**Quick guide to the Robot Code**

Prerequisites:

* Some kind of access to SSH, this can be through Command Prompt, Mac Terminal, Visual Studio Code, or a Linux Virtual Machine/Terminal
* git, not github, just git
* Build tools for C++ and/or python
* Basic understanding of Linux and the Linux terminal, e.g. Bash

Preface:

It is recommended to build the code while connected to the Robots while connected via SSH, the Robots should already have almost all of the build tools required to build the software stack. Your first step should be to connect the Robots to a Wifi network and find it’s IP address, this can be achieved multiple ways, easiest way might be to just plug a monitor into the HDMI output of the robots and reading the IP address from the Network settings.

If you want to install the entire setup elsewhere, you will need to follow one of the countless

Once you have the IP address of the Robot you are working with, you can then attempt to SSH into the robot from a different computer on the same Wifi network. Do this with whatever SSH client you want. I will use the command line option, this will work with most ssh clients as well.

IP Address of robot: 128.208.244.208

username of robot: nano

❯ ssh nano@128.208.244.208

The authenticity of host '128.208.244.208 (128.208.244.208)' can't be established.

ECDSA key fingerprint is SHA256:J5yw7Rc+YBhNPmUanYGSuILCBjTkEEU5h/lDrmC7aB4.

Are you sure you want to continue connecting (yes/no/[fingerprint])?

If this is the first time you are logging into the robot, it will probably complain that the authenticity can’t be established, I typically say type in yes at this step, it will typically only ask you this once or when network settings change drastically.

If you connection is a success, it will ask for a password, simply type in the password and the following will be the output:

❯ ssh nano@128.208.244.208

Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.9.140-tegra aarch64)

\* Documentation: https://help.ubuntu.com

\* Management: https://landscape.canonical.com

\* Support: https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are

not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

310 packages can be updated.

79 of these updates are security updates.

To see these additional updates run: apt list --upgradable

Last login: Wed Apr 7 16:02:01 2021 from 128.208.244.170

Next step will be to download the code into a folder using git and build it using the ROS build systems.

The software for the robots can be found in the following git respository for this project:

<https://github.com/CAMP-Project/CAMP.git>

To download this onto the robots you can use the following command:

❯ git clone https://github.com/CAMP-Project/CAMP.git

This command will output the following:

❯ git clone https://github.com/CAMP-Project/CAMP.git

Cloning into 'CAMP'...

remote: Enumerating objects: 3677, done.

remote: Counting objects: 100% (993/993), done.

remote: Compressing objects: 100% (603/603), done.

remote: Total 3677 (delta 637), reused 670 (delta 328), pack-reused 2684

Receiving objects: 100% (3677/3677), 43.51 MiB | 10.70 MiB/s, done.

Resolving deltas: 100% (2157/2157), done.

This will create a folder called CAMP, to list the contents of the current directory, you can use the following command to confirm that the CAMP folder was created:

❯ ls

CAMP

Next we will build the code inside of this folder

Change directory into the CAMP folder:

❯ cd CAMP

Before initiating the build, we need to initialize the git sub modules, which can be done with the following command:

❯ git submodule update --init

Submodule 'src/multimaster\_fkie' (https://github.com/CAMP-Project/multimaster\_fkie.git) registered for path 'src/multimaster\_fkie'

Cloning into '/home/manpreet/testdir/CAMP/src/multimaster\_fkie'...

Submodule path 'src/multimaster\_fkie': checked out 'b10e41b1abd4d2c4bd8f097d63246e596908a2d8'

And then if you have everything installed correctly, you can simply type in the following command to initiate the build:

❯ catkin\_make

Base path: /home/manpreet/testdir/CAMP

Source space: /home/manpreet/testdir/CAMP/src

Build space: /home/manpreet/testdir/CAMP/build

Devel space: /home/manpreet/testdir/CAMP/devel

Install space: /home/manpreet/testdir/CAMP/install

Removing symlink "/home/manpreet/testdir/CAMP/src/CMakeLists.txt" which points to non-existing file

Creating symlink "/home/manpreet/testdir/CAMP/src/CMakeLists.txt" pointing to "/opt/ros/noetic/share/catkin/cmake/toplevel.cmake"

####

#### Running command: "cmake /home/manpreet/testdir/CAMP/src -DCATKIN\_DEVEL\_PREFIX=/home/manpreet/testdir/CAMP/devel -DCMAKE\_INSTALL\_PREFIX=/home/manpreet/testdir/CAMP/install -G Unix Makefiles" in "/home/manpreet/testdir/CAMP/build"

####

-- The C compiler identification is GNU 9.3.0

-- The CXX compiler identification is GNU 9.3.0

-- Check for working C compiler: /usr/bin/cc

-- Check for working C compiler: /usr/bin/cc -- works

-- Detecting C compiler ABI info

-- Detecting C compiler ABI info - done

-- Detecting C compile features

-- Detecting C compile features - done

-- Check for working CXX compiler: /usr/bin/c++

-- Check for working CXX compiler: /usr/bin/c++ -- works

-- Detecting CXX compiler ABI info

-- Detecting CXX compiler ABI info - done

-- Detecting CXX compile features

-- Detecting CXX compile features - done

-- Using CATKIN\_DEVEL\_PREFIX: /home/manpreet/testdir/CAMP/devel

-- Using CMAKE\_PREFIX\_PATH: /opt/ros/noetic

-- This workspace overlays: /opt/ros/noetic

-- Found PythonInterp: /usr/bin/python3 (found suitable version "3.8.5", minimum required is "3")

-- Using PYTHON\_EXECUTABLE: /usr/bin/python3

-- Using Debian Python package layout

-- Found PY\_em: /usr/lib/python3/dist-packages/em.py

-- Using empy: /usr/lib/python3/dist-packages/em.py

-- Using CATKIN\_ENABLE\_TESTING: ON

-- Call enable\_testing()

-- Using CATKIN\_TEST\_RESULTS\_DIR: /home/manpreet/testdir/CAMP/build/test\_results

-- Forcing gtest/gmock from source, though one was otherwise available.

-- Found gtest sources under '/usr/src/googletest': gtests will be built

-- Found gmock sources under '/usr/src/googletest': gmock will be built

-- Found PythonInterp: /usr/bin/python3 (found version "3.8.5")

-- Found Threads: TRUE

-- Using Python nosetests: /usr/bin/nosetests3

-- catkin 0.8.10

-- BUILD\_SHARED\_LIBS is on

-- BUILD\_SHARED\_LIBS is on

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

-- ~~ traversing 21 packages in topological order:

-- ~~ - camp\_pathfinding

-- ~~ - turtlebot3 (metapackage)

-- ~~ - turtlebot3\_msgs

-- ~~ - turtlebot3\_navigation

-- ~~ - coop\_test

-- ~~ - kalman\_filter

-- ~~ - localizer\_dwm1001

-- ~~ - ddynamic\_reconfigure

-- ~~ - brian

-- ~~ - hls\_lfcd\_lds\_driver

-- ~~ - lidar\_listener

-- ~~ - roomba\_navigation

-- ~~ - camp\_goto

-- ~~ - camp\_map

-- ~~ - camp\_multiagent

-- ~~ - turtlebot3\_bringup

-- ~~ - turtlebot3\_example

-- ~~ - turtlebot3\_gazebo

-- ~~ - turtlebot3\_slam

-- ~~ - turtlebot3\_teleop

-- ~~ - turtlebot3\_description

-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

-- +++ processing catkin package: 'camp\_pathfinding'

-- ==> add\_subdirectory(camp\_pathfinding)

-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy

-- camp\_pathfinding: 1 messages, 0 services

-- +++ processing catkin metapackage: 'turtlebot3'

-- ==> add\_subdirectory(turtlebot3/turtlebot3)

-- +++ processing catkin package: 'turtlebot3\_msgs'

-- ==> add\_subdirectory(turtlebot3\_msgs)

-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy

-- turtlebot3\_msgs: 3 messages, 0 services

-- +++ processing catkin package: 'turtlebot3\_navigation'

-- ==> add\_subdirectory(turtlebot3/turtlebot3\_navigation)

-- +++ processing catkin package: 'coop\_test'

-- ==> add\_subdirectory(coop\_test)

-- +++ processing catkin package: 'kalman\_filter'

-- ==> add\_subdirectory(kalman\_filter)

-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy

-- +++ processing catkin package: 'localizer\_dwm1001'

-- ==> add\_subdirectory(dwm1001\_ros)

-- Using these message generators: gencpp;geneus;genlisp;gennodejs;genpy

-- localizer\_dwm1001: 2 messages, 5 services

-- +++ processing catkin package: 'ddynamic\_reconfigure'

-- ==> add\_subdirectory(ddynamic\_reconfigure)

-- +++ processing catkin package: 'brian'

-- ==> add\_subdirectory(brian)

.

.

. . . and so on

I cannot possibly anticipate every single error you might run into while typing to build the code. But if you are working with the same Linux image on the robots that we had used, all of these steps should work, and some of them may have already been done for you. If you do end up running into build error, you will need to figure out what is broken and fix it yourself, can’t do much from the past, sorry.

To launch the robot code, you have multiple options to do so, you can use the Bash script we’ve put together, or what might be better for the long run, learn how to launch ROS Nodes using Lunch files. Most of the basic stuff for launching things can be learned from both the ROS Textbook and the Turtlebot 3 guides found here:

Turtlebot 3 Manual: <https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/>

ROS Textbook: <http://wiki.ros.org/Books/ROS_Robot_Programming_English>