

## **APPENDIX**

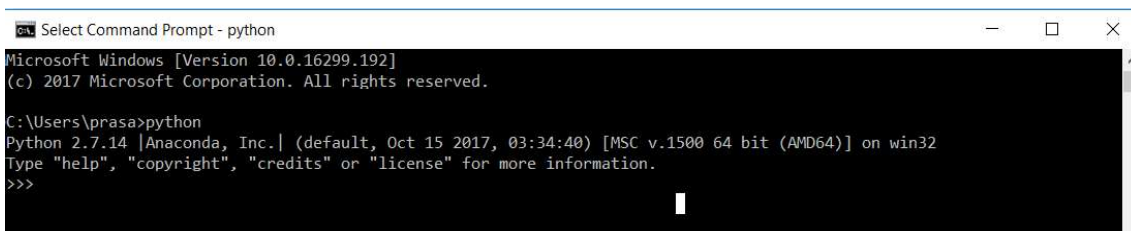
## APPENDIX A – HOW TO WORK WITH RADAR

The initial configuration for the RADAR as per the datasheet is as follows and be sure to verify the same in the Device Manager and the main program since communication with the Radar may fail if they are not correct.

- ✓ *Bits per second: 115200*
- ✓ *Data bits: 8*
- ✓ *Parity: None*
- ✓ *Stop bits: 1*
- ✓ *Flow control: None* [37]

To work with Radar, primarily we require *python 2.7* and all the next steps required to get the distance from radar are as provided below:

1. Go to <https://www.python.org/downloads/> and click on **Download Python 2.7.14** button.
2. Open the downloaded file and install Python 2.7 by following the on-screen wizard prompt.
3. Open command prompt by typing **cmd** in the Start application
4. Once the installation is done, to verify that it was installed correctly, on command line execute the command ***python --version***
5. Next, type ***python*** and press enter. You should see a window like the one shown in the screenshot below.



6. Depending on the version of python installed, you might have to manually install *pip* as it is already installed if you're using *python 2 >=2.7.9* or *python 3 >=3.4* binaries downloaded from *python.org* [36]
7. Next, check if *pip* is installed by typing ***pip*** into the terminal window. Some text should be shown about *pip* or else, install *pip* from their official website. *Proceed if pip is installed.* [36]
8. Next in the same command prompt and type the following commands to install all the dependencies:
  - ***pip install pyserial***
  - ***pip install numpy***
  - ***pip install scipy***
  - ***pip install pandas***
9. These are all the external libraries that we are going to use to communicate with the radar which are easily installed through the python package installer.
10. Then, go to *Device Manager* (steps mentioned Appendix E) and note down the *port number* (usually **COMXX**) of **USBSerialDevice**.
11. Open ***rad.py*** program in any editor (preferably python IDLE) and change the ***port number in line 14***.

12. These steps are one-time setup only. When you are done with all these steps, you're ready to communicate with the radar.
13. To communicate with the radar, open rad.py in python IDLE and run the program by pressing F5. Else, from the source directory, type ***python rad.py*** to generate the output onto the console window.

**ONE SHOULD GET THE RESULT in (m) ON THE CONSOLE EVERYTIME THE PROGRAM IS RUN.**

## APPENDIX B – HOW TO COMMUNICATE WITH IMU

There are many ways to communicate and read sensor outputs from IMU. **IAR workbench** is required to program the IMU component. One can also use the **MotionLink** software to view real-time readings. IAR Workbench is not a free software. They come *in trial edition and full version*. The trial is enough to run and debug the program. The following are the one-time setup steps to be done to initiate communication with IMU.

Steps to setup the IAR Workbench:

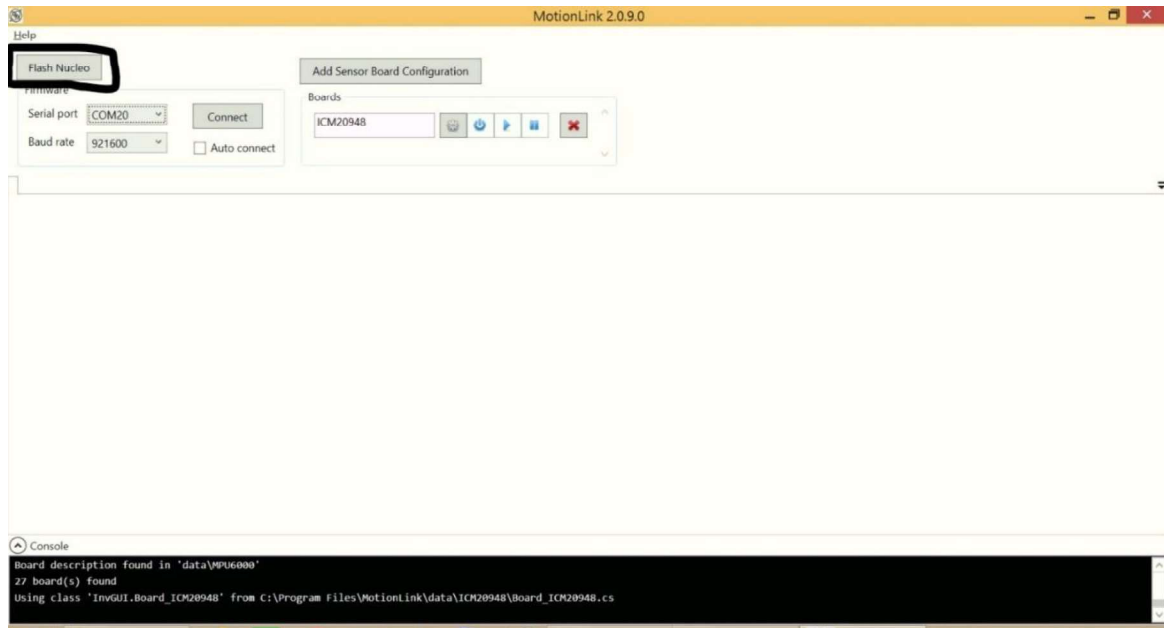
1. Download the **IAR Embedded Workbench for ARM** from [www.iar.com/iar-embedded-workbench](http://www.iar.com/iar-embedded-workbench), by choosing the architecture as “**ARM**”.
2. Once completed, please run the setup program to install the workbench and click **Install IAR Embedded Workbench for Arm** button to start the installation.
3. Click next to choose location of installation. Then make sure **STM** is checked in the next page where you’re asked to *select drivers to install*.
4. Next, download the **STM32 ST-LINK utility**, from <http://www.st.com/en/development-tools/stsw-link004.html>. *Note: You will need to create an account with [www.st.com](http://www.st.com) to access these files.*
5. *Extract* the above downloaded file and open *STM32 ST-LINK Utility v4.2.0 setup.exe* to install **ST-Link**.
6. Install the drivers downloaded from <http://www.st.com/en/development-tools/stsw-link009.html> by choosing *the appropriate architecture x86/64*. Extract the above downloaded file and open *dpinst\_amd64.exe* for amd64 architecture computer and operating system to install required USB drivers.
7. An optional step involves the firmware upgrade using the software downloaded from <http://www.st.com/en/development-tools/stsw-link007.html>. This step is not a compulsion as the firmware we have is updated to the current latest version v15.0. In future if there are any firmware upgrades by STM, one must download and install the file from the above link. [26]

To run **MotionLink**:

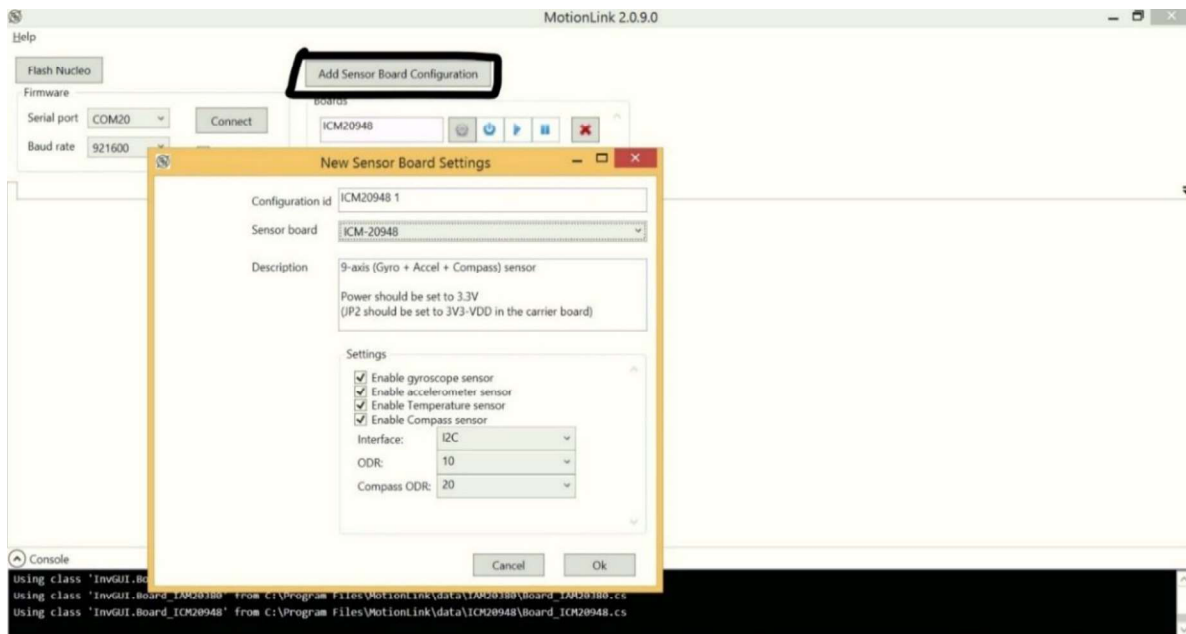
1. Go to <https://www.invensense.com/developers/software-downloads/> . Download and install **MotionLink**.
2. *Right click* on **MotionLink** icon and click on **Run as Administrator**.
3. Click on **Flash NUCLEO** button as shown in the picture below to flash the IMU with binary file

Refer to the next page for pictures and further directions.

The following figure shows the **MotionLink** software main window. It is a very useful tool which provides tools to read, log and save data from the different sensors of the IMU device.

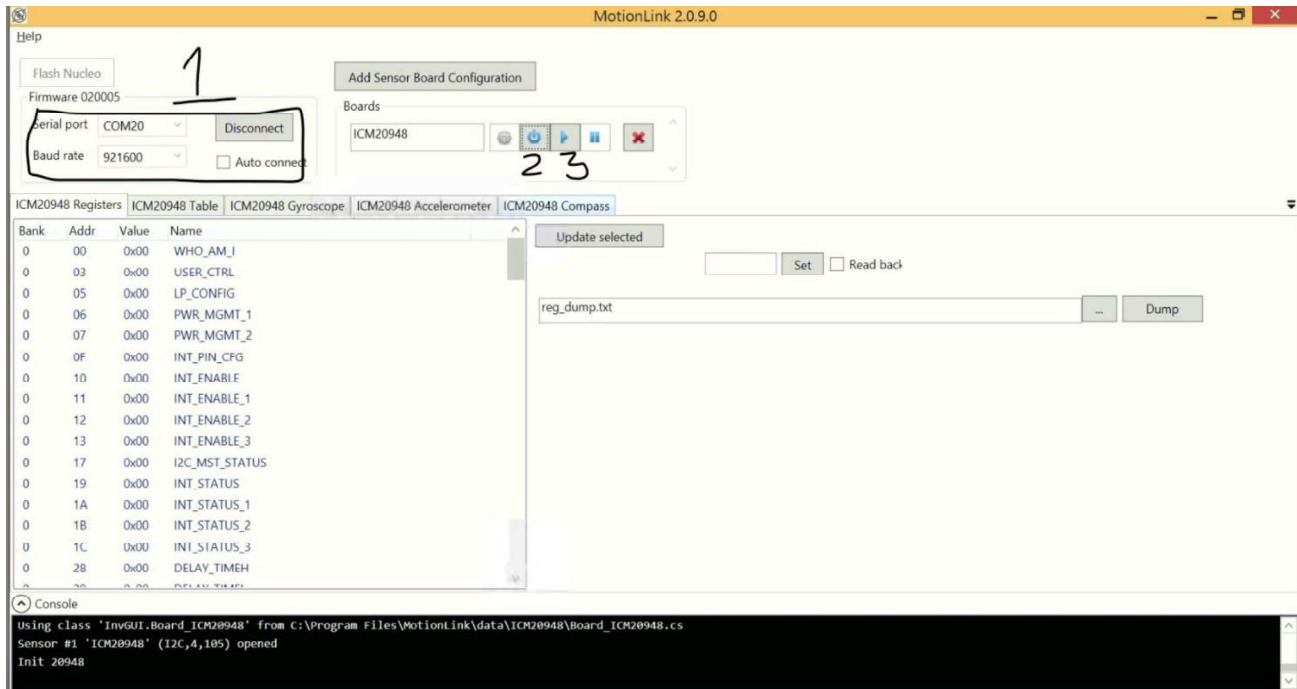


- Click on Add Sensor Board Configuration button as shown in the picture below and choose **ICM20948** from the list and check all the options as shown below.



- You can also change ODR (Output Data Rate) in this screen. You can also select which sensors to activate at this point.
- Make sure that the selected port number is correct, and the baud rate is set to *921600*.

7. Click on **Connect** and *power up* the ICM by clicking on power button and then play button as highlighted in the picture below in the same order, as shown by steps 1, 2, 3 in the figure below.



8. Click on different sensors to get real time readings of each one. The following picture shows the output of Gyroscope. In a similar way, we can get real time readings for all the other available sensors as well.



9. To store these readings in a text file, go to **ICM20948 Table tab**, select the *location* in which you're interested in storing the text file, and *click enable* whenever you're ready. We can use the text file to do various analytics on the obtained data.

**To run Sensor-cli (Cube Program):**

**sensor-cli** is a command line application used to control InvenSense device from a PC for evaluation and testing purpose.

*Included features, non-exhaustive list:*

- a) Start/stop/configure sensors for an InvenSense devices
- b) Display data on the screen
- c) Log sensor data to file

The optimal steps are provided below:

1. Go to <https://www.invensense.com/developers/software-downloads/> and download **Embedded MotionDriver (eMD) ICM-20948 v1.0 for Nucleo Board** and then extract the downloaded file. These extracted files in the folder are the tools we need.
2. Open *command prompt* (cmd) and navigate to *tools* (Wherever you extracted the files) folder using *cd* (Change directory) command. It is located deep within the extracted folder under  
...\\ICM20948\_eMD\_nucleo\_1.0\\invn\\firmware\\emd-mcu\\nucleo\\icm20948\\iar-cmd-fpu\\1.0.0\\tools  
The syntax for *cd* command is

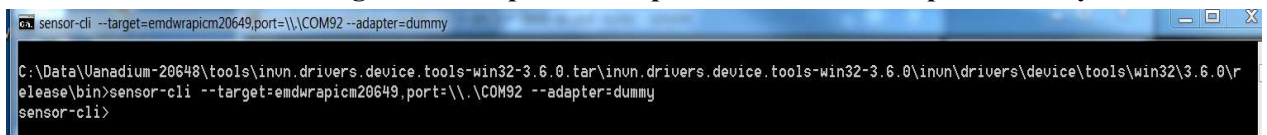
***cd <space> <path\_to\_tools\_folder>***

If you've extracted on the desktop, navigating to tools folder will look something like the picture below:



3. Now type

**sensor-cli --target=emdwrapicm20x48,port=\\.\COMxx --adapter=dummy**



in the command prompt and press enter. *If correctly configured, you will be taken to sensor-cli prompt.*

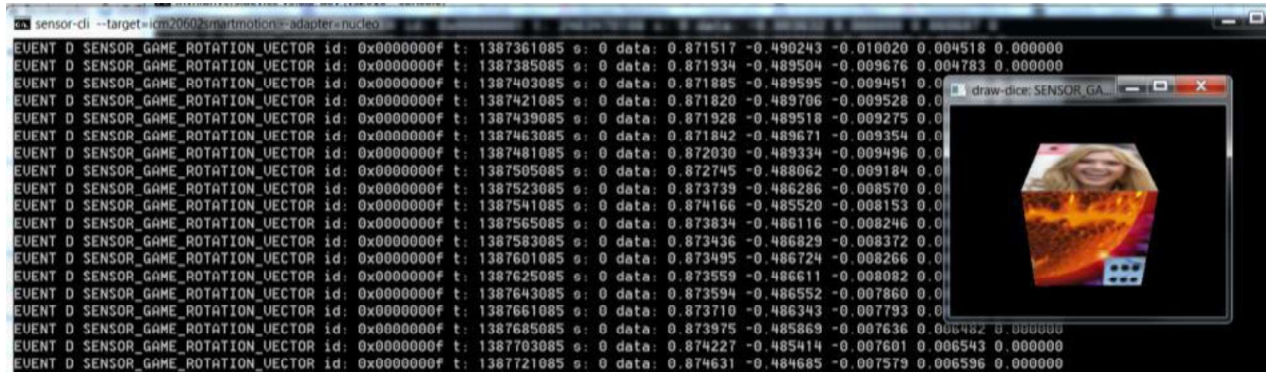
4. Now enable the cube display for *Game Rotation Vector Sensor* (GRV):

**sensor-cli> cube on grv**

*A new window displaying a cube will be opened.*

5. Now enable the GRV with a data output period of 20 ms by typing:

**sensor-cli> en grv 20**



Game Rotation Vector's data will be displayed, and the cube will move according to the IMU motion.

6. Then, to stop the sensor, use the following command:

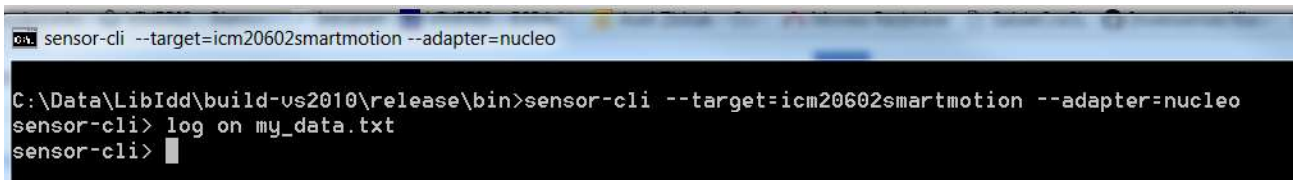
**sensor-cli> dis grv**

*Data output will stop after this command.*

7. We can also enable each sensor individually and log the data to a text file for further processing.  
8. First, to enable logging to a pre-selected text file, use the following command.

**sensor-cli> log on my\_data.txt**

*If you do not type full path to my\_data.txt, the file will be created in the current directory*



9. Then, you can *enable each sensor individually* with specific data rate interval (in milliseconds).  
10. For example, to enable accelerometer with 10 ms data rate interval, use the following command

**sensor-cli> en acc 10**

*Output from accelerometer sensor will start flowing in at the specified rate.*

11. After performing some tests, we can *disable* the sensor by using the following command:



**sensor-cli> dis acc 10**

*This command disables the sensor and output will stop flowing.*

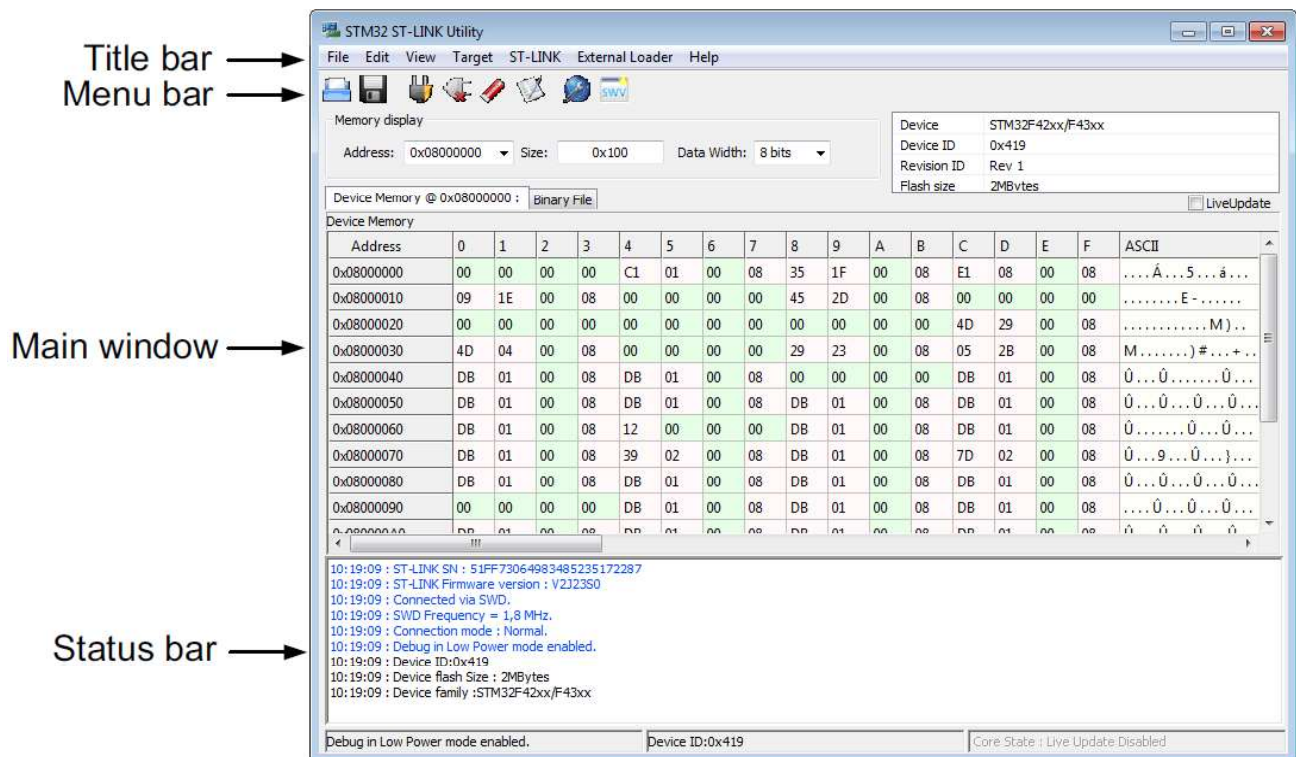
12. Finally, we can check the data from **my\_data.txt** file.

```
my_data.txt
1 D SENSOR_ACCELEROMETER 0x00000001 0 2059052085 -0.018799 -0.002563 0.883972 0
2 D SENSOR_ACCELEROMETER 0x00000001 0 2059052085 -0.022644 -0.005981 0.868591 0
3 D SENSOR_ACCELEROMETER 0x00000001 0 2059060085 -0.020081 -0.005127 0.872437 0
4 D SENSOR_ACCELEROMETER 0x00000001 0 2059068085 -0.019653 -0.006836 0.877991 0
5 D SENSOR_ACCELEROMETER 0x00000001 0 2059076085 -0.021362 -0.003845 0.876709 0
6 D SENSOR_ACCELEROMETER 0x00000001 0 2059086085 -0.021790 -0.005554 0.869446 0
7 D SENSOR_ACCELEROMETER 0x00000001 0 2059096085 -0.022217 -0.002991 0.877991 0
8 D SENSOR_ACCELEROMETER 0x00000001 0 2059106085 -0.023071 -0.010254 0.875427 0
9 D SENSOR_ACCELEROMETER 0x00000001 0 2059116085 -0.019226 -0.005127 0.872864 0
10 D SENSOR_ACCELEROMETER 0x00000001 0 2059126085 -0.020508 -0.007690 0.869873 0
```

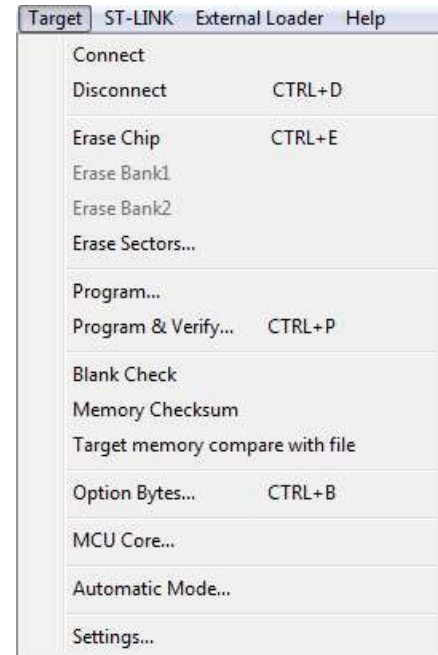
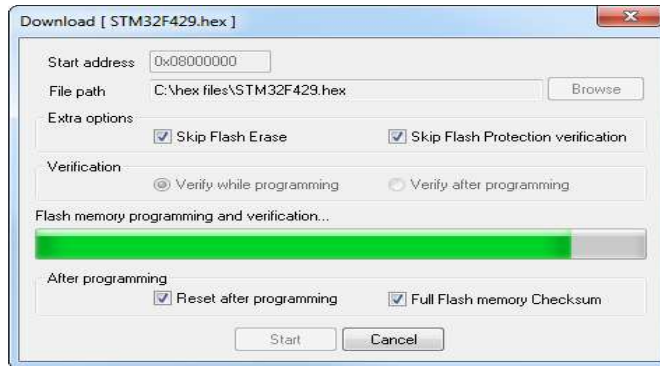
To run **STM32 ST-LINK Utility**:

The STM32 ST-LINK utility software facilitates *fast in-system programming* of the *STM32 microcontroller* families in development environments. This software can be used to *flash the NUCLEO board* with pre-compiled binary files, without the use of Integrated Development Environment(IDE) like IAR Workbench.

The main window of ST-Link looks something like the picture below:



1. From the *Menu* bar, click on **Target** and click **Connect**
2. This connects the IMU with the software and is now ready to be *flashed*.
3. From the same *Target* menu, select **Program** to flash the Nucleo board. Then, choose the **bin/hex file** and click on start.
4. You should see the flashing process begin and complete like shown below.



5. This will *flash the Nucleo board with the selected file and is now ready to operate*.

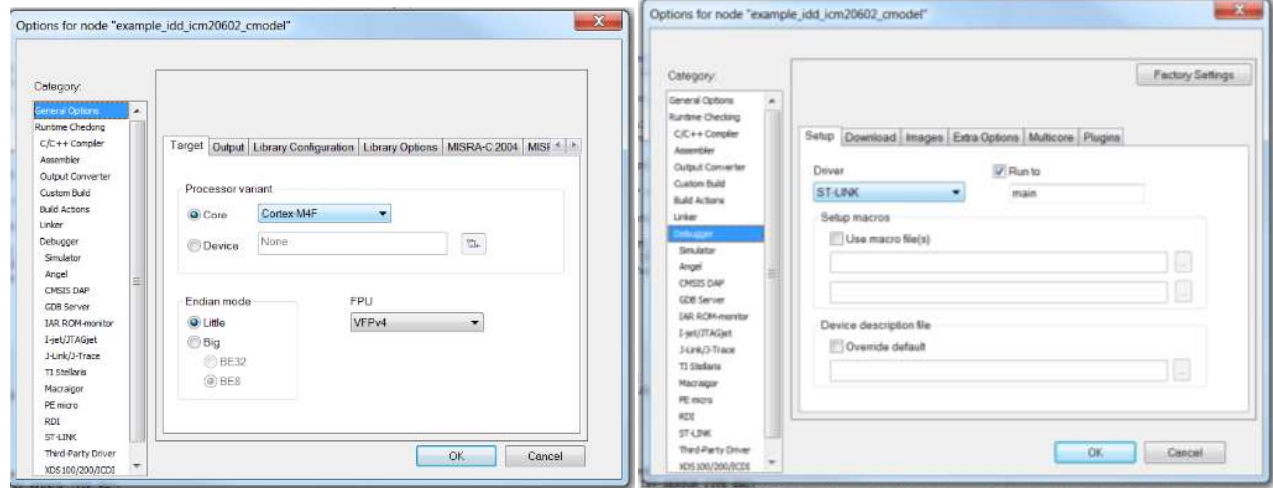
Please refer to the documentation for further details like *firmware upgrades, hex editing and many other advanced services*, which can be found here

[http://www.st.com/content/ccc/resource/technical/document/user\\_manual/e6/10/d8/80/d6/1d/4a/f2/CD00262073.pdf/files/CD00262073.pdf/jcr:content/translations/en.CD00262073.pdf](http://www.st.com/content/ccc/resource/technical/document/user_manual/e6/10/d8/80/d6/1d/4a/f2/CD00262073.pdf/files/CD00262073.pdf/jcr:content/translations/en.CD00262073.pdf)

#### To run **IAR Workbench**:

*IAR Workbench* is an *IDE* to develop and debug the source code to work with IMU. The steps on how to download IAR Workbench has been discussed previously in this document.

1. Once the software is installed, we should be able to open the project saved in **EWB** file format in the IDE.
2. On the left side, we see all the available *source files including headers*. The high-level code can be found in **example.c** file.
3. To check the configuration of the project, left click on project options in “**General Options**” => “**Target**” that the core is properly set to **Cortex-M4F**. And in **debugger Category**, check that **ST-LINK** is selected.



4. To compile and debug the application, first connect the micro USB to the PC linked to the IMU.
5. On the Menu bar of IAR IDE, select **Download and Debug**.

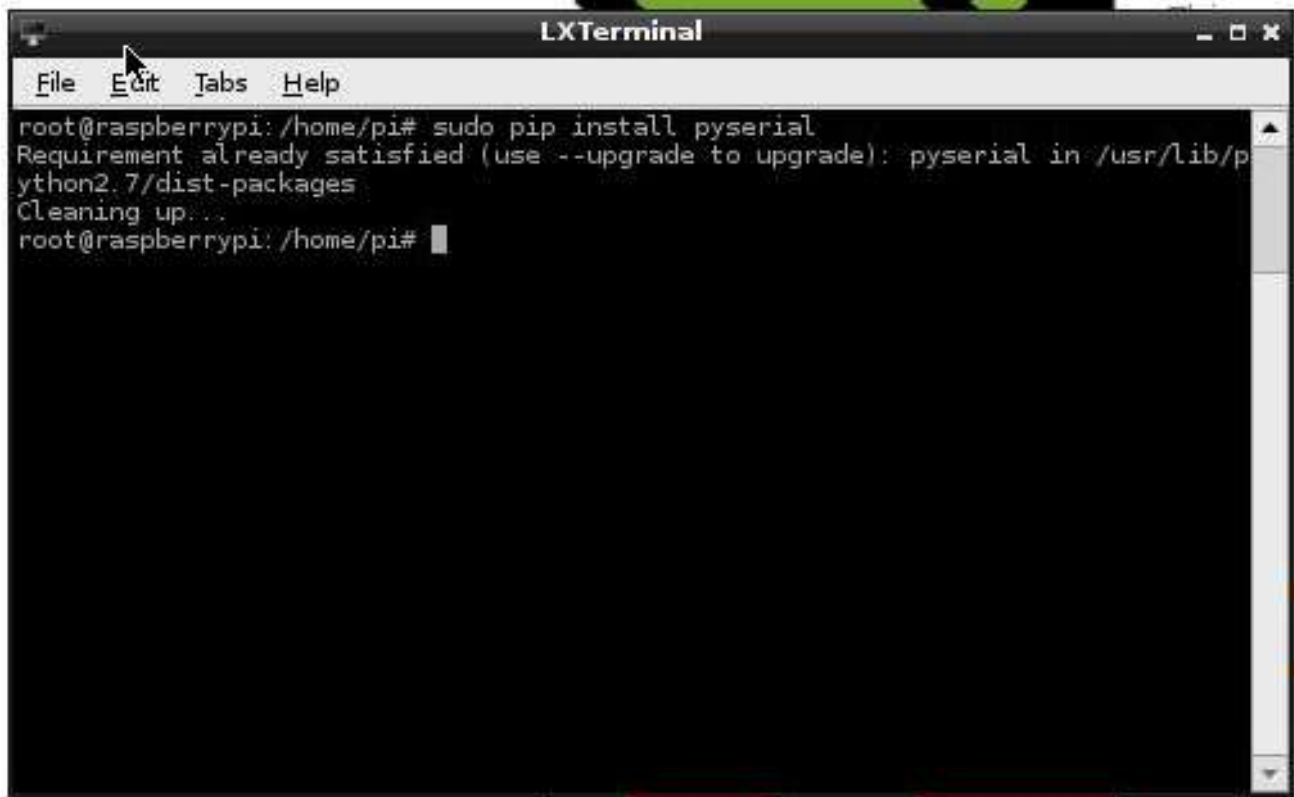


6. This will download the application on the ST Nucleo board. We can debug the software line-by-line or block-by-block and fix bugs and errors.
7. Please refer to IAR Workbench *documentation* for further details.

## APPENDIX C – HOW TO WORK WITH BOTH RADAR AND IMU SIMULTANEOUSLY

*Raspberry Pi* comes pre-installed with Python. The final python script is titled **app.py**. This script can produce real time readings from both IMU and radar simultaneously. Prerequisites for the said python script is a python library called **pySerial**. To install pySerial, open terminal (cmd) and type the following command.

*sudo pip install pyserial*



```
LXTerminal
File Edit Tabs Help
root@raspberrypi: /home/pi# sudo pip install pyserial
Requirement already satisfied (use --upgrade to upgrade): pyserial in /usr/lib/python2.7/dist-packages
Cleaning up...
root@raspberrypi: /home/pi#
```

To run the python script, open terminal and type following command,

*python app.py*

Where the app.py holds the consolidated program to measure real-time readings and can be found on a supporting Compact Disk or Web resource.

## APPENDIX D – DEVELOPING CUSTOMIZED SOLUTIONS

### General Instructions

If one seeks to develop the software with custom requirements fulfilled, then one should use the *Development pathway* described below to complete the task.

- Downloading *Anaconda* –
  1. Proceed to Anaconda's download website to equip with the IDE (Integrated Development Environment) required by us to configure and use the RADAR.
  2. Select python 2.7 and the Anaconda's package will be loaded.
  3. The website is <https://www.anaconda.com/download/?lang=en-us>
  4. Documentation: <https://conda.io/docs/user-guide/install/index.html>
- Follow the steps described in the documentation of anaconda or the onscreen setup wizard to complete the installation.
- Verification of installation
  1. Once installed, open a console window on windows, by searching for it on Start Menu.
  2. We should verify if conda was installed by typing "conda" in command prompt.
  3. If installed, it should show some auto-generated text.
  4. If you see error, please visit the troubleshooting section of the guide. Easy fix: Possible error with PATH environment variable. Consoles need to be restarted to access conda.
- Installing *pyserial*:
  1. To install *pyserial*, we should re-open a console window, and type the command  
*conda install pyserial*
- Launching *spyder* (Python IDE)
  1. Launch *spyder* either from command prompt or from anaconda navigator.
- Checking device COM port & initialization:
  1. Follow the steps provided below to start *Device Manager on Windows*.
  2. In the *Device Manager*, note the port of the *USB Serial Device* attached, which reads the RADAR Components device ID or name.
  3. In the sample code, note the initialization of the serial library and COM port. Here you must change the COM port to the one noted in step 2.
- Reading data from RADAR
  1. Setup the device to begin acquiring readings.
  2. Next, navigate to the folder containing sample program file "rad.py"
  3. Hold *Shift + Right* click to open the context menu in the folder containing the file. Next, select open Command Prompt here.
  4. Once inside the CMD line, enter "python rad.py" to receive the distance as the output, in meters.

This next section provides the details of using the RADAR. We demonstrate the code related to setting up and using the RADAR Data. The RADAR data requires post-processing to generate any real data. Hence, we use the following imports in our rad.py python program. Some of the imports can be skipped if the calibration equation has been derived previously.



```

#####-I-M-P-O-R-T-S---#####
#####
import time #built-in Time library, sleep function to pause CPU

#####
import math #built-in Math library, math functions like sine, cos etc

#####
# py-serial library provides various support functions for working with serial port devices which help automate our job.
import serial

#####
# pandas is a data analysis support library that provides the necessary tools such as data structures and math functions on such
# strpductures
import pandas as pd

#####
# SciPy provides a discrete fourier transform library
from scipy.fftpack import fft

#####
# mathPlot library provides a function for plotting values on a graph
import matplotlib.pyplot as plt

#####
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along
# with a large collection of high-level mathematical functions to operate on these arrays.
import numpy as np

#####
# SciPy is an open source Python library used for scientific computing and technical computing.
import scipy
#####

```

Next, we setup the *pyserial* object with appropriate port number (**port-number/baudrate can be set/verified in *Control Panel/Device Manager***), along with its baudrate. These values are available in the data sheets of the devices.

```

#####
#Setting up the Port and Data channel speed. Initialization of communications port.
ser = serial.Serial(
    port='COM4',
    baudrate=115200,
)
#####
# Printing the connection to serial port status, if OK, then proceed.
print ser.isOpen()
#####

```

Then, we initialize the variables required by the program.

```
#####
# Initialize a small list of commands to be sent over the serial connection, which help to initialize, sweep, trace and collect
# the radar data.
list_of_commands = [
    'hard:syst rs3400w',
    'hard:syst ?',
    'INIT',
    'SWEEP:MEASURE on',
    'SWEEP:NUMBERS 1',
    'TRIG:ARM',
    'TRACE:DATA ?',
    'TRACE:FFT:CALCulate',
    'FREQUENCY:IMMEDIATE ?',
    'TRACe:READ:DIStance ?',
    'Trace:read:FINDeX ?',
    'Trace:read:PINDeX ?'
]
#####
# Initialize certain variables
measuredDist = 0.0 #Floating value initialization
#####
```

Finally, here is the sample code for the main portion of the program:

```
#####
#Start a try-catch block to handle exceptions and prevent program crash
try:
    # Loop over the list of commands one by one
    for command in list_of_commands:
        ser.write(command + '\r\n') # Push the command to RADAR device and expect to receive feedback from it.
        out = ''
        print command # Display the current command to the user for feedback.
        if(command == 'TRACE:DATA ?'):
            print 'if' # Tracing for debugging purposes
            time.sleep(1)
        else:
            print 'else' # Tracing for debugging purposes
            time.sleep(.4 )
        # Check status and wait to receive the output from the RADAR.
        while ser.inWaiting() > 0:
            out += ser.read(1)
        print out # This object holds the output data strings.
        # Showing the measuredData from RADAR.
        if(command == 'TRACe:READ:DIStance ?'):
            out = out.replace("\r\n", "")
            x = float(out)
        else:
            print out
    # The serial library allows to close the serial port connection once done
    ser.close()
    # Accurate distance generation using curve fitted calibration formula
    dist = -0.003754*x*x - 1.391992*x + 128.851509
    print 'dist:::'
    print dist
except:
    #Error handling
    print "error"
    ser.close()
```

This program code is just for reference, the actual curve fitted calibration equation needs to be calculated one time using some basic tests as mentioned in the report. The structure of the program remains the same but there might be some changes required to produce the expected results.

## APPENDIX E – PYTHON LIBRARY DETAILS

### ✓ *pandas*

pandas is a [Python](#) package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal. [33]

### ✓ *numpy*

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the [BSD license](#), enabling reuse with few restrictions. [32]

### ✓ *scipy*

SciPy refers to several related but distinct entities:

- The *SciPy ecosystem*, a collection of open source software for scientific computing in Python.
- The *community* of people who use and develop this stack.
- Several *conferences* dedicated to scientific computing in Python - SciPy, EuroSciPy and SciPy.in.
- The [SciPy library](#), one component of the SciPy stack, providing many numerical routines. [32]

### ✓ *py-serial*

This module encapsulates the access for the serial port. It provides backends for [Python](#) running on Windows, OSX, Linux, BSD (possibly any POSIX compliant system) and IronPython. The module named “serial” automatically selects the appropriate backend. [35]

### ✓ *matplotlib*

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](#) shell, the [jupyter](#) notebook, web application servers, and four graphical user interface toolkits. [34]



## Device Manager

These can be setup in the Device Manager section of the Control Panel in Windows. To access the Device Manager, please follow these steps.

- Click the bottom-left Start button on desktop, type “*device manager*” in the search box and tap **Device Manager** on the menu.
- Or else, Open **Device Manager** from Quick Access Menu. Press **Windows** + X to open the menu and choose **Device Manager** on it.
- One might also Access **Device Manager** in Control Panel. [Google]

Once device manager is visible, we can find the device under **Ports (COM & LPT)**. The RADAR should be listed under this section only.

**Note:** Device Manager is used to setup both the devices.