Setting up MOOS Development Environment

Version 0.7

06/14/2021

Outline

Ubuntu 20.04

1. Installing and Building MOOS system from source

-Downloading the moos-ivp and building

```
svn co https://oceanai.mit.edu/svn/moos-ivp-aro/trunk/ moos-ivp
cd moos-ivp
svn update
sudo apt-get install  g++  cmake  xterm
sudo apt-get install  libfltk1.3-dev  freeglut3-dev  libpng-dev  libjpeg-dev
sudo apt-get install  libxft-dev  libxinerama-dev   libtiff5-dev
./build-moos.sh
(let it build)
./build-ivp.sh
(let it build)
```

-Modifying the bash file to run the script within MOOS

```
cd ~
vim .bashrc
export PATH=”$PATH:/home/moos/moos-ivp/bin”
export MOOSROOT=/home/moos/moos-ivp
source .bashrc
cd $MOOSROOT/ivp/missions/s1_alpha
pAntler alpha.moos
```

- If the “pAntler alpha.moos” can run, and the map with a vehicle shows up, the MOOS has been installed successfully

2. Downloading the moos-ivp-extend tree for our own application

```
svn co https://oceanai.mit.edu/svn/moos-ivp-extend/trunk moos-ivp-extend
cd moos-ivp-extend
./build.sh
export PATH="$PATH:/home/moos/moos-ivp-extend/bin"
```

3. Creating your own MOOSapp in moos-ivp-extend --- An Odometry MOOS App

-In order to use GenMOOSApp_AppCasting , the following path need to be included

```
export PATH="$PATH:/home/moos/moos-ivp/scripts"
GenMOOSApp_AppCasting  Odometry p "Jane Doe"
```

-Add Your New Application to the Build System

```
vim moos-ivp-extend/src/CMakeLists.txt
```

```
#============================================================================
# List the subdirectories to build...
#============================================================================
ADD_SUBDIRECTORY(lib_behaviors-test)
ADD_SUBDIRECTORY(pXRelayTest)
```

```
ADD_SUBDIRECTORY(pExampleApp)
ADD_SUBDIRECTORY(pOdometry)                    <-- Add this line
```

-Build the new created application

```
cd moos-ivp-extend
./build.sh
```

-Verify the pOdometry is in your shell path with:

```
which pOdometry
/home/you/moos-ivp-you/bin/pOdometry
```

--Write the application

```
cd moos-ivp-extend/src/pOdometry
vim Odometry.h (see the figure "Odometry.h")
vim Odometry.cpp (see the code "Odometry.cpp")
```

Odometry.h

```
/************************************************************/
/*    NAME: Shuai Dong                                      */
/*    ORGN: MIT                                             */
/*    FILE: Odometry.h                                      */
/*    DATE:                                                 */
/************************************************************/

#ifndef Odometry_HEADER
#define Odometry_HEADER

#include "MOOS/libMOOS/MOOSLib.h"

class Odometry : public CMOOSApp
{
 public:
   Odometry();
   ~Odometry();

 protected: // Standard MOOSApp functions to overload
   bool OnNewMail(MOOSMSG_LIST &NewMail);
   bool Iterate();
   bool OnConnectToServer();
   bool OnStartUp();
   bool  m_first_reading;
   double m_current_x;
   double m_current_y;
   double m_previous_x;
   double m_previous_y;
   double m_total_distance;
 protected:
   void RegisterVariables();

 private: // Configuration variables

 private: // State variables
};

#endif
```

Odometry.cpp

```
#include <iterator>
#include "MBUtils.h"
#include "Odometry.h"

using namespace std;

//---------------------------------------------------------
// Constructor
```

```
Odometry::Odometry()
{
  m_first_reading= 1;
  m_current_x = 0 ;
  m_current_y = 0;
  m_previous_x = 0;
  m_previous_y = 0;
  m_total_distance = 0;
}
// Destructor


Odometry::~Odometry()
{
}

//---------------------------------------------------------
// Procedure: OnNewMail

bool Odometry::OnNewMail(MOOSMSG_LIST &NewMail)
{
  MOOSMSG_LIST::iterator p;

  for(p=NewMail.begin(); p!=NewMail.end(); p++) {
    CMOOSMsg &msg = *p;

#if 0 // Keep these around just for template
    string key   = msg.GetKey();
    string comm  = msg.GetCommunity();
    double dval  = msg.GetDouble();
    string sval  = msg.GetString();
    string msrc  = msg.GetSource();
    double mtime = msg.GetTime();
    bool   mdbl  = msg.IsDouble();
    bool   mstr  = msg.IsString();
#endif

    string key = msg.GetKey();
    if(key == "NAV_X")
    {
         m_previous_x = m_current_x;
         m_current_x = msg.GetDouble();
    }

    if(key == "NAV_Y")
        {
        m_previous_y = m_current_y;
        m_current_y = msg.GetDouble();
        }
  }
   return(true);

}

//---------------------------------------------------------
// Procedure: OnConnectToServer
```

```
bool Odometry::OnConnectToServer()
{
   RegisterVariables();
   return(true);
}
// Procedure: Iterate()
//            happens AppTick times per second

bool Odometry::Iterate()
{
  if(m_first_reading == 1)
  {
    m_total_distance =sqrt(m_current_x*m_current_x+m_current_y*m_current_y);
    Notify( "ODOMETRY_DIST",m_total_distance);
    m_first_reading=0;
  }
  else
  {
    m_total_distance =sqrt((m_current_x-m_previous_x)*(m_current_x-
m_previous_x)+(m_current_y-m_previous_y)*(m_current_y-m_previous_y))+m_total_distance;
    Notify( "ODOMETRY_DIST",m_total_distance);
  }
  return(true);
}

//---------------------------------------------------------
// Procedure: OnStartUp()
//            happens before connection is open

bool Odometry::OnStartUp()
{
  list<string> sParams;
  m_MissionReader.EnableVerbatimQuoting(false);
  if(m_MissionReader.GetConfiguration(GetAppName(), sParams)) {
    list<string>::iterator p;
    for(p=sParams.begin(); p!=sParams.end(); p++) {
      string line  = *p;
      string param = tolower(biteStringX(line, '='));
      string value = line;

      if(param == "foo") {
        //handled
      }
      else if(param == "bar") {
        //handled
      }
    }
  }

  RegisterVariables();
  return(true);
}
// Procedure: RegisterVariables
void Odometry::RegisterVariables()
{
    Register("NAV_X", 0);
```

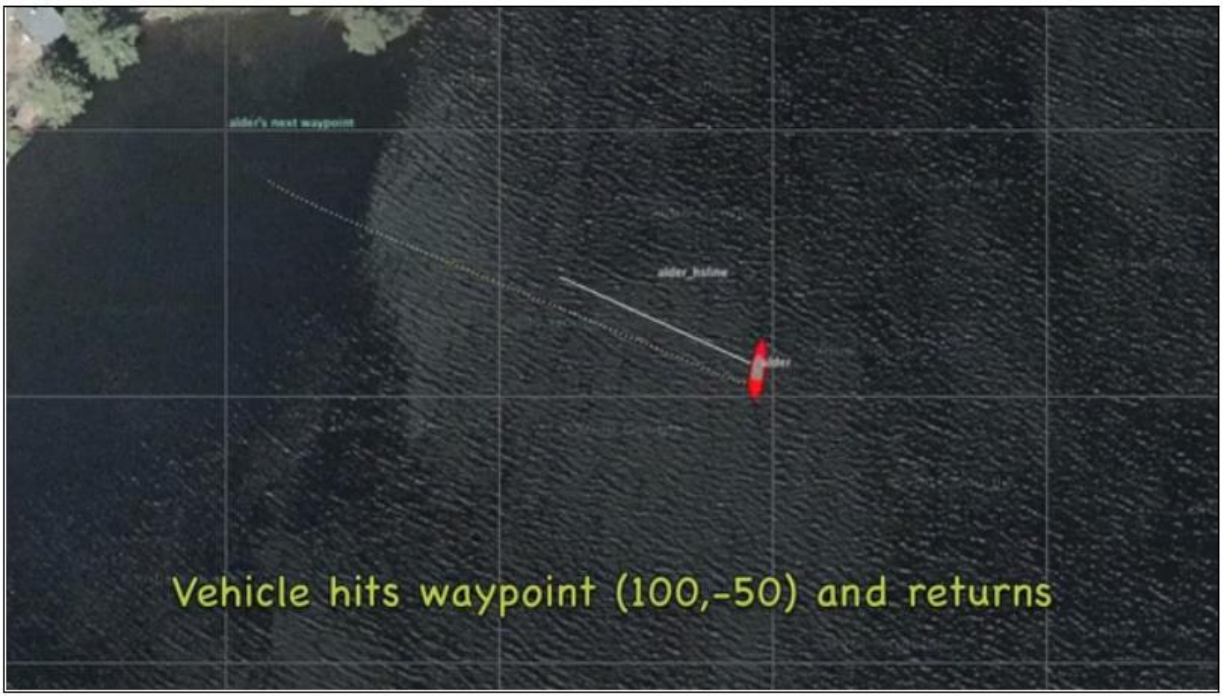
```
    Register("NAV_Y", 0);
}
```

--Every time you modify the code, you need to run the "./build.sh" under the "moos-ivp-extend/"

-Testing the new created pOdometry app in the Alder misstion

--Run the un-modified Alder Mission

```
cd moos-ivp-extend/missions/alder
pAntler --MOOSTimeWarp=10 alder.moos
```



--Run the modified Alder Mission

--Modify the "alder.moos" file

--Add the "Run = pOdometry @ NewConsole = true"

```
// Antler configuration  block
ProcessConfig = ANTLER
{
  MSBetweenLaunches = 200

  Run = MOOSDB          @ NewConsole = false
  Run = uSimMarine      @ NewConsole = false
  Run = pNodeReporter   @ NewConsole = false
  Run = pMarinePID      @ NewConsole = false
  Run = pMarineViewer   @ NewConsole = false
  Run = uProcessWatch    @ NewConsole = false
  Run = pHelmIvP        @ NewConsole = false
  Run = pOdometry       @ NewConsole = true

}
```

--Add pOdometry config block (setting the running frequency)

```
// ******pOdometry config block********

ProcessConfig = pOdometryPublisher
{
  AppTick    = 10
  CommsTick = 10

}
```

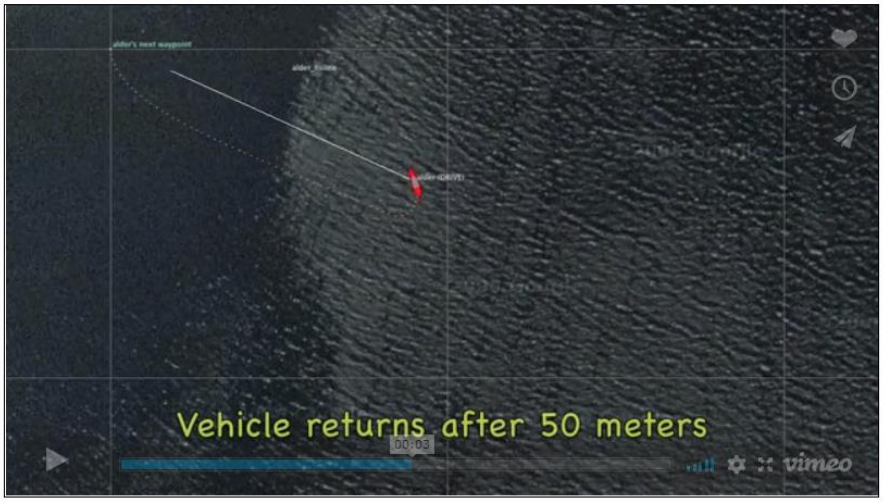
--Modify the alder.bhv file (configurations on Helm)

--Add two conditions into alder.bhv file

```
condition = (ODOMETRY_DIST < 50)

condition = (RETURN = true) or (ODOMETRY_DIST >= 50)
```

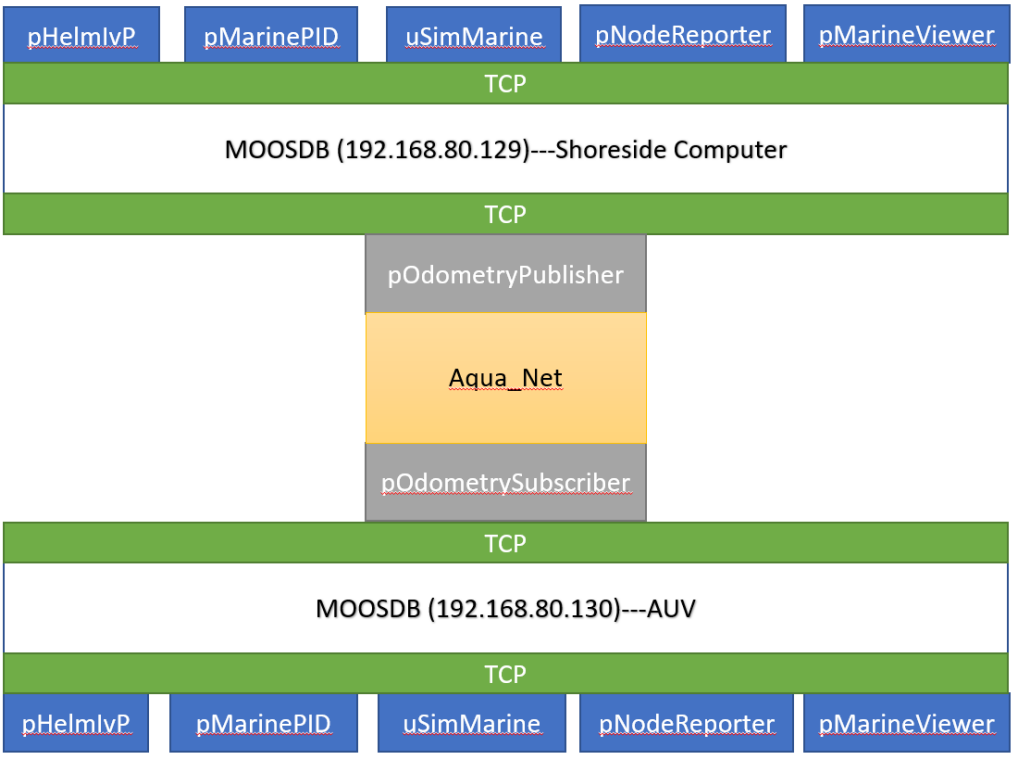
```
//--------    FILE: alder.bhv    -------------

initialize   DEPLOY = false
initialize   RETURN = false

//----------------------------------------------
Behavior = BHV_SimpleWaypoint
{
  name      = waypt_to_point
  pwt       = 100
  condition = RETURN = false
  condition = DEPLOY = true

  condition = (ODOMETRY_DIST < 50)  //add this line

  endflag   = RETURN = true

  speed      = 2.0   // meters per second
  radius     = 8.0
  ptx        = 100
  pty        = -50
}

//----------------------------------------------
Behavior = BHV_Waypoint
{
  name       = waypt_return
  pwt        = 100
  condition = (RETURN = true) or (ODOMETRY_DIST >= 50) //add this line
  condition  = (DEPLOY = true)

  speed      = 2.0
  radius     = 8.0
  point      = 0,0
}
```

--Rebuild the code and run the mission again



4. Integrating AquaNet code into MOOS

In this section, we will talk about how to integrate AquaNet into MOOS communication system. The interface is shown as the following figure.



In order to use the AquaNet socket interface inside MOOS, the following steps should be made: (Two virtual machines, each machine should install the MOOS according to the instructions mentioned above)

-Get and compile the latest aquanet code:

Install subversion:

```
sudo apt-get install subversion
```

-Get the code

```
svn co svn+ssh://dmitrii@hudson.ccny.cuny.edu/var/svn/repos/aquanet aquanet
export AQUANET_FOLDER=/home/ubuntu/aquanet
```

-Install dependencies

```
sudo apt-get install libglib2.0-dev
sudo apt-get install libgsl-dev
```

-Build AquaNet

```
cd $AQUANET_FOLDER/trunk
make
```

Now, the AquaNet is built and downloaded successfully in a separate folder. The step should be executed on both machines.

4.1 Building pOdometry-Publisher on machine A (192.168.80.129)

-Create the pOdometryPublisher moos application

```
GenMOOSApp_AppCasting  OdometryPublisher p "Jane Doe"
```

-Modify the CMakeList

```
vim moos-ivp-extend/src/CMakeLists.txt
```

```
  ADD_SUBDIRECTORY(lib_behaviors-test)
  ADD_SUBDIRECTORY(pXRelayTest)
  ADD_SUBDIRECTORY(pExampleApp)
  ADD_SUBDIRECTORY(pOdometryPublisher)                <-- Add this line
```

-Build the new created application

```
cd moos-ivp-extend
./build.sh
```

-Verify the pOdometryPublisher is in your shell path with:

```
which pOdometryPublisher
/home/you/moos-ivp-you/bin/pOdometryPublisher
```

--Write the application

```
cd moos-ivp-extend/src/pOdometryPublisher
vim OdometryPublisher.h (see the figure "OdometryPublisher.h")
vim OdometryPublisher.cpp (see the code "OdometryPublisher.cpp")
```

--pOdometryPublisher.h could be found at:

```
https://github.com/shuaidong-networking/AquaNet-
MOOS/blob/main/pOdometryPublisher/pOdometryPublisher.h
```

-- pOdometryPublisher.cc could be found at:

```
https://github.com/shuaidong-networking/AquaNet-
MOOS/blob/main/pOdometryPublisher/pOdometryPublisher.cc
```

-In order to use AquaNet API, some header files from aquanet should be link to the pOdometryPublisher folder.

-Header files

```
ln -s $AQUANET_FOLDER/trunk/aquanet_log.h  moos-ivp-extend/src/pOdometryPublisher/aquanet_log.h

ln -s $AQUANET_FOLDER/trunk/aquanet_netif.h  moos-ivp-extend/src/pOdometryPublisher/aquanet_netif.h

ln -s $AQUANET_FOLDER/trunk/aquanet_pdu.h  moos-ivp-extend/src/pOdometryPublisher/aquanet_pdu.h

ln -s $AQUANET_FOLDER/trunk/aquanet_socket.h  moos-ivp-extend/src/pOdometryPublisher/aquanet_socket.h

ln -s $AQUANET_FOLDER/trunk/aquanet_time.h  moos-ivp-extend/src/pOdometryPublisher/aquanet_time.h
```

-Configuration files

```
ln -s $AQUANET_FOLDER/trunk/test_example/mesh/node1/config_add.cfg  moos-ivp-
extend/src/pOdometryPublisher/config_add.cfg

ln -s $AQUANET_FOLDER/trunk/test_example/mesh/node1/config_arp.cfg  moos-ivp-
extend/src/pOdometryPublisher/config_arp.cfg

ln -s $AQUANET_FOLDER/trunk/test_example/mesh/node1/config_conn.cfg  moos-ivp-
extend/src/pOdometryPublisher/config_conn.cfg
```

```
ln -s $AQUANET_FOLDER/trunk/test_example/mesh/node1/config_net.cfg  moos-ivp-extend/src/pOdometryPublisher/config_net.cfg
```

-Compile the new created application

```
cd moos-ivp-extend
./build.sh
```

-Copy alder.moos file to pOdometryPublisher folder and modify it

```
cd moos-ivp-extend/missions/alder
vim alder.moos
```

-Code for alder.moos files could be found at

```
https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometryPublisher/alder.moos
```

4.2 Building pOdometry-Subscriber on machine B (192.168.80.130)

-Create the pOdometrySubscriber moos application

```
GenMOOSApp_AppCasting  pOdometrySubscriber p "Jane Doe"
```

-Modify the CMakeList

```
vim moos-ivp-extend/src/CMakeLists.txt
```

```
  # List the subdirectories to build...
  #============================================================================
  ADD_SUBDIRECTORY(lib_behaviors-test)
  ADD_SUBDIRECTORY(pXRelayTest)
  ADD_SUBDIRECTORY(pExampleApp)
  ADD_SUBDIRECTORY(pOdometrySubscriber)                  <-- Add this line
```

-Building the new created application

```
cd moos-ivp-extend
./build.sh
```

-Verify the pOdometryPublisher is in your shell path with:

```
which pOdometryPublisher
/home/you/moos-ivp-you/bin/pOdometrySubscriber
```

--Write the application

```
cd moos-ivp-extend/src/pOdometrySubscriber
vim pOdometrySubscriber.h (see the figure “pOdometrySubscriber.h”)
vim pOdometrySubscriber.cpp (see the code “pOdometrySubscriber.cpp”)
```

--Code for pOdometrySubscriber.h can be found at:

```
https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometrySubscriber/pOdometrySubscriber.h
```

--Code for pOdometrySubscriber.cc can be found at:

```
https://github.com/shuaidong-networking/AquaNet-MOOS/blob/main/pOdometrySubscriber/pOdometrySubscriber.cc
```

-In order to use AquaNet API, some header files from aquanet shoulde be link to the pOdometryPublisher folder.

-Header files

```
ln -s $AQUANET_FOLDER/trunk/aquanet_log.h  moos-ivp-extend/src/pOdometrySubscriber/aquanet_log.h

ln -s $AQUANET_FOLDER/trunk/aquanet_netif.h  moos-ivp-extend/src/ pOdometrySubscriber /aquanet_netif.h

ln -s $AQUANET_FOLDER/trunk/aquanet_pdu.h  moos-ivp-extend/src/ pOdometrySubscriber /aquanet_pdu.h

ln -s $AQUANET_FOLDER/trunk/aquanet_socket.h  moos-ivp-extend/src/ pOdometrySubscriber /aquanet_socket.h

ln -s $AQUANET_FOLDER/trunk/aquanet_time.h  moos-ivp-extend/src/ pOdometrySubscriber /aquanet_time.h
```

-Configuration files

```
ln -s $AQUANET_FOLDER/trunk/test_example/mesh/node2/config_add.cfg  moos-ivp-extend/src/ pOdometrySubscriber
/config_add.cfg

ln -s $AQUANET_FOLDER/trunk/test_example/mesh/ node2/config_arp.cfg  moos-ivp-extend/src/ pOdometrySubscriber
/config_arp.cfg

ln -s $AQUANET_FOLDER/trunk/test_example/mesh/ node2/config_conn.cfg  moos-ivp-extend/src/ pOdometrySubscriber
/config_conn.cfg

ln -s $AQUANET_FOLDER/trunk/test_example/mesh/ node2/config_net.cfg  moos-ivp-extend/src/ pOdometrySubscriber
/config_net.cfg
```

-Compile the new created application

```
cd moos-ivp-extend
./build.sh
```

-Moving the alder.moos file to pOdometrySubscriber floder and modify

```
cd moos-ivp-extend/missions/alder
vim alder.bhv
vim alder.moos
```

-Alder.bhv

```
https://github.com/shuaidong-networking/AquaNet-
MOOS/blob/main/pOdometrySubscriber/alder.bhv
```

-Alder.moos

```
https://github.com/shuaidong-networking/AquaNet-
MOOS/blob/main/pOdometrySubscriber/alder.moos
```

5. Running AquaNet with MOOS

5.1 Running aquanet on the machine A

-Running AquaNet Virtual Modem Server to emulate L1

```
cd $AQUANET_FOLDER/trunk
./bin/aquanet-vmds 2021
```

-Create a bash-script inside the pOdometryPublisher folder to start the layers from L2-L4:

```
cd moos-ivp-extend/src/pOdometryPublisher
touch publisher.sh
```

-Insert the following code into publisher.sh file

```
#!/bin/sh
# Initialize L2-L4 modules on localhost

# start the protocol stack

$AQUANET_FOLDER/trunk/bin/aquanet-stack &

sleep 2
# start the VMDM
# modify IP and Port number, if necessary
$AQUANET_FOLDER/trunk/bin/aquanet-vmdc 192.168.80.129 2021 1 20 20 20 &

sleep 4

# start the MAC
$AQUANET_FOLDER/trunk/bin/aquanet-uwaloha &
sleep 2
$AQUANET_FOLDER/trunk/bin/aquanet-sroute &
sleep 2
# start the transport layer
$AQUANET_FOLDER/trunk/bin/aquanet-tra &
```

-Run the script

```
chmod +x publisher.sh
./publisher.sh
```

5.2 Running aquanet on the machine B

-Create a bash-script inside the pOdometrySubscriber folder to start the layers from L2-L4:

```
cd moos-ivp-extend/src/pOdometrySubscriber
touch subscriber.sh
```

-Insert the following code into subscriber.sh file

```
#!/bin/sh
# Initialize L2-L4 modules on localhost

# start the protocol stack

$AQUANET_FOLDER/trunk/bin/aquanet-stack &

sleep 2
# start the VMDM
# modify IP and Port number, if necessary
$AQUANET_FOLDER/trunk/bin/aquanet-vmdc 192.168.80.129 2021 2 20 20 20 &

sleep 4

# start the MAC
$AQUANET_FOLDER/trunk/bin/aquanet-uwaloha &
sleep 2
$AQUANET_FOLDER/trunk/bin/aquanet-sroute &
sleep 2
# start the transport layer
$AQUANET_FOLDER/trunk/bin/aquanet-tra &
```

-Run the script

```
chmod +x subscriber.sh
./ subscriber.sh
```

5.3 Runing publisher on machine A

-Start the pOdometryPublisher

```
cd moos-ivp-extend/src/pOdometryPublisher
pAntler alder.moos
```

5.4 Running subscriber on machine B

-Start the pOdometrySubscriber

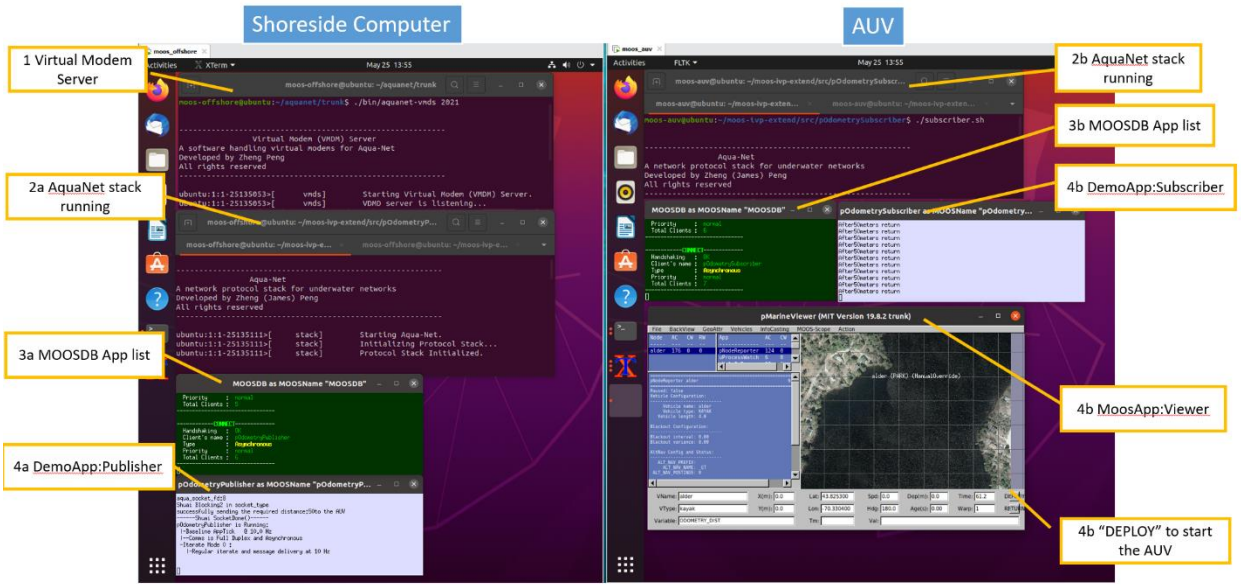
```
cd moos-ivp-extend/src/ pOdometrySubscriber
pAntler alder.moos
```

5.5 Running results is showing as the following figure

When all the programs have been executed, several terminals will be shown as the following figure. The execution order of the program is:

1. Running virtual modem server on shoreside pc side
2. Running AquaNet stack on both shoreside pc and AUV side by using the script mentioned earlier
3. Running the alder.moos file on both ends separately
4. MOOSDB and our demo will be started

Demo Setup



-Click the DEPLOY button and you will see the AUV change the next waypoint at 50 meters instead of 100 meters.

6. pShare based Inter-vehicle communication in MOOS (Two Machines)

In this step, we create two separate MOOS communities: *a shoreside* community on machine A and *an alpha* community on machine B by creating two separate .moos file. In the shoreside community there will be a MOOSDB and pMarineViewer. In the alpha community will be a MOOSDB and everything but pMarineViewer. In both community you will also need to add a pShare configuration block and add pShare to the Antler configuration block.

Machine A IP address: 192.168.80.130

Machine B IP address: 192.168.80.129

6.1 Machine A (shoreside computer) side

-Create a folder for Alpha Mission's MOOS file

```
cd moos-ivp-extend/missions/
mkdir alpha
```

-Make a copy of the Alpha Mission into the MOOS-IVP-EXTEND

```
cd ~
cp -rp moos-ivp/ivp/missions/s1_alpha  moos-ivp-extend/missions/alpha/
```

-Replace the name of alpha.moos file with shoreside.moos

```
cd moos-ivp-extend/missions/alpha/s1_alpha
mv alpha.moos shoreside.moos
```

-Modify the shoreside.moos file

```
vim shoreside.moos
```

-- copy the following content to overwrite the original content in shoreside.moos

```
ServerHost = localhost
ServerPort = 9005
Community  = shoreside

MOOSTimeWarp = 20
TERM_REPORTING = true
LatOrigin  = 43.825300
LongOrigin = -70.330400

// Antler configuration  block
ProcessConfig = ANTLER
{
  MSBetweenLaunches = 200

  Run = MOOSDB            @ NewConsole = true
  Run = pMarineViewer     @ NewConsole = false
  Run = pShare           @ NewConsole = true //adding this line
}
//------------------------------------------
// adding pShare config block by Shuai

ProcessConfig = pShare //pShare configuration
```

```
{
AppTick  = 4
CommsTick = 4
input = route = localhost:9201
output = src_name=MOOS_MANUAL_OVERRIDE, route=192.168.80.129:9200
output = src_name=DEPLOY, route=192.168.80.129:9200 //sending status of RETURN and
DEPLOY and MOOS_MANUAL_OVERRIDE
output = src_name=RETURN, route=192.168.80.129:9200
}

//------------------------------------------
// pMarineViewer config block

ProcessConfig = pMarineViewer
{
  AppTick    = 4
  CommsTick  = 4
  tiff_file            = forrest19.tif
  set_pan_x            = -90
  set_pan_y            = -280
  zoom                 = 0.65
  vehicle_shape_scale  = 1.5
  hash_delta           = 50
  hash_shade           = 0.22
  hash_viewable        = true
  trails_point_size   = 1

  // Appcast configuration
  appcast_height       = 75
  appcast_width        = 30
  appcast_viewable     = true
  appcast_color_scheme = indigo
  nodes_font_size      = xlarge
  procs_font_size      = xlarge
  appcast_font_size    = large
  right_context[return] = DEPLOY=true
  right_context[return] = MOOS_MANUAL_OVERRIDE=false
  right_context[return] = RETURN=false
  scope  = RETURN
  scope  = WPT_STAT
  scope  = VIEW_SEGLIST

  scope  = VIEW_POINT
  scope  = VIEW_POLYGON
  scope  = MVIEWER_LCLICK
  scope  = MVIEWER_RCLICK

  button_one = DEPLOY # DEPLOY=true
  button_one = MOOS_MANUAL_OVERRIDE=false # RETURN=false
  button_two = RETURN # RETURN=true
  button_three = SLOWER # WPT_UPDATE=speed=1.5
  button_four = FASTER # WPT_UPDATE=speed=3.5

  action  = MENU_KEY=deploy # DEPLOY = true # RETURN = false
  action+ = MENU_KEY=deploy # MOOS_MANUAL_OVERRIDE=false
  action  = RETURN=true
  action  = UPDATES_RETURN=speed=1.4
```

```
}
```

6.2 Machine B (vehicle) side

-Do the same operations as machine A side except the step of modifying the content of alpha.moos file.

-Copy the following content to overwrite the original one

```
ServerHost = localhost
ServerPort = 9005
Community  = alpha

MOOSTimeWarp = 20
TERM_REPORTING = true

// Forest Lake
LatOrigin  = 43.825300
LongOrigin = -70.330400

// MIT Sailing Pavilion (use this one)
// LatOrigin  = 42.358456

// LongOrigin = -71.087589

//------------------------------------------
// Antler configuration  block
ProcessConfig = ANTLER
{
  MSBetweenLaunches = 200
  Run = MOOSDB            @ NewConsole = false
  Run = pLogger         @ NewConsole = false
  Run = uSimMarine        @ NewConsole = false
  Run = pMarinePID        @ NewConsole = false
  Run = pHelmIvP @ NewConsole = false
  Run = uProcessWatch     @ NewConsole = false
  Run = pNodeReporter     @ NewConsole = false
  Run = pShare   @ NewConsole = false  //add pShare configuration
}
//------------------------------------------

// adding pShare config block by Shuai

ProcessConfig = pShare

{
AppTick  = 4
CommsTick = 4
input = route = localhost:9200
output = src_name=NODE_REPORT_LOCAL, dest_name=NODE_REPORT, route=192.168.80.130:9201
output = src_name=VIEW_SEGLIST, route=192.168.80.130:9201
output = src_name=VIEW_POINT, route=192.168.80.130:9201
}


//------------------------------------------uSimMarine
// pLogger config block
```

```
ProcessConfig = pLogger
{
  AppTick   = 8
  CommsTick = 8

  AsyncLog = true

  // For variables that are published in a bundle on their first post,
  // explicitly declare their logging request
  Log = IVPHELM_LIFE_EVENT @ 0 NOSYNC
  Log = REPORT @ 0 NOSYNC
  Log = BHV_SETTINGS @ 0 NOSYNC

  LogAuxSrc = true

  WildCardLogging = true
  WildCardOmitPattern = *_STATUS
  WildCardOmitPattern = DB_VARSUMMARY
  WildCardOmitPattern = DB_RWSUMMARY
  WildCardExclusionLog = true
}

//--------------------------------------
// uProcessWatch

ProcessConfig = uProcessWatch
{
  AppTick   = 4
  CommsTick = 4

  watch_all = true
        nowatch   = uPokeDB*
        nowatch   = uXMS*
        nowatch   = uMAC*
}

//--------------------------------------
// uSimMarine config block
ProcessConfig = uSimMarine
{
  AppTick        = 4
  CommsTick      = 4
  start_pos = x=0, y=-20, heading=180, speed=0
  prefix         = NAV

  turn_rate     = 40
  thrust_map    = 0:0, 20:1, 40:2, 60:3, 80:4, 100:5
  thrust_reflect = true
}

//--------------------------------------
// pHelmIvP config block

ProcessConfig = pHelmIvP
{
  AppTick    = 4
  CommsTick  = 4
```

```
  behaviors  = alpha.bhv
  domain     = course:0:359:360
  domain     = speed:0:4:41
}

//--------------------------------------------
// pMarinePID config block

ProcessConfig = pMarinePID
{
  AppTick    = 20
  CommsTick  = 20

  verbose       = true
  depth_control = false

  // SIM_INSTABILITY = 20

  // Yaw PID controller
  yaw_pid_kp              = 1.2
  yaw_pid_kd              = 0.0
  yaw_pid_ki              = 0.3
  yaw_pid_integral_limit = 0.07

  // Speed PID controller
  speed_pid_kp            = 1.0
  speed_pid_kd            = 0.0
  speed_pid_ki            = 0.0
  speed_pid_integral_limit = 0.07

  //MAXIMUMS
  maxrudder    = 100
  maxthrust    = 100

  // A non-zero SPEED_FACTOR overrides use of SPEED_PID
  // Will set DESIRED_THRUST = DESIRED_SPEED * SPEED_FACTOR
  speed_factor = 20
}

//--------------------------------------------
// pMarineViewer config block

ProcessConfig = pMarineViewer
{
  AppTick    = 4
  CommsTick  = 4

  tiff_file            = forrest19.tif

  set_pan_x            = -90
  set_pan_y            = -280
  zoom                 = 0.65
  vehicle_shape_scale  = 1.5
  hash_delta           = 50
  hash_shade           = 0.22
  hash_viewable        = true
```

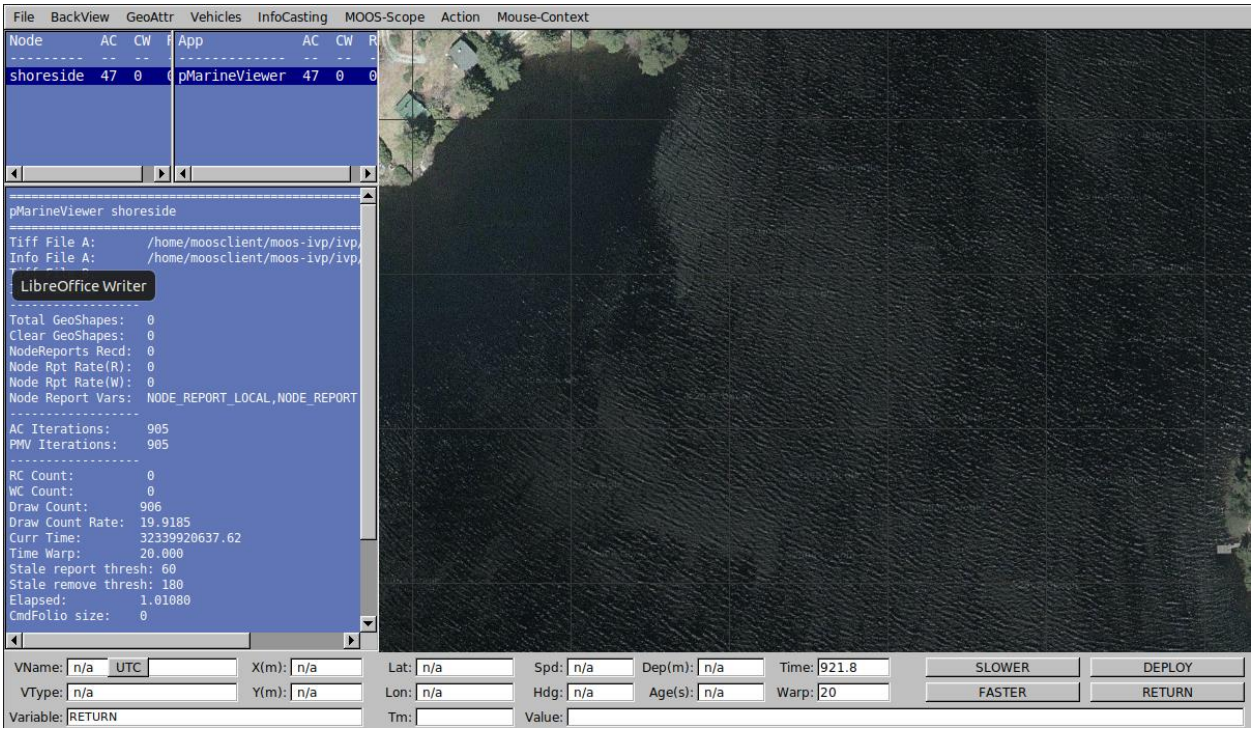
```
  trails_point_size   = 1

  // Appcast configuration
  appcast_height       = 75
  appcast_width        = 30
  appcast_viewable     = true
  appcast_color_scheme = indigo
  nodes_font_size      = xlarge
  procs_font_size      = xlarge
  appcast_font_size    = large

  right_context[return] = DEPLOY=true
  right_context[return] = MOOS_MANUAL_OVERRIDE=false
  right_context[return] = RETURN=false

  scope  = RETURN
  scope  = WPT_STAT
  scope  = VIEW_SEGLIST
  scope  = VIEW_POINT
  scope  = VIEW_POLYGON
  scope  = MVIEWER_LCLICK
  scope  = MVIEWER_RCLICK

  button_one = DEPLOY # DEPLOY=true
  button_one = MOOS_MANUAL_OVERRIDE=false # RETURN=false
  button_two = RETURN # RETURN=true
  button_three = SLOWER # WPT_UPDATE=speed=1.5
  button_four = FASTER # WPT_UPDATE=speed=3.5

  action  = MENU_KEY=deploy # DEPLOY = true # RETURN = false
  action+ = MENU_KEY=deploy # MOOS_MANUAL_OVERRIDE=false
  action  = RETURN=true
  action  = UPDATES_RETURN=speed=1.4
}

//------------------------------------------

// pNodeReporter config block

ProcessConfig = pNodeReporter
{
  AppTick    = 2
  CommsTick       = 2

  platform_type   = kayak
  platform_color  = yellow
  platform_length = 4
}
```

6.3 Test communication between two moos community

-Launch shoreside.moos file on machine A

```
cd moos-ivp-extend/missions/alpha/s1_alpha
pAntler shoreside.moos
```

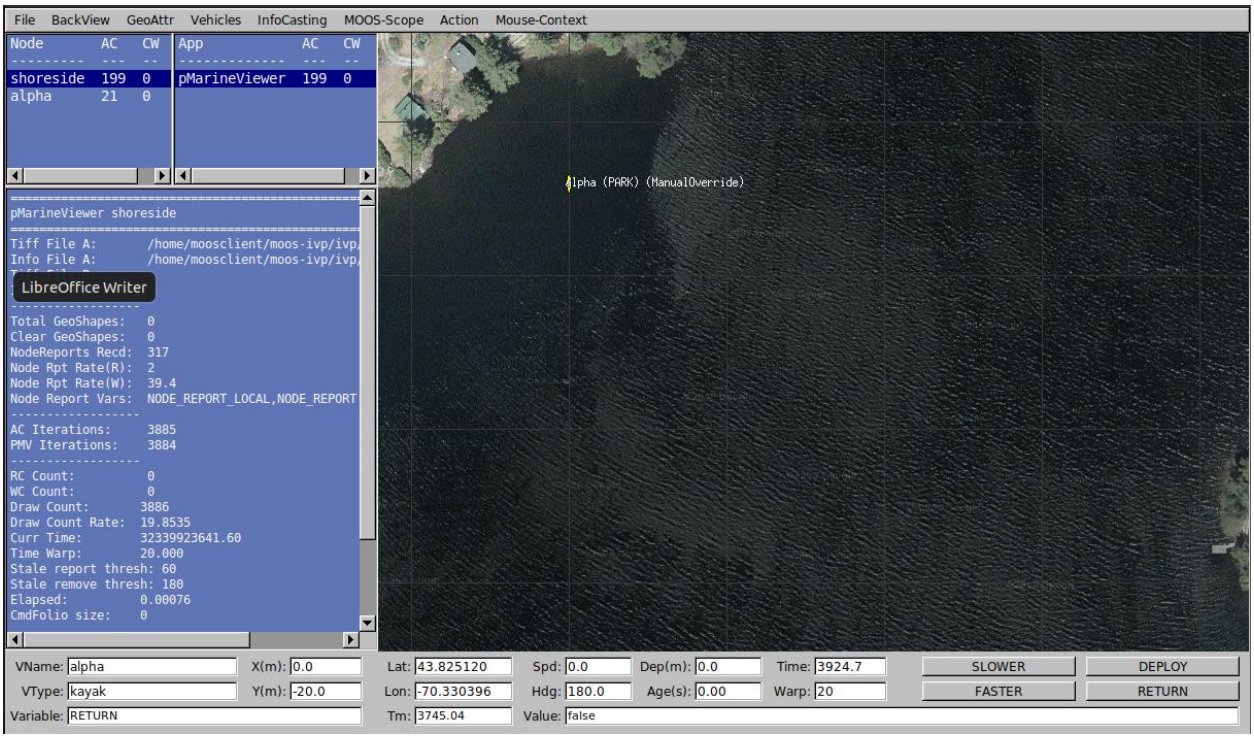
-After the shoreside.moos file is executed, you will see the following figure



-Launch the alpha.moos file on machine B

```
cd moos-ivp-extend/missions/alpha/s1_alpha
pAnIter alpha.moos
```

-After the alpha.moos is executed, a vehicle will be shown on machine A's map.



-Click the “DEPLOY” button, you will see that the vehicle is moving

7. Creating our own share application---AquaShare

-Creating an application under “moos-ivp-extend/src/”

```
cd moos-ivp-extend/src
GenMOOSApp_AppCasting  AquaShare p "Author_Name"
```

-Make the pAquaShare application executable

-Modify the CMakeList

```
vim moos-ivp-extend/src/CMakeLists.txt
```

```
  ADD_SUBDIRECTORY(lib_behaviors-test)
  ADD_SUBDIRECTORY(pXRelayTest)
  ADD_SUBDIRECTORY(pExampleApp)
  ADD_SUBDIRECTORY(pOdometry)
   ADD_SUBDIRECTORY(pAquaShare) //add this line
```

- Delete all files in pAquaShare folder

```
cd pAquaShare
rm -f *
```

-Copy all the files in pShare into pAquashare

```
cp -rf /home/moos/moos-ivp/MOOS_Dec3120/MOOSEssentials/Essentials/pShare/* /home/moos/moos-ivp-extend/src/pAquaShare/
```

-Modify the CMakeLists.txt in pAquaShare. Changing the line2 and line7 as the following figure

```
mv pShareMain.cpp pAquaShare.cpp
vim CMakeLists.txt
```

```
1  #this builds the pScheduler application
2  set(EXECNAME pAquaShare)
3
4  find_package(MOOS 10.0)
5
6  #what files are needed?
7  SET(SRCS   Share.cpp Listener.cpp Route.cpp ShareHelp.cpp pAquaShareMain.cpp)
8
9  include_directories( ${${EXECNAME}_INCLUDE_DIRS} ${MOOS_INCLUDE_DIRS} ${MOOS_DEPEND_INCLUDE_DIRS})
10 add_executable(${EXECNAME} ${SRCS} )
11 target_link_libraries(${EXECNAME} ${MOOS_LIBRARIES} ${MOOS_DEPEND_LIBRARIES})
12
13 INSTALL(TARGETS ${EXECNAME}
14   RUNTIME DESTINATION bin
15 )
16
17 #copy resources
18 #file(COPY ${CMAKE_CURRENT_SOURCE_DIR}/pshare_test_scripts DESTINATION ${CMAKE_RUNTIME_OUTPUT_DIRECTO
19
20
21 #add_custom_command(TARGET pShare POST_BUILD
22 ##                   COMMAND ${CMAKE_COMMAND} -E copy_directory
23 #                       ${CMAKE_SOURCE_DIR}/pshare_test_scripts ${CMAKE_RUNTIME_OUTPUT_DIRECTORY})
```

-Replace the pShare with the new created pAquaShare on machine A and machine B

-For machine A side: adding “run = pAquaShare @ NewConsole = true” in the ANTLER block, and add a new block for pAquaShare as the following figure

```
21
22 //------------------------------------------
23 // Antler configuration  block
24 ProcessConfig = ANTLER
25 {
26   MSBetweenLaunches = 200
27
28   Run = MOOSDB          @ NewConsole = true
29   Run = pLogger          @ NewConsole = false
30   Run = uSimMarine      @ NewConsole = false
31   Run = pMarinePID      @ NewConsole = false
32   Run = pHelmIvP        @ NewConsole = false
33   Run = uProcessWatch   @ NewConsole = false
34   Run = pNodeReporter   @ NewConsole = false
35   //Run = pShare               @ NewConsole = true    //---------add pShare application--internet
36   Run = pAquaShare           @ NewConsole = true    //---------add pShare application
37 }
38 //------------------------------------------
39 // adding pShare config block
40 /*
41 ProcessConfig = pShare
42 {
43 AppTick   = 4
44 CommsTick = 4
45 input = route = 192.168.22.139:9200
46 output = src_name=NODE_REPORT_LOCAL, dest_name=NODE_REPORT, route=localhost:9201
47 output = src_name=VIEW_SEGLIST, route=192.168.22.142:9201
48 output = src_name=VIEW_POINT, route=192.168.22.142:9201
49 }
50 */
51
52 //Create our own share application
53 ProcessConfig = pAquaShare
54 {
55 AppTick   = 4
56 CommsTick = 4
57 input = route = localhost:9200
58 output = src_name=NODE_REPORT_LOCAL, dest_name=NODE_REPORT, route=192.168.22.142:9201
59 output = src_name=VIEW_SEGLIST, route=192.168.22.142:9201
60 output = src_name=VIEW_POINT, route=192.168.22.142:9201
61 }
```

-For machine B, the same procedure.

Until now, the new created compliable pAquaShare application used for message exchange between two MOOSDB has been run successfully based on internet socket.

Next step will replace the internet socket with the AquaNET socket