VISA Communication Tool Project Progress Report

Chenliang Wang[1], Wenbo Hou[1] and Lucien Armand Tamno[1]

[1]Oregon State University

February 17, 2017

Abstract

This document describes the current state of the project, VISA Communication Tool. Each team members will state their contributions to this project. The team will also release the alpha version of the project, and necessary details will be in this documentation.

Contents

1 Project Overview

The VISA Communication Tool is an instruments manage software designed for Tektronix. It is used to replace the old manage software that has limited access features and long responding time. The new software offer code auto-complete and a querying tool, so that users can get rid of the lengthy programmer manual. It also allows users to access block data to debug their script.

2 Weekly Progress

2.1 Week One

In first week, the team met with Kevin to review previous work. Then, the software source structure was created and uploaded to GitHub. Also, team members helped each other to set up the developing environment.

2.2 Week Two

In this week, we met with our new TA, Shivani to talk about the future plan. Shivani mentioned that the team should release the alpha version in week six. Consequently, the team decided to code.

2.3 Week Three

The main task in this week was configuring IDE for group members. Team members take advantages of PyCharm built-in package install tool to add PyVISA and PyQt to our project. According to Weekly blog, all team members completed at least one module of their tasks

2.4 Week Four

The developing group continued working on their tasks. Shivani, our TA required that each team members should contribute to the team repository.

2.5 Week Five

Week five was also a coding week. Unfortunately, the GitHub Repository did not have much changes. The team notebook on OneNote was publishe, and its content kept updating.

3 Individual Contributions

3.1 Code Auto-complete and Querying Tool —– Wenbo Hou

3.1.1 Responsibility Review

From the project proposal and discussions with clients, one of concerns of developing the new VISA communication software is to free users from the programmer manual. Our clients, Tektronix engineers stated that VISA command syntax is too complex to memorize. It was not efficient to referring the coding documentation when users operate

instruments through the old software. Inspired by main-streaming IDE, our client want a new software with a command auto-complete plug-in and a querying tool.–Figure cac1–

According information from QT Documentation, the code complete function includes two parts: providing command reminder and completing inline code. To accomplish that, the program first need to read the input and query it through the built-in documentation. Then, it will display reminders or error messages on GUI. The querying tool is fairly simple, the developer can implement searching functions inside python standard libraries based on their performances including, speed and accuracy

3.1.2 Current Task Progress

First of all, the software's source management need to be settled done. The team decided to organize the program source by purposes. As an extension of software GUI, developers need to specify a uniform GUI package for this project that is PyQt. At the beginning, the developer chose to code this program based on the context viewpoint: Input -> Compute -> Out. Considering the difficulty of each section, the developer adjust the coding order to: Input -> Out -> Compute.

The starting point is taking input from users. Since the code complete is only work when users is typing, it cannot be active as the software running. Then, the developer take advantages of **textChanged()** function built-in PyQt to trigger the code complete function. This function will also pass the current input to the code complete function.

```
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *

......

def trigger(self):
  self.entry.textChanged[str].connect(self.code_ac)
  # code_ac is the code complete function
  # called when the program detects textChanged event

......
```

Listing 1: connect the code complete function to textChnaged Event

Currently, the code complete function only contains a simple search function to generate results for testing output. One convenient feature in this module was assigning symbols to different types of commands. As a result, users can use it as a hint to track commands. The height of the list changed based on the item count so that the user interface can be succinct.

The developer also wrote test cases for above features. The first step was to create an entry to check if the input-getting module can get the correct input, which is an unity test. The **textChanged()** function was tested, either. The output module test has tow parts: fixed row items and dynamic row items. The developer filled the list with both values inside the source code and values from another file to simulate different situations. The final integrated test putted the input module and the output module in the trigger function and added a connection between them. Consequently, the program pipeline was formed and tested.

3.1.3 Problems ad Solutions

The first problem was configuring IDE. Unfortunately, PyQt package is not a standard library built-in Python. PyCharm has trouble recognizing certain attributes, not to mention compiling the program. According to information from Stack Overflow, users could directly install packages through PyCharm package import feature. PyQt works in this way, either.

The second technical issue occurred in the output module. The unit test showed that the space between rows was larger than the developer predicated when import values from another file. To solve this issue, the developer first

checked if the row height setting was affected. After finding no bug in that part, the developer evaluated two different test cases (value inside the program and value from another file) and decided to check if the external file contained bad characters. The developer eventually figured out that the **new line** character caused extra space.

Another technical challenge is about the program's performance. As described before, the output list will show a small icon as type identifier. However, the program became significantly slow when loading one of those icons. Then, the developer compared those icons and found that the problem icon's size (124KB) is much bigger than others (around 20KB).Thus, the developer searched a similar icon with smaller size and tested again. The problem was solved.

The developer fixed another issue in this Monday. The PyQt object, **QListWidget** has an default height value so that it exists on the main window whenever the code complete function is running or not. Then, the developer set a list height flag associated with **code_ac** function. when the code complete function is working, it will adjust the height of the drop list based on the item count. The maximum height is five rows and the minimum is zero.

```
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *

......

  if text == line[:len(text)]:
     lstitem = QListWidgetItem(line[:-1])
     # avoid the last new line character
     lstitem.setIcon(QIcon("..\..\Src\Img\O.png"))
     # set icon associated with each row
     # need to assign different icons with different command sets
     self.list.insertItem(index, lstitem)
     index += 1
      ref.close()
  if index < 5:
      self.list.setFixedHeight(font_size * index)
      # adjust list height when item count is less than 5
  else:
      self.list.setFixedHeight(120)
      # give a fixed 5 row height to the drop list
      # when item count is greater or equal 5
......
```

Listing 2: avoid newline character and adjust list height

3.1.4 Future Work

The primary task is to completing the code auto-complete service. The developer will accomplish the command-matching work and the KeyPress events handler. Tests will be associated during the developing period. After that, the developing group should integrate all code blocks together to release a alpha demo to the client and the course instructors.

Another task is querying tool. The developer shall provide a quick and accurate querying service to the software users. As proposed in Design Document, the developer needs to figure out a effective searching algorithm and data structure to query and store information. Then, the developer must communicate with clients to get related documentation.Depending on the group progress, developers need to integrate code blocks again to get a newer version of the software.

When complete assign tasks, the developer will focus on the performance optimization. For example, the developer could replace an old, slow algorithm/tool with a quicker one. Modularizing code is another optimize choice.

Besides basic coding work, developers shall allow the clients evaluate the software and make changes based on client's

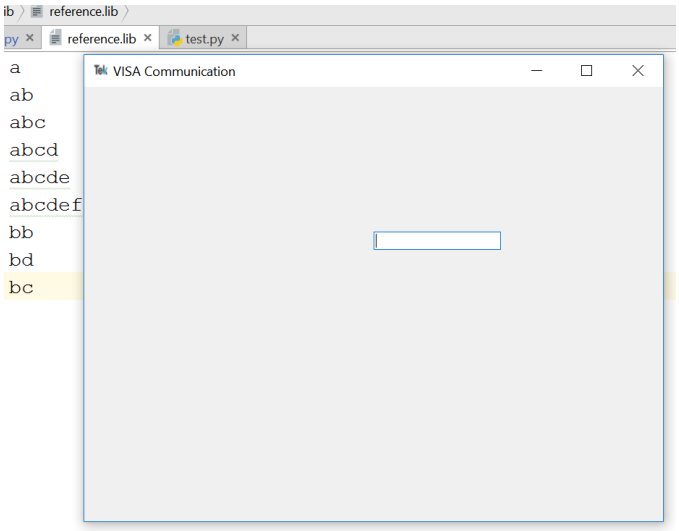

Figure 1: Reference and GUI with deactivated code complete program

feedback. If the beta version was done at the end of the winter term, the group could start a blind test.

3.1.5 GitHub Check Points

Date	Message	Commit Code
Jan 13,17	Set Up Project	cddcbece113aa02438df028f5ef1c65d90c311a0
Jan 13,17	Configuration Help	8bbe6908149e8863cbf32903ffeca4a4397683e2
Feb 02,17	Initial Software Icon	21f8057bb8588634aa8140ab6b11b1d7df4f2f65
Feb 05,17	Test Script and First attempt	ff7fea8408ef8da6e5d53e48572633a2ac9b16fc
Feb 06,17	Fix extra space in list	6af5aa863112208513c4b2bad73403621321dc22
Feb 06,17	Set inline icons	f175edf96163a993be25aaf557202b21bf25ad9d

Table 1: Personal Milestones in VISA Communication

3.1.6 Quick View

Here are screen shoots showing current functionality.

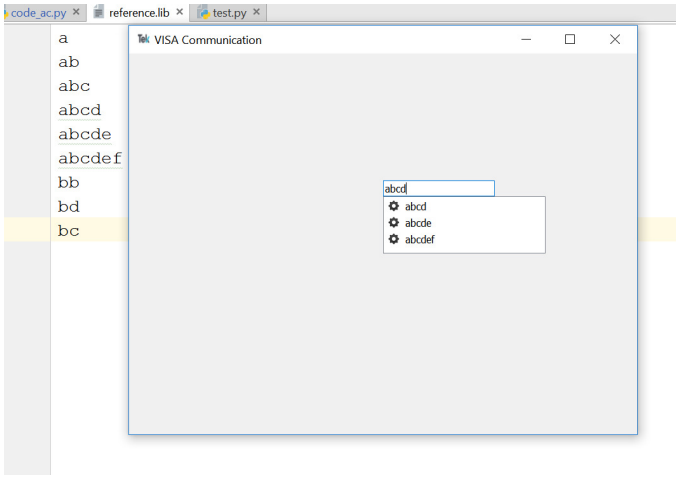

Figure 2: Reference and GUI with activated code complete program

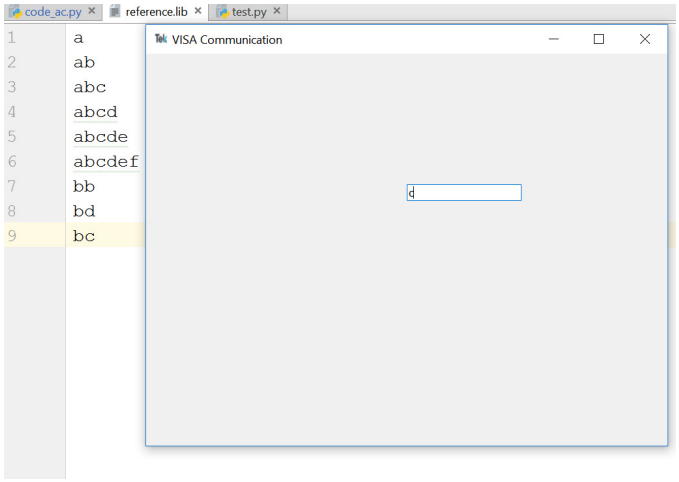

Figure 3: Reference and GUI with deactivated code complete program.

3.2 Progress of Project —— Chenliang Wang

3.2.1 Responsibility Review

Last term, we catch the project from Tektronix. We need to create a new VISA communication tool based on python. From the requirement, we understand there are couple things we need to do. Such as GUI, input command, output, display, command automatic, grammar check, and etc. In order to do this project well, we separate the project to several parts, each member will focus on some parts, and we combine at the end to done this whole project.

3.2.2 Details of task progress

For our project, there are three parts for my task. GUI design, GUI implement and function test.

- GUI design

  The first task for me is the GUI design. Last term, I already draw the design of GUI and email client to check it. They gave back some feedback and I changed it. This term, I try to compete the GUI based on the design. For this part, it is not difficult. After I sent to client, they only point out few small problems what they didn't like. All in all, I think the client also agree with the first user study, they keep the almost idea, and the change is such as add some function button and change the display method. Therefore, this part is looks complete and it maybe change after I coding, but it not a big deal. Also, I did this design with Balsamiq Mockups. It not a really GUI, but it can be click and show more clearly about GUI. I will code based on design and show the clients after coding.

- GUI implement

  The second task is for GUI implement. From this term, I begin to coding for the GUI of our project. Based on research last term, I use PyCharm and PyQt5. It not looks very hard, but it really makes me a lot of trouble as I never use it before. Setup the environment should be very easy but it really cost me a long time. Until week for this term, I do a lot of research online, and almost know the all functions of GUI how to do it. To until the week 5, The GUI is created with few labels, texts, buttons. For the functions, I already done with the navigation bar, the command input with write and query, the command output with last history and separate the write and query, the IP connection window that show the IP in the new sub window. So, I done some basic function of the tool but it not really called a complete GUI. I just know how to do almost function, but I didn't consider about the exterior of GUI itself. As we need to submit the alpha version nearly, I plan to focus on the functions and modify the exterior later.

- Function test

  The third task is the function test. It is an important part even if I put this for the third part. Until now, I just did the human test in order to make should each functions look work well. The human test is absolutely not enough, I will do the unit test and test from other ways later. The unit test is the professional way to test the code.

All in all, the GUI design is almost done, and it already accept by our client. The GUI implement is working right now. Some function is already done, and some will be finish later. The function test will begin to do it soon.

3.2.3 Future task

For the future plan, it generally 4 parts need to do.

- Continue other functions.
- Organize the code.

- Modify the UI.
- Unit Test part by part and debug.

These things will be done and display on the beta version. In order to done these things one by one, I set a flexible plan. First, finish all functions before week 7. Second, the layout problem and code organization will be done before week 8. Then, I will send the UI to client for the final confirmation. Last two weeks to test and debug. The expo board will be doing during these 4 weeks.

3.2.4 Interface design first user study

This part is only for interface design of first user study. As I said, the GUI design is almost done last term. After I send the GUI design to client, they gave some feedback. As beginning design, I design only one command send button with both write and query. User can use that switch the mode between write and query. Also there will be a "W" or "Q" in front of the command line to separate the mode is write or query. However, client think it is not friendly, they gave the advance to put two button one is for write, another for query. Therefore, I change to two button and the image you can see later is the new one. All, in all, based on the GUI design, I think they agree with this design and should have a good first user study. However, for the really GUI, I didn't finish all parts now. I will improve my speed, done the GUI as soon as possible, send to client, and get the first user study for the really GUI product with coding.

3.2.5 Problems and solutions

- Set up the environment

  As I have no experiment to do a project use PyCharm, Python3 and PyQt5. It hard for me to set up the coding environment at begin of the term. After I told this to Wenbo who is one member of our group, he wrote a document on GitHub that make I solve this question quickly, and also make sure we use for the same version of the coding environment.

- Some function is hard to do

  During the coding time, one function is hard for me to implement get the goal. I research online found different resources and solve the problem at end. In the future, it might happen again, I will get help from Ta, instructor, and client.

- Slow process

  After I write the report and presentation, I realize that I need improve my speed of process. It already week six but I didn't done enough parts as my plan at beginning. From now, I will improve my speed and do the plan to done the rest part week by week.

3.2.6 Quick view

These are some images for the project.

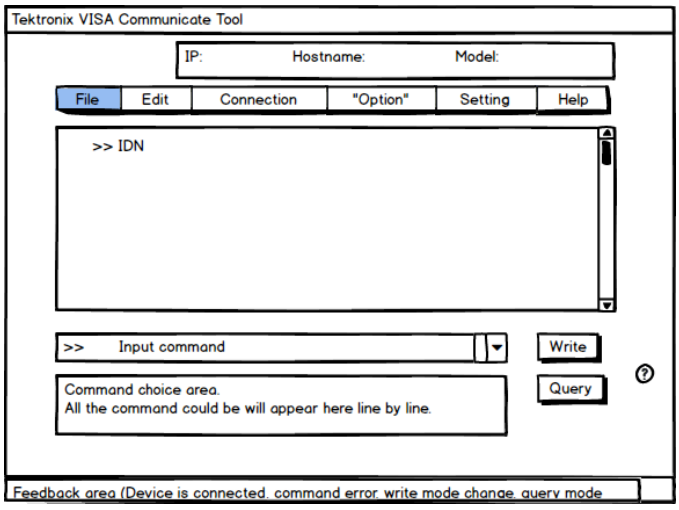

Figure 4: The GUI design

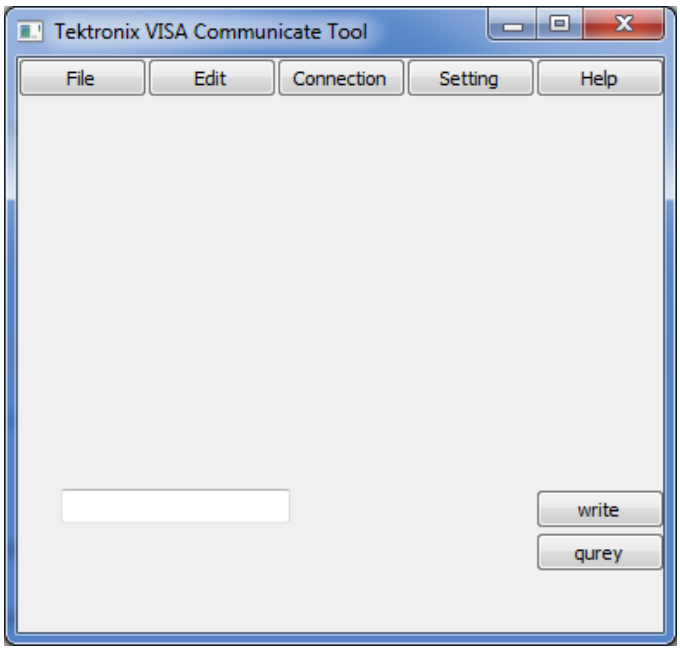

Figure 5: The GUI by coding

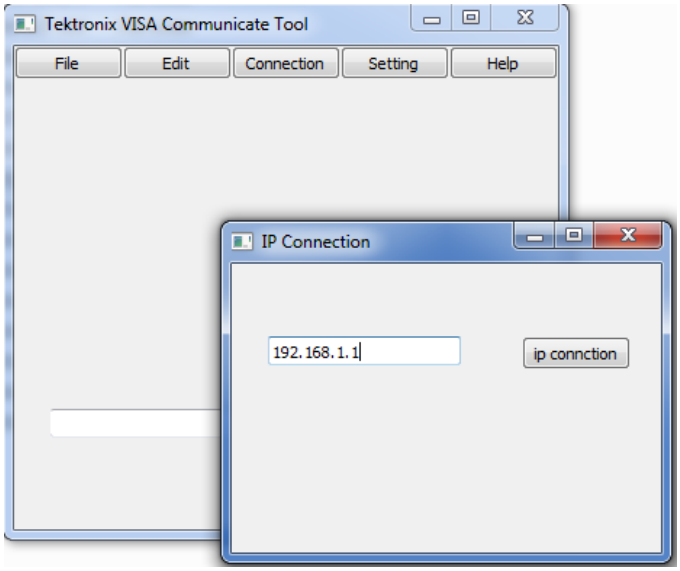

Figure 6: The IP connection

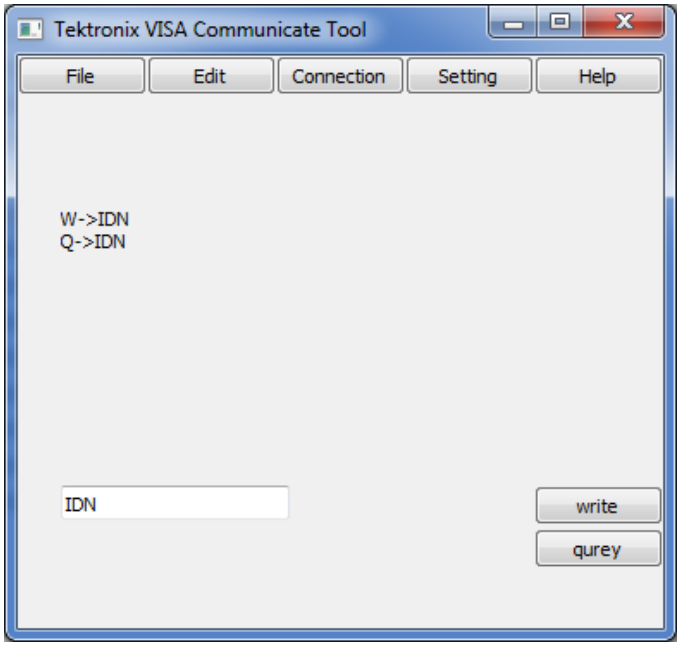

Figure 7: The input and output with different write and query

3.3 Lucien*****Progress Report on the coding's VisaCommunicationTool & adds-on

3.3.1 week 1

Essentially, this first week of the winter term 2017 was about to have some talks with instructor Kevin, in order to determine the direction to be taken for the coding part of the **VisaCommunicationTool** software. To do so, the first thing was to dive back into the project designed document and to work on the details of the coding process, we locked down our weekly day meeting with the new TA: Shivani.

Further, this first week was seen as a week marking a crossover point, between the fall term seen as the ending moment over the designed document, which in turn serves as a prelude of the coding side of the Project and on the other hand, the beginning of the winter term depicting the phase of coding and testing.

3.3.2 week two

this week,featured in good terms is the starting point of elaboration of how the coding will be. In which tool can efficiently get the job done, and prior to that, i dive back into the designed document to fetch and figure out what are the various pieces of the module to be implemented. In this case, the communication API needed among others IDE(s) pycharm that has been revealed being a very effective IDE used in the community of developers all these last 5 to 8 years. And talking about pycharm, this integrated development environment seems to work pretty well with python, for it can easily pull into a single development space, all what we need to essentially attain the task we want to carry out in this case: coding the **VisaCommunicationTool** application. And having figure out that the pycharm was the best tool for us to work with, we place our main focus on it to know how to effectively use it. and by doing so, the team left out any other IDE like Eclipse which is well known by the community of developers.

3.3.3 Week Three

The third week has the marks of gradually turning on all writings done so far into **VisaCommunicationTool** seed which in the coming weeks will be dynamic measurement instrument tool that our client Tektronix needs to daily carry out its innovative business culture on the market and to make it easy for the users to speedily do their job. pycharm hasn't been an all-in-one integrated environment. Instead, for me to write the first draft the code as displayed under week five' session, i needed to integrate pyQt, pyVisa as the others members of the team has already point it out. But beyond that others integration maybe follow on the coming weeks because for the software to talk to any hardware via the communication API, the software will need library not found either on PyQt or PyVisa. for instance the NI DAQmx which i am still accessing how it can work in conjunction with other parts of pycharm IDE.

3.3.4 Week Four

this week can be termed as a week of trials and failures. In effect, during week four, i tried to write code in python which i am learning as programing language and it wasn't something to sort out overnight because writing the code and not to have flaws on it wasn't fun.I have to navigate between the documentation, the write instruction to put in there and beyond those two first,the compliance with the designed pattern i have made on the designed document. Also this was the single occasion to me to encounter limitation i had in terms of documentation and to fix that, i was forced to get the client on-board, reason for me to have the present the right documentation which lines up with the client expectation. Documentation which has been finally approved and for certain, i will dive into it to make good use and keep structuring the code as the one presented hereafter.

3.3.5 Week Five

sample python code of read/write of various type of files:

Listing 3: read/write on the buffer + manage the appropiate bus

```
infile = "myFile.txt"
retrievedFile = open(infile,  mode='r', encoding="utf-8")
if retrievedFile.mode == "r":
    retrievedFile = (open(infile, 'r') or StringIO("some␣initial␣text␣data"))
elif retrievedFile.mode == "rb":
   // retrievedFile = (open(infile, "rb") or BytesIO(b"some␣initial␣binary␣data:␣\x00\x01"))
else:
    print("this␣file␣is␣not␣readable")
    exit(0)
for line in retrievedFile:
        outfile = line.readall()
        outfile.BufferedWriter()

        # module controller of the communication between application and
        # various buses ( e.g GPIB, RS232, USB, Ethernet)
        rm = visa.ResourceManager()
        rm.list_resources()
        ('ASRL1::INSTR', 'ASRL2::INSTR', 'GPIB0::12::INSTR')
        inst = rm.open_resource('GPIB0::12::INSTR')
        print(inst.query("*IDN?"))
        outfile.flush()
retrievedFile.isatty()
retrievedFile.close()
```