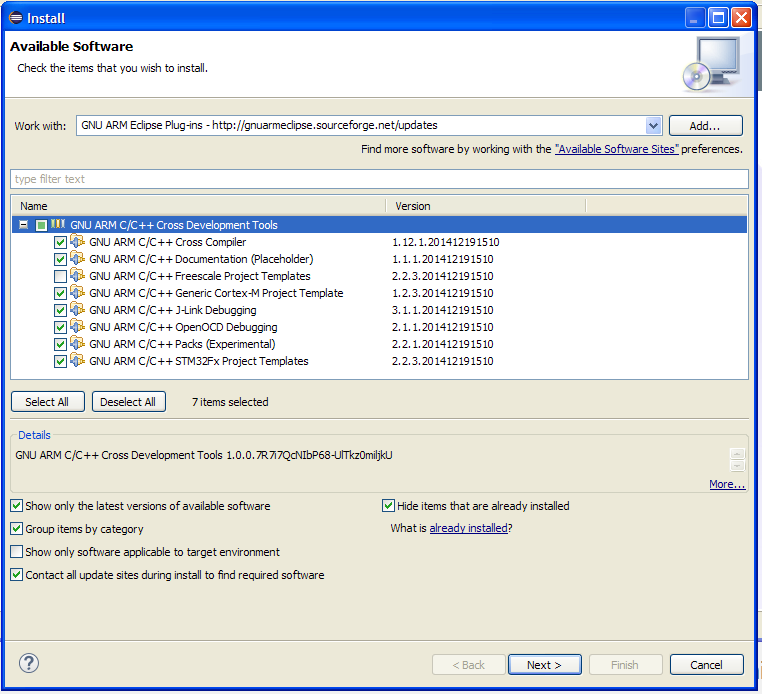
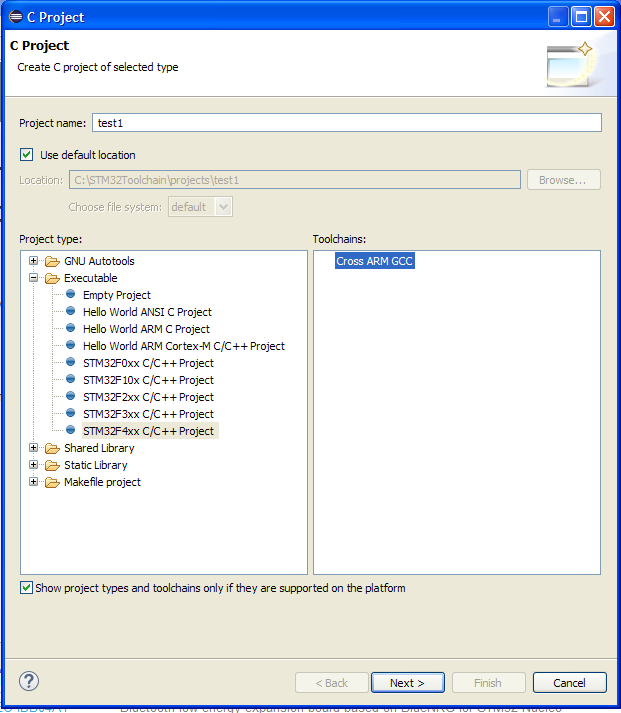
Hello! Email me at [villadn@rose-hulman.edu](mailto:villadn@rose-hulman.edu) if there are any questions or problems with the links

Setting up Eclipse to work with the STM32F446RE

Please name your files/folders better than I did.

1. Make sure your Java is up to data. It should be, but just double check.
2. Download Eclipse for C/C++. This is different than the one you use for CSSE 220/230 and is a separate download. Download the 64-bit version for windows from here: <https://www.eclipse.org/downloads/packages/release/2019-03/r/eclipse-ide-cc-developers>
3. Expand that ZIP into a folder somewhere. I put it in my program files but put it whatever you want
4. Open/run the executable eclipse.exe. It’s in there somewhere. The first time it starts it’s going to ask for a projects folder. I have a folder for all robotics stuff, so either make one or put it somewhere you’ll remember it.
5. Go to Help-> Install New Software, then select the C/C++ Development Tools and check only “CDT Main Feature”. It should look something like this:



1. Click next and go through the rest of the installation. Be patient and restart it when asked.
2. Now to do the plug-ins. In eclipse, go to Help->Install New Software… and click Add… The name is GNU ARM Eclipse Plug-ins and the location is http://gnuarmeclipse.sourceforge.net/updates then click OK
3. Select the plug-ins so that they match the picture’s 
4. Click Next, follow the instructions and restart the IDE when asked
5. Now you need to install the GCC ARM tool-chain. This is very confusing and very inefficient, but it works. Download version 8-2018-q4-major AND the 7-2018-q2-update from <https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads> yes I know this is weird
6. Install them both somewhere you know where it is. For me, they are in my program files (x86) under GNU Tools ARM Embedded and GNU Tools ARM Embedded New respectively. Just put them somewhere when you can find them. Uncheck the box for “Add path to environment variable”
7. For the build tools, we’re going to use MinGW because it doesn’t hate me like some of the other ones do. Download the installer here: <https://sourceforge.net/projects/mingw-w64/>
8. Run the executable and put it a folder somewhere (Mine is C:\MINGW). Open the installer and mark a lot of stuff for installation by clicking on the box and selecting that option. I have the entirety of the MinGW Base System installed as well as the MinGW Libraries -> MinGW Standard folder. Be patient and follow the instructions.
9. Now to install the ST Link Driver. Download the STSW-LINK009 package from <https://www.st.com/en/evaluation-tools/nucleo-f401re.html#tools-software> and expand it. Make sure you know where it is.
10. Download CubeMX. We’ll use this to flash the board. It can be found here: <https://www.st.com/en/development-tools/stm32cubemx.html#overview> you want the STM32CubeMX download. Accept the license agreement, put in your name and rose email but make sure you unsubscribe to their emails since they get annoying. Expand the file and put it somewhere and save an icon to your desktop since this is what we’ll use to flash binaries to the STM32.
11. Now you’re going to create a test project to set up some of the settings. Open eclipse (restart it if it was still open) and go to file->new->C project. Use these configuration options:  call the project whatever you want, uncheck default location and put it somewhere you’ll remember and click next.
12. The chip family for us is STM32F446xE (or something similar), flash size is 512, external clock should be 8000000, content is Blinkey, use freestanding system calls, trace output to ARM ITM, and check the boxes for checking some warnings, using -Og on debug, using newlib nano, exclude unused. The other 3 should not be checked.
13. Click next and use the default parameters until you are asked for a tool-chain. For the name, use GNU Tools for ARM Embedded Processors (arm-none-eabi-gcc) and for the path use <location of 8-2018-q4-major>/8 2018-q4-major/bin and click finish
14. Go to project-> build to build it (Ctrl-B also works). It will fail, that’s fine it’s supposed to. Go to Window->Preferences then C/C++->Build->Environment. Click Add, the name is PATH (make sure it’s capitalized) and the path is <location of MinGW>\msys\1.0\bin;${Path} then click OK, APPLY, APPLY and CLOSE
15. Go to Project->properties then C/C++ Build->Environment. If there is already a variable there named Path, select it, if not, click add. The name should be PATH (capitalized again) and value should be <location of 7 2018-q2-update>\7 2018-q2-updata\bin;<location of MinGW>\msys\1.0\bin;${Path} click OK, APPLY, APPLY and CLOSE
16. Build the project again. This time it should work. Go to the directory this test project is located (this is location you selected in step 16), go into the project folder, go into Debug and look at the files. If there is an elf file there with a recent date modified (the time you built it) then it works. If there is a non-empty hex file that is also good but it’s also ok if it’s completely empty. If something doesn’t work, let me know and I’ll look into it.
17. You will need more libraries to access any of the sensors or get input, but those haven’t been finalized yet