

Avocado Quickstart

Wildcat Robot Design Studio, Winter/Spring 2018

Akash Borde, Tsong Chen, Rachel Hughes, Weilin Ma, Ryan Miller, Nathan Shelly, Allen Tang

# Materials:

###### In your box:

* Avocado
* Power supply (24V, 12.50A) and cables
* Mounting bracket and M4 mounting screws

###### You will need:

* Standard wall power outlet
* Tiva TM4C1294XL microcontroller with LaunchPad by Texas Instruments
* Code Composer Studio software package Texas Instruments
* Jumper cables for serial communication
* Chips and Guacamole (optional)

# Setting up:

###### Wiring:

1. Take the 3-pin Molex communication cable and plug the Molex connector into the 3-pin Molex plug in the Avocado. On the other side, plug the red wire into PC4 of a Tiva TM4C1294XL (or other compatible device) and the green wire into PC5.
2. Plug the Tiva LaunchPad into your computer and open Code Composer Studio.
3. Connect the white power connector into the power supply, and plug the power supply into the wall. If there is a green light on the PCB, you have powered it on successfully! See Figure 1 (below) for what it should look like.

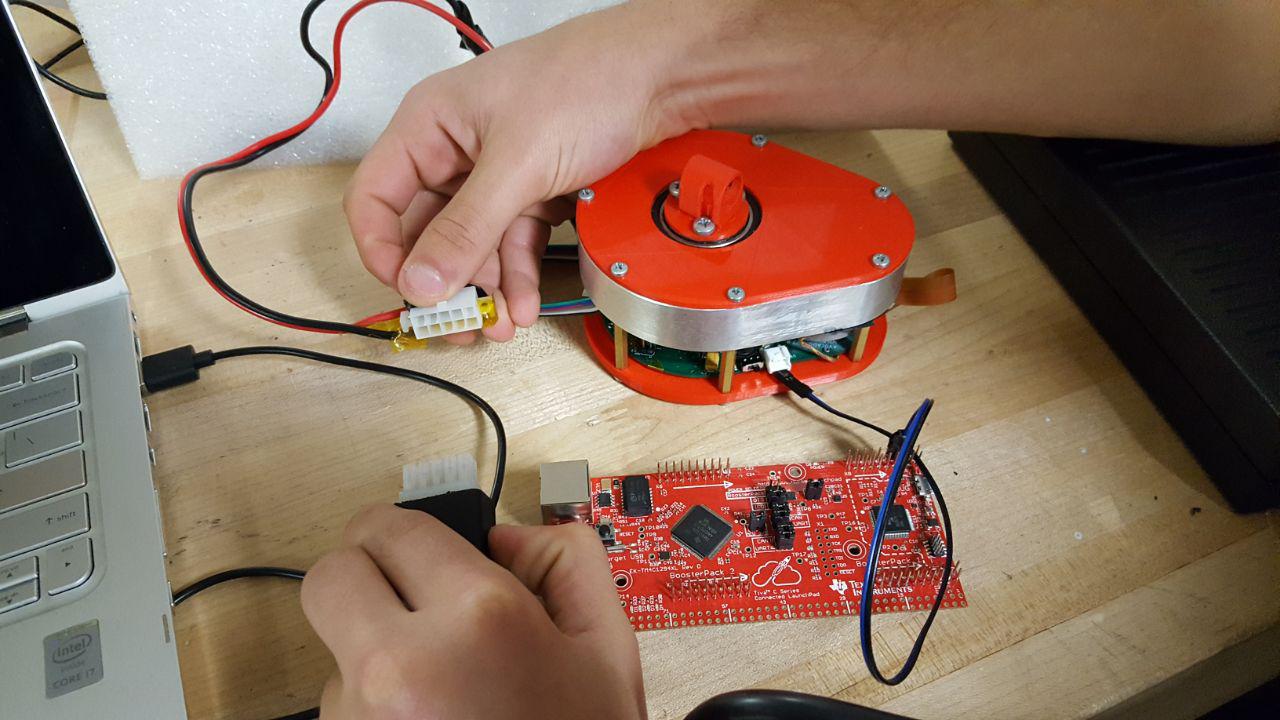
Detailed View:

Tiva Launchpad Pins:

(PC5 | PC4 | GND)

Corresponding Avocado Pins:

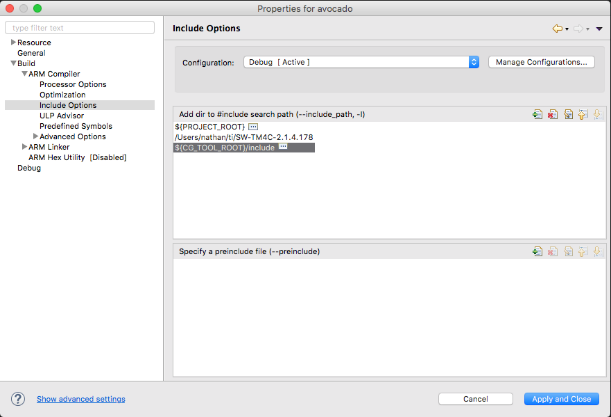
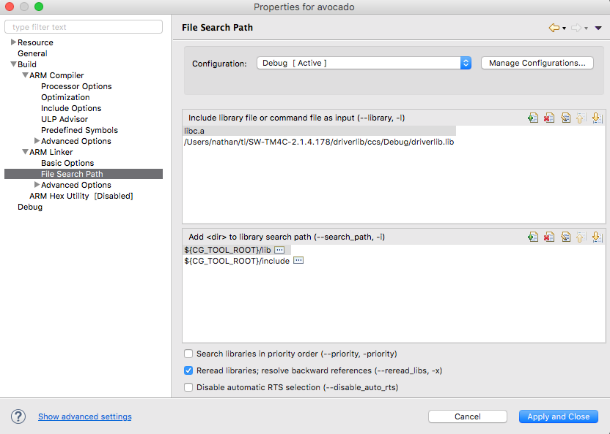
(Data A | Data B | GND)



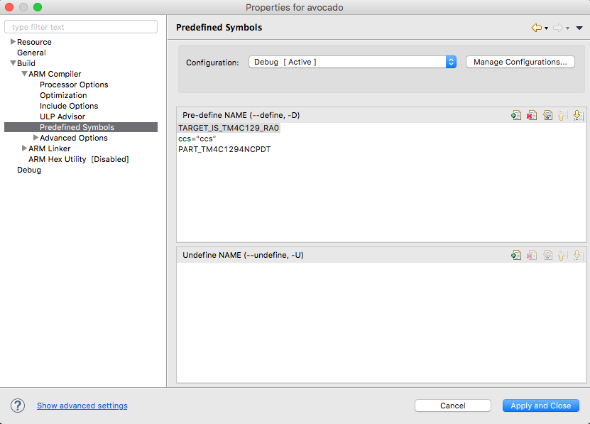
*Figure 1:*

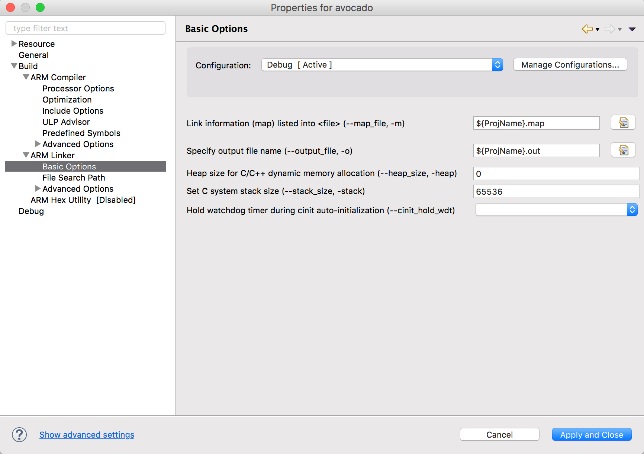
*Wiring your Avocado*

# Configuring Code Composer Studio:

1. Make a new empty CCS project
   * Target: Tiva C Series - Tiva TM4C1294NCPDT
   * In project templates select Empty Project
2. Move all files from your working repo into the project folder (including the hidden .git and .gitignore). Once these files have been copied over you can delete the cloned folder.
3. [](https://raw.githubusercontent.com/Avocado-Actuator/embedded/assets/images/include_options.png)The project folder is now your git repo. Use git commands there like normal (the .gitignore file will ignore CCS configuration files). If you run git status you should not have any unstaged changes (if you see an unstaged targetConfigs folder feel free to delete it)
4. Now we need to set a few CCS project properties:
   * [](https://raw.githubusercontent.com/Avocado-Actuator/embedded/assets/images/file_search_path.png)Under Build -> ARM Compiler -> Include Options in the Add dir to #include search path pane add your TivaWara C Series folder (the folder that contains your examples, driverlib, etc.). For me this folder is ~/ti/SW-TM4C-2.1.4.178 (Note: the folder name may be different in Windows, an example path is C:\ti\TivaWare\_C\_Series-2.1.4.178)

Next under Build -> ARM Linker -> File Search Path in the Include library file or command file as input pane add the full path to your driverlib.lib file. For me this path is ~/ti/SW-TM4C-2.1.4.178/driverlib/ccs/Debug/driverlib.lib, on Windows an example path might be C:\ti\TivaWare\_C\_Series-2.1.4.178\driverlib\ccs\Debug\driverlib.lib.

[](https://raw.githubusercontent.com/Avocado-Actuator/embedded/assets/images/predefined_symbols.png)Penultimately, under Build -> ARM Compiler -> Predefined Symbols in the Pre-define NAME pane add TARGET\_IS\_TM4C129\_RA0.

Finally, under Build -> ARM Linker -> Basic Options set the Set C system stack size option to 65536 . [](https://raw.githubusercontent.com/Avocado-Actuator/embedded/assets/images/stack_size.png)

Check out our GitHub repository at <https://github.com/Avocado-Actuator> for in-depth resources about running your Avocado!

# Running Your First Program:

The c\_library repo holds code for a master controller communicating with avocado actuators. The [export](https://github.com/avocado-actuator/c_library/tree/export) branch contains the library in a pair of header and source files - avocomms.c & avocomms.h. Currently to use this library one should copy these two files to the root of their CCS project (see the [setup guide](https://github.com/Avocado-Actuator/embedded/blob/master/setup.md) in the embedded repo for project setup steps).

If you'd prefer to write raw messages yourself feel free to examine the protocol [here](https://github.com/Avocado-Actuator/embedded/blob/master/protocol.md).

###### Minimal Usage

Copy over avocomms.c & avocomms.h, then update your ...startup\_ccs.c to enable the interrupts needed by our library. If you have no other interrupts feel free to copy over tm4c1294ncpdt\_startup\_ccs.c to your project, otherwise you'll specifically want to add external declarations for UARTIntHandler, ConsoleIntHandler & Timer0IntHandler (see lines 59-61 of our ...startup\_ccs.c file for a reference). Then make sure to add these interrupts in the appropriate locations in the vector table (again see ...startup\_ccs.c for guidance here).

You should now be able to communicate with your avocado! The first thing you'll want to do is set its address with the function setAddress. Once you've done that you can begin setting position, velocity, current & more.

Here is a minimal main function that moves the avocado to an angle of 60°.

int main(void) {

// set clocking to run directly from the crystal at 120MHz

g\_ui32SysClock = MAP\_SysCtlClockFreqSet((SYSCTL\_XTAL\_25MHZ | SYSCTL\_OSC\_MAIN

| SYSCTL\_USE\_PLL | SYSCTL\_CFG\_VCO\_480), 120000000);

ROM\_IntMasterEnable(); // enable processor interrupts

// initialize communication across UART

CommsInit(g\_ui32SysClock);

// initialize timers for heartbeats

TimerInit(g\_ui32SysClock);

// optional, only if you want debugging readouts

ConsoleInit(); // initialize UART0 for debugging output using UARTStdio

// set avocado's address to 1

setAddress(1);

// rotate avocado with address 1 (what we just set) to 60°

rotateToPosition(1, 60);

}

# 

