FRC Sofware Installation Notes

Dell Inspiron 7559 Laptop

System Details

Hardware

1. Dell Inspiron 15” laptop (7559)
   - $793 from Amazon
   - NVIDIA GTX 960M Graphics card (4 GB RAM)
   - 8 GB RAM
   - Intel I5 Processor
   - 256 GB Internal SSD Drive
2. Samsung 256 GB SSD second internal drive
   - $96 from Amazon
   - Installation (easy):
     - Remove single screw and open back cover
     - Insert drive into empty slot and connect cable
     - close back cover and reattach screw

Software

1. Windows 10 (pre-installed on internal SSD drive)
2. UEFI (Unified Extensible Firmware Interface) BIOS Installed

Windows and Linux Dual Boot

1. Create Linux Installation media
   - On a Linux computer, Download Ubuntu 16.04 “.iso” image
     - www.ubuntu.com/download/desktop
   - Transfer image file to a 4GB micro SSD card
     - Determine device name of SSD card

            - sudo fdisk -l
                - record initial /dev ids (e.g. /dev/sda1,/dev/sda2 ..)
            - Insert SSD card into USB card reader and connect to a USB port
            - sudo fdisk -l
                - note any new partitions (e.g. /dev/sdg1 ..)
                - record the partition name (sans any numbers) e.g. “sdg”
        - Use dd to transfer downloaded image to card
            - sudo dd -i <path-to-image> -o /dev/<dev-name-of-card>
            - e.g. sudo dd -i ~/Downloads/ubuntu-16.04.1-desktop-amd64.iso -o /dev/sdg

2. Configure Windows
    - Disable “Fast boot”
        - Control Panel→ Power Options
3. Configure BIOS
    - Boot to BIOS (press f2 when booting)
    - Tab to Boot Options
    - Disable “Secure boot”
    - Enable “Legacy boot options”
    - keep UEFI boot mode
    - Save and Exit (f10)
4. Install Ubuntu on second 256 GB SSD drive
    - Insert SSD with Ubuntu installer image in card reader into USB port of Inspiron
    - reboot laptop
    - press f12 (boot options) at start of boot cycle (be quick)
        - should see a boot device list on a blue background
    - Choose “USB .. ” from list and press return
        - note: sometime not present in list (if so, power cycle and try again)
    - When Ubuntu installer boots should see a (maroon) screen with 2 small icons at the bottom

    - press return to get list of install options
    - <u>important: </u>press “Advanced” options (f6) and disable ACPI (press esc to exit)
        - otherwise Ubuntu installer will hang on startup
    - Choose “Try Ubuntu” from options list
    - When desktop comes up press the “Install Ubuntu Icon”
    - Install Ubuntu on Second drive (sdb)
    - choose sdb for boot partition in pulldown menu (normally defaults to sda)
        - Otherwise installation will fail to complete (no way to exit except power cycle)
    - Set swap size to 0
        - Recommended for SSD drives to prevent write cycle wear
    - Remove installation media
5. Boot to Ubuntu (after running Windows)
    - Reboot laptop
    - Press f12 when booting
    - choose “Second HDD drive” from list (this new entry should now be present)
    - log into Ubuntu (using user and password supplied during installation process)
6. Fix ACPI startup problem in Ubuntu kernel loader
    - Causes the boot sequence to fail (hang) with several messages like “failed to unregister from ACPI system”, “bbwsitch: cannot find ACPI handle for VGA device”
        - The problem started after changing “Additional Drivers” tab in Software & Updates window from “Nouveau” to NVIDIA
    - After the failure the only way to recover was to reboot and press the ‘e’ key at just the right time during the startup cycle (took dozens of tries to stop the boot cycle from hanging)
    - Once in edit mode then needed to remove acpi=off from the kernel launch command line
        - press ctrl-x to continue boot
    - When (If) Ubuntu now boots up do the following to make the change permanent
        - $ sudo gedit /etc/default/grub
        - Add a comment in front of: GRUB_CMDLINE_LINUX="acpi=off"

        - #GRUB_CMDLINE_LINUX="acpi=off"
    - $ sudo update-grub

7. Boot to Windows (after running Ubuntu)
    - Reboot laptop
    - don't press f12 when booting
    - default boot will launch Windows

Ubuntu 16.04

Configure Desktop

1. Install gnome flashback

- Gave "Unity" yet another shot for a while but in the end I still like the older "flashback" interface better
    - I also got annoyed when I found out that the Unity "designers" decided to remove the ability to move window controls to the right in the latest versions of ubuntu (many complaints about this on-line but the response was a smug "take it or leave it" answer)
- To install flashback:
    - $ sudo apt-get update
    - $ sudo apt-get install gnome-session-flashback
    - Log out
    - Log in and press icon on upper right of login window
    - choose "Gnome Flashback Metacity"

Install/build FRC Software (2016)

References

1. https://wpilib.screenstepslive.com/s/4485/m/23353/l/228979-installing-frcsim-manually-ubuntu

Components

1. Java (OpenJDK 1.8.0_91)
    - sudo apt-get install default-jdk
2. Eclipse (Neon)
    - Download tar file from Eclipse download site

        - https://www.eclipse.org/downloads/download.php?file=/oomph/epp/neon/R/eclipse-inst-linux64.tar.gz
    - Install
        - $ tar -xf eclipse-inst-linux64.tar.gz
        - $ cd ~/Downloads/eclipse-installer
        - $ ./eclipse-inst
            - set install path to /opt (previously write-enabled)
        - add to path in ~/.profile : /opt/eclipse/cpp-neon/eclipse
    - install plugins
        - Open Eclipse and install plugins as described in reference 1
            - creates ~/wpilib/cpp
            - note: Prior to mid Oct. 2016 only the Java plugin was installable

3. install Gazebo
    - http://gazebosim.org/tutorials?tut=install_ubuntu
    - latest version (8.0 as of 4/17)
        - $ curl -ssL http://get.gazebosim.org | sh
        - requires Ubuntu 16 or later
    - older versions
        - follow “alternate installation” instructions
        - For Ubuntu 14.04 (install version 7.6)
            - replace: sudo apt-get install gazebo8 with: sudo apt-get install gazebo7
            - replace: sudo apt-get install libgazebo8-dev with: sudo apt-get install libgazebo7-dev

4. check out MentorRepository
    - mkdir Robotics && cd Robotics
    - git clone https://github.com/FRCTeam159/MentorRepository
5. merge patched allwpilib with wpi git repo
    - cd MentorRepository/Software/workspace

- tar -cf allwpilib.tar allwpilib
  - save patches from team’s repo
- rm -fr allwpilib
- git clone https://usfirst.collab.net/gerrit/p/allwpilib.git
  - adds allwpilib directory with upstream wpi git repo
- tar -xf allwpilib.tar
  - restores team patches

6. Build frc_gazebo_plugins
   - patch allwpilib source files
     - $ cd ~/Robotics/MentorRepository/Software/workspace/allwpilib/simulation
     - add #include <boost/algorithm/string.hpp> at line 16 of gz_msgs/msgs.h.in
       - fixed error with “replace_all” not supported by boost
     - replace “boost” with “std” on line 32 of frc_gazebo_plugins/rangefinder/src/rangefinder.cpp
     - replace “boost” with “std” on line 12 of frc_gazebo_plugins/limit_switch/src/external_limit_switch.cpp
   - cd ~/Robotics/MentorRepository/Software/workspace/allwpilib
   - ./gradlew :wpilibc:build -PmakeSim
7. Build libwpilibcSim.so
   - cd ~/Robotics/MentorRepository/Software/workspace/allwpilib
   - $ sudo apt-get install cmake libprotobuf-dev libprotoc-dev protobuf-compiler -y
   - ./gradlew :wpilibc:build -PmakeSim
8. Built libntcore.so (for simulation)
   - linker problem with libntcore.so

   When FRCUserProgram is compiled using the pre-built allwpilib libntcore.so (downloaded from maven repo) the following error is generated in the link step:

```
../build/install/simulation/lib/libwpilibcSim.so: undefined reference to
`nt::Value::MakeStringArray(std::vector<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >,
std::allocator<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > > >&&)'
```

```
../build/install/simulation/lib/libwpilibcSim.so: undefined reference to
`nt::Value::MakeStringArray(llvm::ArrayRef<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > >)'
collect2: error: ld returned 1 exit status
makefile:47: recipe for target 'FRCUserProgram' failed
```

- The origin of the link error appears to come from the following code

  `MakeStringArray` defined in nt_value.h

  ```
  Commands/Scheduler.cpp
  m_table->PutValue("Names", nt::Value::MakeStringArray(commands));
  ```

  ```
  SmartDashboard/SendableChooser.cpp
  nt::Value::MakeStringArray(std::move(keys)));
  ```

- MakeStringArray is clearly included in the maven built and downloaded libntcore.so library (found using “strings”)
- link also fails using the prebuilt library in ~/wpilib/simulation/lib (this looked to have worked once but could not repeat)

9. Install frc-toolchain (arm compiler)
   - Ubuntu 16.04 (xenial) isn’t yet officially supported by frc (11/1/16) but as a workaround a package for an older version (vivid, wiley) can be installed instead
     - sudo apt-add-repository ppa:wpilib/toolchain
     - sudo apt update
     - sudo gedit /etc/apt/sources.list.d/wpilib-ubuntu-toolchain-xenial.list
     - change out where it says 'xenial' (without the apostrophes) and replace it with 'vivid' (also without the apostrophes)
     - sudo apt-get update
   - previous releases require libisl10 which is not included with Ubuntu 16 so must be downloaded and installed manually
     - http://packages.ubuntu.com/trusty/amd64/libisl10/download
     - sudo dpkg -i libisl10_0.12.2-1_amd64.deb
   - Installation

     ```
     sudo apt install frc-toolchain
     ```

10. build libntcore.so from source
    - sudo apt-get install gcc-multilib g++-multilib
    - obtain source from wpi github project

        - git clone https://github.com/wpilibsuite/ntcore
    - cd ntcore
    - ./gradlew build [-PskipArm]

      simulation library built at: ./native/ntcore/build/libs/ntcore/shared/x64/libntcore.so
    - arm library built at: ./arm/ntcore/build/libs/ntcore/shared/libntcore.so

11. Create system wide directories for build headers and libraries
    - Make soft links in /usr/include to mask gazebo and sdformat revision numbers
        - cd /usr/include
        - $ sudo ln -s gazebo-7 gazebo
        - $ sudo ln -s gazebo-7 gazebo-6.5
        - $ sudo ln -s sdformat-4.1 sdformat
        - $ sudo ln -s sdformat-4.1 sdformat-3.7
12. create a system directory for wpi simulation objects
    - sudo mkdir -p /usr/local/wpi/arm
    - cd ~/Robotics/MentorRepository/Software/workspace/allwpilib
    - gradlew build -PmakeSim
    - copy over files from allwpilib simulation build
        - gradlew simulation:zip -PmakeSim
            - creates ./simulation/build/distributions/simulation-trusty.zip
        - sudo unzip ./simulation/build/distributions/simulation-trusty.zip -d /usr/local/wpi/sim
    - copy over gz_msgs header file (msgs.h) from allwpilib build
        - sudo cp -R ./build/simulation/gz_msgs/generated/simulation /usr/local/wpi/sim/include/
    - copy over libntcore.so from ntcore simulation build
    - cd ~/Robotics/MentorRepository/Software/workspace/ntcore
        - sudo cp /native/ntcore/build/libs/ntcore/shared/x64/libntcore.so /usr/local/wpi/sim/lib
13. create a system directory for wpi arm objects
    - sudo mkdir -p /usr/local/wpi/arm

    - cd ~/Robotics/MentorRepository/Software/workspace/allwpilib
    - gradlew build
    - gradlew wpilibcZip
        - creates: ./wpilibc/build/wpilibc.zip
    - copy over files from allwpilib arm build
        - sudo unzip ./wpilibc/build/wpilibc.zip -d /usr/local/wpi/arm
    - copy over libntcore.so from ntcore arm build
        - cd ~/Robotics/MentorRepository/Software/workspace/ntcore
        - sudo cp ./arm/ntcore/build/libs/ntcore/shared/libntcore.so /usr/local/wpi/arm/lib

14. modify ~/wpilib to use global files instead of the files created by the Eclipse frc plugin installation
    - installing the Eclipse frc c++ plugin creates ~/wpilib/simulation and ~/wpilib/cpp/current which have several problems
        - The “release” plugin branch doesn’t support neon and Ubuntu 16
        - The “development” branch works with Neon but has no support for CAN devices (e.g. no CANTalon.h in ~/wpilib/cpp/current/include)
    - modify ~/wpilib as follows:
        - cd ~/wpilib
        - rm -fr simulation
        - ln -s /usr/local/wpi/sim simulation
        - cd cpp/current
        - rm -fr lib include
        - ln -s /usr/local/wpi/arm/lib lib
        - ln -s /usr/local/wpi/arm/include include

Eclipse FRCUserProgram build settings

linux_simulation

1. C++ include paths
    - "${workspace_loc:/${ProjName}}/src"

    - ${WPILIB}/simulation/include
        - linked to: /usr/local/wpi/sim/include
    - /usr/include
    - /usr/include/gazebo-6.5
        - linked to: /usr/include/gazebo-7
    - /usr/include/ignition/math2
    - /usr/include/sdformat-3.7
        - linked to: /usr/include/sdformat-4.1
2. Linker libraries
    - ntcore
    - wpilibcSim
    - gz_msgs
    - gazebo_client
    - boost_system
3. Linker paths
    - /usr/lib/x86_64-linux-gnu
    - ${WPILIB}/simulation/lib
        - linked to /usr/local/wpi/sim/lib
4. Linker misc (-rpath)
    - -rpath ${WPILIB}/simulation/lib
        - linked to /usr/local/wpi/sim/lib

Debug (arm)

1. C++ include paths
    - "${workspace_loc:/${ProjName}/src}"
    - "${WPILIB}/cpp/current/include"
        - linked to: /usr/local/wpi/arm/include
2. Linker libraries
    - wpi

3. Linker paths
    - "${WPILIB}/cpp/current/lib"
        - linked to: /usr/local/wpi/arm/lib

Gazebo Simulation Test

1. Environment
    - `Set soft link for sim_ds`
        - `$ sudo ln -s ~/wpilib/simulation/sim_ds /usr/bin/sim_ds`
    - in ~/.bashrc add:

      export ALLWPIDEV=$HOME/Robotics/MentorRepository/Software/workspace/allwpilib

      export SWEXPORTS=$HOME/Robotics/MentorRepository/Solidworks/Exported

      export GAZEBO_PLUGIN_PATH=/usr/local/lib/frcsim/plugins:/usr/lib/x86_64-linux-gnu/gazebo-7/plugins
2. Model
    - world file: 2016-Robot-field.world
    - robot model:2016-Robot
    - eclipse project: SimVisionTest
    - launch script: gazebo-2016-Robot-field

      ```
      export LD_LIBRARY_PATH=/usr/local/lib/frcsim/lib:/usr/lib:/usr/lib/x86_64-
      linux-gnu/:${GAZEBO_PLUGIN_PATH}

      export SWMODEL=$SWEXPORTS/2016-Robot_Exported

      export GAZEBO_MODEL_PATH=$SWMODEL:
      $WPILIB/simulation/models:/usr/share/gazebo-7/models

      gazebo --verbose $SWMODEL/2016-Robot-field.world
      ```
3. Run simulation
    - Connect gamepad to USB port
    - Open a terminal tab and start gazebo launch script
        - cd ~/Robotics/MentorRepository/Software/workspace/SimVisionTest
        - ./gazebo-2016-Robot-field
    - open a second tab and start up sim_ds
        - sim_ds
    - open a third tab and start FRCUserProgram

        - cd linux_simulate
        - ./FRCUserProgram
    - Press “Teleop Enable” in sim_ds and use the gamepad to drive the robot around the field
        - Buttons X and B raise and lower the shooter (or loader depending on the mode)
        - Button Y fires (or loads) the boulder
        - Button A toggles between “shoot” and “load” modes

4. Fix problem with “Reset Model Poses”

   In Gazebo 6.5 pressing Edit→Reset Model Poses restored the positions of all component models to their original places (first captured by “save world” in an element called “state”)

   In 7.3 it looks like “state” is only read when the world is first loaded and reset poses (or world) sets the components to the positions they were in when they were first dragged into the scene

    - A solution/workaround was to copy the “pose” line for each component in the state element and paste it immediately before the first link in the corresponding model element in the main body of the world file
    - when this has been done with all components the state element can be removed from the world file

FRC Software (2017)

References

1. https://wpilib.screenstepslive.com/s/4485

Build and Install Components

Allwpilib

1. Obtain source clone (new) git repo

   git clone https://github.com/wpilibsuite/allwpilib

2. build

   > gradlew :wpilibc:build -PmakeSim

3. Fix build bugs (compile errors)

- wpilibc/sim/include/Counter.h
  undefined symbol “Mode”: in Counter.h (Mode mode = kTwoPulse not defined)

```
@@ -30,7 +30,7 @@ class Counter : public SensorBase,
                public CounterBase,
                public LiveWindowSendable {
```

```
  public:
-  explicit Counter(Mode mode = kTwoPulse);
+  //explicit Counter(Mode mode = kTwoPulse);
```

4. fix simulation bugs (still broken in 2017 source code)

- Notifier.cpp

```
@@ -151,6 +151,9 @@ void Notifier::InsertInQueue(bool reschedule) {
     if ((*i)->m_expirationTime > m_expirationTime) {
       timerQueue.insert(i, this);
       m_queued = true;
+                        // BUG 1: wpi version doesn't break here so it keeps inserting in front of all elements
+                        // with expiration times > current element
+                        break;
     }
   }

@@ -158,7 +161,8 @@ void Notifier::InsertInQueue(bool reschedule) {
    * element was greater than the new entry.
    */
   if (!m_queued) {
-    timerQueue.push_front(this);
+    // BUG 2: wpi version uses "push_front" which is wrong since it adds the longest time to the front
+                timerQueue.push_back(this);
```

- wpilibc/simulation/src/PIDController.cpp

```
@@ -83,6 +83,8 @@ PIDController::PIDController(double Kp, double Ki, double Kd, double Kf,
   m_controlLoop = std::make_unique<Notifier>(&PIDController::Calculate, this);
   m_controlLoop->StartPeriodic(m_period);
+  m_setpointTimer.Start();
+
   static int instances = 0;
   instances++;
@@ -172,6 +174,17 @@ void PIDController::Calculate() {
     }
     pidOutput->PIDWrite(result);
+
+                // BUG: 2016 WPI code does not do this
+            // Update the buffer.
+
+            m_buf.push(m_error);
+            m_bufTotal += m_error;
+            // Remove old elements when buffer is full.
+            if (m_buf.size() > m_bufLength) {
+              m_bufTotal -= m_buf.front();
+              m_buf.pop();
+            }
   }
```

```
 }

@@ -193,7 +206,7 @@ void PIDController::Calculate() {
 double PIDController::CalculateFeedForward() {
   if (m_pidInput->GetPIDSourceType() == PIDSourceType::kRate) {
     return m_F * GetSetpoint();
-  } else {
+  } else if( m_F>0) {
     double temp = m_F * GetDeltaSetpoint();
     m_prevSetpoint = m_setpoint;
     m_setpointTimer.Reset();
+} else
+    return 0;
@@ -507,8 +520,11 @@ void PIDController::SetToleranceBuffer(int bufLength) {
 bool PIDController::OnTarget() const {
   std::lock_guard<priority_recursive_mutex> sync(m_mutex);
-  if (m_buf.size() == 0) return false;
-  double error = GetError();
+       double error;
+       if (m_buf.size() == 0)
+               error = GetError();
+       else
+               error = GetAvgError();
```

- wpilibc/sim/src/simulation/SimContinuousOutput.cpp

```
@@ -16,7 +16,8 @@ SimContinuousOutput::SimContinuousOutput(std::string topic) {
   std::cout << "Initialized ~/simulator/" + topic << std::endl;
 }
-void SimContinuousOutput::Set(double speed) {
+void SimContinuousOutput::Set(double _speed) {
+  speed=_speed;
```

GearsBot Test

Build errors

1. SetDistanceToBoxPIDOutput and DriveStraight
- problem: unimplemented function in pure abstract class PIDOutput
- fix: replace PIDWrite(float) with PIDWrite(double)
2. Link error (nt)
- fix: rebuilt ntcore fom latest source

Run Test

1. Obtain and install GearsBot models as described in following reference https://wpilib.screenstepslive.com/s/4485/m/23353/l/228979-installing-frcsim-manually-ubuntu
2. Run Gearsbot simulation

- > frcsim ~/wpilib/simulation/worlds/GearsBotDemo.world
- > sim_ds
- > cd ../linux_simulate
- >FRCUserProgram
- Simulation works (can drive robot around with gamepad in teleop mode)

CANtalon Simulation Support

1. As of 2017 wpilib no longer directly supports CAN devices

Windows 10

Install FRC Software