

USB DEVICE CONNECTIVITY USING PYTHON

by

Mohamad Alif bin Mohd Roki
(Supervisor: Vijanth Sagayan a/l Asirvadam)

Dissertation submitted in partial fulfillment of
the requirement for the
Bachelor of Engineering (Hons)
(Electrical and Electronic Engineering)

SEPTEMBER 2013

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

DESIGN OF IMPROVED GRID FOR TURTLE ROBOT

by

Mohamad Alif bin Mohd Roki

A project dissertation submitted to the
Department of Electrical and Electronic Engineering
in Partial Fulfillment of the Requirement
for the Degree Bachelor of Engineering (Hons)
Electrical and Electronic Engineering

Approved by

(Vijanth Sagayan a/l Asirvadam)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
September 2013

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own concept as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

(MOHAMAD ALIF BIN MOHD ROKI)

ABSTRACT

This thesis is the final manifestation of the final year project that aims to determine how the Universal Serial Bus (USB) work in serial and parallel connection with coding that had been design. By using Python 2.7.5 software allows the project running smoothly, whether to send or transfer any data to or from the hardware devices. The communication of this project was recorded in the Python 2.7.5 software itself as a strong result.

ACKNOWLEDGEMENT

All praise due to Him, the most merciful for giving this opportunity to successfully complete my Final Year Project (FYP) in 6 month. I am proud and glad to express my utmost appreciation to those who have successfully guided me throughout my project finishing.

I take this opportunity to express my profound gratitude and deep regards for exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given time to time shall carry me a long way in the journey of life on which I am about to embark. Special thanks to these following people:

AP Dr. Vijanth Sagayan a/l Asirvadam

Associate Professor in Electrical and Electronic Department

Supervisor for my FYP

Mr. Patrick Sebastian

Lecturer in Electrical and Electronic Department

Co-Supervisor of my FYP

I am obliged to committee of FYP in Electrical and Electronic Department in Universiti Teknologi Petronas (UTP) for the valuable information provided by them. I am grateful for their cooperation during the period of my project. This project would be never has become a reality without the effort of FYP committee for conducting and facilitating the course. Finally, I thank my parents, brother, sisters and friends for their constant encouragement without which this assignment would not be possible.

<u>NO</u>	<u>TABLE OF CONTENT</u>	<u>PAGE</u>
	CERTIFICATION OF APPROVAL	ii
	CERTIFICATION OF ORIGINALITY	iii
	ABSTRACT	iv
	ACKNOWLEDGEMENT	v
	Table of Content	vi
	List of Figures	vii
	List of Tables	viii
	List of Abbreviation	viii
1	INTRODUCTION	
1.1	Project Background	1
1.2	Problem Statement	1
1.3	Objectives	1
2	LITERATURE REVIEW	
2.1	Python as Base Language Programming	2
2.2	USB Connection	3
3	METHODOLOGY	
3.1	Research Methodology	6
3.2	Key Milestone	18
3.3	Gantt Chart	19
3.4	Tools	20
4	RESULT AND DISCUSSION	21
5	CONCLUSION	27
6	REFERENCES	28
7	APPENDICES	29

List of Figures

Figure 1: USB port	4
Figure 2: Serial port and Parallel port	4
Figure 3: USB Basic Communication flow	5
Figure 4: Flow of Methodology	6
Figure 5: Step 1 Python 2.7.5 Installation	8
Figure 6: Step 2 Python 2.7.5 Installation	8
Figure 7: Step 3 Python 2.7.5 Installation	9
Figure 8: Step 4 Python 2.7.5 Installation	9
Figure 9: Step 5 Python 2.7.5 Installation	10
Figure 10: Step 6 Python 2.7.5 Installation	10
Figure 11: Step 7 Python 2.7.5 Installation	11
Figure 12: Step 8 Python 2.7.5 Installation	11
Figure 13: Step 9 Python 2.7.5 Installation	12
Figure 14: Example how to find the Device using USB Module	14
Figure 15: USBview software that views the Component of USB Devices	15
Figure 16: USBview shows the Bulk Type Transfer of Thumb Drive	17
Figure 17: The Methodology of the Project	18
Figure 18: Python 2.7.5 Software Interface	20
Figure 19: Thumb Drive and PC that use for Project	20
Figure 20: Coding for Detect USB Device in Python Interface	21
Figure 21: Result for the Detection of USB Device in Python Interface	22
Figure 22: Coding for Reading the USB Thumb Drive	23
Figure 23: Expected Result of Reading USB Device	24
Figure 24: Result shown of Failure in Listing the Information from USB	24
Figure 25: Coding for Writing in USB Thumb Drive	25
Figure 26: Result shown of Failure in Communication of USB	26

List of Table

Table 1: Comparison of Python and Java Programming Languages	2
Table 2: Description of PyUSB Modules	13
Table 3: Gantt Chart for the Whole Semester Project Progress	19

List of Abbreviations

ACK	Acknowledge
API	Application Programming Interface
FYP	Final Year Project
ISA	Industrial Standard Architecture
OS	Operating System
PC	Personal Computer
PCI	Peripheral Component Interconnect
UTP	Universiti Teknologi Petronas
USB	Universal Serial Bus

INTRODUCTION

1.1 Background of Study

The project go deep to looks into how functional programming using python which is used to interact with one or many hardware(s). The python programming is used to read and send signal via USB connection which can be in form of serial (one way communication) and parallel (with feedback) connections. In this project was use Python 2.7.5 software that interacts with Windows 7 64bits operating system (OS).

1.2 Problem Statement

The project problem formulation involves studying two form of USB connections – serial and parallel using python programming which are different in paradigm. The USB serial connection uses 8 bits single transmission whereas parallel connection sending and receiving information in form packets. The aim of these is project diagnostics toolkits which can be record activity of USB devices. The diagnostic tool is also able to interact with USB devices using serial (one way communication) and parallel (with feedback). As we know USB got many problems, based on the designer, they said USB have communications issues which are lack of latency time problem and sometime the USB port is not found or unrecognized[1]. The diagnostic toolkit is important to solve these problems.

1.3 Objective

The objectives of this project are as stated below:

1. To understand Python based USB connectivity
2. To diagnose (send or receive control signal) to/from USB serial devices and record the activity.
3. To investigate USB parallel connection using Python which involved packet based information transfer.

LITERATURE REVIEW

2.1 Python as base language programming

Easiest Language Design

Programming language requires a lot of patience to learn. As a beginner language learner, It is required a lot of patience to learn, but in Python it has a easiest way to learn the language compare to other programming languages like Java, Cobra, Curl, Dylan, Z++ and others.

This is because python are easily used in an object oriented and functional design patterns, from these method users will easily access and edited the programming languages by user's intention. From the previous programmer, it said that the python are fully featured and practical; this is because the power of iterators and function-as-variables makes many users to easily determine the concise and intuitive solutions[2].

In python, the also have a modules and functions are passed as an object. Users from other language specialist will simply understand the python concept of programming languages. Comparing with the other programming languages such as Java, python programming language are more simple and powerful. The table below shows as research had been done toward it.

Variables	Python Programming Language	Java Programming Languages
Language Variable	Dynamic typing	Static typing
Defining the function	Indentation	Braces
Speed and portable	Faster but not portable	Slow but portable

Table 1: Comparison of Python and Java Programming Languages

As shown above, the python are more likely useful and easy for beginner programmer[2]. From the research that been done, the Java is entirely use by today world, but the process to create a coding is too complicated and long. But, python are direct and easily to use.

Libraries and Modules

The available of libraries and modules means that, Python have very much libraries and modules that can be extend with installing them. For example, pyusb, beautifulsoup, markupsafe, numpy, jinja2 and others[3]. These libraries usually come from third party and sometimes they are not suitable with some Python version. This will have a compatibility issues on that particular interface.

2.2 USB Connection

How USB work

USB (Universal Serial Bus) is to be considered simple, easier and flexible connections between computer and the lots of peripheral. The peripheral connection of USB includes many ways and many methods which are supported by motherboard's BIOS. For example of the device that connect using a USB port are; mouse, keyboard, printer, speaker, and etc. In USB there are 4 different transfer's modes. Which are;[4]

1. Interrupt : Transfer little amount of data but need fast response (e.g. mouse, keyboard)
2. Bulk : Which receive big packet of data (e.g. printer)
3. Isochronous : Requires streaming process (e.g. speaker, webcam)
4. Control : Short, simple command to the devices, and status responses.

The unique design of USB is, it is intended to eliminate the need for the addition of an ISA expansion card to a computer or the PCI bus, and improves the ability to plug-and-play to allow the equipment exchanged or added to the system without the need to reboot the computer. When the USB is installed, it is immediately recognized and the computer system to process the necessary device drivers to run it.[5]



Figure 1: USB Port

Before the USB device is fully used, there are serial port and parallel port that continuous used by users at that time. The serial port connection takes a byte of data and transmits the 8 bits in the byte one at a time[6] and transmit in a single row. Whereas the parallel connection work by sending 8 bits of data (1byte) at a time and transmitted parallel to each other[7]. So, from this advance of technology, people move to use USB connection for better speed compare to the serial port connection and parallel port connection is a bit slow.

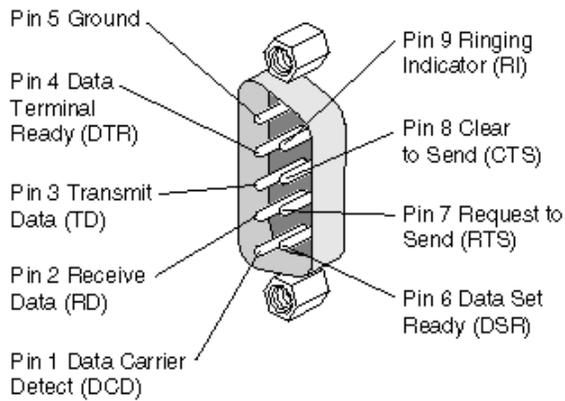


Figure 2: Serial Port and Parallel Port

All USB data is sent serial connection, the process is starting from a least significant bit (LSB) then go to the High significant bit (HSB). USB will transfer data in form of packets of data which it sent back and forth between the host and the peripheral devices. All the packets of data are sent from the host to the devices via a root hub. In reply, some packets are sending to the device directly. Each of transfers USB consists of:

- 1) Token Packet
- 2) Data Packet
- 3) Status or handshake Packet

Token packet is generated by the host to describe what is to follow and whether the data transfer will be a read or write. The second packet is data packet which it is carrying the content information. For the last packet it is a status or handshake packet which used to acknowledge transactions and to provide a means of error correction.

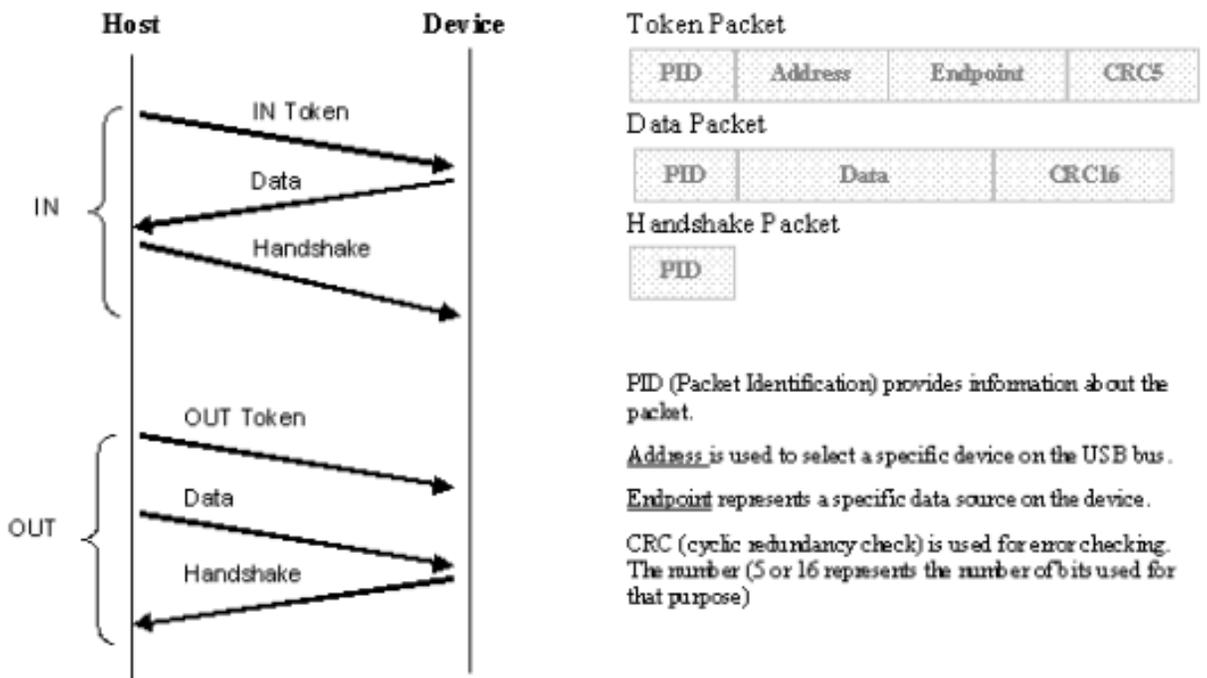


Figure 3: USB Basic Communication Flow

METHODOLOGY

3.1 Research Methodology

To smoothness the flow of this project, the Gantt chart has been established in the pre-project stage in order to have specific dateline indicator for the each task ahead that is required by the course.

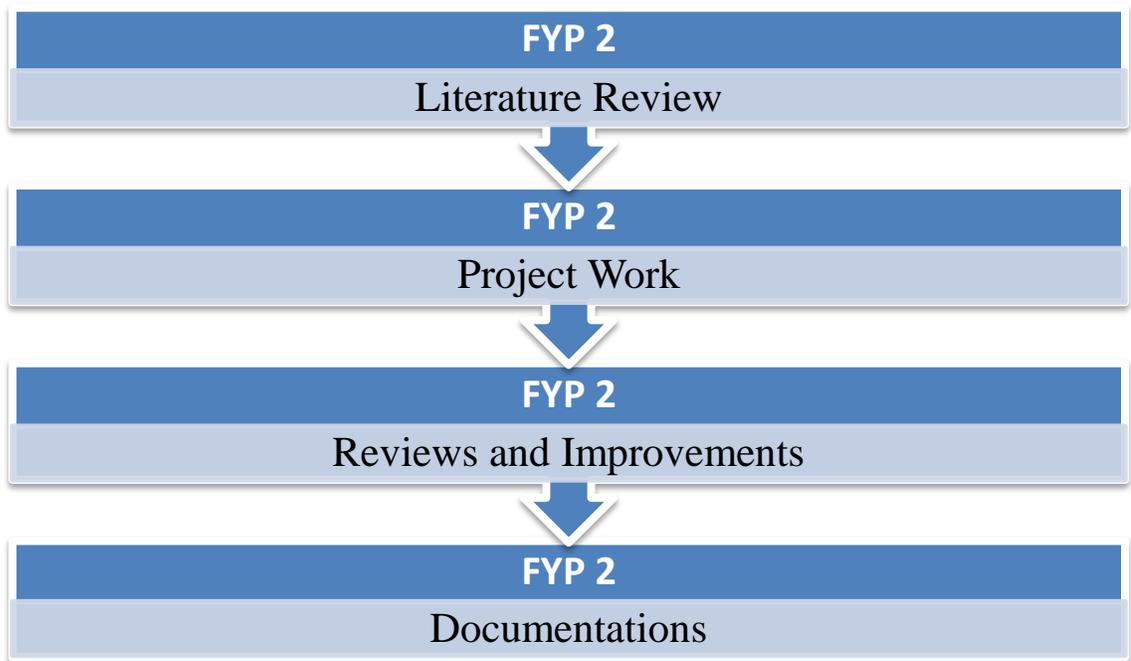


Figure 4: Flow of Mmethodology

Literature Review

For the literature review, all the information are searching and then the collected information and resources are gathered to make a beautiful and complete literature review. This information and resource mainly focus on the USB connectivity and Python programming language. The information are collected from the book, thesis paper, internet sources and others resource.

Project Work

In this stage the collected information and resource had been analyze to see the problem occur. Within it, the coding for python programming language are create based on the literature review and other analyzing result.

Review and Improvement

During the coding was created, the review and any improvement are made to meet an expectation. This project was guided by supervisor for better improvement. It is very important for us to know the originality of the project itself.

Documentation

Documentation is a last stage for compilation. At this stage all the create project and result are compile in one file to submitted to the organizer.

Python 2.7.5

As a beginner, to run Python software in our computer is first to know how to install it. The software can be downloads from many source that provide it. But, in this time the latest version of the Python software is Python 2.7.5. As know, python is open source software, where with it people can used, changed, and distribute the software to anyone with licensed.

Below shows the step on how to install the Python 2.7.5 and its module with are PyUSB 1.0, pywinusb-0.3.2, libusb1-1.1.0 and pyserial-2.7. The installed modules are needed to be installing in OS for communication and access to USB.

For the installation of the Python 2.7.5 software it is very common, mostly we can install directly with the given instruction in that software. But for the module it is a bit complicated.

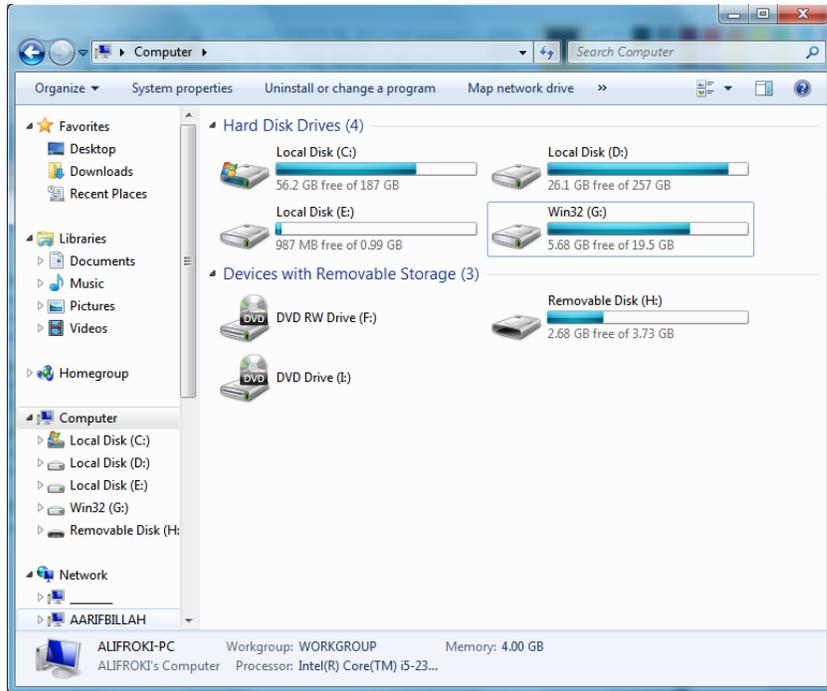


Figure 5: Step 1 Python 2.7.5 Module Installation

Open my computer and then click at the system properties tab.

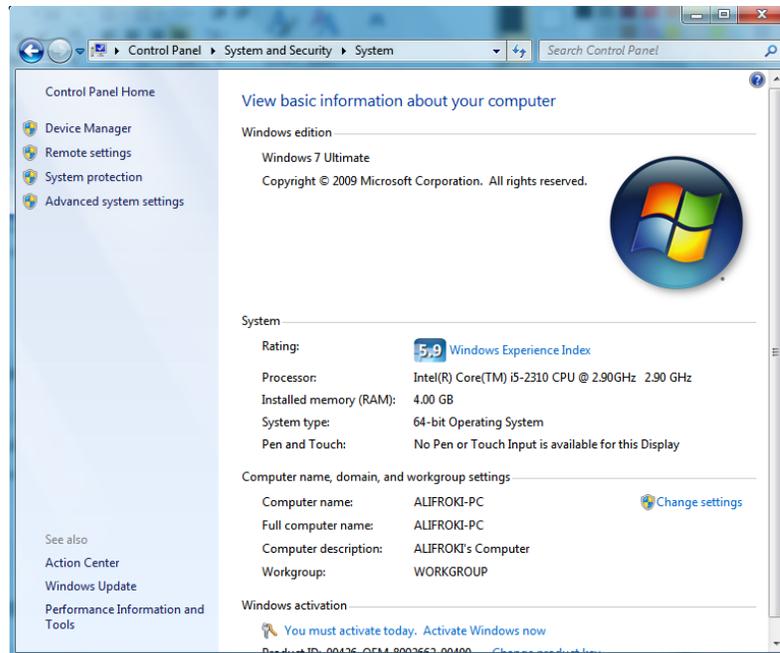


Figure 6: Step 2 Python 2.7.5 Module Installation

Then click at advanced system setting tab at the left column. The below windows should pop up.

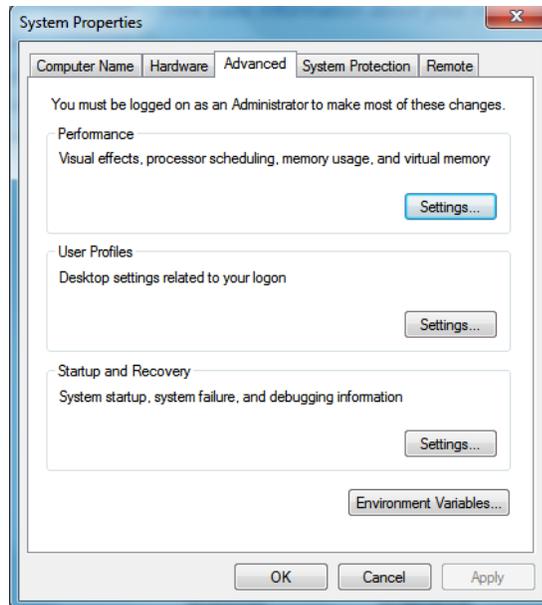


Figure 7: Step 3 Python 2.7.5 Module Installation

Then click at the environment variable at the advanced tab of the System Properties window. The Environment Variables should show.

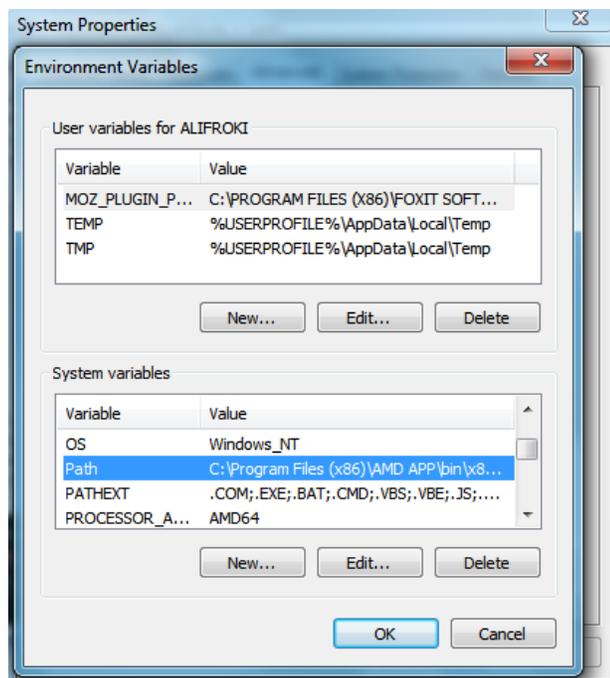


Figure 8: Step 4 Python 2.7.5 Module Installation

Follow exactly as shown in the figure 5. Then, click at System variables place to find the 'path' variable. Then click on it.

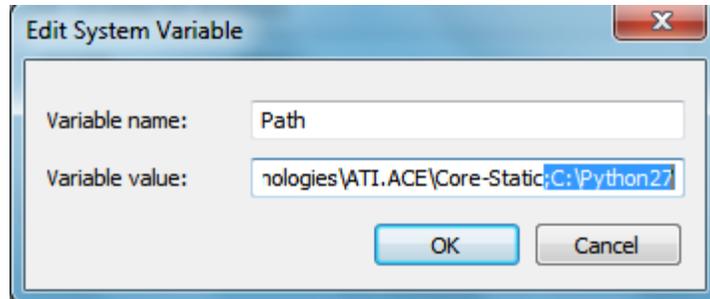


Figure 9: Step 5 Python 2.7.5 Module Installation

At the end of the variable value, add the ‘;C:\Python27’ as shown. Or add type where are your Python 2.7.5 file destination as your install at the beginning of the installation here. Then click ‘OK’.

In the next step, go to the Start button of your computer. Then type ‘cmd’ and then enter. This will appear.

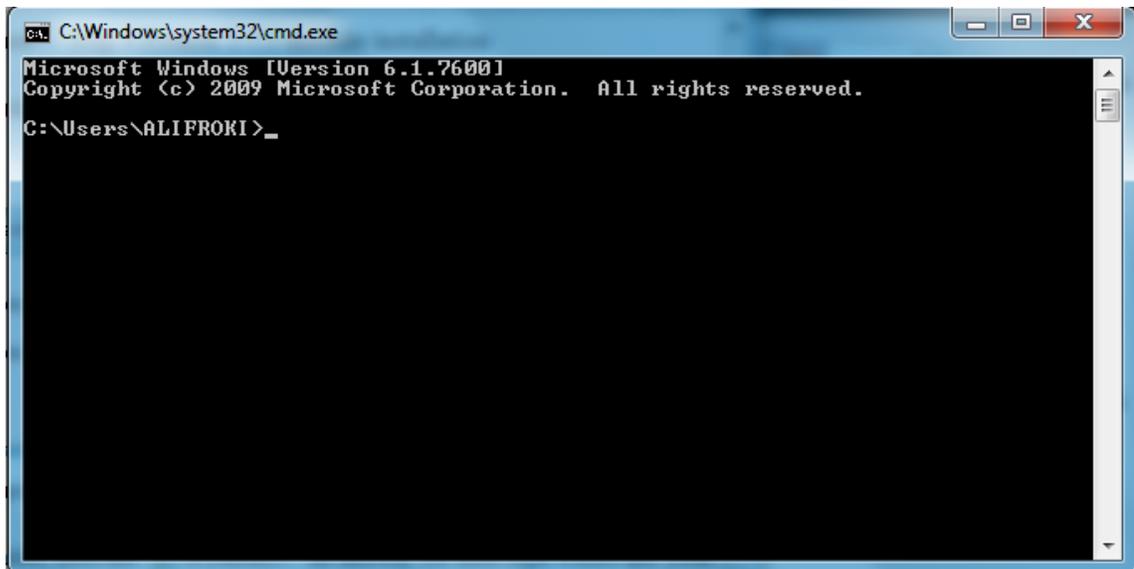
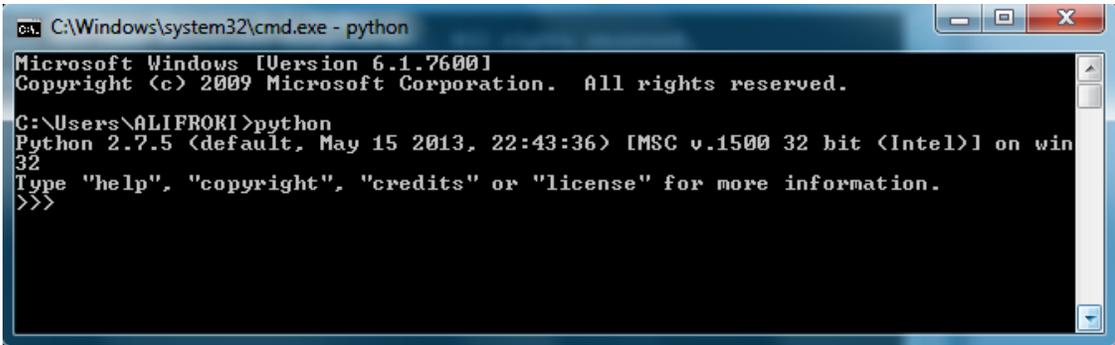


Figure 10: Step 6 Python 2.7.5 Module Installation

By typing ‘python’ on the cmd. We will be directly in the Python 2.7.5 interface.

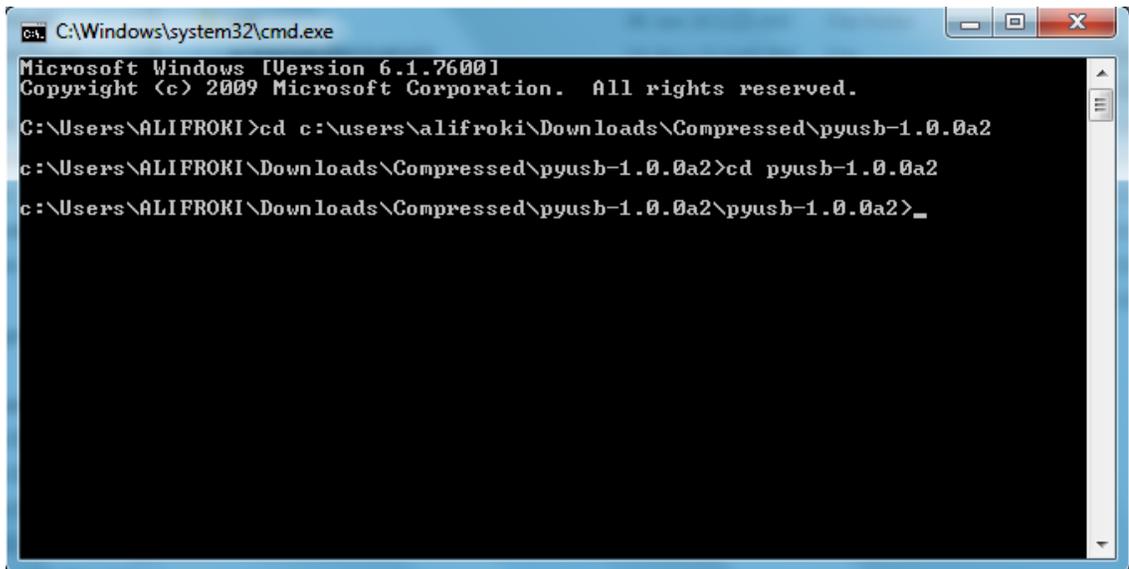


```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ALIFROKI>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 11: Step 7 Python 2.7.5 Module Installation

Then, to install the module, it needs to install in this cmd windows. For example, installing a PyUSB 1.0.0a2 (need to be downloading from website) shows below.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ALIFROKI>cd c:\users\alifroki\Downloads\Compressed\pyusb-1.0.0a2
c:\Users\ALIFROKI\Downloads\Compressed\pyusb-1.0.0a2>cd pyusb-1.0.0a2
c:\Users\ALIFROKI\Downloads\Compressed\pyusb-1.0.0a2\pyusb-1.0.0a2>_
```

Figure 12: Step 8 Python 2.7.5 Module Installation

Type as shown or where the PyUSB folder it located. For example “c:\users\alifroki\Downloads\Compressed\pyusb-1.0.0a2\pyusb-1.0.0a2”

Then the last command needs to write “python setup.py install”. The installation process is made by itself.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ALIFROKI>cd c:\users\alifroki\Downloads\Compressed\pyusb-1.0.0a2
c:\Users\ALIFROKI\Downloads\Compressed\pyusb-1.0.0a2>cd pyusb-1.0.0a2
c:\Users\ALIFROKI\Downloads\Compressed\pyusb-1.0.0a2\pyusb-1.0.0a2>python setup.
py install
running install
running build
running build_py
creating build
creating build\lib
creating build\lib\usb
copying usb\control.py -> build\lib\usb
copying usb\core.py -> build\lib\usb
copying usb\legacy.py -> build\lib\usb
copying usb\util.py -> build\lib\usb
copying usb\_debug.py -> build\lib\usb
copying usb\_interop.py -> build\lib\usb
copying usb\_init__.py -> build\lib\usb
creating build\lib\usb\backend
copying usb\backend\libusb01.py -> build\lib\usb\backend
copying usb\backend\libusb10.py -> build\lib\usb\backend
copying usb\backend\openusb.py -> build\lib\usb\backend
copying usb\backend\_init__.py -> build\lib\usb\backend
running install_lib
running install_egg_info
Removing C:\Python27\Lib\site-packages\pyusb-1.0.0a2-py2.7.egg-info
Writing C:\Python27\Lib\site-packages\pyusb-1.0.0a2-py2.7.egg-info
c:\Users\ALIFROKI\Downloads\Compressed\pyusb-1.0.0a2\pyusb-1.0.0a2>
```

Figure 13: Step 9 Python 2.7.5 Module Installation

For the other module, the method for installation is same as above. So, with this installation we can be access the USB device with some command shown below in the PyUSB 1.0 section.

PyUSB 1.0

To be understood further, a research had been made based on the PyUSB 1.0, from understanding, PyUSB 1.0 are library and module for programmer to access the USB device in their project. These PyUSB 1.0 is very helpful in term of communication to the USB device which programmer can access the information in or out from the USB. It is also let programmer feel less painful because to program and communicate via USB device is playing with complex protocol[8]. There are some features that make PyUSB 1.0 is easily to handle which are:

- 1) Portability
- 2) Easiness
- 3) Support Isochronous Transfers
- 4) 100% written in Python language

Creating Coding

First of all, to creating a coding it needs to understand the behavioral of USB peripheral and complex protocol of USB itself. From the PyUSB 1.0 modules it will reduce the complex part and replace with easier way to access the USB device.

The information was gathering from PyUSB 1.0 tutorial website[9].

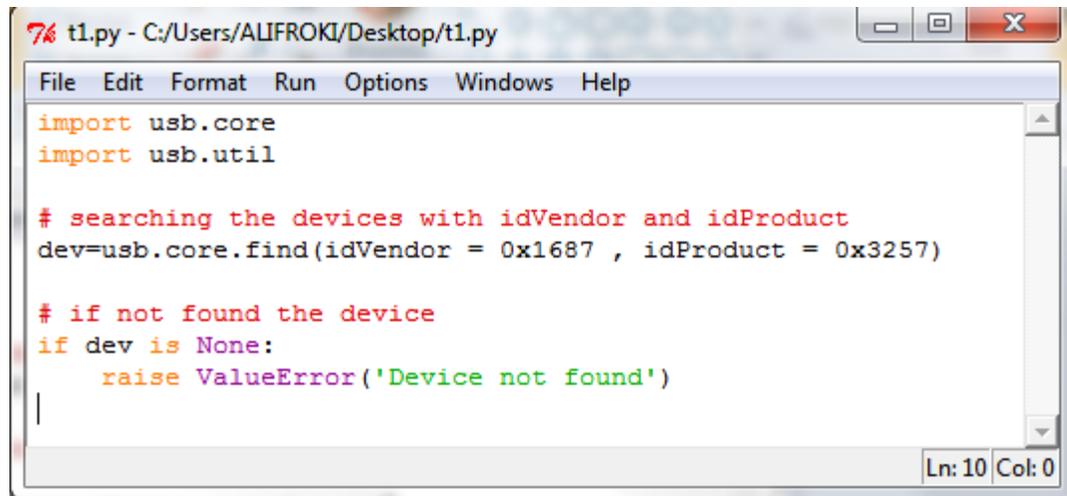
Content	Description
usb.core	Main of USB module
usb.util	Utility functions
control	Standard control requests
legacy	The 0.x version compatibility layer
backend	A sub package containing the built-in backbends

Table 2: Description of PyUSB Modules.

Based on the tutorial of pyUSB[9], the example is given on how to import the core module:

```
>>> import usb.core
>>> dev = usb.core.find()
```

As shown above, the imported module is an usb.core, this will allow a programmer to access any kind of information in or out of the USB devices. Then, for the dev = usb.core.find() is for searching a devices in the Python 2.7 software interface, whether it find or not is depending on the USB Id product and Id vendor. The example of coding in python language is shown below on how to access the USB devices. The main content of the programming shown below,

A screenshot of a Python IDE window titled 't1.py - C:/Users/ALIFROKI/Desktop/t1.py'. The window contains the following Python code:

```
import usb.core
import usb.util

# searching the devices with idVendor and idProduct
dev=usb.core.find(idVendor = 0x1687 , idProduct = 0x3257)

# if not found the device
if dev is None:
    raise ValueError('Device not found')
```

The status bar at the bottom right of the window shows 'Ln: 10 Col: 0'. The IDE has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Windows', and 'Help'.

Figure 14: Example how to find the Device using USB Module.

From the first two lines we can see that, the coding is importing PyUSB package modules. `usb.core` and `usb.util` which is the main module and contains utility functions respectively. The next line showed that `#searching the devices with idVendor and idProduct`, which mean that we need to find our USB devices that connect to the PC. For example the `idproduct` is `0x1687` and the `idvendor` is `0x3257`, Python will look it up until it finds the specific number that had been assigning. To know our device `idproduct` and `idvendor`, we can find it form `usbview` software. Which can be downloaded here[10]. But there are other method of coding for searching all the devices without putting the `idVendor` and `idProduct`. Such as

```
dev = usb.core.find(find_all=True)
```

In the second command which is “if dev is None”, to determine the coding if not found. Then the “Device not found” statement will display.

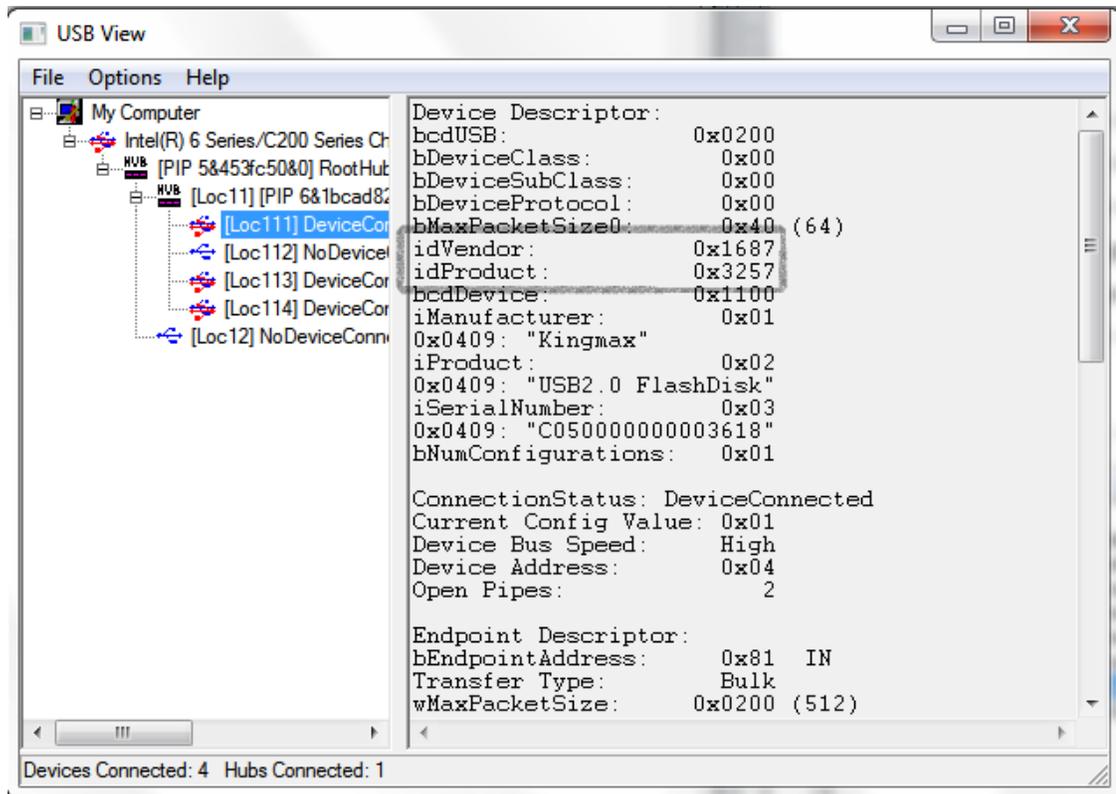


Figure 15: USBview Software that views the Component of USB Devices.

USB to Communicate

As mentions at the previous session, to communicate with USB devices first things first we should know the USB has four flavors of transfers: which are Interrupt, Bulk, Isochronous and Control.

In short, for the interrupt type of transfer, USB devices send and receive small amounts of data infrequently or in an asynchronous time frame. This type of transfer are surely guarantees a maximum service period and also will make a re-attempted delivery in the next period if there is an error happen on the bus. It also can operate for low, full and high speed devices. The maximum packets are 8 byte or less for the low-speed devices, 64bytes or less for full-speed devices, and 1024 bytes or less for high-speed devices.

Besides, for the bulk transfer is typically used by all the devices which transferring a huge data, such as printer and scanner that used any available bandwidth. From the

transfer itself, this transferring data are actually can allow access to the bus on an 'as-available' basis and also providing an error checking if there are something bad happen. The maximum packet size for the bulk transfers is can be whether 8, 16, 32, or 64byte.

The Isochronous transfer type is the most commonly used by time dependent information, for example as a telecommunication system and others multimedia transfer. This transfer may not support the low-speed devices. And the last is the control transfer type, Control transfers are primarily intended to support configuration, command and operational status of the software on the host and the device.

From this information, the coding is design based on the transfer; the coding is shown in the result section. For the USB thumb drive that used in this project is a bulk type of transfer as shown in the usbview software. Once, we know the type of transfer we are easily to derive the coding.

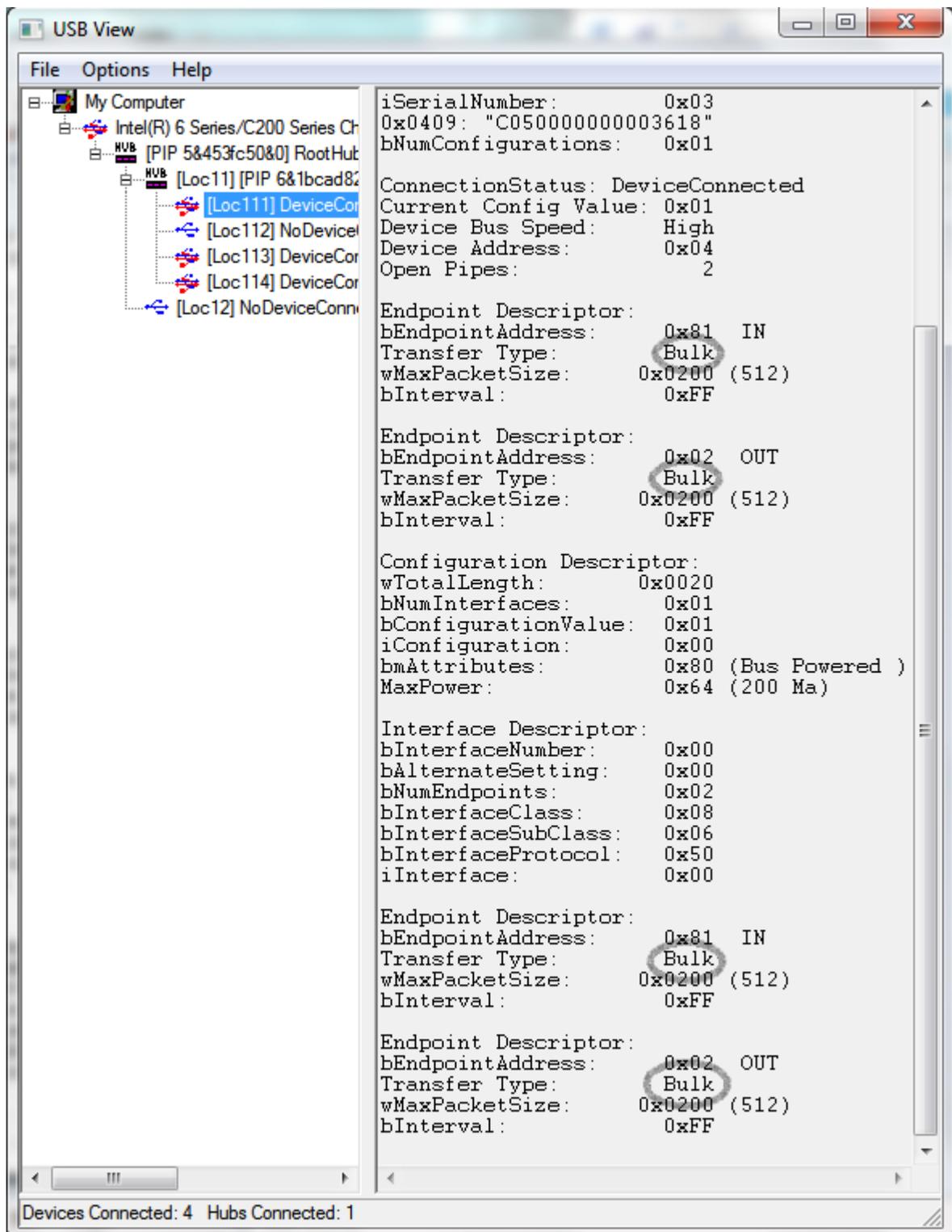
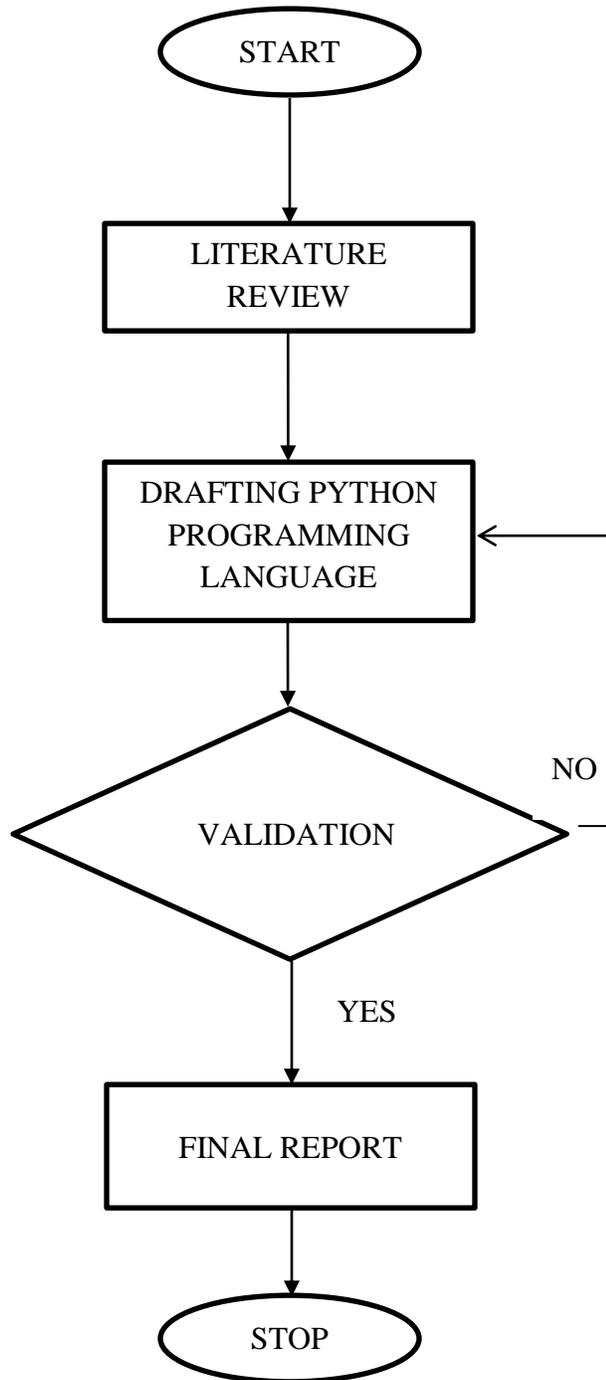


Figure 16: USBview shows the Bulk Type Transfer of Thumb Drive

3.2 Key Milestone



-
- Identify the problem occurring in USB and Python
 - Find the information and thesis about the previous project.
-

- Verify the information
 - Prepare the tools and equipment.
 - Analyze the data to validate the result.
 - If not validate, repeat the process.
-

- Report on the result and finding based on the programming had been done.
 - Recommendation for further research and development.
-

Figure 17: The Methodology of the Project

3.3 Gantt Chart

Department	Activities	FYP I																			
		MONTH	May		Jun			July				August				September					
		WEEK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
Electrical and Electronic Engineering	First title release																				
	Second title release																				
	Third title release																				
	FYP 1 Title confirmation (USB Connectivity Using Python)																				
	Extended Proposal																				
	– Background Study																				
	– Objectives																				
	– Scope of work																				
	– Problem Statement																				
	– Literature review																				
	– Methodology																				
	– Conclusion																				
	Submission of Extended Proposal																				
	Proposal Defence																				
	Project in Progress																				
	Submission of Interim Draft Report																				
	Submission of Interim Report																				
			FYP II																		
			MONTH	September			October			November				Disember				January			
			WEEK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
		Project Work Continues																			
		Submission of Progress Report																			
		Project Work Continues																			
		Pre-EDX																			
		Submission of Draft Report																			
		Submission of Dissertation (soft bound)																			
	Submission of Technical Paper																				
	Oral Presentation																				
	Submission of Project Dissertation (Hard bound)																				

Table 3: Gantt chart for the whole semester project progress

3.4 Tools

- 1) Python 2.7.5 software for windows 7 64bits.
- 2) Thumb Drive as USB devices
- 3) Desktop Personal Computer

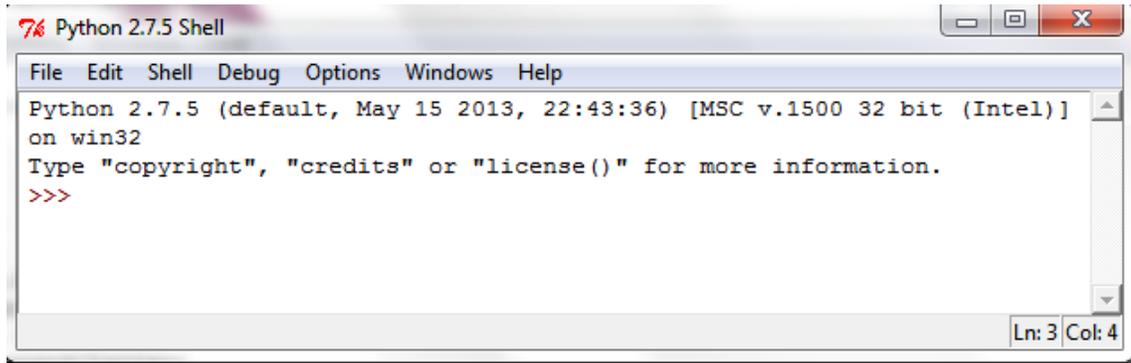


Figure 18: Python 2.7.5 software interface

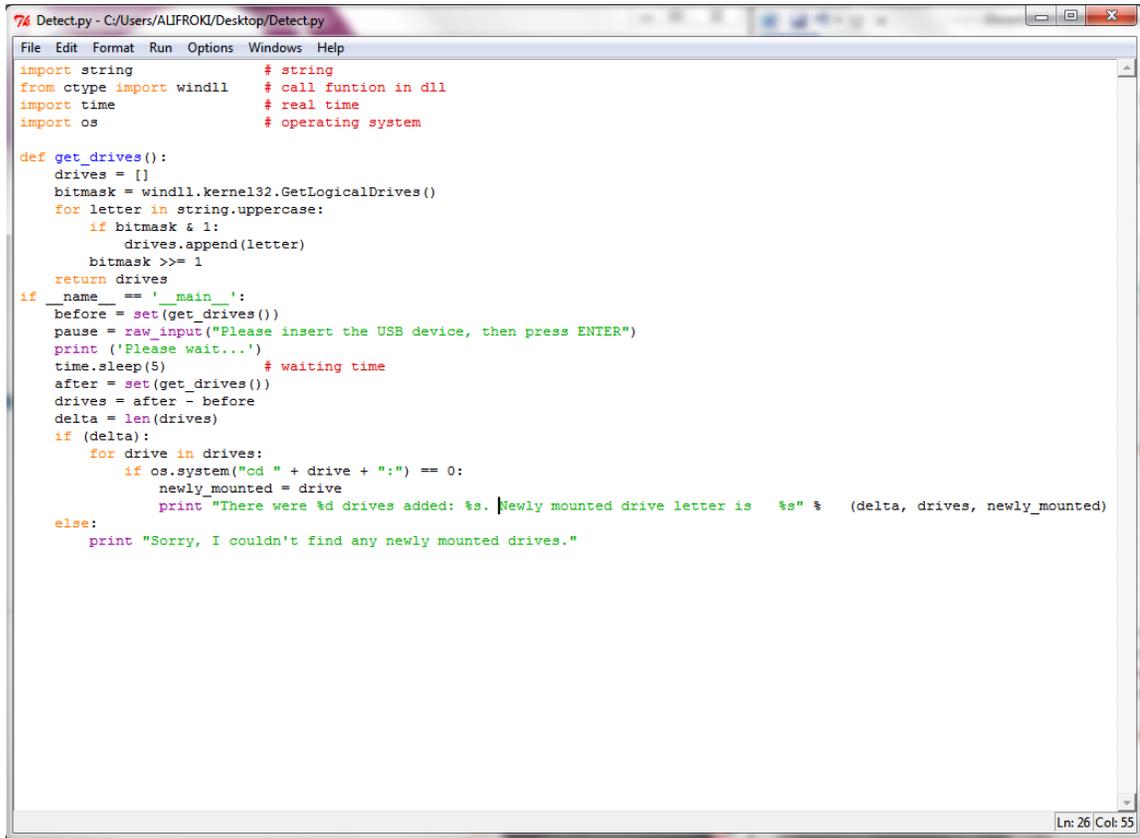


Figure 19: Thumb drive and PC that use for this project

RESULT AND DISCUSSION

Result and discussion

The first step of the coding was made to communicate with USB. The coding shows below:



```
7% Detect.py - C:/Users/ALIFROK/Desktop/Detect.py
File Edit Format Run Options Windows Help
import string          # string
from ctypes import windll # call funtion in dll
import time           # real time
import os             # operating system

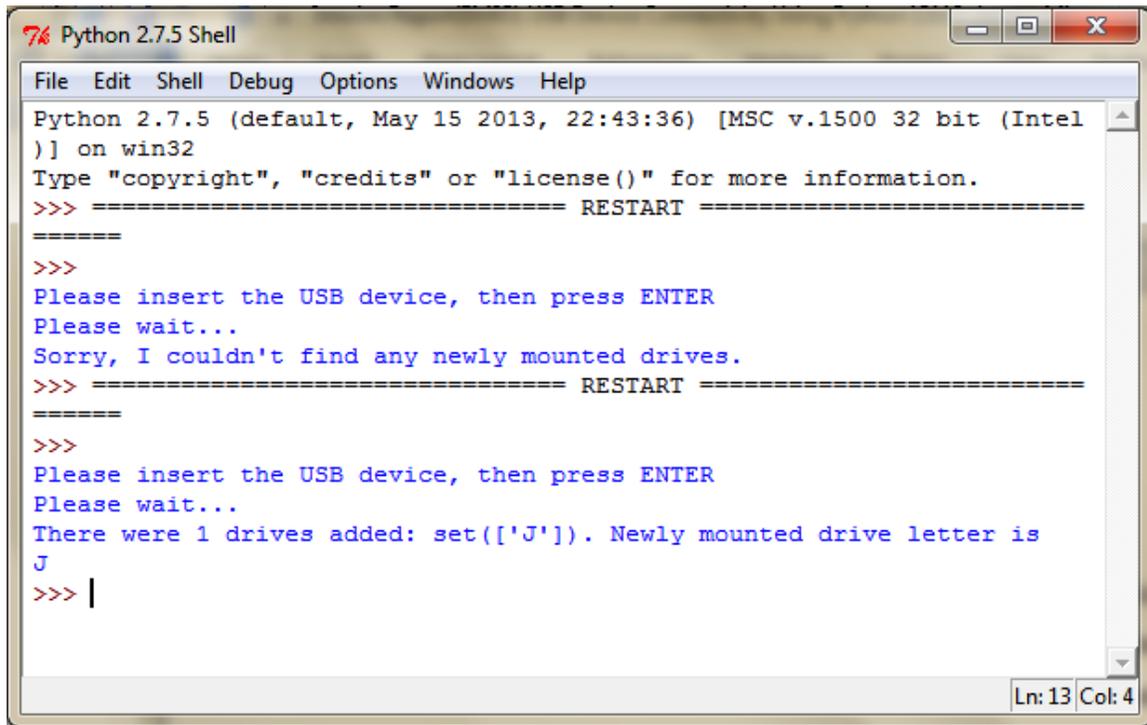
def get_drives():
    drives = []
    bitmask = windll.kernel32.GetLogicalDrives()
    for letter in string.uppercase:
        if bitmask & 1:
            drives.append(letter)
            bitmask >>= 1
    return drives

if __name__ == '__main__':
    before = set(get_drives())
    pause = raw_input("Please insert the USB device, then press ENTER")
    print ('Please wait...')
    time.sleep(5) # waiting time
    after = set(get_drives())
    drives = after - before
    delta = len(drives)
    if (delta):
        for drive in drives:
            if os.system("cd " + drive + ":") == 0:
                newly_mounted = drive
                print "There were %d drives added: %s. Newly mounted drive letter is %s" % (delta, drives, newly_mounted)
    else:
        print "Sorry, I couldn't find any newly mounted drives."

Ln: 26 Col: 55
```

Figure 20: Coding for detect USB device in python interface

The result shows below when I insert the USB device:

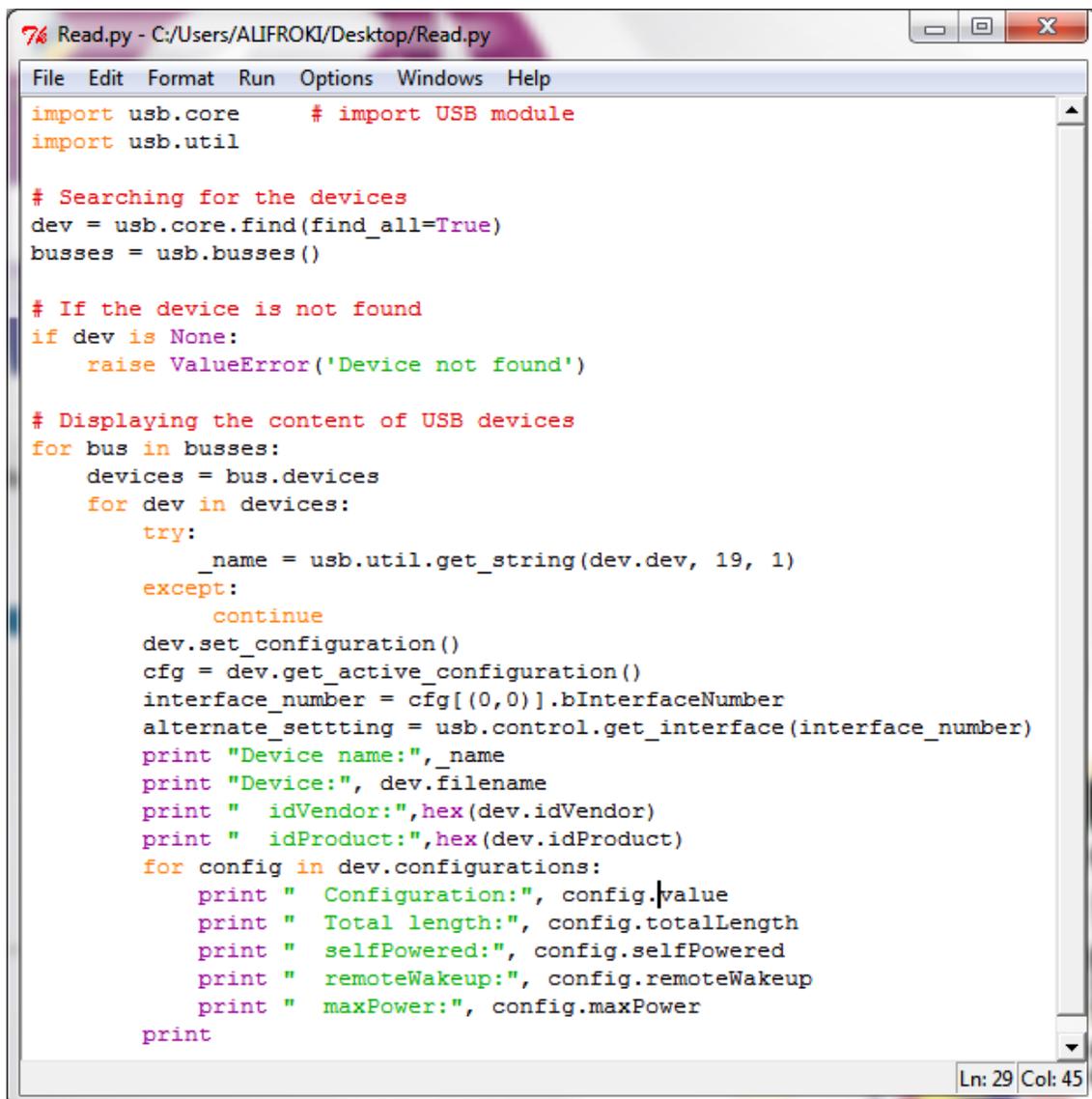


```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel
)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
>>> Please insert the USB device, then press ENTER
Please wait...
Sorry, I couldn't find any newly mounted drives.
>>> ===== RESTART =====
>>>
>>> Please insert the USB device, then press ENTER
Please wait...
There were 1 drives added: set(['J']). Newly mounted drive letter is
J
>>> |
```

Figure 21: Result for the detection of USB device in python interface

Below show the coding that we are trying to listing all the information in the USB device (reading USB device).

Coding:

A screenshot of a Python IDE window titled "Read.py - C:/Users/ALIFROKI/Desktop/Read.py". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The main area contains Python code for finding and reading USB devices. The code imports the "usb" module, searches for devices, and if found, iterates through buses and devices to print their configuration details. The status bar at the bottom right shows "Ln: 29 Col: 45".

```
import usb.core      # import USB module
import usb.util

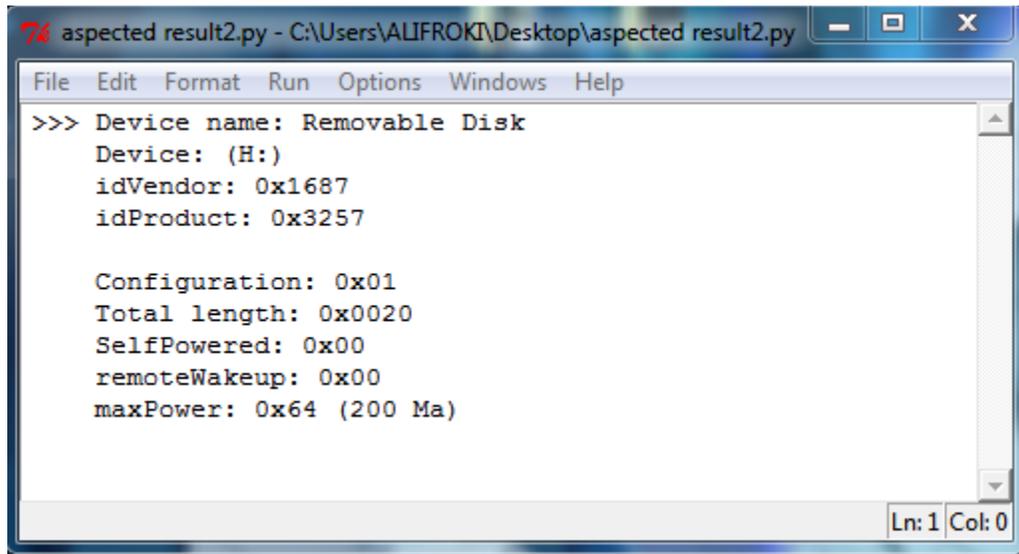
# Searching for the devices
dev = usb.core.find(find_all=True)
busses = usb.busses()

# If the device is not found
if dev is None:
    raise ValueError('Device not found')

# Displaying the content of USB devices
for bus in busses:
    devices = bus.devices
    for dev in devices:
        try:
            _name = usb.util.get_string(dev.dev, 19, 1)
        except:
            continue
        dev.set_configuration()
        cfg = dev.get_active_configuration()
        interface_number = cfg[(0,0)].bInterfaceNumber
        alternate_setting = usb.control.get_interface(interface_number)
        print "Device name:", _name
        print "Device:", dev.filename
        print " idVendor:", hex(dev.idVendor)
        print " idProduct:", hex(dev.idProduct)
        for config in dev.configurations:
            print " Configuration:", config.value
            print " Total length:", config.totalLength
            print " selfPowered:", config.selfPowered
            print " remoteWakeup:", config.remoteWakeup
            print " maxPower:", config.maxPower
        print
```

Figure 22: Coding for reading the USB thumb drive.

Expected Result:

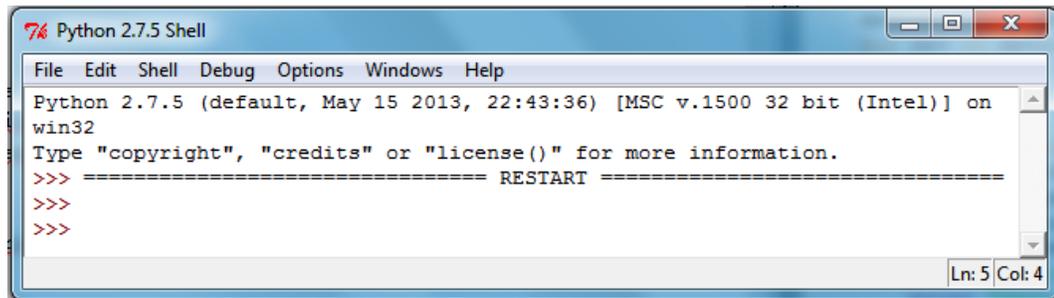


```
aspected result2.py - C:\Users\ALIFROKI\Desktop\aspected result2.py
File Edit Format Run Options Windows Help
>>> Device name: Removable Disk
Device: (H:)
idVendor: 0x1687
idProduct: 0x3257

Configuration: 0x01
Total length: 0x0020
SelfPowered: 0x00
remoteWakeup: 0x00
maxPower: 0x64 (200 Ma)
Ln: 1 Col: 0
```

Figure 23: Expected Result of reading USB devices

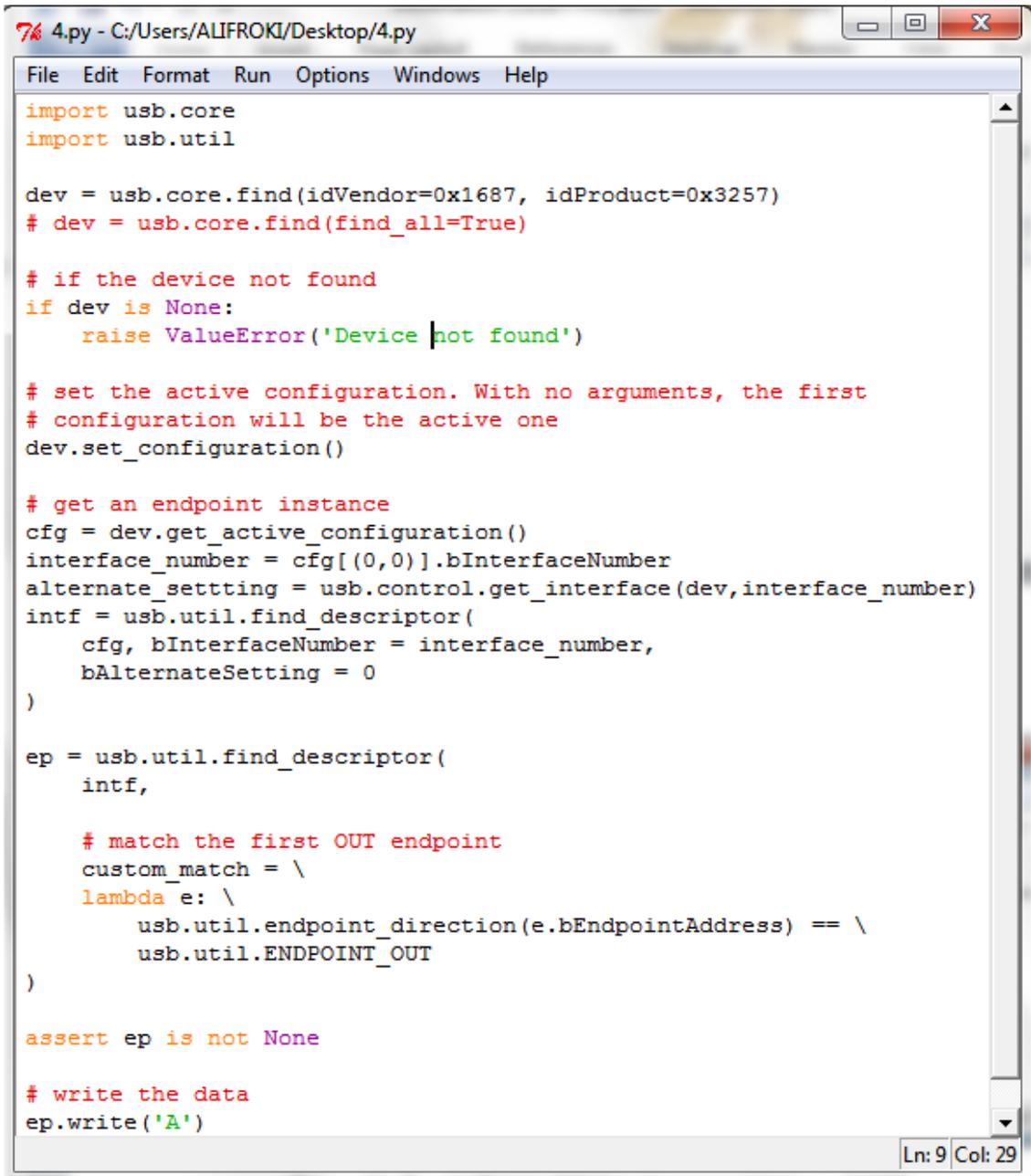
The real result shown below:



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
>>>
Ln: 5 Col: 4
```

Figure 24: Result shown of failure in listing the information from USB

Then, we are trying to communicate with USB device (writing in USB device):

A screenshot of a Python IDE window titled "4.py - C:/Users/ALIFROKI/Desktop/4.py". The window contains Python code for finding a USB device and writing to it. The code includes imports for 'usb.core' and 'usb.util', a function to find a device by vendor and product ID, error handling for device not found, configuration setting, and endpoint finding. The code ends with an assertion and a write operation.

```
import usb.core
import usb.util

dev = usb.core.find(idVendor=0x1687, idProduct=0x3257)
# dev = usb.core.find(find_all=True)

# if the device not found
if dev is None:
    raise ValueError('Device not found')

# set the active configuration. With no arguments, the first
# configuration will be the active one
dev.set_configuration()

# get an endpoint instance
cfg = dev.get_active_configuration()
interface_number = cfg[(0,0)].bInterfaceNumber
alternate_setting = usb.control.get_interface(dev, interface_number)
intf = usb.util.find_descriptor(
    cfg, bInterfaceNumber = interface_number,
    bAlternateSetting = 0
)

ep = usb.util.find_descriptor(
    intf,

    # match the first OUT endpoint
    custom_match = \
        lambda e: \
            usb.util.endpoint_direction(e.bEndpointAddress) == \
            usb.util.ENDPOINT_OUT
)

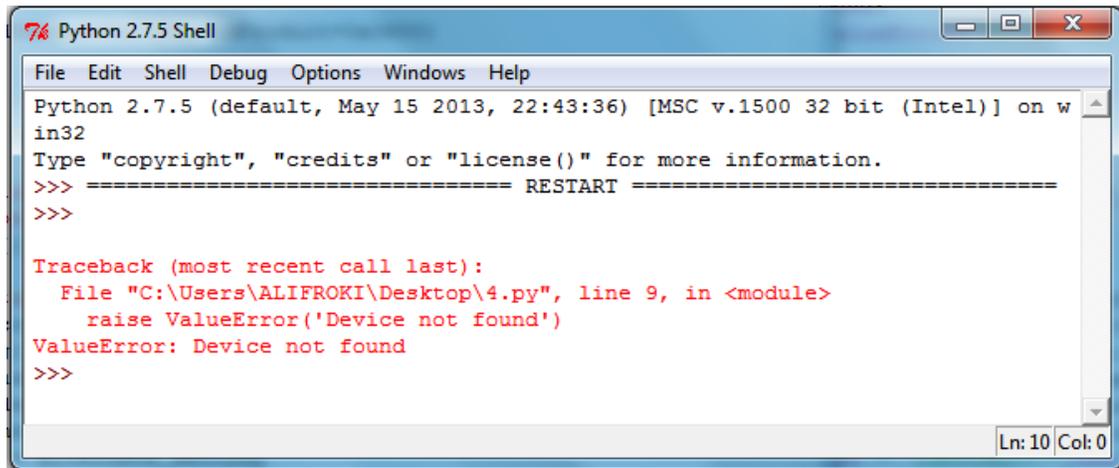
assert ep is not None

# write the data
ep.write('A')
```

Ln: 9 Col: 29

Figure 25: Coding for writing in USB thumb drive

The real result shown below:



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on w
in32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Traceback (most recent call last):
  File "C:\Users\ALIFROKI\Desktop\4.py", line 9, in <module>
    raise ValueError('Device not found')
ValueError: Device not found
>>>
```

Figure 26: Result shown of failure in communication of USB.

As obtaining the result, we are searching to the error that show in figure 5, we found out there is no such solution for this error, there are probably cause of the Python 2.7 software itself. Then the project cannot be proceeding unless using another method that makes us to discover the perfect technique to determine it compatibility. Based on research that made, we understand this is causing of the compatibility with the windows 7 64bits. Some software is not compatible with these windows. As supported by literature review, the library and module of pyUSB 1.0 come from third party, and it support with python 2.7.5 but it is not supported with Windows 7 64bit. Previous project had done by researcher not in Windows 7 64bit. So, we conclude that with Windows 7 64bit cannot be done a USB device connectivity using python. Within time constraint we suggest for another researcher to do this project or just continue to research on it in another time. From our experience we are suggesting to use another method that use to communicate with USB devices.

CONCLUSION

In conclusion, this project research aims to understand Python based USB connectivity. By the first objective of the project it is achievable, which is to understand the Python and USB connectivity. But, as a result stated from these project, it show that the programming itself have a difficulty to determine the solution. In which these case there are no such solution for that error. According to the key milestone and project details it is already set in order to achieve the objectives. So, by our conclusion and the hypothesis, to make a better USB connectivity using python programming language, it is better to make by using another method. The programming interface itself is stable but the content of the library or module is not stable for the windows 7 64bit. We are suggesting this project need to be continuing in another future to prove that the working coding with the perfect method which trying to do in OS windows 7 32bits or any better solution. [11]

REFERENCES

- [1] J. Hyde, *USB Design by Example: A Practical Guide to Building I/O Devices with CD-ROM*: Intel Press, 2002.
- [2] K. Mikoluk. (30 December). *Python vs Java: Key Differences* [Online]. Available: <https://www.udemy.com/blog/python-vs-java/>
- [3] (6 January). *Third-party Libraries in Python 2.7* [online]. Available: <https://developers.google.com/appengine/docs/python/tools/libraries27>
- [4] W.-J. Wu, "USB connection-detection circuitry and operation methods of the same," ed: Google Patents, 2001.
- [5] R.-Y. Shieh, G. Tsai, Y.-A. Chen, and C.-L. Chang, "Externally connection type USB2. 0 interface flash card reader," ed: Google Patents, 2001.
- [6] J. Tyson, "How Serial Ports Work," 09 February 2001.
- [7] J. Tyson, "How Parallel Ports Work," 03 october 2000.
- [8] J. Axelson, *USB complete: everything you need to develop custom USB peripherals*: lakeview research llc, 2005.
- [9] (30 November). *Programming with PyUSB 1.0* [Online]. Available: <http://pyusb.sourceforge.net/docs/1.0/tutorial.html>
- [10] (11 November). *USBView* [Online]. Available: <http://msdn.microsoft.com/en-us/library/windows/hardware/ff560019%28v%3Dvs.85%29.aspx>
- [11] (11 November). *libusb-win32* [Online]. Available: <http://sourceforge.net/apps/trac/libusb-win32/wiki>

HowTo Install LibUSB on Windows 7

LibUSB 1.2.1

Pinguino need libusb to communicate with your computer. Do not install a previous version of LibUSB on windows 7, only use the version 1.2.1 with a windows 7 computer.

Download this compressed file:

<http://sourceforge.net/projects/libusb-win32/files/libusb-win32-releases/1.2.1.0/libusb-win32-src-1.2.1.0.zip/download>

and extract it in a new folder in your personal folder.

Disconnect your pinguino.

Then you need to know what is the processor in your computer.

If your computer is a **X86 32 bits system**:

- go to the libusb-win32-bin-1.2.1.0\bin\x86 folder,
- rename the file `libusb0_x86.dll` to `libusb0.dll`,
- copy this new file in the `c:\Windows\system32\` folder,
- copy the file `libusb0.sys` to the `c:\Windows\system32\drivers\` folder,

If your computer is a **X86 64 bits system**:

- go to the libusb-win32-bin-1.2.1.0\bin\x86 folder,
- rename the file `libusb0_x86.dll` to `libusb0.dll`,
- copy this new file in the `c:\Windows\syswow64\` folder,
- copy the file `libusb0.sys` to the `c:\Windows\system32\drivers\` folder,

If your computer is an **AMD 64 bits system**:

- go to the libusb-win32-bin-1.2.1.0\bin\amd64 folder,
- copy this new file in the `c:\Windows\system32\` folder,
- copy the file `libusb0.sys` to the `c:\Windows\system32\drivers\` folder,

If your computer is an **intel IA64 bits system**:

- go to the libusb-win32-bin-1.2.1.0\bin\ia64 folder,
- copy the file `libusb0.dll` in the `c:\Windows\system32\` folder,
- copy the file `libusb0.sys` to the `c:\Windows\system32\drivers\` folder,

Pinguino driver

Then you can download the windows 7 Pinguino driver on hackinglab:

<http://www.hackinglab.org/pinguino/download/driver%20pinguino%20windows/driver%20pinguino%20w7.zip>

extract this file in the examples folder of libusb-win32-bin-1.2.1.0 folder.

Pinguino

Connect your Pinguino, and go to the control panel, system ,Device Manager

Pinguino is a non recognised device (with the warning icon).

Right click on the device, then click properties.

You must now install the driver for Pinguino. Click on the update driver button, then go to the folder with Pinguino driver (the one you downloaded before).

Then click OK.